TAG+6

Proceedings of the Sixth International Workshop on Tree Adjoining Grammars and Related Frameworks

Universitá di Venezia
20–23 May 2002

TAG+6 has been sponsored by:

Dipartimento di Elettronica e Informatica
Universitá di Padova

Dipartimento di Scienze del Linguaggio
Universitá Ca' Foscari, Venezia

Institute for Research in Cognitive Science

Institute for Scientific and Technological Research

Hello Venezia

Preface

The papers collected in this volume were presented at the Sixth International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+6), held at the University of Venice in May 2002. Previous TAG workshops took place at Schloß Dagstuhl (1990), at the University of Pennsylvania (1992, 1998), and at the University of Paris 7 (1994, 2000).

The Tree Adjoining Grammar (TAG) formalism has been studied for some time, both for its mathematical properties and computational applications, as well as for its role in constructing grammatical theories and models of language processing. Over the years, these lines of inquiry have fed off of one another: empirical consequences have been derived from TAG's mathematical restrictiveness, and extensions to the TAG formalism have been motivated by the exigencies of grammatical analysis. One of the main goals of TAG+6 was to bring together the full range of researchers interested in the TAG formalism, continuing the productive interactions that have been the hallmark of TAG research. The success of the meeting can be judged by the range of topics explored by the papers collected here, including linguistic theory, mathematical properties of grammar formalisms, computational and algorithmic studies of parsing and generation, psycholinguistic modeling, and applications to natural language processing systems.

It has been observed for some time that a range of grammatical frameworks, for example minimalist syntax, categorial grammar, dependency grammars, HPSG, and LFG, share with TAG a number of significant properties, including the lexicalization of syntactic structure, a conception of syntactic derivation rooted in generalized transformations, a simple notion of locality for grammatical dependencies, and mildly context sensitive generative capacity. A second main goal of TAG+6, and the reason for the '+' in the workshop's name, was to better understand the connections between TAG and other related grammatical frameworks. Such connections are explored in a number of the papers in this volume. In addition to these contributed papers, interframework connections were further elucidated during the workshop by three invited speakers each representing a different grammatical framework: Joan Bresnan, Guglielmo Cinque, and Jan Hajič. Hajič's contribution is preserved here in written form.

This workshop would not have been possible without the hard work of the program committee and the organizing committee. Members of the program committee were Anne Abeillé, William Badecker, Srinivas Bangalore, Tilman Becker, Tonia Bleam, Mark Dras, Fernanda Ferreira, Claire Gardent, Anthony Kroch, Seth Kulick, David Lebeaux, Larry Moss, Gertjan van Noord, Richard Oehrle, Martha Palmer, Owen Rambow, Norvin Richards, James Rogers, Ed Stabler, Mark Steedman, Yuka Tateisi, Juan Uriagereka, K. Vijay-Shanker, and David Weir. The organizing committee consisted of Rodolfo Delmonte and Giorgio Satta (co-chairs), Julia Akhramovitch, Antonella Bristot, David Chiang, Aravind K. Joshi, Alberto Lavelli, Carlo Minnaja, Laura Paccagnella, Luisella Romeo, Anoop Sarkar, and Trisha Yannuzzi. My gratitude to all for their excellent work. A special thanks to Trisha for her gift of her vast experience and for her herculean efforts at a number of crucial points in the process.

Finally, I would like to thank the University of Padua, the University of Venice, the Institute for Research in Cognitive Science at the University of Pennsylvania, and the Institute for Scientific and Technological Research (ITC-IRST) for their financial support.

Robert Frank
Program Chair

Conference Program

Monday, May 20

4:00-4:15	Presentation of the Conference
4:15-4:45	*Compositional Semantics for Relative Clauses in Lexicalized Tree Adjoining Grammars* Chung-Hye Han
4:45-5:30	*Putting Some Weakly Context-Free Formalisms in Order* David Chiang
5:30-6:00	*Supertagging for Combinatory Categorial Grammar* Stephen Clark

Tuesday, May 21

9:30-10:00	*Learning languages from positive examples with dependencies* Jérôme Besombes and Jean-Yves Marion
10:00-10:45	*Towards a Dynamic Version of TAG* Vincenzo Lombardo and Patrick Sturt
10:45-11:15	*Resumptive Pronouns, Wh-island Violations, and Sentence Production* Cassandra Creswell
11:15-11:45	Coffee Break
11:45-12:15	*Statistical Morphological Tagging and Parsing of Korean with an LTAG Grammar* Anoop Sarkar and Chung-Hye Han
12:15-1:15	**Invited Lecture** *Hard and Soft Constraints in Syntax: An Approach to Person/Voice Interactions in Stochastic Optimality Theory* Joan Bresnan
1:15-3:00	Lunch
3:00-3:30	*Notes on the Complexity of Complex Heads in a Minimalist Grammar* Jens Michaelis
3:30-4:00	*Learning Mirror Theory* Gregory M. Kobele, Travis Collier, Charles Taylor and Edward Stabler
4:00-5:00	**Poster Session**
	Defining a Lexicalized Context-Free Grammar for a Subdomain of Portuguese Language Cinthyan Renata Sachs C. de Barbosa, Davidson Cury, José Mauro Volkmer de Castilho and Celso de Renna Souza
	Practical, Template-Based Natural Language Generation with TAG Tilman Becker
	Relative Clause Attachment and Anaphora: A Case for Short Binding Rodolfo Delmonte
	A Left Corner Parser for Tree Adjoining Grammars Victor J. Diaz, Vicente Carrillo and Miguel A. Alonso
	Context-Free Parsing of a Tree Adjoining Grammar Using Finite-State Machines Alexis Nasr, Owen Rambow, John Chen and Srinivas Bangalore
	How to Prevent Adjoining in TAGs and its Impact on the Average Case Complexity Jens Woch

5:00-5:30 *Quantification Over Possible Worlds in LTAG: Some Constraints*
Maribel Romero

5:30-6:00 *One More Perspective on Semantic Relations in TAG*
James Rogers

6:00-6:30 *Using an Enriched TAG Derivation Structure as Basis for Semantics*
Laura Kallmeyer

8:00- Banquet at *Al Gatto Nero*, Burano
Departure at 7:00 from Fondamente Nuove

Wednesday, May 23

9:30-10:00 *A Proof System for Tree Adjoining Grammars*
Adi Palm

10:00-10:30 *Tree-Adjoining Grammars as Abstract Categorial Grammars*
Philippe de Groote

10:30-11:00 *Residuation, Structural Rules, and Context Freeness*
Gerhard Jäger

11:00-12:00 **Poster Session**

A Note on the Complexity of Associative-Commutative Lambek Calculus
Christophe Costa Florêncio

Turning Elementary Trees into Feature Structures
Alexandra Kinyon

On the Affinity of TAG with Projective, Bilexical Dependency Grammar
Tom B.Y. Lai, Changning Huang and Robert W.P. Luk

The Theory of Control Applied to the Prague Dependency Treebank (PDT)
Jarmila Panevová, Veronika Řezníčková and Zdeňka Urešová

Systematic Grammar Development in the XTAG Project
Carlos Prolo

A Formal Proof of Strong Equivalence for a Grammar Conversion from LTAG to HPSG-style
Naoki Yoshinaga, Yusuke Miyao and Jun'ichi Tsujii

12:00-12:30 *Parsing MCS languages with Thread Automata*
Éric Villemonte de la Clergerie

12:30-1:00 *Evaluation of LTAG Parsing with Supertag Compaction*
Olga Shaumyan, John Carroll and David Weir

1:00-3:00 Lunch

3:00-3:30 *Korean-English MT and S-TAG*
Mark Dras and Chung-Hye Han

3:30-4:30 **Invited Lecture**

Tectogrammatical Representation: Towards a Minimal Transfer in Machine Translation
Jan Hajič

4:30-5:00 Coffee Break

5:00-5:30 *Clustering for Obtaining Syntactic Classes of Words from Automatically Extracted LTAG Grammars*
Tadayoshi Hara, Yusike Miyao and Jun'ichi Tsujii

5:30-6:00 *A New Metagrammar Compiler*
B. Gaiffe, B. Craibbé and A. Roussanaly

6:00-6:30 *DTAG?*
Kim Gerdes

Table of Contents

Compositional Semantics for Relative Clauses in Lexicalized Tree Adjoining Grammars

Chung-hye Han
Simon Fraser University

1. Introduction

This paper proposes a compositional semantics for relative clauses in Lexicalized Tree Adjoining Grammars (LTAG). As explicated in (Joshi and Vijay-Shanker, 1999; Joshi and Kallmeyer, 2000), in the phrase-structure based compositional semantics the meaning of a sentence is computed as a function of meaning of each node in the tree. On the other hand, in LTAG based compositional semantics, the meaning of a sentence is computed as a function of meaning of elementary trees put together to derive the sentence. This is because in LTAG, the elementary objects are lexicalized trees that encapsulate all syntactic/semantic arguments of the associated lexical item (i.e., the anchor). Each elementary tree is associated with a semantic representation, and given the history of how the elementary trees are put together to form a sentence, its semantics can be computed by combining the semantic representations of the elementary trees. In other words, semantics in LTAG can be defined to operate on bigger objects than in a phrase-structure based approach, without violating the principle of compositionality. One could naturally compose the full derived tree for the sentence at the end of the derivation process, and then compute the semantics on each node in in the full derived tree. However, this has two major disadvantages: first, there is no correspondence between semantic composition and the syntactic operations of substitution and adjunction; and secondly, it is impossible to compute semantic interpretation incrementally and monotonically for partial derivations. This suggests that compositional semantics in TAG should be done on the derivation tree, not on the derived tree.

There are two ways of doing semantics on the derivation tree: (i) synchronous TAG as in (Abeillé, 1994), and (ii) flat semantics as in (Joshi and Vijay-Shanker, 1999; Joshi and Kallmeyer, 2000). In this paper, I pursue the flat semantics approach (also known as minimal recursion semantics), in which the main operation for semantic composition is the conjunction of the semantic representations associated with each elementary tree along with the unification of variables contributed by each semantic representation. Doing flat semantics on relative clauses is particularly interesting because it involves defining an alternative semantic role for the relative pronoun to the phrase-structure based approach, in which the relative pronoun has been argued to be an operator that turns the relative clause into a function of a predicate type (Heim and Kratzer, 1998). In addition, it involves defining a relationship between the head noun and the *wh* relative pronoun, which turns out to be non-trivial.

I will start the paper with an illustration of an LTAG-based compositional semantics for a simple sentence with an attributive adjective in section 2. This will allow us to understand how semantic composition in general and modification in particular work in LTAG semantics. In section 3, using a relative clause containing a genitive relative pronoun (e.g., *whose*), a case of pied-piping, I will first present a couple of approaches that do not work. This will allow us to clarify the necessary components for a proper analysis. I then propose my analysis of relative clauses that accounts for these components. Section 4 discusses how the proposed analysis can be generalized to relative clauses with a simple relative pronoun, adjunct relative clauses and relative clauses whose relative pronoun is deeply embedded in a recursive genitive NP. The discussion on recursive genitive NPs will lead to a slight modification of the proposed analysis. In general, I follow the English grammar developed in (The XTAG-Group, 2001) for the syntax of various constructions discussed in this paper (although in some cases, where convenient, I differ from the XTAG analysis to produce the appropriate semantics).

2. LTAG-based Compositional Semantics for a Simple Sentence with an Attributive Adjective

The elementary trees to generate the derived and the derivation tree for sentence in (1), and their corresponding semantic representations are given in Figure 1.

(1) John solved a difficult problem.

* I am indebted to Aravind Joshi for many discussions on this topic. I also wish to thank Tonia Bleam, Mark Dras, Maribel Romero, Anoop Sarkar, Alexander Williams and the members of the XTAG group at the University of Pennsylvania. All errors are mine.

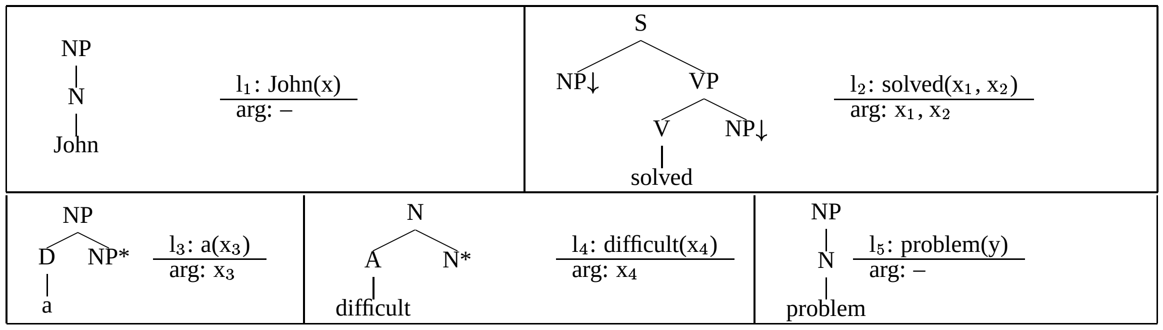
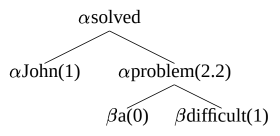


Figure 1: Elementary Trees and Semantic Representations

The symbols l_i label each semantic representation. The elementary tree anchoring *solved* contains two substitution sites, each corresponding to subject and object argument. This is associated with a semantic representation with the predicate *solved* and two argument variables. The *Arg* slot contains two variables x_1 and x_2, indicating that they each must be unified with variables contributed by the subject and the object in the semantic composition. The elementary trees anchoring *a* and *difficult* are adjunction trees. They are each associated with a semantic representation with the predicate corresponding to the anchor and one argument variable. The *Arg* slot contains a variable which must be unified with a variable that is contributed by the adjoining noun (or NP). The elementary trees anchoring *John* and *problem* are associated with semantic representations that each contributes a variable. The argument slot is empty, reflecting the fact that the elementary tree is an initial tree with no substitution sites. The derivation tree and the semantic composition for (1) are given in Figure 2.

Assuming a bottom-up semantic composition, first the semantics for *a* and *problem* combine, unifying the argument variable of *a* with the variable contributed by *problem*. Further, the semantics for *difficult* and *problem* combine, unifying the argument variable of *difficult* with the variable contributed by *problem*. And then the semantics for *John* and *problem* combine with the semantics for *solved*. This results in unifying the argument variables of *solved* with the variables contributed by *John* and *problem*. The final semantics for (1) is a conjunction of each labeled semantic representations in Figure 2.



l_3: a(y)
l_4: difficult(y)
l_5: problem(y)
l_2: solved(x,y)
l_1: John(x)

Figure 2: Derivation Tree and Semantic Composition

3. LTAG-based Compositional Semantics for Relative Clauses

The example in (2) will be used throughout this section to illustrate the analysis for an LTAG-based semantics for relative clauses.

(2) A problem whose solution is difficult

It will be shown that the main source of the problem is that in a relative clause there's actually two variables that must be kept track of: a variable corresponding to the gap in the relative clause and the variable corresponding to the *wh* relative pronoun. In order to get the correct predicate/argument relation and the semantics that a relative clause is a modifier of a head noun, the variable for the relative pronoun (*whose*) must unify with the head noun (*problem*), and the variable for the gap must come from the head of the pied-piped structure (*solution*). In the simple case with no pied-piping, the two variables are the same. But as soon as the relative pronoun occurs in a pied-piped structure, the two variables are not the same, and since the *wh* is embedded, its variable cannot directly unify with the variable from the head noun, creating a locality problem. In this section, I will first present a couple of approaches that do not work in subsections 3.1 and 3.2 to illustrate the issues just described and motivate the analysis proposed in subsection 3.3.

3.1. Trial 1

As a first try, let's consider the elementary trees and their corresponding semantic representations in Figure 3. The relative clause tree has a substitution site designated as NP[WH]. I am using this notation for convenience to represent the assumption that relative clause trees are encoded with a [WH] feature that requires a phrase dominating a relative pronoun to be substituted into this position in the course of derivation.

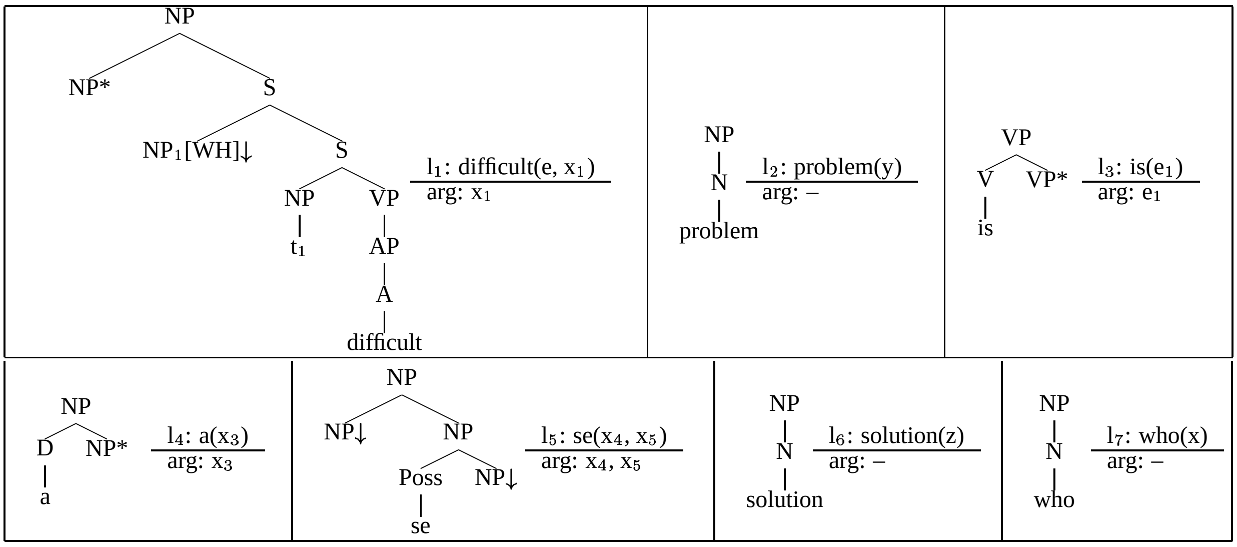
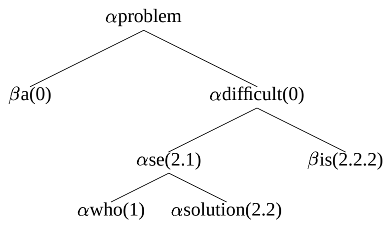


Figure 3: Elementary Trees and Semantic Representations

The semantics for *a, problem, solution* and *who* are straightforward. I have defined the semantics of the auxiliary verb *is* to be a proposition composed of a predicate and an event argument. This event argument is needed to unify with the event argument contributed by the adjoining verb (or adjective). Consequently, the semantics for the relative clause tree anchoring *difficult* is defined to contribute an event variable. Further, the semantics for the relative clause tree is defined to require an argument variable which must unify with the variable contributed by the head noun (i.e., *problem*). Moreover, assuming that *se* in *whose* is equivalent to genitive *'s*, *se* anchors an elementary tree with two substitution sites for the possessor and the possessee. This corresponds to a semantic representation with the predicate *se* and two argument variables that must unify with the variables contributed by the substituting NPs. The derivation tree for (2) and the semantic composition under this approach are given in Figure 4.[1]



l_2: problem(y)
l_4: a(y)
l_1: difficult(e,y) *** *wrong semantics*
l_3: is(e)
l_5: se(x,z)
l_6: solution(z)
l_7: who(x)

Figure 4: Derivation Tree and Semantic Composition

The problem with this approach is that it derives the incorrect meaning that it is the *problem* that is difficult, not the *solution.* Another problem, which is related to the first problem, is that there is no way to define the relationship between the relative pronoun and the head noun.

1. Notice that I am assuming multiple modification analysis given in (Schabes and Shieber, 1994).

3.2. Trial 2

This thus takes us to the second approach. In this approach, I define an operator called LINK that enforces the unification of the variables contributed by the *wh* relative pronoun and the head noun.

The LINK operation does the similar job as predicate modification in phrase-structure based compositional semantics, as defined in (3) (Heim and Kratzer, 1998). When applied to a relative clause and its head noun, which are both predicate types, the predicate modification ensures that both of them are predicates over the same variable. This in turn effectively derives the interpretation of the relative clause as a modifier of the head noun. The LINK operation is intended to perform the same function.

(3) Predicate Modification

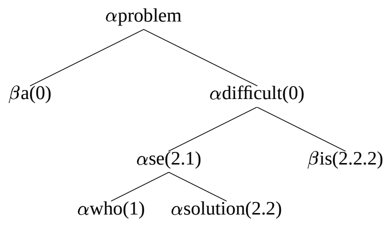
If α has the form $\begin{array}{c}\alpha\\ \widehat{\beta\quad\gamma}\end{array}$, and $[\![\beta]\!]^s$ and $[\![\gamma]\!]^s$ are both in $D_{<e,t>}$, then $[\![\alpha]\!]^s = \lambda x_e [\![\beta]\!]^s(x) \wedge [\![\gamma]\!]^s(x)$.

The semantic representations under the second approach are given in Figure 5. The only difference between the first and the second approaches is in the semantics for the relative clause elementary tree anchoring *difficult*. Here, x_1 stands for the variable for the *wh* relative pronoun, and x_2 stands for the variable for the head noun. We can think of [WH] feature encoded in the relative clause tree to be responsible for contributing the variable for the relative pronoun. This approach again derives wrong semantics for (2): the *problem* incorrectly ends up being *difficult*, as shown in Figure 6.

l_1: [difficult(e,x_1) ∧ LINK(x_1,x_2)] arg: x_1, x_2	l_2: problem(y) arg: –	l_3: is(e_1) arg: e_1

l_4: a(x_3) arg: x_3	l_5: se(x_4,x_5) arg: x_4, x_5	l_6: solution(z) arg: –	l_7: who(x) arg: –

Figure 5: Semantic Representations



l_2: problem(y)
l_4: a(y)
l_3: is(e)
l_5: se(x,z)
l_6: solution(z)
l_7: who(x)
l_1: [difficult(e,x) ∧ LINK(x,y)] *** *wrong semantics*

Figure 6: Derivation Tree and Semantic Composition

Changing the semantics for the relative clause as in Figure 7 will not help. *Difficult* is a predicate over the variable for the head noun, and so it will again derive the incorrect interpretation that the *problem*, and not the *solution*, is difficult.

l_1: [difficult(e,x_2) ∧ LINK(x_1,x_2)]   arg: x_1, x_2

Figure 7: Semantic Representation for a Relative Clause Anchoring *difficult*

3.3. Trial 3: A proposal

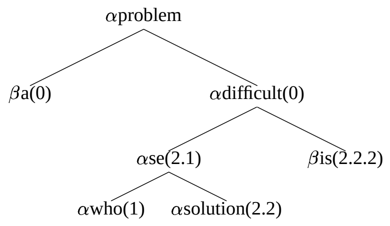
In order to derive the correct semantics for (2), *difficult* must be a predicate over a variable associated with *solution*. As a way of ensuring this, I define three argument variables for relative clauses: one for the *wh* relative pronoun, another for the head noun, and another for the head of NP[WH]. The semantics under this approach is given in Figure 8.

l_1: [difficult(e,x_0) ∧ LINK(x_1,x_2)] arg: x_0, x_1, x_2	l_2: problem(y) arg: –	l_3: is(e_1) arg: e_1

l_4: a(x_3) arg: x_3	l_5: se(x_4,x_5) arg: x_4, x_5	l_6: solution(z) arg: –	l_7: who(x) arg: –

Figure 8: Semantic Representations

In the semantics for the relative clause anchoring *difficult*, three argument variables are defined: x_0 must unify with the variable contributed by the head of NP[WH] (i.e., *solution*), x_1 must unify with the variable contributed by the *wh*-word (i.e., *who*), and x_2 must unify with the variable contributed by the head noun (i.e., *problem*). The semantic composition under this approach is given in Figure 9. This semantics correctly derives the meaning of the relative clause in (2): the *solution* is difficult, and this *solution* is in a possession relation with *problem*, as forced by the LINK operation unifying the variables for *who* and *problem*.



l_2: problem(y)
l_4: a(y)
l_3: is(e)
l_5: se(x,z)
l_6: solution(z)
l_7: who(x)
l_1: [difficult(e,z) ∧ LINK(x,y)]

Figure 9: Derivation Tree and Semantic Composition

4. Generalizing

In this section, we will see how the proposed analysis can be generalized to relative clauses whose relative pronoun is the head of NP[WH], adjunct relative clauses, and relative clauses containing a recursive genitive NP[WH].

4.1. Relative clauses whose relative pronoun is the head of NP[WH]

The proposed approach straightforwardly extends to the simple case where the relative clause contains a relative pronoun which is the head of the NP[WH], as in (4).

(4) The solution which is difficult

The semantics for the elementary trees are given in Figure 10, and the derivation tree and the corresponding semantic composition are given in Figure 11. In this case, since the head of NP[WH] is the relative pronoun itself, x_0 for the relative pronoun and x_1 for the head of NP[WH] in the relative clause tree semantics both unify with the variable x from the relative pronoun tree. By the LINK operation, x is unified with the variable from *solution*, giving us the correct interpretation that it is the solution that is difficult.

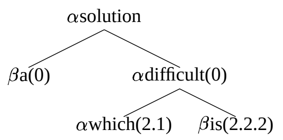
l_1: [difficult(e,x_0) ∧ LINK(x_1,x_2)] arg: x_0, x_1, x_2	l_2: solution(y) arg: –

l_3: is(e_1) arg: e_1	l_4: a(x_3) arg: x_3	l_5: which(x) arg: –

Figure 10: Semantic Representations

4.2. Adjunct relative clauses

We now discuss how the proposed analysis can be extended to handle the semantics of adjunct relative clauses as in (5). We will consider two possible approaches: (i) an approach based on the assumption that the adjunct



l_2: solution(y)
l_4: a(y)
l_3: is(e)
l_5: which(x)
l_1: [difficult(e,x) ∧ LINK(x,y)]

Figure 11: Derivation Tree and Semantic Composition

phrase *in which* substitutes into the relative clause tree; and (ii) an approach based on the assumption that the adjunct phrase adjoins onto the relative clause tree.

(5) The place in which John lives is expensive.

4.2.1. Substitution approach

Under the substitution approach, the elementary tree for the adjunct relative clause anchoring *lives* has two substitution sites: one for the subject NP and the other for the PP that will contain the relative pronoun in the course of the derivation. The corresponding semantic representation is given in the first box in Figure 12. Here, *lives* takes an event argument variable (e) and a variable (x_0) for the subject. Further, the variable for the *wh* relative pronoun (x_1) and the variable for the head noun (x_2) are forced to unify by the LINK operation as before. The derivation requires a PP initial tree anchoring *in*. The semantics for this tree is given in l_5 in Figure 12: *in* is a predicate taking an event variable, and another variable for the substituting NP. Substituting this PP into the relative clause tree will allow the event variable from the PP tree to unify with the event variable from the relative clause tree. This will have the interpretive effect that the PP is modifying the verb *lives*.

The derivation tree and the corresponding semantic composition for (5) are given in Figure 13. We correctly end up with the interpretation that the place is expensive, and John lives in this place.

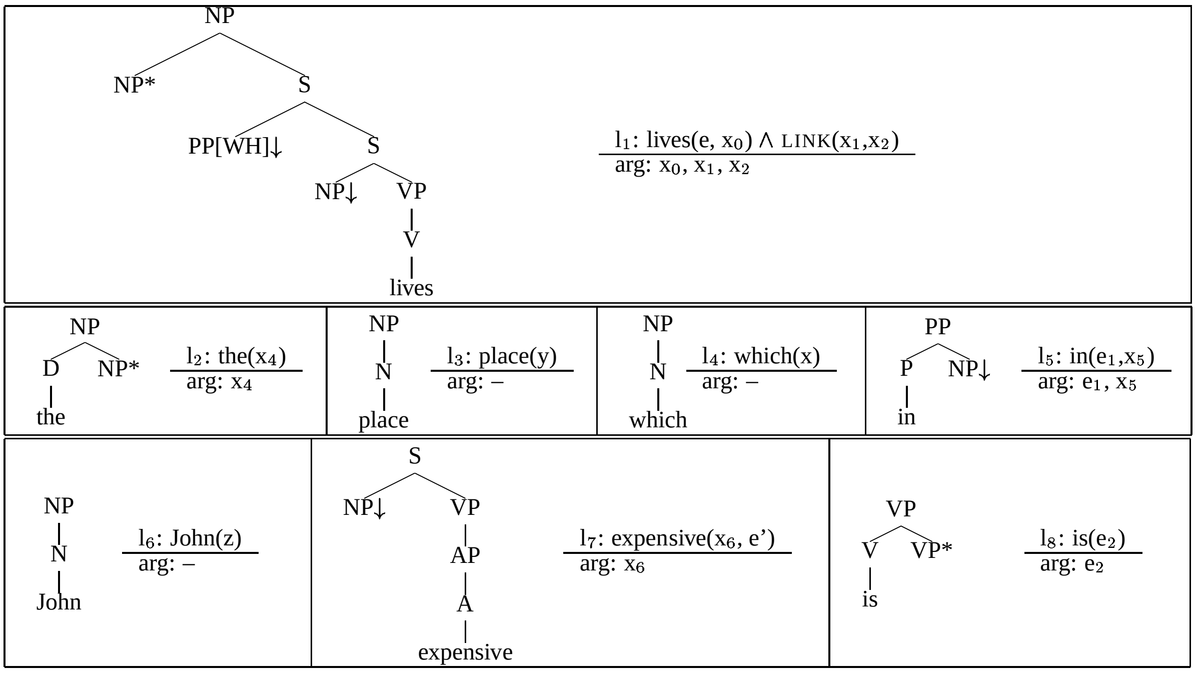
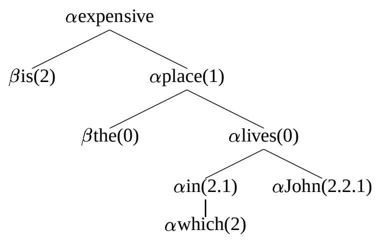


Figure 12: Elementary Trees and Semantic Representations

4.2.2. Adjunction approach

Under the assumption that adjunct phrase *in which* is adjoined to the relative clause tree, the elementary tree and the corresponding semantic representation for the adjunct relative clause tree anchoring *lives* can be specified

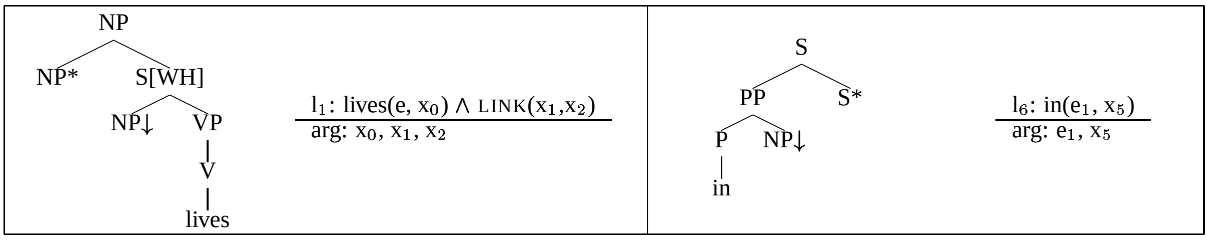


l_4: which(x)
l_5: John(z)
l_3: place(y)
l_2: the(y)
l_1: lives(e,z) ∧ LINK(x,y)
l_6: in(e,x)
l_7: expensive(y,e’)
l_8: is(e’)

Figure 13: Derivation Tree and Semantic Composition

as in the first box in Figure 14. The semantics for the adjunct relative clause is as before: *lives* takes an event argument variable (e) and a variable (x_0) which will unify with the subject, and the variable for the head noun (x_2) is forced to be unified with the variable for the *wh* relative pronoun (x_1). Although there is no syntactic position designated for a relative pronoun in the relative clause tree, we can motivate a variable for it with the assumption that the tree is encoded with a [WH] feature that requires a relative pronoun containing phrase to be adjoined onto the S node. Further, the derivation under the adjunction approach requires an S-rooted auxiliary tree anchoring *in*, which has an NP node that will be substituted with a *wh* relative pronoun. Its semantics is represented in l_6 in Figure 14: e_1 is an event argument variable that will unify with the event variable from the adjoining S, and x_5 will unify with the variable from the substituting NP. All other elementary trees and their semantics necessary for the derivation are as same as in Figure 12.

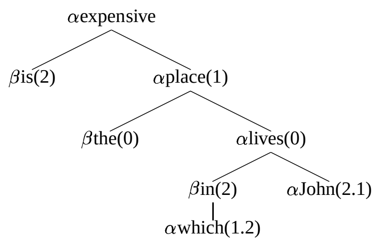
The derivation tree and the corresponding semantic composition for (5) are given in Figure 15. This results in the correct interpretation that the place is expensive and John lives in that place.



l_1: lives(e, x_0) ∧ LINK(x_1,x_2)
arg: x_0, x_1, x_2

l_6: in(e_1, x_5)
arg: e_1, x_5

Figure 14: Elementary Trees and Semantic Representations



l_4: which(x)
l_5: John(z)
l_6: in(e,x)
l_3: place(y)
l_2: the(y)
l_1: lives(e,z) ∧ LINK(x,y)
l_7: expensive(y,e’)
l_8: is(e’)

Figure 15: Derivation Tree and Semantic Composition

At the current stage of understanding, the adjunction approach seems to be preferable to the substitution approach. This is because under the substitution approach, adjunct PPs enter into the derivation through substitution. However, in all other cases, while adjunct PPs are represented with auxiliary trees that enter into the derivation through adjunction, argument PPs are represented with initial trees and enter into the derivation through substitution. The adjunction approach allows us to maintain this dichotomy between arguments and adjuncts.

4.3. Relative clauses containing a recursive genitive NP[WH]

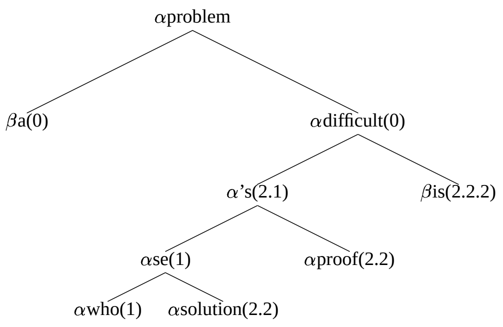
In the derivation of relative clauses with a recursive genitive NP[WH] as in (6), each genitive contributes an elementary tree with two substitution sites. They are each associated with the semantic representation in which *se* or *'s* is a predicate requiring an argument variable for the possessor and an argument variable for the possessee. These are represented as l_5 and l_9 in Figure 16.

Let's see what happens if we use the semantics given in Figure 16 and the derivation tree given in (17) to do the compositional semantics for (6). The semantics for the relative clause tree and other elementary trees are similar to the the ones we used in section 3.3. But now we have a problem. Although the resulting interpretation gets the right predicate/argument relation between *difficult* and *proof*, and the possession relation between *solution* and *proof* and *who* and *solution*, the variable for *who* cannot be unified with x_1 in the relative clause semantics. This is because *who* is deeply embedded and so its variable cannot pass all the way up to the relative clause semantics. Thus, there is no way to enforce the unification between the variable from *who* and x_1, and the meaning that *problem* is the possessor of *solution* is lost and the meaning that the relative clause is the modifier of the head noun cannot be represented.

(6) A problem whose solution's proof is difficult

l_1: [difficult(e,x_0) ∧ LINK(x_1,x_2)] arg: x_0, x_1, x_2	l_2: problem(y) arg: –	l_3: is(e_1) arg: e_1
l_4: a(x_3) arg: x_3	l_5: 's(x_4,x_5) arg: x_4, x_5	l_6: solution(z) arg: –
l_7: who(x) arg: –	l_8: proof(v) arg: –	l_9: se(x_6,x_7) arg: x_6, x_7

Figure 16: Semantic Representations



l_2: problem(y)
l_4: a(y)
l_3: is(e)
l_9: se(x,z)
l_6: solution(z)
l_8: proof(v)
l_5: 's(z,v)
l_7: who(x)
l_1: [difficult(e,v) ∧ LINK(x_1,y)]

Figure 17: Derivation Tree and Semantic Composition

Here, I will sketch two possible approaches to address this problem: one is to exploit feature unification (Vijay-Shanker and Joshi, 1991), and the other is to use set-local multi-component TAG (MC-TAG) (Weir, 1988).

Under the feature unification approach, we need to make the assumption that a *wh* feature is encoded in relative pronoun trees as well as in relative clause trees, and that these features are syntactically constrained to be the same. This syntactic constraint is instantiated as the semantic constraint that the variable for the *wh* relative pronoun in the semantics of the relative clause tree and the variable in the semantics of the relative pronoun tree be the same.

The semantics for the relative clause anchoring *difficult* now looks as in the first box in Figure 18. x_0 will unify with the variable from *proof*, the head of NP[WH], and x_2 will unify with the variable from the head noun *problem*. And x_w is the variable for the relative pronoun, which is motivated by the *wh* feature encoded in the relative clause tree. This feature is syntactically constrained to be the same as the feature on the relative pronoun tree. This means that semantically, the relative pronoun *who* contributes the same variable, x_w.

l_1: [difficult(e,x_0) ∧ LINK(x_w,x_2)] arg: x_0, x_2	l_7: who(x_w) arg: –

Figure 18: Modified Semantics for Relative Clause and Relative Pronoun Trees

The semantic composition using these semantics will give us the correct interpretation, as shown in Figure 19: *problem* is the possessor of *solution*, *solution* is the possessor of *proof* and *proof* is *difficult*.

l_2:problem(y)
l_4:a(y)
l_3:is(e)
l_9:se(z,v)
l_6:solution(z)
l_8:proof(v)
l_5:'s(x_w,z)
l_7:who(x_w)
l_1:difficult(e,v) ∧ LINK(y,x_w)

Figure 19: Semantic Composition

Under the set-local approach, we need to assume three sets of trees as shown in Figure 20. One set contains an NP tree anchoring *who* and a degenerate auxiliary tree S*, another set contains a relative clause tree and an NP tree anchoring *'s*, and the other set contains NP trees anchoring *se* and *problem* respectively. The first set is for the relative pronoun and can be naturally motivated: the NP tree anchoring *who* corresponds to the contribution of *who* to the predicate/argument structure, and S* contributes to the scope of *who*. The other two sets, however, are not a linguistically natural set, although it will be shown that postulation of these sets are necessary in resolving our problem.

The syntactic derivation will proceed as follows: S* adjoins to S_1 in the relative clause tree, and NP anchoring *who* substitutes into the specifier of *se* tree. And *solution* tree substitutes into the complement of *se* tree, which will substitute into the specifier of *'s* tree. The complement of *'s* tree is substituted with *proof* tree. And then *'s* tree substitutes into NP[WH] node of the relative clause tree. The derivation tree is given in Figure 21

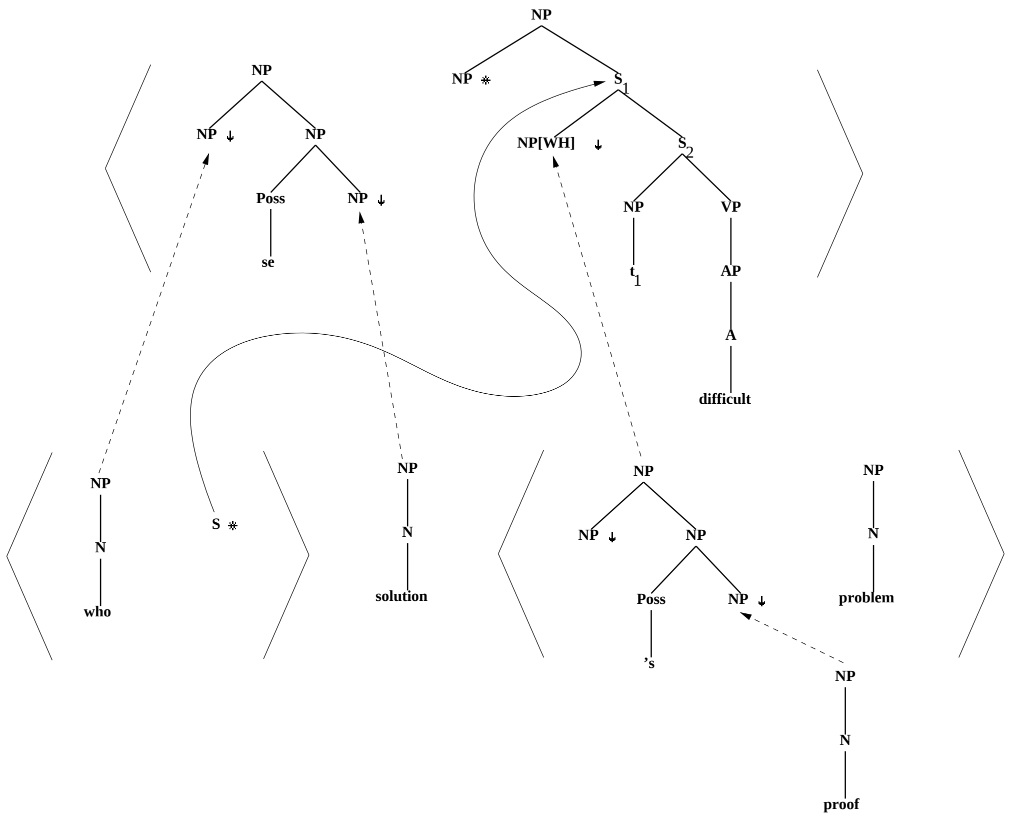


Figure 20: Derivation in Set-Local MC-TAG

The only new thing we need to do for semantics is to redefine the semantics for *who*, as in Figure 22, and the rest will look exactly the same as in Figure 16. The semantics in $l_{7[1]}$ is for the elementary tree anchoring *who*, and the semantics in $l_{7[2]}$ is for the degenerate S* tree. The variable from $l_{7[1]}$ will unify with the variable for the

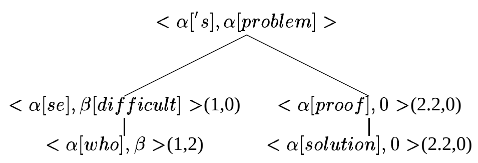

$< \alpha['s], \alpha[problem] >$
$< \alpha[se], \beta[difficult] >$(1,0)
$< \alpha[proof], 0 >$(2.2,0)
$< \alpha[who], \beta >$(1,2)
$< \alpha[solution], 0 >$(2.2,0)

Figure 21: Derivation tree

$l_{7[1]}$: who(x)	$l_{7[2]}$: x
arg: –	arg: –

Figure 22: Modified Semantics for Relative Pronoun Trees

possessor in l_9, and the variable from $l_{7[2]}$ will unify with the variable for the *wh* relative pronoun in l_1. This has the desirable result that *who* is the possessor of *solution* and that the relative clause is the modifier of the head noun *proof*.

While both feature unification and set-local approaches give us the correct semantics, there are problems with both. In feature unification approach, the variable for *who* ends up being LINKed to the variable for *problem*, not through a direct variable unification, but because the *wh* features encoded in the relative clause elementary tree and in the relative pronoun tree are stipulated to translate to the same variable, x_w. In the set-local approach, variable unification in semantics works without resorting to any stipulation, but the cost to syntax is too much. From an implementational point of view, it seems that feature unification approach is preferable, given its relative simplicity.

5. Conclusion

I have shown that an LTAG-based compositional semantics for relative clauses can be done by defining three argument variables for the semantics of relative clause elementary trees: one for the *wh* relative pronoun, one for the head of NP[WH] and the other for the head noun. I have introduced an operator, LINK, that forces variable unification between the *wh* relative pronoun and the head noun. We have seen that the proposed analysis handles relative clauses with a simple relative pronoun as well as those with a relative pronoun in pied-piping structure, and adjunct relative clauses. I have also pointed out a potential problem in variable unification in relative clauses with a deeply embedded relative pronoun, and suggested two possible ways of addressing this problem: exploiting feature unification and using set-local MC-TAG. All this ensures the unification between the variables from the head noun and the relative pronoun, no matter how deeply embedded the relative pronoun is, deriving the desirable predicate/argument relations and the interpretation that the relative clause is a modifier of the head noun. It remains to be seen how the proposed analysis can be extended to relative clauses with long distance relativization (e.g., *the solution which John said Mary thinks is difficult*).

References

Abeillé, Ann. 1994. Syntax or semantics? Handling nonlocal dependencies with MCTAGs or synchronous TAGs. *Computational Intelligence*.

Heim, Irene and Angelika Kratzer. 1998. *Semantics in Generative Grammar*. Blackwell Publishers.

Joshi, Aravind K. and Laura Kallmeyer. 2000. Factoring predicate argument and scope semantics: underspecified semantics with LTAG. Ms. Univeristy of Pennsylvania and UFRL, University Paris 7.

Joshi, Aravind K. and K. Vijay-Shanker. 1999. Compositional semantics with lexicalized tree-adjoining grammar (LTAG): How much underspecification is necessary? In *Proceedings of the 3rd International Workshop on Computational Semantics*, pages 131–145, The Netherlands.

Schabes, Yves and Stuart M. Shieber. 1994. An Alternative Conception of Tree-Adjoining Derivation. *Computational Linguistics*, 20(1):91–124.

The XTAG-Group. 2001. A Lexicalized Tree Adjoining Grammar for English. Technical Report IRCS 01-03, University of Pennsylvania.

Vijay-Shanker, K. and Aravind Joshi. 1991. Unification Based Tree Adjoining Grammars. In *Unification-based Grammars*. MIT Press, Cambridge, Massacusetts.

Weir, D. 1988. *Characterizing mildly context-sensitive grammar formalisms*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, August.

Putting Some Weakly Context-Free Formalisms in Order

David Chiang
University of Pennsylvania

1. Introduction

A number of formalisms have been proposed in order to restrict tree adjoining grammar (TAG) to be weakly equivalent to context free grammar (CFG): for example, tree substitution grammar (TSG), tree insertion grammar (TIG), and regular-form TAG (RF-TAG); in the other direction, tree-local multicomponent TAG (TL-MCTAG) has been proposed as an extension to TAG which is weakly equivalent to TAG. These formalisms have been put to use in various applications. For example, Kroch and Joshi (1987) and others use TL-MCTAG for linguistic description; Bod (1992) uses TSG, and Chiang (2000) uses TIG, for statistical parsing; Shieber (1994) proposes to use TSG for generating synchronous TAG derivations for translation, and Dras (1999) uses RF-TAG for the same purpose. Although it is understood that these formalisms are useful because they have greater strong generative capacity (henceforth SGC) than their weakly-equivalent relatives, it is not always made clear what this actually means: sometimes it is understood in terms of phrase structures, sometimes in terms of a broader notion of structural descriptions (so Chomsky (1963)).

We take the latter view, and follow Miller (1999) in seeing phrase structures as just one of many possible interpretations of structural descriptions (alongside, for example, dependency structures). For Miller, structural descriptions themselves should never be compared (since they vary widely across formalisms), but only their interpretations. Thus SGC in the phrase-structure sense is one of several ways of testing SGC in the broader sense. However, not much effort has been made to demonstrate precisely how formalisms compare in these other ways.

In this paper we examine four formalisms—CFG, TIG, RF-TAG, and what we call *component-local scattered context grammar* (CL-SCG)—under four different interpretations, and find that TIG, RF-TAG, and CL-SCG all extend the expressivity of CFG in different ways (see Figure 1). These results show that it is possible to make formally precise statements about notions of generative capacity other than weak generative capacity (henceforth WGC), as a step towards articulating desiderata of formal grammars for various applications.

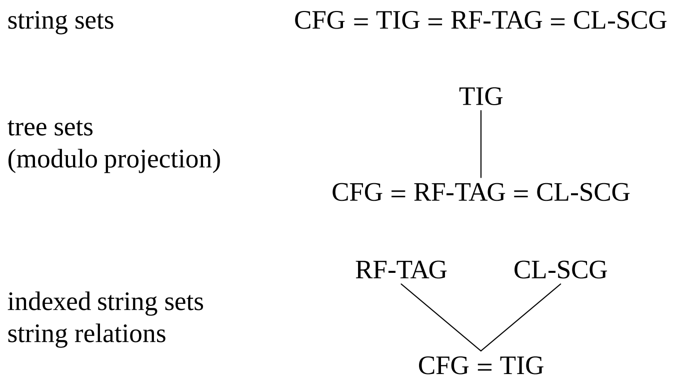


Figure 1: Summary of results. Edges denote strict inclusion (lower ⊂ higher); = denotes equality.

2. Definitions

We assume familiarity with CFGs and TAGs, and proceed to define two restrictions on TAGs:

Definition 1. A *left* (or *right*) *auxiliary tree* is an auxiliary tree in which every frontier node to the right (resp., left) of the foot node is labeled with the empty string. A *tree insertion grammar* (Schabes and Waters, 1995) is a TAG in which all auxiliary trees are either left or right auxiliary trees, and adjunction is constrained so that:

- no left (right) auxiliary tree can be adjoined on any node that is on the spine of a right (left) auxiliary tree, and
- no adjunction is permitted on a node that is to the right (left) of the spine of a left (right) auxiliary tree.

Definition 2. We say that a TAG is in *regular form* (Rogers, 1994) if there exists some partial ordering $\preceq$ over nonterminal symbols such that if β is an auxiliary tree whose root and foot nodes are labeled X, and η is a node labeled Y on β's spine where adjunction is allowed, then $X \preceq Y$, and $X = Y$ only if η is a foot node. Thus adjunction at the foot node is allowed freely, adjunction at the middle of a spine is allowed only to a bounded depth, and adjunction at the root is not allowed at all.[1]

Next we define CL-SCG, first introducing the notion of an indexed string, which we will make use of in several places in this paper.

Definition 3. An *indexed string* is a pair $(w; I_w)$, where w is a string and I_w is an equivalence relation over string positions of w. An *indexed string n-tuple* is an $(n+1)$-tuple $(w_1, \ldots, w_n; I_w)$, where $w_1, \ldots, w_n$ are strings and I_w is an equivalence relation over string positions of the w_i. We notate these equivalence relations using boxed indices.

Definition 4. A *local scattered context grammar*[2] is a tuple $G = (N, T, P, S)$, where

- N and T are finite, disjoint sets of nonterminal symbols and terminal symbols, respectively,
- $S \in N$ is the start symbol, and
- P is a finite set of productions of the form

$$(A_1, \ldots, A_n) \to (\alpha_1, \ldots, \alpha_n; I_\alpha)$$

  where $n \geq 1$, $A_i \in N$ and $(\alpha_i, \ldots, \alpha_n; I_\alpha)$ is an indexed tuple of strings over $(N \cup T)^*$.

We write $(\gamma; I_\gamma) \Rightarrow_G (\delta, I_\delta)$, where $(\gamma; I_\gamma)$ and $(\delta; I_\delta)$ are indexed strings, if and only if there exists a production $(A_1, \ldots, A_n) \to (\alpha_1, \ldots, \alpha_n; I_\alpha) \in P$ such that

$$\gamma = \gamma_0 A_1 \gamma_1 \cdots A_n \gamma_n$$

where $A_1, \ldots, A_n$ comprise an equivalence class of I_γ, and

$$\delta = \gamma_0 \alpha_1 \gamma_1 \cdots \alpha_n \gamma_n,$$

where any nonterminal instances in the γ_i whose corresponding instances in γ are equivalent under I_γ are also equivalent under I_δ, as are any nonterminal instances in the α_i which are equivalent under I_α, and nothing else.

Let I_S be the equivalence relation on string positions of S that relates S to itself. Then

$$L(G) = \{w \mid (S; I_S) \overset{*}{\Rightarrow}_G (w; I_w) \text{ for some } I_w\}.$$

We say that a local scattered context grammar is *component-local* if for each production $(A_1, \ldots, A_n) \to (\alpha_1, \ldots, \alpha_n; I_\alpha) \in P$, a nonterminal instance in α_i and a nonterminal instance in α_j are equivalent under I_α only if $i = j$.

We call this restriction "component-local" by analogy with tree-local MCTAG (Weir, 1988), because a production simultaneously rewrites multiple nonterminals with multiple components, but all those nonterminals must have come from the same component.

3. The formalisms considered as string-rewriting systems

Proposition 1. *CFG, TIG, RF-TAG, and CL-SCG are weakly equivalent.*

Proof. The weak equivalence of TIG to CFG was shown by Schabes and Waters (1995). The basic idea is to flatten each elementary tree into a CFG production, discarding every foot node, but adding a nonterminal to the left (right) of every node at which left-adjunction (resp., right-adjunction) is possible.

1. Note that this definition is stricter than Rogers' original definition, which allows "redundant" elementary trees. His parsing algorithm does not produce all possible derivations under the original definition, but does under the stricter definition.
2. This definition is based on local unordered scattered context grammar (Rambow and Satta, 1999), but is simplified in two ways: first, our scattered contexts are ordered rather than unordered; second, our productions explicitly specify which sets of nonterminals may be rewritten. We do not believe either of these simplifications affects the results shown here.

The weak equivalence of RF-TAG to CFG was shown by Rogers (1994). The basic idea is to break each elementary tree into CFG productions, augmenting the nonterminal alphabet with a stack of bounded depth to keep track of non-foot adjunctions.

For any CL-SCG, a weakly equivalent CFG can be obtained by a construction analogous to that for tree-local multicomponent TAG (Weir, 1988). Given a CL-SCG $G = (N, T, P, S)$, let f be the maximum number of components of (the left- or right-hand side of) any production in P, and let $N' = (\bigcup_{i=1}^{f} N^i \times \{1, \ldots f\})$. We want to break each production of P into its component CFG productions, using this augmented nonterminal alphabet to ensure that all the component productions are used together. To do this, construct P' from P as follows:

- For every nonterminal occurrence A_i on a left-hand side $(A_1, \ldots A_n)$, replace A_i with $(A_1, \ldots, A_n, i)$.
- For every nonterminal occurrence A_i on a right-hand side $(\alpha_1, \ldots, \alpha_n; I_\alpha)$, if A_i is the ith member of an equivalence class of I_α whose members are $A_1, \ldots, A_n$, in that order, replace A_i with $(A_1, \ldots, A_n, i)$.

Then construct P'' from P' as follows: if $(A'_1, \ldots, A'_n) \to (\alpha'_1, \ldots, \alpha'_n) \in P'$, then $A'_i \to \alpha'_i \in P''$ for all i between 1 and n. Then the CFG $(N', T, P'', (S, 1))$ is weakly equivalent to G. □

4. The formalisms considered as tree-rewriting systems

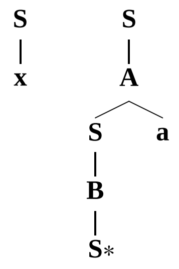
CFG generates only local sets, whereas CL-SCG and RF-TAG generate some recognizable sets which are not local. However, any recognizable set can be made from a local set by projection of labels (Thatcher, 1967). If we factor out this distinction, then CL-SCG and RF-TAG are no more powerful than CFG:

Proposition 2. *For any RF-TAG (or CL-SCG) G, there is a CFG G' and a projection of labels π such that $T(G) = \{\pi(t) \mid t \in T(G')\}$.*

Proof. The constructions used to prove the weak equivalence of these formalisms to CFG also preserve trees, modulo projection of labels. □

Proposition 3. *TIG can generate a tree set which is not recognizable.*

Proof. As was observed by Schabes and Waters (1995), the following TIG generates a non-recognizable set:



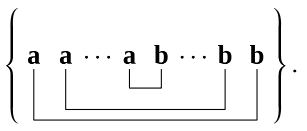
When the path set is intersected with $\{\mathbf{SA}\}^*\mathbf{S}\{\mathbf{BS}\}^*\mathbf{x}$, the result is $\{(\mathbf{SA})^n\mathbf{S}(\mathbf{BS})^n\mathbf{x} \mid n \geq 0\}$, a non-regular set. □

5. The formalisms considered as linking systems

We now define derivational generative capacity (henceforth DGC, introduced by Becker et al. (1992)), which measures the generative capacity of what Miller (1999) calls "linking systems."

Definition 5. We say that a grammar G *index-generates* an indexed string $(a_1 \cdots a_n; I_w)$ (see Definition 3) if G generates $a_1 \cdots a_n$ such that a_i and a_j are equivalent under I_w if and only if a_i and a_j are contributed by the same derivation step. The *derivational generative capacity* of a grammar G is the set of all indexed string sets index-generated by G.

In this section we notate indexed strings by drawing links (following Joshi (1985)) between all the positions of each equivalence class. Thus the CFG $\mathbf{X} \to \mathbf{aXb} \mid \epsilon$ index-generates the familiar-looking indexed string set



We saw in the previous section that with respect to SGC (in the sense of phrase structures), TIG was more powerful than CFG, whereas RF-TAG and CL-SCG were no more powerful than CFG. With respect to DGC, the opposite obtains.

Proposition 4. *CFG and TIG are derivationally equivalent.*

Proof. The construction given by Schabes and Waters (1995) preserves derivations, and therefore preserves indexed strings. □

On the other hand, RF-TAG and CL-SCG both have greater DGC than CFG (and TIG). Moreover, they extend CFG in different ways, because each is able to generate an indexed string set that the other is not.

Lemma 5 (indexed pumping lemma). *Let L be an indexed string set generated by a CFG (or CL-SCG). Then there is a constant n such that if $(z; I_z)$ is in L and $|z| \geq n$, then z may be rewritten as $uvwxy$, with $|vx| > 0$ and $|vwx| \leq n$, such that for all $i \geq 1$, there is an equivalence relation I_z^i such that $(uv^iwx^iy; I_z^i)$ is in L and I_z^i does not relate any positions in w to any positions in u or y.*

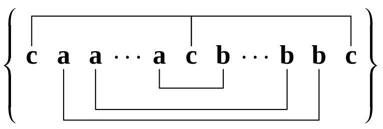
Proof. The proof is analogous to that of the standard pumping lemma (Hopcroft and Ullman, 1979). However, since the grammar cannot be put into Chomsky normal form, we let $n = m^k$ instead of 2^k, where k is the size of the nonterminal alphabet and m is the maximum number of symbols on any right-hand side. The key difference from the standard proof is the observation that since, for each i, the derivation of uv^iwx^iy can be written as

$$S \overset{*}{\Rightarrow} uAy \overset{*}{\Rightarrow} uv^iAx^iy \overset{*}{\Rightarrow} uv^iwx^iy$$

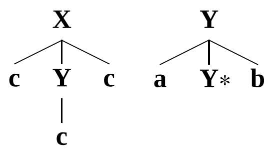
for some nonterminal A, no position in w can be contributed by the same derivation step as any position in u or y.

The generalization to CL-SCG is straightforward, since a CL-SCG G can be converted into a CFG G' which generates the same trees. G' will not generate the same indexed strings as G; nevertheless, the equivalence relations index-generated by G can only relate terminal instances which are first cousins in the derived tree, so for $i \geq 1$, it remains the case that no position in w is related to any position in u or y. □

Proposition 6. *The following indexed string set is index-generable by an RF-TAG but not by any CL-SCG:*

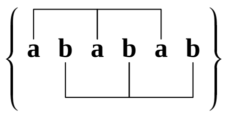
$L_1 =$


Proof. The following RF-TAG generates L_1:



But suppose L_1 is index-generated by some CFG or CL-SCG G. For any n given by the indexed pumping lemma, let $z = \mathbf{ca}^n\mathbf{cb}^n\mathbf{c}$ satisfy the conditions of the pumping lemma. It must be the case that v and x contain only **a**'s and **b**'s, respectively, or else $uv^iwx^iy \notin L_1$. But then u, w, and y would each have to contain one of the **c**'s, and since the **c**'s are all related, this contradicts the pumping lemma. □

Proposition 7. *The following indexed string set (consisting of a single string) is generable by a CL-SCG but not by any RF-TAG, nor indeed by any TAG:*

$L_2 =$


Proof. The following CL-SCG generates L_2:

$$(\mathbf{S}) \rightarrow (\mathbf{aB}_{\boxed{1}}\mathbf{aB}_{\boxed{1}}\mathbf{aB}_{\boxed{1}})$$

$$(\mathbf{B}, \mathbf{B}, \mathbf{B}) \rightarrow (\mathbf{b}, \mathbf{b}, \mathbf{b})$$

But L_2 cannot be generated by any TAG. In general, if $a_1 \cdots b \cdots a_2$ is index-generable by a TAG such that a_1 and a_2 are related, then either the tree that contributes a_1 and a_2 (call it β_a) adjoins into the tree that contributes b (call it β_b) such that its foot node dominates b, or else β_b adjoins into β_a. In the case of L_2, suppose that the **a**'s are contributed by β_a, and the **b**'s are contributed by β_b. If β_a adjoins into β_b, its foot node must dominate both of the **b**'s, which is a contradiction; similarly if β_b adjoins into β_a. □

6. The formalisms considered as local synchronous systems

In a *local synchronous system* (Aho and Ullman, 1969; Shieber, 1994; Rambow and Satta, 1996), two grammars are constrained to generate pairs of strings via isomorphic derivations (up to relabeling and reordering of sisters). Although translation via a synchronous grammar is not really an "interpretation" in Miller's sense, nevertheless, because a synchronous derivation in effect realizes a single derivation structure in two different ways, we expect the set of string relations generated by a local synchronous system to reveal something about the relationship between derivations and strings that weak generative capacity does not.

Definition 6. A *synchronous CFG* is a tuple $G = (N, T, P, S)$, where N, T, and S are as in ordinary CFG, and P is a set of productions of the form

$$(A : A') \rightarrow (\alpha : \alpha'; I_\alpha)$$

where $A, A' \in N$, $\alpha, \alpha' \in (N \cup T)^*$, and I_α is a bijection between nonterminal instances in α and nonterminal instances in α'.

We write $(\gamma : \gamma'; I_\gamma) \Rightarrow_G (\delta : \delta'; I_\delta)$, where $(\gamma : \gamma'; I_\gamma)$ and $(\delta : \delta'; I_\delta)$ are indexed string pairs, if and only if there exists a production $(A : A') \rightarrow (\alpha : \alpha'; I_\alpha) \in P$ such that

$$\gamma = \gamma_0 A \gamma_1 \qquad\qquad \gamma' = \gamma'_0 A' \gamma'_1$$

where A and A' are related under I_γ, and

$$\delta = \gamma_0 \alpha \gamma_1 \qquad\qquad \delta' = \gamma'_0 \alpha' \gamma'_1$$

where any nonterminal instances in γ_0, γ_1, γ'_0, or γ'_1 whose corresponding instances in γ are equivalent under I_γ are also equivalent under I_δ, as are any nonterminal instances in α and α' which are equivalent under I_α, and nothing else.

Let I_S be the equivalence relation on string positions of $(S : S)$ which relates both instances of S to each other. Then the *weak generative capacity* of G is the string relation

$$L(G) = \{(w : w') \mid (S : S; I_S) \overset{*}{\Rightarrow}_G (w : w'; I_w)\}.$$

The definition of synchronous TAG (Shieber, 1994) is analogous, but with bijections between adjunction sites instead of bijections between nonterminal instances; synchronous TIG and synchronous RF-TAG are just restrictions of synchronous TAG. The definition of synchronous CL-SCG is also analogous, but with bijections between equivalence classes of nonterminal instances instead of bijections between nonterminal instances. These four synchronous formalisms relate to each other in the same way as the linking systems of the previous section.

Proposition 8. *Synchronous CFG and synchronous TIG are weakly equivalent.*

Proof. The construction given by Schabes and Waters (1995) preserves derivations, and therefore preserves string pairs. □

Lemma 9 (synchronous pumping lemma). *Let L be a string relation generated by a synchronous CFG (or synchronous CL-SCG). Then there is a constant n such that if $(z : z')$ is in L and $|z| \geq n$ and $|z'| \geq n$, then $(z : z')$ may be written as $(uwy : u'w'y')$, and there exist strings v, x, v', x', such that $|vxv'x'| > 0$, $|vwx| \leq n$, $|v'w'x'| \leq n$, and for all $i \geq 0$, $(uv^iwx^iy : u'v'^iw'x'^iy')$ is in L.*

Proof. The proof is again analogous to that of the standard pumping lemma: let $G = (N, T, P, S)$ be a synchronous CFG generating L. We choose n as the proof of the standard lemma would if the nonterminal alphabet were $N \times N$. This guarantees the existence of a pair of corresponding paths in the derivation of $(z : z')$ such that the same pair of nonterminals $(A : A')$ occurs twice:

$$(S : S) \overset{*}{\Rightarrow} (uAy : u'A'y') \overset{*}{\Rightarrow} (u_1vAxy_1 : u'_1v'A'x'y'_1) \overset{*}{\Rightarrow} (u_1vwxy_1 : u'_1v'w'x'y'_1)$$

If we let $u = u_1v$ and $y = xy_1$, and likewise $u' = u'_1v'$ and $y' = x'y'_1$, then $(z : z') = (uwy : u'w'y')$, and for all $i \geq 0$, $(uv^iwx^iy : u'v'^iw'x'^iy') \in L$.

The CL-SCG case is similar but quite messy. A CL-SCG derivation tree has the same height as its derived tree, minus one. Therefore we can choose n such that any derivation of $(z : z')$ where $|z| \geq n$ and $|z'| \geq n$ must have a pair of corresponding paths such that the same set of nonterminals occurs twice (new material underlined for clarity):

$$\begin{aligned}(S : S) &\overset{*}{\Rightarrow} (u_0A_1u_1 \cdots A_ku_k \cdots A_nu_n : u'_0A'_1u'_1 \cdots A'_{k'}u_{k'} \cdots A'_{n'}u'_{n'})\\ &\overset{*}{\Rightarrow} (u_0\underline{v_1}u_1 \cdots \underline{v_{k0}A_1v_{k1} \cdots A_nv_{kn}}u_k \cdots \underline{v_n}u_n : u'_0\underline{v'_1}u'_1 \cdots \underline{v'_{k'0}A'_1v'_{k'1} \cdots A'_{n'}v'_{k'n'}}u'_{k'} \cdots \underline{v'_{n'}}u'_{n'})\\ &\overset{*}{\Rightarrow} (u_0v_1u_1 \cdots v_{k0}\underline{w_1}v_{k1} \cdots \underline{w_n}v_{kn}u_k \cdots v_nu_n : u'_0v'_1u'_1 \cdots v'_{k'0}\underline{w'_1}v'_{k'1} \cdots \underline{w'_{n'}}v'_{k'n'}u'_{k'} \cdots v'_{n'}u'_{n'})\end{aligned}$$

If we let

$$\begin{aligned}u &= u_0v_1u_1 \cdots v_{k-1}u_{k-1} & u' &= u'_0v'_1u'_1 \cdots v'_{k'-1}u'_{k'-1}\\ v &= v_{k0}v_1v_{k1} \cdots v_{k-1}v_{k,k-1} & v' &= v'_{k'0}v'_1v'_{k'1} \cdots v'_{k'-1}v'_{k',k'-1}\\ w &= v_{k0}w_1v_{k1} \cdots w_nv_{kn} & w' &= v'_{k'0}w'_1v'_{k'1} \cdots w'_{n'}v'_{k'n'}\\ x &= v_{k,k+1}v_{k+1} \cdots v_nv_{kn} & x' &= v'_{k',k'+1}v'_{k'+1} \cdots v'_{n'}v'_{k'n'}\\ y &= u_kv_{k+1}u_{k+1} \cdots v_nu_n & y' &= u'_{k'}v'_{k'+1}u'_{k'+1} \cdots v'_{n'}u'_{n'}\end{aligned}$$

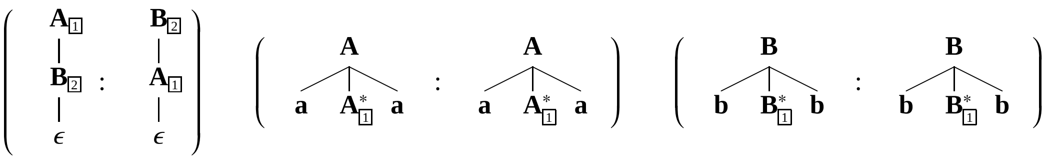
then $(z : z') = (uwy : u'w'y')$, and for all $i \geq 0$, $(uv^iwx^iy : u'v'^iw'x'^iy') \in L$. □

Proposition 10. *The string relation*

$$L_3 = \{(\mathbf{a}^m\mathbf{b}^n\mathbf{c}^n\mathbf{d}^m : \mathbf{b}^n\mathbf{a}^m\mathbf{d}^m\mathbf{c}^n)\}$$

is generable by a synchronous RF-TAG but not by any synchronous CL-SCG.

Proof. The following synchronous RF-TAG generates L_3:



But suppose L_3 can be generated by some CL-SCG G. For any n given by the pumping lemma, let $(z : z') = (\mathbf{a}^n\mathbf{b}^n\mathbf{c}^n\mathbf{d}^n : \mathbf{b}^n\mathbf{a}^n\mathbf{c}^n\mathbf{d}^n)$ satisfy the conditions of the pumping lemma. Then $vxv'x'$ must contain only **a**'s and **d**'s, or only **b**'s and **c**'s, otherwise $(uv^iwx^iy : u'v'^iw'x'^iy')$ will not be in L. But in the former case, $|vwx| > n$, and in the latter case, $|v'w'x'| > n$, which is a contradiction.[3] □

Proposition 11. *There is a string relation which is generable by a synchronous CL-SCG but not by any synchronous RF-TAG, nor indeed by any synchronous TAG.*

Proof. Define $L_4(k)$ to be

$$\{(w_1 \cdots w_{2n} : w_{\pi^k_n(1)} \cdots w_{\pi^k_n(2n)}) \mid w_i \in \{\sigma(i)\}^*, n \geq 1\},$$

where

$$\sigma(i) = \begin{cases}\mathbf{a} & \text{if } i \text{ is odd,}\\ \mathbf{b} & \text{if } i \text{ is even,}\end{cases}$$

3. An analogous result for synchronous regular-form two-level TAG was shown by Chiang et al. (2000).

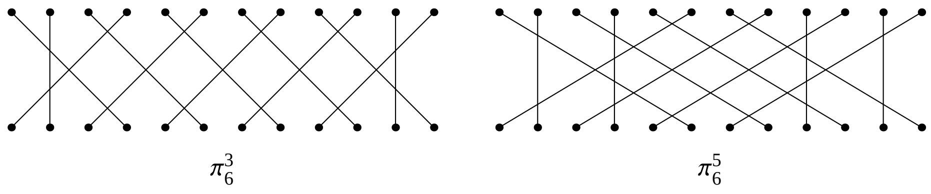


Figure 2: Example diagrams of π_n^k.

and

$$\pi_n^k(i) = \begin{cases} i+k & \text{if } i \text{ is odd and } i+k \leq 2n, \\ i-k & \text{if } i \text{ is even and } i-k \geq 1, \\ i & \text{otherwise.} \end{cases}$$

See Figure 2 for examples of π_n^k. The following synchronous CL-SCGs generate $L_4(3)$ and $L_4(5)$, respectively:

$$\begin{aligned}
(\mathbf{S} : \mathbf{S}) &\rightarrow (\mathbf{X}_{\boxed{1}}\mathbf{A}_{\boxed{2}}\mathbf{Y}_{\boxed{1}} : \mathbf{X}_{\boxed{1}}\mathbf{A}_{\boxed{2}}\mathbf{Y}_{\boxed{1}}) \\
(\mathbf{X}, \mathbf{Y} : \mathbf{X}, \mathbf{Y}) &\rightarrow (\mathbf{X}_{\boxed{1}}\mathbf{A}_{\boxed{2}}\mathbf{Y}_{\boxed{1}}, \mathbf{B}_{\boxed{3}} : \mathbf{X}_{\boxed{1}}\mathbf{B}_{\boxed{3}}\mathbf{Y}_{\boxed{1}}, \mathbf{A}_{\boxed{2}}) \\
&\mid\ (\epsilon, \mathbf{B}_{\boxed{1}} : \epsilon, \mathbf{B}_{\boxed{1}}) \\
(\mathbf{A} : \mathbf{A}) &\rightarrow (\mathbf{aA}_{\boxed{1}} : \mathbf{aA}_{\boxed{1}}) \ \mid\ (\epsilon : \epsilon) \\
(\mathbf{B} : \mathbf{B}) &\rightarrow (\mathbf{bB}_{\boxed{1}} : \mathbf{bB}_{\boxed{1}}) \ \mid\ (\epsilon : \epsilon)
\end{aligned}$$

$$\begin{aligned}
(\mathbf{S} : \mathbf{S}) &\rightarrow (\mathbf{X}_{\boxed{1}}\mathbf{A}_{\boxed{2}}\mathbf{Y}_{\boxed{1}}\mathbf{A}_{\boxed{3}}\mathbf{Z}_{\boxed{1}} : \mathbf{X}_{\boxed{1}}\mathbf{A}_{\boxed{2}}\mathbf{Y}_{\boxed{1}}\mathbf{A}_{\boxed{3}}\mathbf{Z}_{\boxed{1}}) \\
(\mathbf{X}, \mathbf{Y}, \mathbf{Z} : \mathbf{X}, \mathbf{Y}, \mathbf{Z}) &\rightarrow (\mathbf{X}_{\boxed{1}}\mathbf{A}_{\boxed{2}}\mathbf{Y}_{\boxed{1}}\mathbf{A}_{\boxed{3}}\mathbf{Z}_{\boxed{1}}, \mathbf{B}_{\boxed{4}}, \mathbf{B}_{\boxed{5}} : \mathbf{X}_{\boxed{1}}\mathbf{B}_{\boxed{4}}\mathbf{Y}_{\boxed{1}}\mathbf{B}_{\boxed{5}}\mathbf{Z}_{\boxed{1}}, \mathbf{A}_{\boxed{2}}, \mathbf{A}_{\boxed{3}}) \\
&\mid\ (\epsilon, \mathbf{B}_{\boxed{1}}, \mathbf{B}_{\boxed{2}} : \epsilon, \mathbf{B}_{\boxed{1}}, \mathbf{B}_{\boxed{2}}) \\
(\mathbf{A} : \mathbf{A}) &\rightarrow (\mathbf{aA}_{\boxed{1}} : \mathbf{aA}_{\boxed{1}}) \ \mid\ (\epsilon : \epsilon) \\
(\mathbf{B} : \mathbf{B}) &\rightarrow (\mathbf{bB}_{\boxed{1}} : \mathbf{bB}_{\boxed{1}}) \ \mid\ (\epsilon : \epsilon)
\end{aligned}$$

But suppose $L_4(3)$ is generated by a synchronous CFG G. Let r be the order of G, that is, the maximum number of nonterminals in either half of the right-hand side of any production. By means of closure properties it can be shown that there is a synchronous CFG G' of order r which generates the language

$$\{(w_1 \cdots w_{2n} : w_{\pi_n^3(1)} \cdots w_{\pi_n^3(2n)}) \mid w_i \in \{\mathbf{c}_i\}^*\},$$

where $n = \max\{3, \lceil \frac{r+1}{2} \rceil\}$ and π_n^k is as above. But this leads to a contradiction by means of the same argument used by Aho and Ullman (1969) to show that the synchronous CFGs of order $r+1$ properly include the synchronous CFGs of order r. The basic idea is as follows: we say that a production *covers* $\mathbf{c}_i$ if its right-hand side derives an unbounded number of $\mathbf{c}_i$'s. Then it can be shown that any production which covers two of the $\mathbf{c}_i$ must cover all of them. So there would have to be a production covering all $2n$ of the $\mathbf{c}_i$, with $2n$ nonterminals on its right-hand side, each of which gets rewritten with a production covering only one of the $\mathbf{c}_i$. But since $2n > r$, this is a contradiction.

For the TAG case, suppose $L_4(5)$ is generated by a synchronous TAG G, and again let r be the order of G. By means of closure properties it can be shown that there is a synchronous TAG G' of order r which generates the language

$$\{(w_1 \cdots w_{2n} : w_{\pi_n^5(1)} \cdots w_{\pi_n^5(2n)}) \mid w_i \in \{\mathbf{c}_i\}^*\},$$

where $n = 4r+1$ and π_n^k is as above. This case is more difficult because unlike a CFG nonterminal, a TAG auxiliary tree has a hole in its span created by its foot node. But it can be shown that any elementary tree pair which covers three of the $\mathbf{c}_i$, such that none of them are separated by a foot node in either the domain or the range, must cover all the $\mathbf{c}_i$ except perhaps one. Given our choice of n, this suffices to show that G' cannot exist.

$\square$

7. Conclusion

Joshi (2000) poses the question, "How much strong generative power can be squeezed out of a formal system without increasing its weak generative power?" The shifting relations between these four formalisms under different interpretations (see Figure 1) show that there is more than one way to answer this question.

First, there is more than one way to measure strong generative power. That TIG generates more tree sets than RF-TAG or CL-SCG but fewer indexed string sets demonstrates a point noted by Becker et al. (1992): that SGC (in the sense of phrase structures) and DGC are orthogonal notions. This is because SGC (in the sense of phrase structures) is based on the tree yield function, which can be chosen somewhat independently of the string yield function. On the other hand, DGC and synchronous WGC, which are both based on the string yield function, order our four formalisms in the same way.

Second, there is more than one way to squeeze a formal system. RF-TAG and CL-SCG are incommensurate with respect to both DGC and synchronous WGC; that is, each is able to do something the other is not. Thus, even under a particular interpretation of strong generative power, the question is not only, how much strong generative power can be squeezed, but also, in what ways? Characterizing the different ways in which strong generative power can be both measured and squeezed is a task for future research.

Acknowledgements

This research was supported in part by NSF grant SBR-89-20230. I would like to thank Anoop Sarkar, William Schuler, Aravind Joshi, and the three anonymous reviewers for their valuable comments. *S. D. G.*

References

Aho, A. V. and J. D. Ullman. 1969. Syntax Directed Translations and the Pushdown Assembler. *J. Comp. Sys. Sci.*, 3:37–56.

Becker, Tilman, Owen Rambow and Michael Niv. 1992. The derivational generative power of formal systems, or, Scrambling is beyond LCFRS. Technical Report IRCS-92-38, Institute for Research in Cognitive Science, University of Pennsylvania. Presented at MOL3.

Bod, Rens. 1992. Data-oriented parsing. In *Proceedings of the Fourteenth International Conference on Computational Linguistics (COLING-92)*, Nantes.

Chiang, David. 2000. Statistical parsing with an automatically-extracted tree adjoining grammar. In *Proceedings of the 38th Annual Meeting of the Assocation for Computational Linguistics*, pages 456–463, Hong Kong.

Chiang, David, William Schuler and Mark Dras. 2000. Some remarks on an extension of synchronous TAG. In *Proceedings of the Fifth International Workshop on TAG and Related Formalisms (TAG+5)*, pages 61–66.

Chomsky, Noam. 1963. Formal properties of grammars. In R. Duncan Luce, Robert R. Bush and Eugene Galanter, editors, *Handbook of Mathematical Psychology*. Wiley, New York, pages 323–418.

Dras, Mark. 1999. A meta-level grammar: redefining synchronous TAG for translation and paraphrase. In *Proceedings of the 37th Annual Meeting of the Assocation for Computational Linguistics*, pages 80–87, College Park, MD.

Hopcroft, John E. and Jeffrey D. Ullman. 1979. *Introduction to automata theory, languages, and computation*. Reading, MA: Addison-Wesley.

Joshi, Aravind K. 1985. Tree adjoining grammars: How much context-sensitivity is necessary for assigning structural descriptions? In David Dowty, Lauri Karttunen and Arnold Zwicky, editors, *Natural Language Parsing*. Cambridge University Press, Cambridge, pages 206–250.

Joshi, Aravind K. 2000. Relationship between strong and weak generative power of formal systems. In *Proceedings of the Fifth International Workshop on TAG and Related Formalisms (TAG+5)*, pages 107–113.

Kroch, Anthony and Aravind K. Joshi. 1987. Analyzing extraposition in a tree adjoining grammar. In Geoffrey J. Huck and Almerindo E. Ojeda, editors, *Discontinuous Constituency*. Academic Press, Orlando.

Miller, Philip H. 1999. *Strong Generative Capacity: The Semantics of Linguistic Formalism*. CSLI lecture notes, number 103. Stanford: CSLI Publications.

Rambow, Owen and Giorgio Satta. 1996. Synchonous Models of Language. In *Proceedings of the 34th Annual Meeting of the Assocation for Computational Linguistics*, pages 116–123, Santa Cruz, CA.

Rambow, Owen and Giorgio Satta. 1999. Independent parallelism in finite copying parallel rewriting systems. *Theoretical Computer Science*, 223:887–120.

Rogers, James. 1994. Capturing CFLs with tree adjoining grammars. In *Proceedings of the 32nd Annual Meeting of the Assocation for Computational Linguistics*, pages 155–162, Las Cruces, NM.

Schabes, Yves and Richard C. Waters. 1995. Tree insertion grammar: a cubic-time parsable formalism that lexicalizes context-free grammar without changing the trees produced. *Computational Linguistics*, 21:479–513.

Shieber, Stuart M. 1994. Restricting the weak generative capacity of synchronous tree-adjoining grammars. *Computational Intelligence*, 10(4):371–385, November. Special Issue on Tree Adjoining Grammars.

Thatcher, J. W. 1967. Characterizing Derivation Trees of Context-Free Grammars through a Generalization of Finite Automata Theory. *J. Comp. Sys. Sci.*, 1:317–322.

Weir, David J. 1988. *Characterizing Mildly Context-Sensitive Grammar Formalisms*. Ph.D. thesis, Univ. of Pennsylvania.

Supertagging for Combinatory Categorial Grammar

Stephen Clark
Division of Informatics, University of Edinburgh
2 Buccleuch Place, Edinburgh, EH8 9LW
Scotland, UK
`Stephen.Clark@ed.ac.uk`

1. Introduction

Supertagging was introduced for Lexicalised Tree Adjoining Grammar (LTAG) to reduce the number of elementary trees assigned to a word, thereby increasing parsing efficiency (Bangalore and Joshi, 1994). Bangalore and Joshi (1999) have shown that techniques used for POS-tagging can be applied successfully to the supertagging problem.

Parsing efficiency is also an issue for Combinatory Categorial Grammar (CCG, Steedman (2000)) since many words can have many possible CCG categories. We have developed a supertagger for CCG, similar to the POS-tagger of Ratnaparkhi (1996). Maximum entropy models are used to estimate the probability that a word is assigned a particular category, given the local context. These probabilities are then used to select a sub-set of the possible categories for a word.

The next section gives a brief introduction to CCG, and Section 3 describes the category set used in the experiments. Section 4 describes a "single-category" supertagger, and gives figures for its accuracy. Section 5 shows how the supertagger can be adapted to output more than one category per word, to produce a "multi-tagger", and we show the effect the multi-tagger has on the speed and coverage of a CCG parser.

2. Combinatory Categorial Grammar

A grammar in CCG consists of a lexicon, which pairs words with lexical categories, and a set of combinatory rules, which specify how categories combine. Categories are either atomic or complex. Examples of atomic categories include S (sentence), N (noun), NP (noun phrase) and PP (prepositional phrase). Features on the S category can be used to indicate types of sentence (or clause); for example: S[dcl] (declarative), S[to] (to-infinitival), S[b] (bare-infinitival), S[adj] (adjectival).

Complex categories are functors which specify the type and directionality of the arguments, and the type of the result. For example, one of the categories for the verb *likes* specifies that one noun phrase (NP) is required to the right of the verb, and one to the left, resulting in a sentence (as in *John likes sweets*):

likes := (S[dcl]\NP)/NP

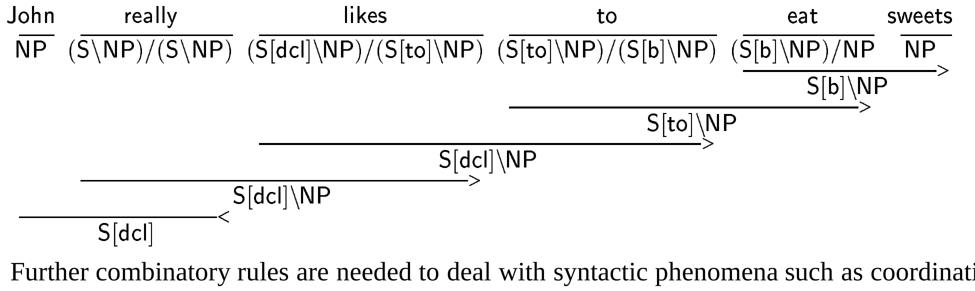
Another category for *likes* specifies that a to-infinitival clause is required to the right of the verb (as in *John likes to eat sweets*):

likes := (S[dcl]\NP)/(S[to]\NP)

Functor categories can also express modification, as in the following adverbial category for *really*:

really := (S\NP)/(S\NP)

The following derivation shows how categories combine. This derivation uses only two combinatory rules: forward application (>) and backward application (<).

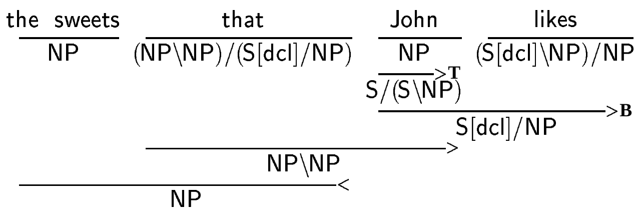


Further combinatory rules are needed to deal with syntactic phenomena such as coordination and extraction.

frequency cut-off	# cat types in cat-set	# cat tokens in 2-21 not in cat-set	# sentences in 2-21 with missing cat	# cat tokens in 00 not in cat-set	# sentences in 00 with missing cat
1	1,206	0	0	11 (0.02%)	11 (0.6%)
5	512	1,157 (0.1%)	1,032 (2.6%)	49 (0.1%)	44 (2.3%)
10	398	1,898 (0.2%)	1,667 (4.3%)	79 (0.2%)	67 (3.5%)
20	304	3,190 (0.4%)	2,756 (7.0%)	123 (0.3%)	104 (5.5%)

Table 1: Category coverage for seen data (sections 2-21) and unseen data (section 00) under various cut-offs

In the following object-extraction example, *type-raising* (>**T**) turns the atomic NP category for *John* into a functor category looking for a verb-phrase to its right; *forward composition* (>**B**) then combines the type-raised category with the category for *likes*:



Note that, in this paper, we assume type-raising is dealt with by the parser rather than at the level of the lexicon. Thus the supertagger does not deal with type-raised categories.

3. The Lexical Category Set

The category set used here has been derived from a treebank of CCG (normal-form) derivations, in which each word is tagged with a lexical category.[1] The treebank, which we call CCG-bank, has been created by Julia Hockenmaier (Hockenmaier and Steedman, 2002a), and derived semi-automatically from the Penn Treebank (Marcus, Santorini and Marcinkiewicz, 1993). Below is an example sentence from the CCG-bank, together with the lexical categories.[2]

Pierre N/N Vinken N , , 61 N/N years N old (S[adj]\NP)\NP , , will (S[dcl]\NP)/(S[b]\NP)
join ((S[b]\NP)/PP)/NP the NP/N board N as PP/NP a NP/N nonexecutive N/N
director N Nov. ((S\NP)\(S\NP))/N 29 N . .

The category set was obtained from sections 2-21 of the CCG-bank (corresponding to the Penn Treebank sections), which contain $39,161$ sentences, $903,661$ category tokens, and $1,206$ category types. Many of the category types occur only a few times in the data, and some arise through noise in the Penn Treebank, or errors introduced by the CCG extraction program. We therefore investigated using a subset of the $1,206$ category types by applying a frequency cut-off.

Table 1 shows how many category tokens in Sections 2-21 do not appear in the category set entailed by the cut-off, and also shows the percentage of sentences that have at least one category token not in the set. The same figures are given for Section 00, which has $1,900$ sentences and $44,544$ category tokens. The figures show that the size of the category set can be reduced by as much as $3/4$, without greatly increasing the percentage of missing category tokens in the data (unseen and seen).

The LTAG supertag set used by Bangalore and Joshi (1999) contained 300 supertags (for their WSJ experiments). To obtain a CCG category set as compact as that requires a cut-off as high as 20. However, Bangalore and Joshi took their supertags from the manually created XTAG grammar, which presumably contains a much cleaner set of supertags than an automatically extracted grammar.

A more useful comparison is with the work of Chen and Vijay-Shanker (2000), who extract an LTAG automatically from the Penn Treebank. A number of strategies are used for extracting the supertags (referred to as *tree*

1. There is a distinction between lexical categories and categories created during a derivation. Since we are only interested in lexical categories in this paper, we may sometimes use *category* to mean *lexical category*.
2. The category for *join* has a PP-complement, which is arguably incorrect. Since the distinction between complements and adjuncts is not always reliably marked in the Penn Treebank, the procedure for identifying complements and adjuncts does lead to some erroneous data in the CCG-bank.

Experiment 1		Experiment 2		Experiment 3		Experiment 4	
Condition	Contextual predicates						
w_i is not rare	$w_i = X$	$\forall w_i$	$t_i = X$	$\forall w_i$	$t_i = X$	$\forall w_i$	$t_i = X$
w_i is rare	X is prefix of w_i, $\|X\| \leq 4$		$t_{i-1} = X$		$t_{i-1} = X$		$t_{i-1} = X$
	X is suffix of w_i, $\|X\| \leq 4$		$t_{i-2} = X$		$t_{i-2} = X$		$t_{i-2} = X$
	w_i contains a number		$t_{i+1} = X$		$t_{i+1} = X$		$t_{i+1} = X$
	w_i contains uppercase character		$t_{i+2} = X$		$t_{i+2} = X$		$t_{i+2} = X$
	w_i contains a hyphen				$w_{i-3} = X$		$c_{i-2}c_{i-1} = XY$
$\forall w_i$	$c_{i-1} = X$				$w_{i+3} = X$		$w_{i-2}w_{i-1} = XY$
	$c_{i-2} = X$				$t_{i-3} = X$		$w_{i+1}w_{i+2} = XY$
	$w_{i-1} = X$				$t_{i+3} = X$		$t_{i-2}t_{i-1} = XY$
	$w_{i-2} = X$						$t_{i+1}t_{i+2} = XY$
	$w_{i+1} = X$						
	$w_{i+2} = X$						

Table 2: Contextual predicates used in the experiments. The predicates shown for experiments 2, 3 and 4 are in addition to those used in 1; the rare word predicates are only used in experiment 1.

frames by Chen and Vijay-Shanker), but the size of the resulting sets range between $2,366$ and $8,996$ supertags. Chen and Vijay-Shanker also experimented with a cut-off, with a value of 3 reducing the size of the supertag sets by at least 1/2, and a cut-off of 9 producing sets ranging between around 800 and 1,800 supertags. These numbers suggest that the CCG category sets extracted from CCG-bank are more compact than Chen and Vijay-Shanker's LTAG supertag sets (for equivalent cut-off values), although we should point out that CCG-bank has received a significant amount of manual clean-up.

4. The supertagger

The supertagger uses conditional maximum entropy models to estimate the probability of words being assigned particular categories. We chose to implement a maximum entropy supertagger, rather than the HMM supertagger used by Bangalore and Joshi (1999), because of the ease with which additional features can be integrated into the model. The use of conditional models, rather than the generative model of the HMM, also makes it easy to define a "multi-tagger", as we show in Section 5.

The probability of a category, c, given a context, h, is defined as follows:

$$p(c|h) = \frac{1}{Z(h)} e^{\sum_i \lambda_i f_i(c,h)} \tag{1}$$

The functions $f_i(c,h)$ define "features" of the category and context, and $Z(h)$ is a normalisation constant. An example feature is as follows:

$$f_j(c,h) = \begin{cases} 1 & \text{if } \texttt{curr\_word\_is\_the}(h) = \textit{true} \ \& \ c = \texttt{NP/N} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

The feature function takes the value 1 if the current word (i.e. word to be tagged) is *the* and the category is $\texttt{NP/N}$. The weight (λ_j) corresponding to the feature contributes to the probability $p(c|h)$ when h contains *the* as the current word and $c = \texttt{NP/N}$. Generalised Iterative Scaling is used to estimate the values of the weights.

The predicate `curr_word_is_the` is an example of what Ratnaparkhi (1998) calls a *contextual predicate*. Contextual predicates identify elements of the context that might be useful in predicting the lexical category. Possible contextual predicates for the supertagger include the current word, certain properties of the current word (such as suffix and prefix information), the POS-tag of the current word, words either side of the current word, and the POS-tags of those words.

We experimented with a number of contextual predicates, and different window sizes for the context. Table 2 gives "templates" for the contextual predicates used in a series of experiments. The first set used in experiment 1 is based on that used by Ratnaparkhi (1998). (The notation in the table is also borrowed from Ratnaparkhi (1998).) The current word, w_i, is *rare* if it appears less than a certain number of times in the training data. A cut-off value

	expt.1	expt.2	expt.3	expt.4
CCG supertagger	88.1	90.4	90.5	90.5
Baseline	71.2			
CCG (Nielsen)	87.7			
CCG (TnT)	87.8			
LTAG (Srinivas)	92.2			
LTAG (Chen)	78.9			

Table 3: Results for the single supertagger

of 5 was used here. If w_i is not rare, the word itself forms a contextual predicate, otherwise contextual predicates based on the suffixes and prefixes (up to 4 characters) are used, plus some others listed in the table. In addition, the two categories before w_i (c_{i-1}, c_{i-2}) form contextual predicates, along with the two words before and after w_i. Thus in experiment 1 a 5-word window is providing the context.

Experiment 2 introduces the POS-tags (t_j) of the words in the 5-word window, but does not use the rare word features, since we found that ignoring the rare word features once the POS-tags had been introduced did not affect the results. The POS-tagger of Ratnaparkhi (1998) was used to provide the POS-tags. Experiment 3 extends the context to a 7-word window. Experiment 4 retains the 5-word window, but forms complex features by considering pairs of categories, words, and POS-tags.

Following Ratnaparkhi (1998), a simple feature selection technique was used for all the experiments: only those features that appear at least 10 times in the training data are considered (except the current word features which only have to appear 5 times).

Initially we developed a supertagger that chooses the most probable sequence of categories, given the sentence. The probability of a category sequence, C, given a sentence S, is defined as follows:

$$p(C|S) = \prod_i p(c_i|h_i) \tag{3}$$

where c_i is the ith category in the sequence and h_i is the context for the ith word. A beam search is used, so that only the top N sequences are retained at each point in the tagging process. A value of $N = 10$ was used here.

The supertagger consults a “tag-dictionary”, which contains, for each word, the set of categories the word was seen with in the data. If a word appears at least K times, the supertagger only considers the categories in the word’s category set. If a word appears less than K times, all categories are considered.

Table 3 gives the results on Section 23 of the CCG-bank (ignoring punctuation) using the category set with 398 types, and a value of $K = 20$ for the tag dictionary.[3] Sections 2-21 were used for training. The baseline result is obtained by choosing the category the current word appears with most frequently in the data, and assigning category N to unseen words.

Results are also given for two HMM supertaggers: Brants’ TnT tagger (Brants, 2000), and a supertagger developed by Nielsen (2001).[4] The Nielsen figure was obtained using a slightly older version of the CCG-bank, and does include punctuation. The Brants’ figure was obtained by training the publically available version of TnT on the latest version of the CCG-bank, and does not include punctuation.[5] The HMM results are similar to those for the maximum entropy experiment 1, which uses a similar set of features. The improvement in experiment 2 over the HMM supertaggers is because the HMM supertaggers did not use POS-tag information.

Results are also given for LTAG supertagging (Bangalore and Joshi, 1999; Chen and Vijay-Shanker, 2000) on the Penn Treebank. The Srinivas supertagger uses a manually created set of elementary trees, whereas the Chen supertagger uses an automatically extracted tag set, which explains the difference in the LTAG results.

We have integrated the supertagger with a CCG parser (Hockenmaier and Steedman, 2002b), and attempted to parse the sentences from the Penn Treebank. If the supertagger assigns a single category to each word, the parser is able to provide an analysis for around 91% of the sentences in Section 23. To try and increase the coverage we investigated assigning more than one category to each word.

3. Very similar results were obtained for values of K between 5 and 100 using Section 00 as a development set.
4. The TnT tagger was designed for POS-tagging, but it can be easily adapted to the supertagging problem by simply training it on supertagged data.
5. Including punctuation increases the figure by around 1%, because the categories for punctuation marks such as comma and period are simply the punctuation marks themselves.

CCG multi-tagger		# cats/word	LTAG multi-tagger	# cats/word
$\beta = 0.1$	96.2	1.6	94.6	1.9
$\beta = 0.05$	97.2	2.0	95.8	2.2
$\beta = 0.01$	98.4	3.8	97.0	3.4

Table 4: Results for the multi-tagger

5. Using the supertagger as a multi-tagger

A feature of the conditional model in equation 1 is that it is very easy to define a multi-tagger: simply assign all categories to a word whose probabilities (given by equation 1) are within some factor, β, of the highest probability category. Note that equation 3 is no longer used, since we are no longer estimating the probability of a sequence of categories. The feature set used in experiment 4 for the single tagger was used (but without the previous category features), with the category set of 398 category types, and $K = 20$. The results on Section 23 are given in Table 4, for various values of β, together with the average number of categories per word.

The LTAG results of Chen, Bangalore and Vijay-Shanker (1999), using a supertag set derived from the XTAG grammar, are given for comparison. Table 3 showed the single tagger results to be better for LTAG, at least for this supertag set (Chen et al. report a similar result), but the multi-tagger results are better for CCG, using a much simpler approach. Chen, Bangalore and Vijay-Shanker (1999) used a clustering technique to assign more than one category to each word.

Using a multi-tagger rather than a single-tagger in conjunction with the Hockenmaier and Steedman (2002a) parser greatly increases the coverage. For a β value of 0.01, over 99% of Section 23 could be parsed. To give some indication of how changes in β affect the speed of the parser, we took the first 500 sentences from Section 23 and recorded the parse times for various values of β. The times depend on various parameter settings used by the parser, such as beam width, but the figures reported here are intended only to give a rough indication of how the supertagger can speed up the parser. A β value of 0.0001 (which gives an average of 28 categories per word) was used to provide a baseline figure. Changing β to 0.01 (3.8 categories per word) increased the speed of the parser by a factor of 2.7 over the baseline figure. Changing β to 0.1 (1.6 categories per word) increased the speed of the parser by a factor of 4.7 over the baseline figure (but with a slight loss in coverage).

6. Conclusion

The supertagger described here can be used to significantly speed up a CCG parser. However, assigning a single category to each word in a sentence allows the parser of Hockenmaier and Steedman (2002a) to provide an analysis for only 91% of the sentences from Section 23 of the Penn Treebank. A simple multi-tagging approach can greatly increase the coverage, while keeping the average number of categories to a manageable number. The accuracy of the supertagger compares favourably with existing LTAG supertaggers, particularly those which use an automatically extracted LTAG.

One question which has not been discussed in this paper is whether supertagging is easier for LTAG or CCG. The comparison with the work of Chen and Vijay-Shanker (2000) suggests that it may be easier to do supertagging with an automatically extracted CCG than with an automatically extracted LTAG, although of course this depends crucially on the extracted grammars in each case.

There is also the question of how much work remains to be done by the parser after supertagging has taken place. Bangalore (2000) argues that, for LTAG, simple heuristics can be used to satisfy the dependency requirements encoded in the supertags. Whether such an approach would work for CCG remains an open question.

7. Acknowledgements

This research was funded by EPSRC grant GR/M96889/01. I would like to thank Julia Hockenmaier for providing the CCG-bank and for getting the results for her parser, and Mark Steedman for his guidance and help with this work.

References

Bangalore, Srinivas. 2000. A Lightweight Dependency Analyser for Partial Parsing. *Natural Language Engineering*, 6(2):113–138.

Bangalore, Srinivas and Aravind Joshi. 1994. Disambiguation of Super Parts of Speech (or Supertags): Almost Parsing. In *Proceedings of the 15th COLING Conference*, pages 154–160, Kyoto, Japan.

Bangalore, Srinivas and Aravind Joshi. 1999. Supertagging: An Approach to Almost Parsing. *Computational Linguistics*, 25(2):237–265.

Brants, Thorsten. 2000. TnT - a statistical part-of-speech tagger. In *Proceedings of the 6th Conference on Applied Natural Language Processing*.

Chen, John, Srinivas Bangalore and K. Vijay-Shanker. 1999. New Models for Improving Supertag Disambiguation. In *Proceedings of the 9th Meeting of EACL*, Bergen, Norway.

Chen, John and K. Vijay-Shanker. 2000. Automated Extraction of TAGS from the Penn Treebank. In *Proceedings of IWPT 2000*, Trento, Italy.

Hockenmaier, Julia and Mark Steedman. 2002a. Acquiring Compact Lexicalized Grammars from a Cleaner Treebank. In *Proceedings of the Third International Conference on Language Resources and Evaluation (to appear)*, Las Palmas, Spain.

Hockenmaier, Julia and Mark Steedman. 2002b. Generative Models for Statistical Parsing with Combinatory Categorial Grammar. In *Proceedings of the 40th Meeting of the ACL (to appear)*, Philadelphia, PA.

Marcus, Mitchell, Beatrice Santorini and Mary Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Nielsen, Lief. 2001. Supertagging with Categorial Grammar. Master's thesis, University of Cambridge.

Ratnaparkhi, Adwait. 1996. A Maximum Entropy Part-Of-Speech Tagger. In *Proceedings of the EMNLP Conference*, pages 133–142, Philadelphia, PA.

Ratnaparkhi, Adwait. 1998. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. Ph.D. thesis, University of Pennsylvania.

Steedman, Mark. 2000. *The Syntactic Process*. Cambridge, MA: The MIT Press.

Learning languages from positive examples with dependencies

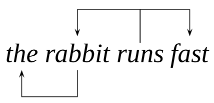
Jérôme Besombes, Jean-Yves Marion
Loria, INRIA Lorraine

Abstract

We investigate learning dependency grammar from positive data, in Gold's identification in the limit model. Examples are dependency trees. For this, we introduce reversible lexical dependency grammars which generate a significant class of languages. We have demonstrated that reversible dependency languages are learnable. We provide a $O(n^2)$-time, in the example size, algorithm. Our objective is to contribute to design and the understanding of formal process of language acquisition. For this, dependency trees play an important role because they naturally appear in every tree phrase structure.

1. An identification paradigm

From Tesniére (Tesnière, 1959) seminal study, and from ideas of Mel'čuk (Mel'čuk, 1988), we propose a two tier communication process between two speakers, see Figure 1. Jean transmits a sentence to Marie. At the first stage, Jean generates a structural sentence, like the following dependency tree



Then, Jean transforms it into a linear phrase, *the rabbit runs fast*, and send it to Marie. Now, Marie has to inverse the two tier process of Jean. For this, she has (i) to recover the structural sentence from the linear sentence (η^{-1}), (ii) to build/update/improve her grammar in order to understand the message, and to generate other messages (θ^{-1}). In the setting of natural language learning, parsers perform the first task of Marie. Roughly speaking, parsing corresponds to inverse η, that is to compute η^{-1}.

We are investigating identification, that is exact learning, of dependency tree languages. The leading idea is that data used to learn are dependency trees. Dependencies are semantic annotation by additional informations which facilitates the learning process. Such data are widely available and come from a lot of computational linguistic formal grammars such as LFG, TAG, categorial grammar and interaction grammar. The relations between those formalism are explained in the special issue of TAL (Kahane, 2000) on dependency grammars.

The data available are positive examples of a language, that is a sequence of dependency trees. Our hypothesis is that the computation of θ is reversible, that is the inputs can always be deduced from the outputs. So, identification corresponds to inverse θ. For this, we give a sufficient condition of reversibility on the grammars (G_0). We show that the class of reversible dependency tree languages is efficiently identifiable in Gold's model (Gold, 1967). That is, given a finite sequence of examples of a reversible language, we determine a reversible grammar that generates it. We refer to (Jain *et al.*, 1999) for further explanations. Our study leans on the work of Angluin (Angluin, 1982) on learning languages produced by deterministic and reversibles finite automaton. It is also closely related to Sakakibara (Sakakibara, 1992) work on reversible context free grammars and to Kanazawa (Kanazawa, 1998) work on rigid categorial grammars.

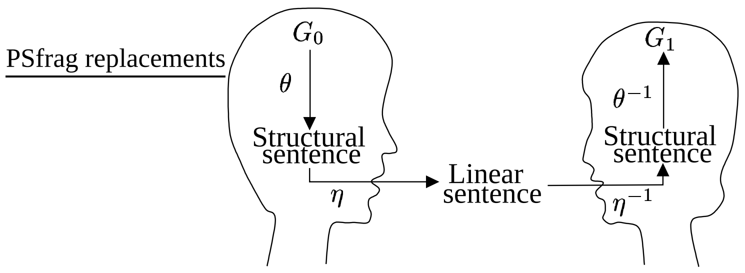


Figure 1: A two tier communication process

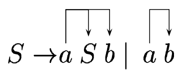
© 2000 . *Proceedings of the Sixth International Workshop on Tree Adjoining Grammar and Related Frameworks (TAG+6)*, pp. 25–29. Universitá di Venezia.

2. Lexical dependency grammar

Following Dikovsky and Modina (Dikovsky and Modina, 2000), we present a class of projective dependency grammars which was introduced by Hays (Hays, 1961) and Gaifman (Gaifman, 1965).

A *lexical dependency grammar* (LDG) Γ is a quadruplet $\langle \Sigma, N, P, S\rangle$, where Σ is the set of terminal symbols, N is the set of non-terminal symbols, $S \in N$ is the start symbol, and P is the set of productions. Each production is of the form $X \to U_1 \dots U_p\ a\ V_1 \dots V_q$, where $X \in N$, each U_i and V_j are in $\Sigma \cup N$. The terminal symbol a is called the *head* of the production. In other words, the head is the root of the flat tree formed by the production right handside. Actually, if we forget dependencies, we just deal with context free grammars.

Example 1. The grammar $\Gamma_0 = \langle \{a, b\}, \{S\}, P, S\rangle$ where P consists

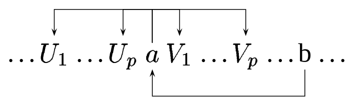


$$S \to a\ S\ b \mid a\ b$$

Partial dependency trees are recursively defined as follows.

1. S is a partial dependency tree generated by Γ.
2. If $\dots X \dots$b$\dots$ is a partial dependency tree generated by Γ, and if

   $X \to U_1 \dots U_p\ a\ V_1 \dots V_q$ is a production of Γ, then

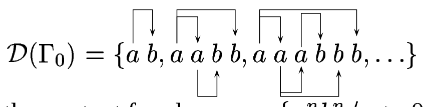


$$\dots U_1 \dots U_p\ a\ V_1 \dots V_p \dots \text{b} \dots$$

   is a partial dependency tree generated by Γ.

A *dependency tree* generated by a LDG Γ is a partial dependency tree of Γ in which all nodes are terminal symbols. The language $\mathcal{D}(\Gamma)$ is the set of all dependency trees generated by Γ.

Example 2. The language generated by Γ_0 of Example 1 is



$$\mathcal{D}(\Gamma_0) = \{a\ b, a\ a\ b\ b, a\ a\ a\ b\ b\ b, \dots\}$$

Without dependencies, we recognize the context free language $\{a^n b^n / n > 0\}$.

3. Reversible LDG

A LDG grammar Γ is *reversible* if the conditions **R1**, **R2** and **R3** of Figure 2 are satisfied. The class of *reversible dependency tree languages* is the class of languages generated by reversible LDG.

4. The learning algorithm

The learning algorithm works as follows. The input is a finite set H of positive examples which are dependency trees. Define $\mathtt{TG}(H)$ as the grammar constructed from all productions outputed by $\mathtt{TG}(w, S)$, for each $w \in H$. The function TG is described in Figure 3.

Stage 0 $G_0 = \mathtt{TG}(H)$.

Stage n+1 The grammar G_{n+1} is defined by applying one of the rule of Figure 2.

R1 If $X \rightarrow \mathbf{U}\ a\ \mathbf{V}$ and if $Y \rightarrow \mathbf{U}\ a\ \mathbf{V}$, then $X = Y$.

R2 If $X \rightarrow \boldsymbol{\alpha}\ Y\boldsymbol{\beta}\ a\ \boldsymbol{\gamma}$ and if $X \rightarrow \boldsymbol{\alpha}\ Z\boldsymbol{\beta}\ a\ \boldsymbol{\gamma}$, then $Y = Z$, where $Y, Z \in N$.

R3 If $X \rightarrow \boldsymbol{\alpha}$ a $\boldsymbol{\beta}\ Y\ \boldsymbol{\gamma}$ and if $X \rightarrow \boldsymbol{\alpha}$ a $\boldsymbol{\beta}\ Z\ \boldsymbol{\gamma}$, then $Y = Z$, where $Y, Z \in N$.

We write $\boldsymbol{\alpha}\ a\ \boldsymbol{\beta}$ for $\alpha_1 \ldots \alpha_p\ a\ \beta_1 \ldots \beta_q$.

Figure 2: Reversibility conditions and reduction rules.

```
Function TG($w, X$)
Inputs : $w$ is a dependency tree,
         $X$ is a non-terminal symbol.
if $w \in \Sigma$
  then Output $X \rightarrow w$

if $w = u_1 \ldots u_p$ a $v_1 \ldots v_q$
  then Take new $U_1, \ldots, U_p$ and $V_1, \ldots, V_q$

       Output $X \rightarrow U_1 \ldots U_p$ a $V_1 \ldots V_q$
       for $i = 1$ to $p$ do TG($u_i, U_i$)
       for $i = 1$ to $q$ do TG($v_i, V_i$)
```

Figure 3: $\mathtt{TG}(w, X)$ recognizes exactly w, i.e. $\mathcal{D}(\mathtt{TG}(w, X)) = \{w\}$

The process terminates at some stage m because the number of grammar productions decreases at each stage. So, put $\phi(H) = G_m$.

Theorem 3. *ϕ learns the class of reversible dependency tree languages.*

The learning algorithm is incremental and runs in quadratic time in the size of the examples. Our algorithm is implemented as a Java prototype which is accessible from `http://www.loria.fr/~ besombes`.

Example 4. $H = \{$ *the rabbit is very fast*, *the rabbit is fast*, *the rabbit is very very fast*$\}$,

The productions of $G_0 = \mathtt{TG}(H)$ are

$$\mathsf{TG}(\textit{the rabbit is very fast}, S) = \begin{cases} S \ \rightarrow X_1 \textit{ is } X_2, \\ X_1 \rightarrow X_3 \textit{ rabbit}, \\ X_2 \rightarrow X_4 \textit{ fast}, \\ X_3 \rightarrow \textit{the}, \\ X_4 \rightarrow \textit{very} \end{cases}$$

$$\mathsf{TG}(\textit{the rabbit is fast}, S) = \begin{cases} S \ \rightarrow X_5 \textit{ is } X_6, \\ X_5 \rightarrow X_7 \textit{ rabbit}, \\ X_6 \rightarrow \textit{fast}, \\ X_7 \rightarrow \textit{the}. \end{cases}$$

$$\mathsf{TG}(\textit{the rabbit is very very fast}\}, S) = \begin{cases} S \ \rightarrow X_8 \textit{ is } X_9, \\ X_8 \rightarrow X_{10} \textit{ rabbit}, \\ X_9 \rightarrow X_{11} \textit{ fast}, \\ X_{10} \rightarrow \textit{the}, \\ X_{11} \rightarrow \textit{very } X_{12}, \\ X_{12} \rightarrow \textit{very} \end{cases}$$

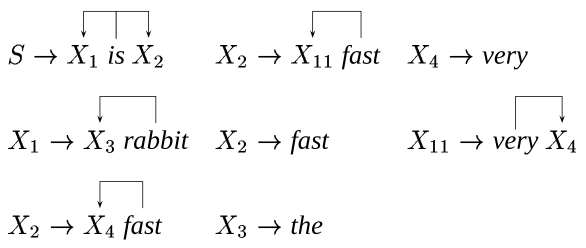
Second, we apply **R1** to identify $X_4 = X_{12}$, $X_3 = X_7 = X_{10}$.

$$\begin{array}{llll} S \ \rightarrow X_1 \textit{ is } X_2 & X_3 \rightarrow \textit{the} & X_5 \rightarrow X_3 \textit{ rabbit} & X_8 \rightarrow X_3 \textit{ rabbit} \\ X_1 \rightarrow X_3 \textit{ rabbit} & X_4 \rightarrow \textit{very} & X_6 \rightarrow \textit{fast} & X_9 \rightarrow X_{11} \textit{ fast} \\ X_2 \rightarrow X_4 \textit{ fast} & S \ \rightarrow X_5 \textit{ is } X_6 & S \ \rightarrow X_8 \textit{ is } X_9 & X_{11} \rightarrow \textit{very } X_4 \end{array}$$

We apply **R1** to identify $X_1 = X_8$ and $X_5 = X_1$.

$$\begin{array}{lll} S \rightarrow X_1 \textit{ is } X_2 & X_4 \rightarrow \textit{very} & S \rightarrow X_1 \textit{ is } X_9 \\ X_1 \rightarrow X_3 \textit{ rabbit} & S \rightarrow X_1 \textit{ is } X_6 & X_9 \rightarrow X_{11} \textit{ fast} \\ X_2 \rightarrow X_4 \textit{ fast} & X_6 \rightarrow \textit{fast} & X_{11} \rightarrow \textit{very } X_4 \\ X_3 \rightarrow \textit{the} & & \end{array}$$

Now, we merge $X_2 = X_6 = X_9$ by applying **R3** on S rules.

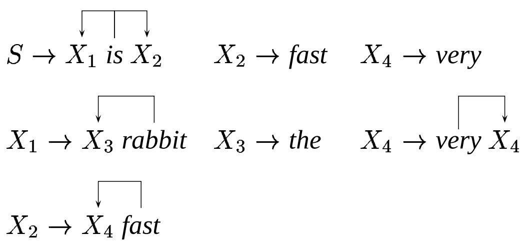


$$S \rightarrow X_1 \textit{ is } X_2 \qquad X_2 \rightarrow X_{11} \textit{ fast} \qquad X_4 \rightarrow \textit{very}$$

$$X_1 \rightarrow X_3 \textit{ rabbit} \qquad X_2 \rightarrow \textit{fast} \qquad X_{11} \rightarrow \textit{very } X_4$$

$$X_2 \rightarrow X_4 \textit{ fast} \qquad X_3 \rightarrow \textit{the}$$

Lastly, we apply **R2** on X_2 rules by merging $X_4 = X_{11}$. We obtain the final grammar:



$$S \rightarrow X_1 \textit{ is } X_2 \qquad X_2 \rightarrow \textit{fast} \qquad X_4 \rightarrow \textit{very}$$

$$X_1 \rightarrow X_3 \textit{ rabbit} \qquad X_3 \rightarrow \textit{the} \qquad X_4 \rightarrow \textit{very } X_4$$

$$X_2 \rightarrow X_4 \textit{ fast}$$

5. Related works

- Sakakibara (Sakakibara, 1992) gives a learning algorithm to infer reversible context free languages from skeleton parse trees. The definition of reversible grammar is very similar to ours. However, we distinguish between both productions $X \rightarrow Y\ a\ b\ Z$ and $X \rightarrow Y\ a\ b\ Z$, unlike (Sakakibara, 1992) in which they are considered identical.
- Kanazawa (Kanazawa, 1998) studies inference of several classes of categorial grammars from functor structures, based on counting the number of categories associated to a terminal symbol. It is not difficult to faithfully translate rigid grammar in reversible dependency grammars.

The defect of learning from structures is that examples usually depend on the implicit grammar that we have to guess. It appears that it is not the case in our approach because we deal with tree languages, and so is seemingly more natural.

References

Angluin, Dana. 1982. Inference of reversible languages. *Journal of the ACM*, 29:741–765.

Dikovsky, A. and L. Modina. 2000. Dependencies on the other side of the curtain. *Traitement automatique des langues*, 41(1):67–96.

Gaifman, H. 1965. Dependency systems and phrase structure systems. *Information and Control*, 8(3):304–337.

Gold, M.E. 1967. Language identification in the limit. *Information and Control*, 10:447–474.

Hays, D.G. 1961. Grouping and dependency theories. In *National symp. on machine translation*.

Jain, J., D. Osherson, J. Royer and A. Sharma. 1999. *Systems that learn*. MIT press.

Kahane, S. 2000. *Les grammaires de dépendance*, volume 41. Hermes.

Kanazawa, M. 1998. *Learnable classes of Categorial Grammars*. CSLI.

Mel'čuk, I. 1988. *Dependency Syntax: Theory and Practice*. The SUNY Press.

Sakakibara, Y. 1992. Efficient learning of context free grammars from positive structural examples. *Information and Computation*, 97:23–60.

Tesnière, L. 1959. *Eléments de syntaxe structurale*. Klincksieck.

Towards a Dynamic Version of TAG

Vincenzo Lombardo and Patrick Sturt
Università di Torino and University of Glasgow

1. Introduction

This paper proposes a syntactic framework that is appropriate for modeling human language processing. The work moves from the largely held assumption that human language processing is incremental, and aims at exploring the consequences of incrementality when the processing strategy is encoded in the operations of a formal system. This makes a syntactic framework consistent with the Transparency Hypothesis (Berwick and Weinberg, 1984), in that a syntactic theory must reflect the behaviour of the human syntactic processor in its formal operations.

Incrementality is a strategy that constrains the processor to analyse the input words from left to right, and to carry out a semantic interpretation of the partial structures (Marslen-Wilson, 1973). A parsimonious version of incrementality is what we call *strong incrementality*, which restricts incrementality to the case in which the parser maintains a fully connected tree at each state (cf. (Stabler, 1994)). There have been some proposals in the literature that claim that structure building occurs incrementally both in derivation and in parsing. Many aspects of the CCG formalism are grounded on incrementality (Steedman, 2000), and Phillips has provided a large amount of evidence that incrementality in the derivation process solves many problems in the definition of constituency (Phillips, 1998). The incremental nature of the formalism is a major issue in Milward's proposal of a *dynamic dependency grammar* (Milward, 1994). In this case, the syntactic formalism is expressed in the terms of a dynamic system, that is a system evolving in time through a number of steps. A *dynamic grammar* views the syntactic process as a sequence of transitions between adjacent states S_{i-1} and S_i while moving from left to right in the input string [1]. Thus, it naturally implements a strongly incremental strategy. The state S_i is a partial tree that spans the input string from the beginning to the i-th word, and at each step the parser tries to attach the word w_i into the current partial tree S_{i-1}, also called the *left context*. So, the definition of a formal dynamic grammar involves the definition of the shape of the partial trees and the formal operations that extend these partial trees.

Tree Adjoining Grammar (Joshi, Levy and Takahashi, 1975) is a well studied framework, that can realize a range of syntactic theories, provided a few constraints on trees and formal operations are satisfied. The goal of this paper is to devise a dynamic version of TAG, that retains the formal characteristics of the framework in terms of basic machinery, while constraining derivation and parsing to be strongly incremental. Of great interest for implementing incrementality are the wide domain of locality introduced by TAG rules, the adjunction operation, and the linguistic motivation for TAG elementary trees. Some of these considerations are shared by (Frank and Badecker, 2001), as points of strength for TAG in language production. Let us consider them in turn.

1) Wide domain of locality
In an empirical study on the Penn Treebank, aimed to discover the amount of non-lexical information which is necessary to implement a fully connected incremental processing strategy, we simulated the sequence of processing steps of an incremental parser on the treebank (Lombardo and Sturt, 2002). Given full connectedness, each new word must be attached to the preceding left context via some syntactic structure, called "connection path". In the simulation, connection paths often needed to be multiply levelled trees. See, e.g., the structure that connects the word "the" to the left context in:

[$_{\text{S}}$ [$_{\text{NP}}$ John] [$_{\text{VP}}$ [$_{\text{V}}$ thinks] [$_{\text{S}}$ [$_{\text{NP}}$ [$_{\text{D}}$ the]]]]]

However, the form of these connection paths is quite predictable from linguistic observations. The use of multiply levelled elementary trees in TAG is an immediate encoding of this requirement. We also found that the introduction and co-indexation of traces needed to be carefully designed for the incremental setting. In TAG, given the wide domain of locality, filler and trace are both present in the same elementary tree, and then they may be separated through a number of adjunctions. This solution is particularly interesting for incrementality, provided that adjunctions are constrained to occur through the insertion of lexical material only to the right of the word whose elementary tree introduced the filler-trace pair.

1. Here we have used the generic term "syntactic process" to indicate both derivation and parsing. In fact, in a dynamic grammar both processes share the same mechanisms, and we will use the two terms interchangeably.

2) Adjunction
The adjunction operation provided by the TAG formalism is vital for the incremental processing of modifiers. In fact, in many cases the correct attachment of a word to the current partial tree requires the guess of an unknown number of left recursive structures. The adjunction operation permits the progressive extension of recursive structures, as they are required to process the input string. A typical use of left recursive structures in English is in possessive constructions. In a sentence like “Mary hated the salesman’s assistant’s hairstyle”, during incremental processing it is impossible to guess in advance the depth of the NP immediately dominating “the salesman”. One solution is to attach this NP immediately as the object of “hated”, and then to embed this NP via adjunction when the possessive is processed. This operation can be repeated an arbitrary number of times to produce an arbitrarily embedded structure (cf. (Thompson, Dixon and Lamping, 1991)).

3) Linguistically motivated elementary trees
TAG, and in particular Lexicalized TAG (Schabes, Abeillé and Joshi, 1988), constrain the form of the elementary trees to be linguistically motivated according to the number and the type of arguments required by some lexical anchor. This is a desirable property also for incremental processing, since in head-initial languages, items that are to occur on the right are often predicted from anchors on the left. Here we are not referring to any processing model, but to a basic requirement for a syntactic formalism that supports incrementality. We must also notice that linguistic constraints are not always enough to guarantee the full connectedness of the partial tree. It can be that the argument structures of two adjacent lexical anchors cannot be directly combined, and so we need some extra structure to guarantee full connectedness. Take again the previous example “John thinks the ...”: the lexical entry for “thinks” is an initial tree that predicts an S node on its right, marked for substitution; the lexical entry for “the” is an auxiliary tree rooted in NP; the extra structure provides the edge between S and NP, and must be included in one of the two elementary trees, even though this edge is not directly licensed by any of the lexical anchors in the partial tree. As a consequence, an elementary tree of a dynamic TAG may be larger than the argument domain of that elementary tree’s anchor. This is because extra structure is needed to guarantee connectivity (in the next section we will see a few examples of this case).

This paper describes the major issues of a *dynamic version of TAG* (called *DV-TAG*), which is suitable for incremental processing. We first describe the consequences that incrementality bears upon the shape of elementary trees and on the form of the attachment operations, together with some definitions. Then we describe the derivation process that produces the so-called derived tree. Finally, in order to illustrate the functioning of the formalism, we provide a few meaningful linguistic examples.

2. Dynamic Version of TAG (DV-TAG)

In this section we describe the basic machinery of DV-TAG. We start with elementary trees, and then we move to the attachment operations, Substitution and Adjunction.

2.1. Elementary trees

A DV-TAG grammar consists of elementary trees, divided into initial trees and auxiliary trees. Most of the definitions provided by TAG are also applicable to elementary trees in DV-TAG. The structure of elementary trees in DV-TAG is constrained by a number of syntactic modules (X-bar theory, case theory, thematic relations theory, some empty category principle(s), see (Kroch and Joshi, 1985)). In particular, we assume that each node has a distinguished head daughter, that each elementary tree is associated with a lexical anchor, and that the elementary tree includes the argument structure of the anchor. Long-distance dependencies have to be stated locally, in a single elementary tree. Auxiliary trees are minimal recursive structures with root and foot nodes identically labelled.

The major difference in comparison with TAG is that the shapes of the elementary trees must be constrained in order to permit the left-to-right derivation (and parsing) of all and only the legal structures. The linguistic motivations mentioned above may not be enough to constrain the elementary trees in a way that guarantees the construction of a fully connected structure [2]. This motivates the addition of extra information to elementary trees. Here we are not claiming that additional information is in some form, and is added on-the-fly while processing, but that elementary trees in DV-TAG are larger than the corresponding elementary trees in TAG.

Even if we ground this requirement on the empirical observations addressed in the Penn Treebank (Lombardo and Sturt, 2002), we can think of some principled way to constrain the extension of elementary trees beyond the modules above. A similar solution, devised again in the context of incremental processing, is the *type raising*

2. Remember that a fully connected structure is required by incremental processing.

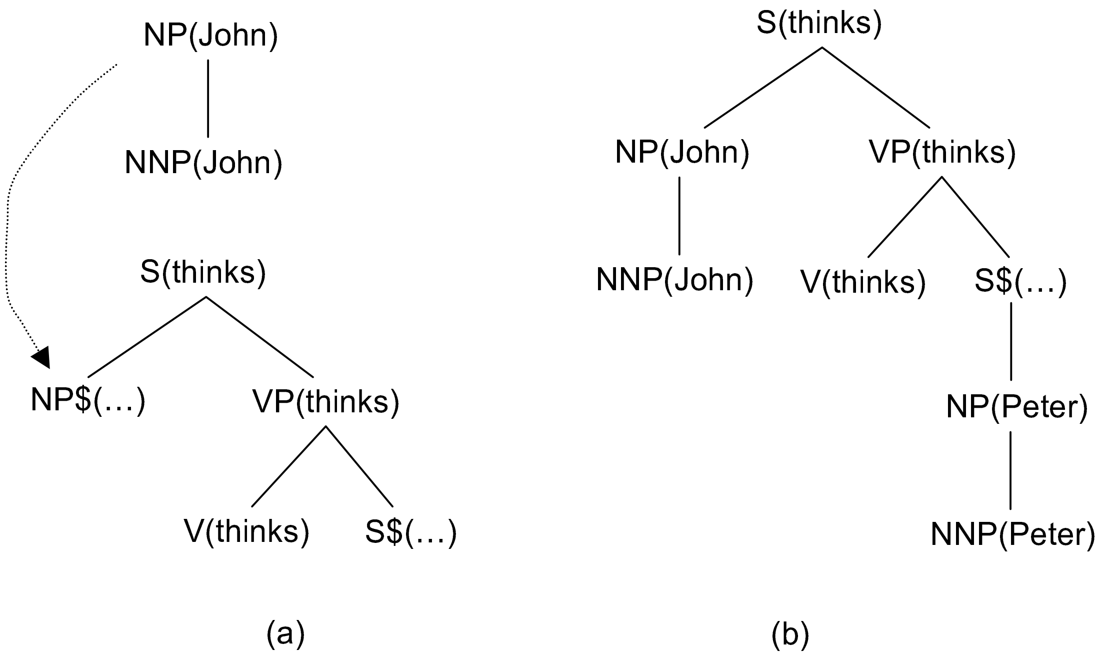


Figure 1: Initial trees and derivation for “John thinks Peter ...”. Node labels include the head word. The symbol “$” indicates “marked for Substitution”. (a) The initial trees for “John” and “thinks”. (b) The derived tree for “John thinks Peter”.

operation provided in the CCG theory (Steedman, 2000). We have not yet addressed such a problem at this stage of development. What may happen is that some portions of this extra information can overlap with the arguments of some lexical anchor which is yet to come (on the right of the current word). So, a DV-TAG grammar can include some redundant information, since the same portions of structure can belong to several elementary trees for different reasons.

In order to illustrate the case for redundant information, consider the sentence “John thinks Peter laughs”. In Figure 1a there are the initial trees for “John” and “thinks”, respectively. “Thinks” subcategorizes for a subject NP and a sentential clause S. The initial tree for “Peter” should be the same as the one for “John”. However, given the substitution indicated by the dotted arrow, we have that the NP “Peter” has to be incorporated into the subject position of the complement clause licensed by “thinks” (Figure 1b). This means that the connection between S and NP must be realized by the elementary tree for “Peter”. Certainly, the edge between S and NP is not projected by the lexical anchor “Peter”, but is part of the initial tree for “Peter”. It is linguistically licensed by “laughs”, and so it must be part of the initial tree for “laughs” as well. As we see below, the attachment operations need to be more sophisticated in DV-TAG, because they also include some checking of structures previously built for connectivity reasons. We call this process *redundancy checking*. The edge existing between the complement S and the subject NP (“Peter”) is redundancy checked when the initial tree for “laughs” is Substituted.

The structure in Figure 2 is more complex. It is an initial tree for a NP that is fronted as a consequence of an object extraction (in the example, the word “beans” in “Beans_1 John likes e_1”). This structure is common to all cases where the filler precedes the trace. It provides both the filler and the trace positions in the same elementary tree, and the two positions are co-indexed (index i). The assumption underlying this structure is that the fronting phenomenon must predict the embedded verb that licenses the fronted NP as an object. There are two nodes (S and NP) marked for Substitution: this means that, in order to have an object extraction we need an actual transitive verbal subcategorization frame and an actual subject. The S node shares the lexical head with the VP and the V nodes (see the index j).

Another underlying assumption that is not obvious in standard TAG is that maximal projections that are marked for substitution can also be internal nodes, and not leaves. This is the case for the internal S node in Figure 2. The nodes marked for substitutions are required to be roots of initial trees. The substitution that will take place here is the initial tree of a transitive verb, e.g. “like”, whose initial tree [S [NP VP [V NP]]] is already part of the structure

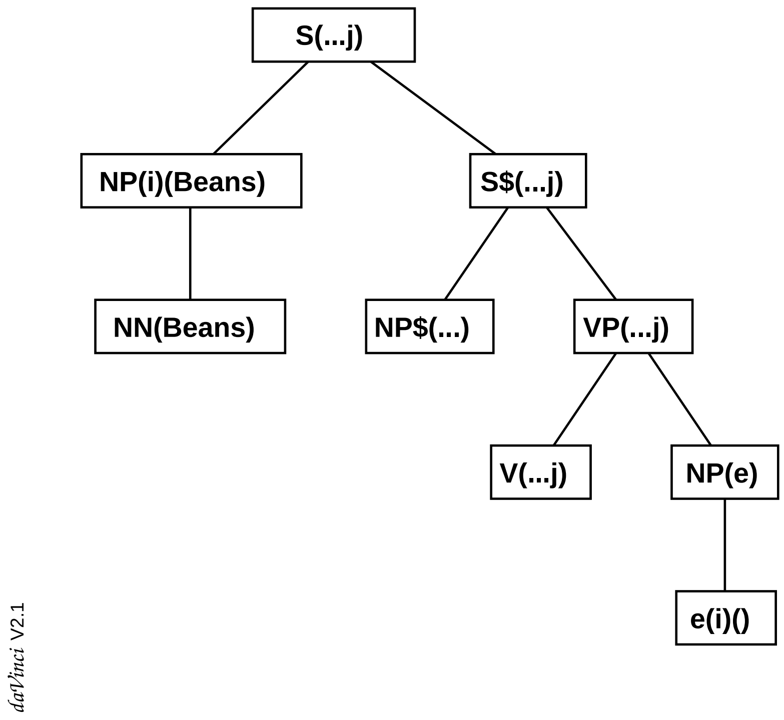


Figure 2: An elementary tree for an extracted object. Notice that it includes the constraints on the structure of the verbal subcategorization frame. Filler and trace nodes are co-indexed with i. The nodes including the j-index in parentheses are constrained to bear the same head daughter.

and will be rooted in “S$(...j)”. This substitution will be a redundancy checking only, since the whole substructure is already in the left context.

Finally, notice that adjunctions to the internal S node permit the insertion of other clauses, that take the filler and the trace further apart (“Beans_1 I know John likes e_1”). Notice that in this case, the elementary tree whose head is “likes” will be discontinuous in the derivation tree; clearly, the actual implementation of the redundancy checking mechanism would require some care to deal with such cases.

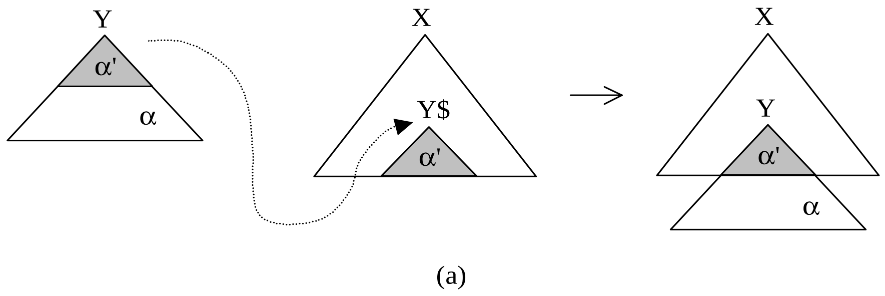
It can also be the case that some elementary tree introduces some other lexical element beyond the lexical head, as may occur, for example, when a verb selects the lexical form of the head of an argument PP (e.g., “go to”). In derivation and parsing, we assume that the pointer on the input string goes to the leftmost lexical head of the elementary tree, and it is possible to adjoin words in between (“go slowly to”). We stipulate that only one of the lexical elements is the lexical anchor of the elementary tree. Redundancy checking must also take into account lexical items.

2.2. Attachment operations

The attachment operations in DV-TAG are substitution and adjunction, with a number of differences on their applicability and functioning in comparison with standard TAG. In particular, their functioning has to implement some form of redundancy checking as mentioned above[3]. Their applicability depends on the legal attachment sites, with respect to left-to-right processing.

Since we proceed incrementally, every time we apply an operation there will be a *left context*, given by the tree spanning all the words from the beginning of the sentence to the word w_{i-1}, and a current elementary tree for the word w_i.

3. In actual implementations redundancy checking should be accompanied with feature upgrading.



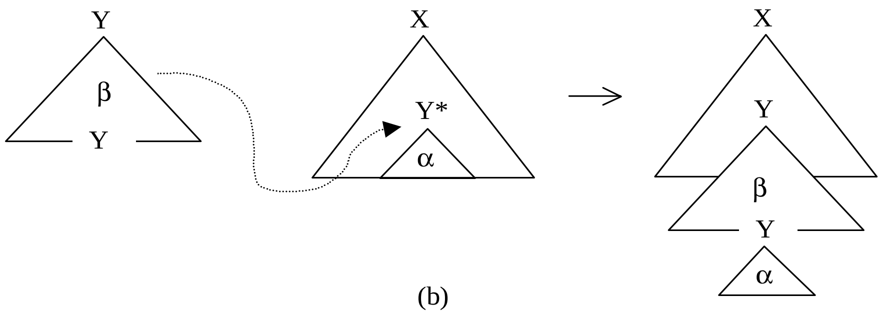


Figure 3: Substitution (a) and Adjunction (b).

- Substitution (see Figure 3(a)):
  In the Substitution operation, either the root node of the current initial tree is merged into the leftmost non-terminal node marked for substitution in the left context, or the root node of the left context is merged into the leftmost non-terminal node marked for substitution in the current initial tree, producing a new tree.

The root node and the substitution node must have the same name. Nodes marked for Substitution are indicated with a $ in the elementary trees. The leftmost relation on nodes marked for Substitution is calculated on the lowest nodes of their respective head projections[4].

- Adjunction (see Figure 3(b)):
  In the Adjunction operation, either the left context, which has the form of an auxiliary tree, is grafted into a non-terminal node marked for adjunction in the current elementary tree, or the current elementary tree, which is an auxiliary tree, is grafted into a non-terminal node marked for adjunction in the left context.

The root and foot nodes of the auxiliary tree, whether it is the left context or the current elementary tree, must match the node at which the auxiliary tree adjoins. It is important to notice that in DV-TAG both the left context and the current elementary tree can act as the auxiliary tree in adjunction. This is to say that either the left context is split because of the adjoining of some auxiliary tree, or some elementary tree is split because of the adjoining of the left context at some node. In the left context, nodes are marked for adjunction according to the definition of the accessible *fringe* (see below for the definition); in the elementary trees, nodes are overtly marked for adjunction.

The foot node of the auxiliary tree is matched with the node which is the site of adjunction. If this node has daughters marked for substitution, then it is necessary to move these daughters to the root node, leaving the co-indexed traces at the foot node. Traces allow us to keep the locality of the subcategorization constraints, while moved elements account for the structural constraints. An example of when this is necessary is in the processing of expressions where a modifier intervenes between a head and its argument, as in “George worried severely about the storm.[5] The Figure 4 depicts this case.

4. The precedence relation is trivially defined when the substitution nodes are not in a dominance relation (remember that we assume a full connectedness of the left context); in case one substitution node dominates another, the leftmost substitution node is defined in relation to nodes on the respective head projections of all the substitution nodes, which by definition must be different.
5. Note that adverbs very frequently intervene between a verb and its object in Romance languages like Italian.

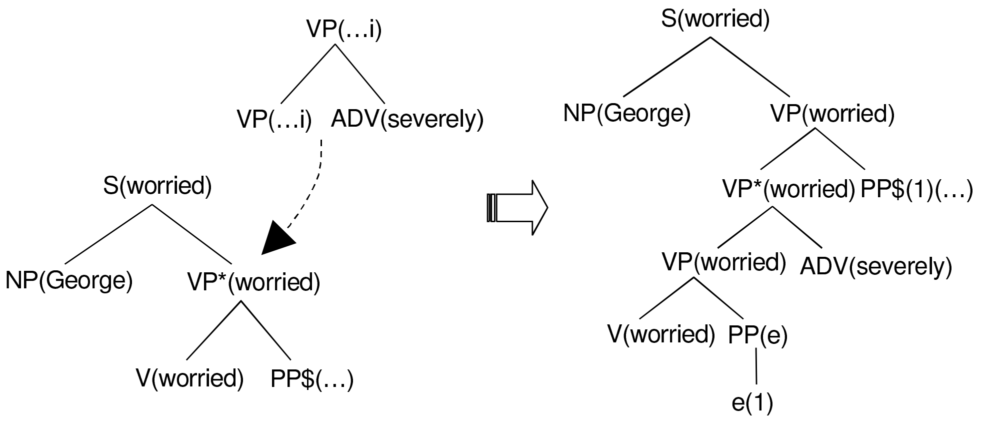


Figure 4: The modifier “severely” occurs between the head “worried” and its argument PP “about the storm”.

The attachment operations involve the redundancy checking procedure. Redundancy checking involves those substructures that are already part of the left context. Starting from the root node that matched, the redundancy checking procedure overlaps the nodes of the elementary tree onto the left context, and verifies that nodes that match are consistent. Nodes that are the results of adjoining operations are neglected during this process. Our solution is to maintain a data structure for each word inserted through an initial tree, that keeps track of the correspondences between the nodes of the original structure, and the actual instantiations of nodes in the current left context.

3. Derivation in DV-TAG

The derivation, as well as the parsing, of syntactic structures, proceeds from left to right, by combining the elementary trees anchored to the words in the string, using the operations of Substitution and Adjoining.

The left-to-right constraint implies that the derivation process proceeds through intermediate states $\mathrm{S_i}$, that exactly advance on the words of the terminal string. At the processing step i, we have a *current word* w_i and a *current state* S_{i-1}, the left context. The derivation process combines an elementary tree for w_i with S_{i-1} to produce S_i, which becomes the current state at the next step.

Each elementary tree typically introduces a few nodes headed by the lexical anchor, and a few nodes that are marked for substitution, and that need to be actually substituted. In derivation, these nodes are called *predicted nodes*. Predicted nodes are introduced both for connectivity requirements and for linguistic motivations. They are typically unheaded; in fact, usually, during the course of a derivation, they are pending non terminals that expect to be substituted by some initial tree, possibly after a number of adjoinings. It can also be the case that a predicted node is headed: such a situation occurs when the node marked for substitution already has a lexical anchor (that is, the word is part of the prediction, like in the case of subcategorized prepositions, see above). This can cause a situation where the derivation of some word at position i occurs before the derivation of some word at position j (<i), which seems in contrast with the incremental assumption. For example, when a verb subcategorizes for a preposition, the elementary tree for the verb can include the preposition too. Verb and preposition can possibly be pushed apart by the adjunction of some adverbial to the VP node. This adjunction occurs in the linear order between the verb and the preposition. Thus, the derivation of the preposition precedes the derivation of the adverbial; the reverse of the linear string order. In case the predicted nodes are headed, the terminal symbol (i.e. the lexical anchor) is checked for redundancy, when it is the current word. A node such that there are no predicted nodes in its subtree is said to be *complete*. In the final tree all nodes must be complete.

The left-to-right order constraint, combined with the standard conditions on trees implies that only a part of the left context is accessible for continuation; that is, not all the nodes can be sites of substitutions or adjunctions. Specifically, elementary trees can be attached only in the strict sequence given by the terminal string of the lexical anchors. The accessible nodes form the so-called *fringe*. In context-free incremental processing, such as left-to-right top-down parsing, the fringe corresponds to the right frontier. However, because of the presence of predicted nodes and the possibility of the adjoining operation, the fringe needs to be a bit more articulated, with nodes that can be the site of adjunction and nodes that can be site of substitution.

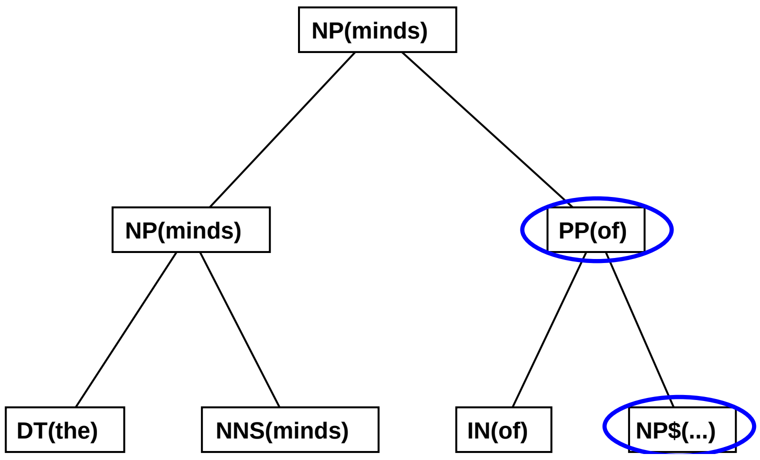


Figure 5: A fringe exemplifying the second item in the definition.

Here is the definition of the fringe in the left context S_{i-1}:

- at the beginning, the derivation process postulates a left context given by the start symbol;
- at the step i, we take the path leading from the preterminal node for the word w_{i-1} to the root:
  - if there are no nodes marked for substitution, all the nodes on this path are marked for adjunction;
  - if there are nodes marked for substitution, take the lowest common ancestor of w_{i-1} and the leftmost node marked for substitution; all the nodes on the two paths to the common ancestor are marked for adjunction.

These constraints on the fringe preserve the linear order of tree nodes with respect to word order. In the figures 5 and 6 there are two examples of fringes. Figure 5 shows the fringe at a point where a node is marked for substitution. Figure 6 shows the fringe after the substitution node has been filled. It can be seen that the number of nodes available for adjunction increases dramatically after the substitution operation.

4. Derivation of cross-serial dependencies

In the following paragraphs we will give a sketch of how the formalism can be used for creating an incremental derivation of a Dutch verb-raising sentence, which is an example of a cross-serial dependency construction. This construction is well-known as a test-bed for TAG-based systems, because it requires greater than context-free expressive power.

The example sentence is below:

(1) Jan Piet Marie zag helpen zwemmen.
Jan Piet Marie saw help swim.
(i.e. *Jan saw Piet help Marie to swim.*)

Assume that the elementary tree for a noun phrase can also include elements of the sentential structure in which the noun phrase appears (this could be seen as an analogue of *type raising* in Combinatorial Categorial Grammar (Steedman, 2000)) so that the elementary tree for “Jan” will be as in Figure 7.

The tree includes information that the noun phrase appears as the subject of a sentence whose verb subcategorizes for a clause, and the verb is extraposed (Kroch and Santorini, 1991).[6] The next word, “Piet” has an elementary tree with an identical format to that of “Jan”. The left context (that is the elementary tree of “Jan”) is adjoined into that of “Piet”[7], creating a new partial derived tree, which is in turn adjoined to the elementary tree for *Marie*, yielding the new partial derived tree shown in Figure 8.

6. Whether so much predicted information would be included in a psychologically plausible elementary tree is debatable. For the purposes of parsing, some forms of underspecification could be used, in which case the relation between parsing and grammatical derivations would be weakened.
7. This is non standard in TAG, the left context behaves like an auxiliary tree, and adjoins into the elementary tree of “Piet”.

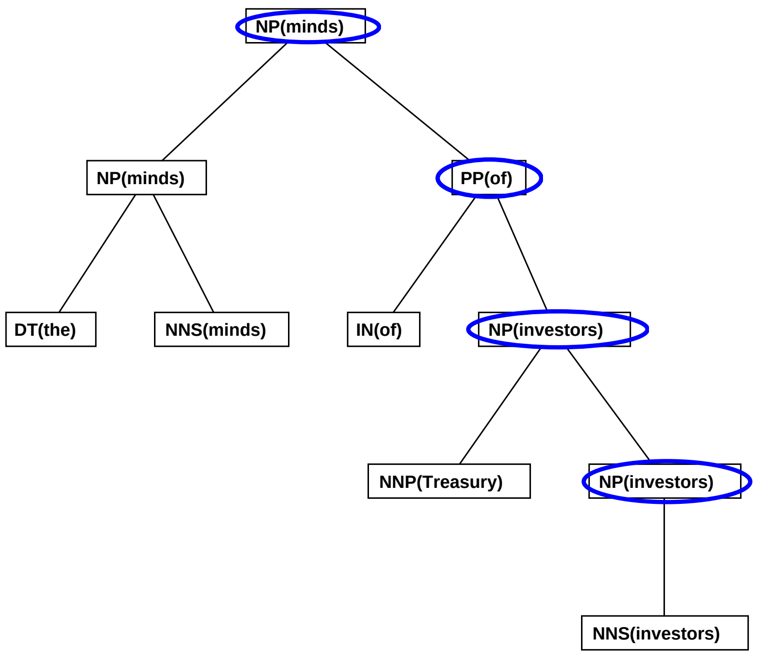


Figure 6: A fringe exemplifying the first item in the definition.

Notice that at this point, the lowest substitution node is close to the root of the tree, so, according to the definition of the fringe, adjunctions could occur anywhere in the dotted line shown on the figure.

The final three verbs of the sentence are in the respective extraposed head positions at the end of the sentence.

5. Conclusions and future extensions

In this paper, we have illustrated informally an approach that makes TAG a dynamic grammar, that is a formalism suitable for the incremental processing of natural language. The dynamic version of the TAG (DV-TAG) involves elementary trees and attachment operations that obey the left-to-right full-connectedness constraint due to incrementality.

The linguistic examples we have shown reveal the major issues of the approach. Beyond the formalization of the approach, two extensions are planned in the near future. On the theoretical side, we are going to model the incremental attachment of extraposed modifiers (such as the phrase “with blond hair” in “A man came in with blond hair”). This type of phenomenon is challenging, since it requires adjunction in two places in the left context simultaneously; one adjunction will represent the surface position of the extraposed element, and the other will represent its trace. If the trace and the extraposed element must be in the same elementary tree, then it may be necessary to add a regular expression to the elementary tree, which can match an arbitrary number of intervening nodes between the surface attachment point and the trace. An alternative solution could be the use of Synchronous TAGs. On the practical side, we aim to extract a (large scale) grammar from a treebank. The extractor will be based on the algorithms presented in (Xia, Palmer and Joshi, 2000), especially in the initial stage, where treebank trees are transformed into derived trees. The resulting grammar can then be used to build models of human language processing on a realistically large scale.

References

Berwick, R. and A. Weinberg. 1984. *The grammatical basis of linguistic performance: language use and acquisition.* MIT Press.

Frank, R. and W. Badecker. 2001. Modeling syntactic encoding with Tree Adjoining Grammar. In *Talk presentd at the CUNY conference on sentence processing.*

Joshi, A., L. Levy and M. Takahashi. 1975. Tree adjunct grammars. *Journal of the Computer and System Sciences*, 10(1):136–163.

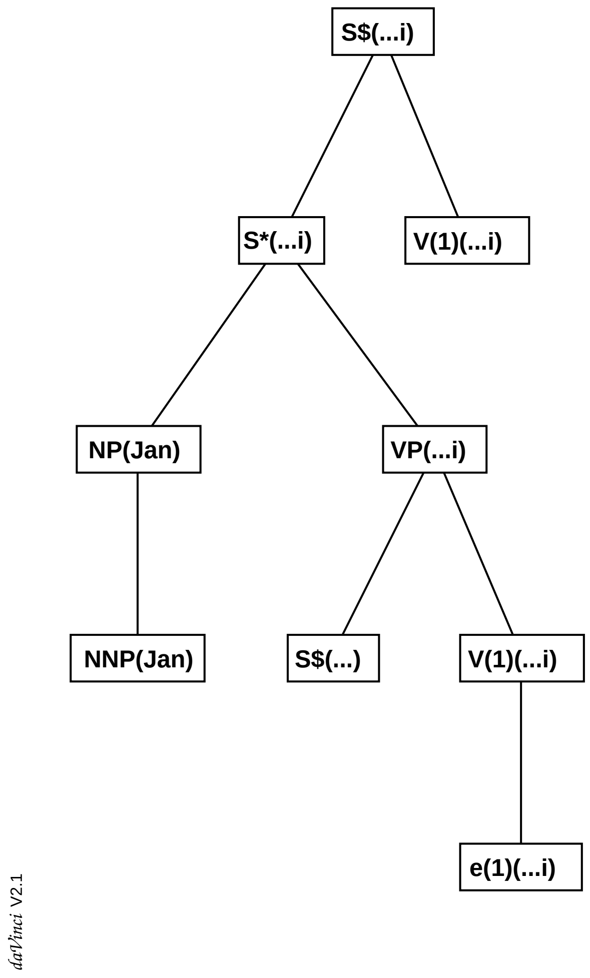


Figure 7: Elementary tree for *Jan*. Notice that the internal S node is marked for Adjunction. In the derivation of the example sentence, it is this node to be adjoined in "Piet" elementary tree.

Kroch, A. and A. Joshi. 1985. The linguistic relevance of Tree Adjoining Grammar. *CIS Technical Report MS-CIS-85-16, University of Pennsylvania.*

Kroch, A. and B. Santorini. 1991. The derived constituent structure of the West Germanic verb-raising construction. In R. Freidin, editor, *Principles and Parameters in comparative grammar*. MIT Press, Cambridge: MA, pages 269–338.

Lombardo, V. and P. Sturt. 2002. Incrementality and lexicalism: A treebank study. In S. Stevenson and P. Merlo, editors, *Lexical Representations in Sentence Processing.* John Benjamins.

Marslen-Wilson, W. 1973. Linguistic structure and speech shadowing at very short latencies. *Nature*, 244:522–533.

Milward, D. 1994. Dynamic Dependency Grammar. *Linguistics and Philosophy*, 17(6).

Phillips, C. 1998. Linear Order and Constituency. *Linguistic Inquiry*, in press.

Schabes, Y., A. Abeillé and A. K. Joshi. 1988. Parsing strategies with "lexicalized" grammars: Applications to tree adjoining grammars. In *12th International Conference in Computational Linguistics*, pages 578–583, Budapest, August.

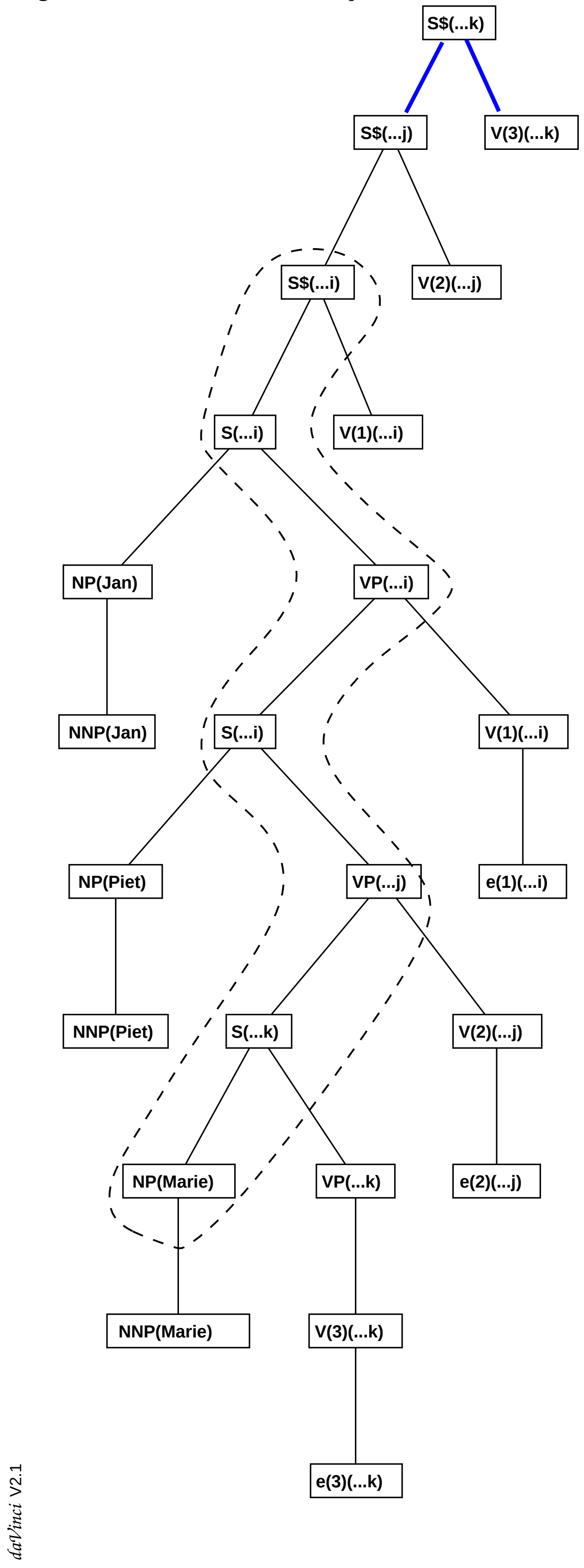
Stabler, E. P. 1994. The finite connectivity of linguistic structure. In C. Clifton, L. Frazier and K. Reyner, editors, *Perspectives on Sentence Processing.* Lawrence Erlbaum Associates, pages 303–336.

Steedman, M. J. 2000. *The syntactic process.* A Bradford Book, The MIT Press.

Thompson, H. S., M. Dixon and J. Lamping. 1991. Compose-reduce parsing. In *Proc. of 29th ACL*, pages 87–97.

Xia, F., M. Palmer and A. Joshi. 2000. A Uniform Method of Grammar Extraction and Its Applications. In *Proc. of the EMNLP 2000.*

Figure 8: Derived tree for the sequence *Jan Piet Marie*



Resumptive Pronouns, Wh-island Violations, and Sentence Production

Cassandre Creswell
University of Pennsylvania

1. Introduction

In spontaneous speech, English speakers produce relative clauses whose structures violate wh-island constraints but where resumptive pronouns appear in place of the gap/trace normally found in a relative clause. It is difficult to explain these island-violation remedying resumptive pronouns in a model of sentence production where only grammatically-licensed trees are available to the speaker's production system. This paper argues that the only explanation for these island-violating relative clauses in such a model of sentence production is that they are fully grammatical constructions, whose marginal status reflects non-syntactic factors.

The paper is structured as follows. The next section outlines the problem that resumptives in island contexts present for a model of sentence production using TAG. Section 3 argues that the problem can be resolved given the existence of resumptive pronouns in English in non-island contexts. Section 4 discusses two potential diffiulties for the analysis. Section 5 outlines some possible implications of the analysis for the typology of relative clauses in general. Finally, Section 6 concludes with suggestions for further investigation.

2. The problem of "remedying" island violations

2.1. Accounting for island violations in TAG

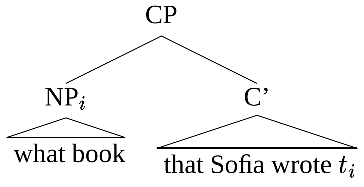
Frank (2002) presents a conception of grammar applying the machinery of tree adjoining grammar within an independent theory of well-formedness of grammatical structures. The domains over which any structural dependencies are expressed are the elementary trees in the system. The TAG operations of substitution and adjunction can then apply to any of the set of well-formed trees. No movement transformations outside the domain of the elementary tree are possible in this system.

In order to account for the well-known phenomena of island violations, rather than stating principles constraining wh-movement (or the location of traces of this movement), Frank's theory instead rules out the elementary and auxiliary trees needed to derive structures with island-violations primarily based on two independent principles: 1) the Theta Criterion: only nodes that are selected for can appear in an elementary tree; and 2) in order to obey the requirement that derivation trees should be context-free, a well-formed derived auxiliary tree rooted in X cannot be derived via substitution of a tree with an X foot node into a tree with an X root node.

For example, extraction from an NP complement, as in (1), is ruled out because in addition to the legitimate elementary tree rooted in *wrote* in (2), it would require the impossible auxiliary tree in (3).

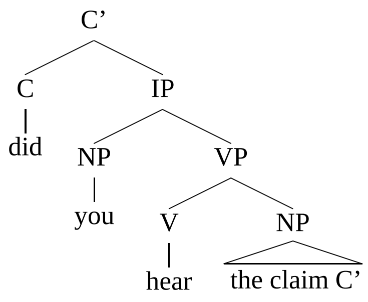
(1) *What book$_j$ did you hear [the claim that Sofia wrote t$_j$]?

(2)



* This paper has benefited from the useful comments of Tonia Bleam, David Chiang, Na-Rae Han, Elsi Kaiser, Kieran Snyder, and two anonymous reviewers, among others.

(3)



The latter is ill-formed because neither of the trees that make it up—an elementary tree rooted in C’ and anchored by *hear* and an elementary DP tree anchored by *claim* taking a C’ complement—are themselves a C’ auxiliary tree. Extraction from wh-islands, relative clauses, adverbial adjuncts, and sentential or DP subjects are ruled out in similar ways.

2.2. Remedying island violations with resumptive pronouns

In unplanned speech speakers of English sometimes produce utterances with a resumptive pronoun in place of the gap or trace which would be an island-violation if a wh-word had been extracted from that position, as shown in the naturally-occuring examples of extraction from a wh-island, (4a), a relative clause, (4b), an adverbial clause, (4c), and a subject in (4d).

(4) a. There are always guests who I am curious about what **they** are going to say. (Prince (1990)’s 3a)

b. That asshole X, who I loathe and despise the ground **he** walks on, pointed out that...(Prince (1990)’s 5a)

c. Apparently, there are such things as bees in the area which if you’re stung by **them**, you die. (Prince (1990)’s 5b)

d. You have the top 20% that are just doing incredible service, and then you have the group in the middle that a high percentage of **those** are giving you a good day’s work... (http://www.ssa.gov/history/WEIKEL.html)

Kroch (1981) argues that such resumptive pronouns are the result of a processing effect. Although a dependency between a wh-word and a gap in these positions is ungrammatical, as an artifact of how sentence production proceeds, forms like those above are uttered. Kroch explicitly rejects any possibility of a formal solution within the grammar to account for these forms. He uses an incremental model of sentence production in which a wh-element is produced in a fronted position before the entire sentence has been planned out, allowing the utterance of fluent speech. Rather than having to wait to create the entire sentence in which the distance between the wh-element and its gap is potentially unbounded, the speaker can begin speaking while the remainder of the structure is being constructed. This wh-element is adjoined to a full clause where it appears again in its base-generated position. When the repeat occurrence is reached in the course of production, it is not pronounced. If the sentence produced has an island-violation, then a pronoun or NP is inserted in the base-generated position. In this model, resumptive pronouns are expressly inserted as a “last resort” to avoid violating the ECP by leaving an ungoverned trace. The resumptive pronoun remedies the violation because an empty category will no longer be present.

2.3. Sentence production using TAG

In a TAG-based model of incremental sentence production (Ferreira, 2000; Frank and Badecker, 2001), elementary trees are the basic “building blocks” of generation. Selecting a lexical item corresponds to selecting an elementary tree. The grammatical properties of that tree’s structure are a natural domain for explaining the planning commitments a speaker makes while producing a sentence. The grammar provides a set of trees to work with, and as sentence planning proceeds, the speaker attempts to select the correct trees to encode the meaning she wishes to express and to put them together in a particular configuration. The only source of production errors is incorrectly combining trees; ill-formed elementary (and auxiliary) trees never arise. To produce a structure with a wh-dependency, the speaker selects the elementary tree headed by the main verb with the appropriate wh-element fronted. In a structure with a long-distance dependency, additional material can be adjoined in between the wh-element and the remainder of the clause it originated in, but the commitment to the main clause structure has already been made.[1]

In this model of production where we assume that a speaker only has grammatical resources with which to work, we can not use Kroch (1981)'s explanation of the appearance of resumptive pronouns in island-violation contexts. The resources needed to produce island-violating structures are not available in the grammar that licenses the set of tree building blocks. On the face of it then, it seems that the existence of resumptive pronouns in island violating contexts would prove devastating for this model of sentence production. Based on the assumptions that 1) the processing system has only grammatically-licensed trees with which to create larger structures and 2) the structures needed to extract from island-violation contexts are not grammatically-licensed, speakers could not be remedying violations that should not even be created given their underlying grammars. As we will argue in the following section, however, the underlying grammar of English speakers independently requires the resources needed to produce these forms. Hence, we can preserve both the assumptions above and still have a grammar that characterizes all the structures that English speakers use.

3. Resumptive pronouns as grammatical resource

3.1. Resumptives in non-island contexts

Resumptive pronouns appear in relative clauses in English in *non*-island violation contexts, as in (5), from Prince (1990).

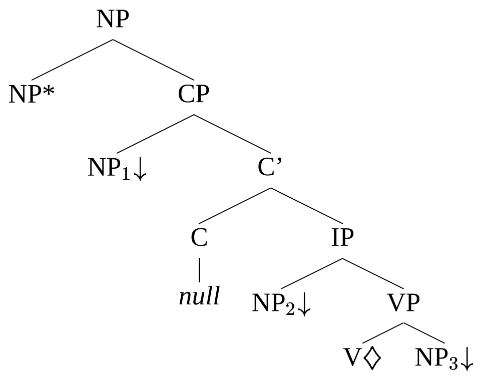
(5) a. My son, God bless him, he married this girl which I like **her**. (Prince's 28a)
b. If there's any message that she can forward **it** to us, then...(Prince's 15b)
c. You get a rack that the bike will sit on **it**. (Prince's 15c)
d. I have a friend who **she** does all the platters. (Prince's 4c)

Prince presents an analysis of the function of this type of resumptive pronoun, claiming that they serve as normal discourse pronouns rather than as a type of bound pronoun or trace. These pronouns appear in contexts where the relative clause is being used simply to predicate additional information about a discourse entity evoked by the head noun, not to assist the hearer's identification of the referent evoked by the head noun. For example, they are far more common with non-restrictive relatives and relatives modifying indefinite NPs. Additional evidence she presents for the "discourse pronoun" analysis are cases where a co-referential demonstrative or full NP or even a non-coreferential (but related) NP appears instead of a resumptive pronoun, as in (6) (Prince's 34(a-d)).

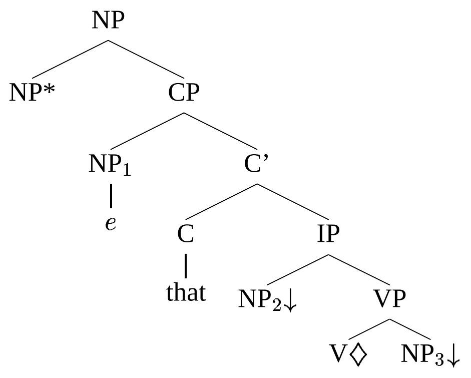
(6) a. I had a handout and notes from her talk that **that** was lost too.
b. He's got this lifelong friend who he takes money from the parish to give to **this lifelong friend.**
c. I have a manager, Joe Scandolo, who **we**'ve been together over twenty years.
d. You assigned me to a paper which I don't know anything about **the subject**.

In order to produce relative clauses like these, speakers must be using structures like those in (7).

(7) a.



1. This does not rule out the possibility that the production process could be in some sense non-deterministic. That is, when there is more than one grammatical way to encode the meaning to be expressed, the speaker may be able to retain and manipulate more than one possible main clause tree during production. The possibility of a non-deterministic production model is relevant to the discussion in Section 5 below.

b.


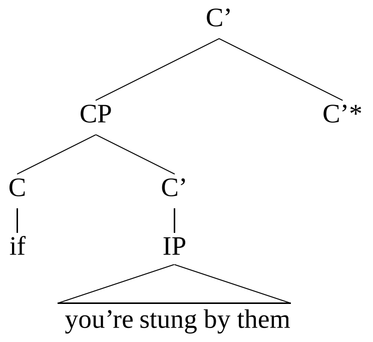
Here NP_1 would have features requiring the substituting NP to be a wh-word. But NP_2 and NP_3 could have any NP substituted within them, including a pronoun coreferential with the NP to which this tree adjoins.

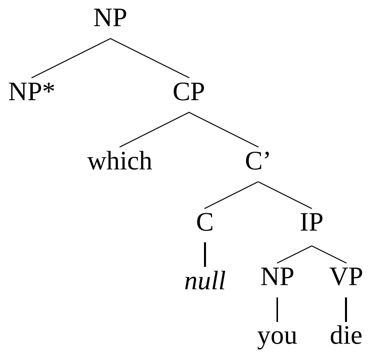
3.2. Generating resumptives in island contexts

Because the production system necessarily allows relative clause auxiliary trees like those in (7), we can now explain where speakers find the grammatical resources to produce relative clauses with island-violating resumptive pronouns. The trees in (7) are projections of a head verb. As such any finite verb can project this (or a related) structure. Any legitimate auxiliary tree can be adjoined into it. There is no syntactic dependency between the relative pronoun and any of the lower NPs. Therefore, there needs to be no tree at any point in the derivation that reflects such a local dependency. For example, in order to derive (8a), we now only need a relative clause auxiliary tree with *die* as its head, into which we substitute *which* and *you*, as in (9b).

(8) a. bees which if you're stung by them, you die

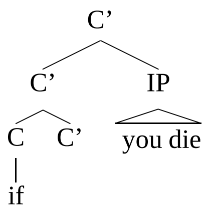
b. * bees which$_j$ if you're stung by t$_j$, you die

(9) a.


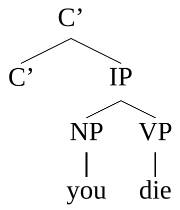
b.


An initial tree headed by *stung* can then be substituted into a tree headed by *if*; the resulting tree, (9a), can be adjoined into the *die* tree, (9b). Neither of the suspect trees which would be required in some derivation of the unacceptable NP (8b), e.g. those in (10), are needed to generate the resumptive pronoun version in (8a).

(10) a.



b.



As a result, the impossibility of generating island-violations without resumptive pronouns is preserved in this analysis.

4. Potential problems

In the previous sections we saw that given the elementary trees needed to generate resumptive pronouns in relative clauses without island-violations, the constraints preventing the generation of island violations need not be altered in order to correctly allow a TAG of English that generates resumptive pronouns in otherwise-island-violating contexts.

This section will discuss two potential difficulties for the syntactic analysis of resumptives in island contexts given above. The first is to explain the apparent overgeneration of such an analysis. The second is to explain why resumptives in island and non-island contexts in English are often characterized as marginally acceptable if in fact the grammar generates them.

4.1. Overgeneration

The relative clause trees given in (7) would overgenerate by allowing non-gap-containing relative clauses, like (11–12):

(11) the police officer who John prefers spinach

(12) the smell that my mom is baking bread

Relatives like (11) could be ruled out with the pragmatic requirement that relative clauses must make a statement about their head NP (Kuno, 1976), in that there is no easily apparent relation between the head and the relative clause. The oddness in English of example (12) is more difficult to explain.

These forms are far more common in topic-oriented languages like Korean and Japanese, as illustrated by the perfectly acceptable relative clause in (13).

(13)	emeni-ka	ppang-ul	kwup-nun	naymsay
	mom-Nom	bread-Acc	bake-Rel	smell

‘smell that my mom is baking bread’ (Na-Rae Han, p.c.)

The pattern here could be explained by a cross-linguistic difference in the pragmatics of structures where a NP is adjoined to a clause, main or relative, without a gap. In English, these structures are possible, as shown in (6d) above, but the relation between the NP and clause is subject to strict pragmatic restrictions. In a language like Korean, however, the discourse conditions such constructions are subject to are less restrictive.[2]

2. This difference in the pragmatic factors conditioning identical or similar syntactic forms is not unknown in the literature. See (Prince, 1999) for a discussion of the different restrictions on the conditions allowing the formally identical constructions of Focus Movement in Standard English and Yiddish-Movement in Yiddish English.

4.2. Marginal acceptability

One issue that remains to be explained in this analysis is the following: if these sentences are created using legitimate parts of the grammar, why then are they regarded as being only marginally acceptable? I will suggest three reasons here; two of which apply to both island- and non-island resumptive pronoun uses.

First, resumptive pronouns in both island and non-island contexts are primarily confined to informal, spoken registers in English. In addition, for reasons to be discussed below, they are infrequent forms. Both of these factors are reasons linguists might be inclined to classify them as only "marginally acceptable." [3]

Secondly, relative clauses are supposed to make a statement "about" their head noun (Kuno, 1976). This functional requirement might make processing island-violating relative clauses difficult for hearers because the resumptive NP appears in a position unlikely to be the theme of the clause; that is, in general, sentences are unlikely to be "about" an NP embedded in an adjunct clause, a relative clause, or other island context.

One final possible reason for the marginal acceptability of resumptives in relative clauses applies to both island- and non-island violating contexts. Given the predominance of gap-containing relative clauses in English, hearers expect to find a gap in a relative clause. This gap allows the semantic relation between the extracted NP and the relative clause to be computed directly from the syntax. When a resumptive pronoun (or other NP) appears in such a slot instead, additional non-syntactic processing is required to compute this relation. This explains why native speakers often judge resumptive pronouns in non-island violating contexts as unacceptable as those in island-violating contexts (although they *produce* both.)

Given the statistical preference in English for gap-containing relatives, for English speakers constructions with resumptive pronouns in both island and non-island contexts could be explained as a result of poor planning commitments during the production process. Under this line of explanation, we would hypothesize that at the point at which they begin selecting trees to encode meaning, speakers have already committed to combining more information into a single utterance than they, under ideal circumstances, would wish to express in just a main clause and a relative clause. Rather than using two separate main clauses to make two separate predications, they are "stuck" with using one main clause and a relative clause tree that is essentially a main clause adjoined in as a modifier of an NP.

5. Implications for the typology of relative clauses

As discussed extensively in the literature, the syntactic strategies for relativizing within a given language are closely tied to the Accessibility Hierarchy (AH). In general, the lower on the AH an NP's syntactic role is, the less likely a language is to use a gap rather than a resumptive when relativizing that NP (Kuno, 1976; Keenan and Comrie, 1977; Prince, 1990). In addition, for any position on the AH, if it can be relativized using a gap, all positions higher than it can too.

Three questions that this analysis might be expected to shed light on with respect to the strategies for relativization will be examined:

- Why are NPs low on the Accessibility Hierarchy harder to relativize with gaps?
- Are gaps always easier to process than resumptives?
- Are resumptives always interpreted as discourse pronouns (and gaps as bound)?

5.1. Relativization strategy and the Accessibility Hierarchy

The syntactic analysis given here predicts only that in non-island contexts either gaps or resumptives should be possible, and in island contexts only the latter are possible. However, in English, relativization of NPs lower on the AH may be more likely to appear as either gaps or resumptives in relatives, as in the datives in (14) [=Prince (1990)'s (7)a-c)].

(14) a. ...the man who this made **him** feel sad...

b. Some of the same judges who we told **them** that if you mess with John Africa...

c. He looks like one of those guys you got to be careful throwing **them** fastballs

3. Although it is an important question, whether such a classification is meaningful will not be further discussed here. Instead, we will simply take this sub-acceptability as a given and explore further reasons for it.

Besides the thematic claim of Kuno (1976) mentioned above, we do not know why the AH is related to relativization. We can only speculate why producing a relative clause where the relativized NP is "unthematic" and gapped is difficult. With respect to production models based in TAG, taking the AH into account in relativization would possibly require the activation of multiple trees in cases where either resumptives or gaps are possible.[4] The lower the NP role is on the AH, the more equally the trees might be weighted probabilistically allowing either to be produced with equal likelihood.

5.2. Relative processing difficulty of gaps and resumptives

English is unusual in its ability to relativize from even rather deeply embedded positions (Keenan, 1985). As discussed above in Section 4.2 the comprehension system of English is likely affected by the relative frequency of gaps vs. resumptive pronouns. Because gap-containing forms are the more frequent, and hence less "marked", form, they should be easier to comprehend. In addition, they should be the preferred form to produce. In other languages where the break between gaps and resumptives falls at a different position on the Accessibility Hierarchy, and so the frequency of resumptive forms is much greater, the relative ease of processing resumptive relative clauses in both production and comprehension may very well differ from that in English.

5.3. Interpretations of gaps and resumptives

In English, resumptive pronouns in non-island contexts have a specific discourse function, in that they serve as discourse pronouns. This appears to correspond to the syntactic analysis given here, because in resumptive relative clauses, any NPs within the relative clause are unbound by the relative pronoun, while in the case of a gap/trace, there is syntactic binding. However, zero pronouns can function as discourse pronouns in numerous languages, and phonologically explicit pronouns can function as bound pronouns in many languages, including English. Therefore, the syntactic analysis here does not necessarily have implications for the semantic/pragmatic interpretation of the two strategies of relativization examined here. Any typological claims would need to be based on corpora of naturally-occurring examples, in order to take into account the properties of their contexts.

6. Conclusion and areas for further investigation

This paper has presented evidence to resolve the paradox that resumptive pronouns in island-violating contexts would otherwise present for TAG-based model of sentence production. I have argued that given the grammatical resources of English, specifically resumptive pronouns in non-island violating relative clauses, resumptive pronouns in island-violating contexts are part of the grammatical competence of speakers of English. Their marginal acceptability is most likely due to frequency factors affecting their processing.

In order to confirm whether these forms are actually the result of "poor planning," psycholinguistic experimentation is needed. A focus in such experimentation on how and why speakers decide to split up information they encode into units (a main clause with attached subordinate vs. two main clauses) would be particularly useful.

An additional linguistic area of exploration would be whether the syntactic analysis here of resumptives in island vs. non-island relatives could be extended to their role in other related structures, like wh-questions and topicalization. Preliminary evidence seems to support an affirmative answer for topicalization structures, as shown by the distribution in (15-18).

(15) [Most of those people]$_i$ I never met $\emptyset_i$. (SSA)[5]

(16) And [those that hadn't]$_i$, I assumed **they**$_i$ were interested in basketball or football, so it was not difficult to figure out a way to get to see them. (SSA)

(17) But [the field office claims of the Claims Manual]$_i$, you tended to learn pretty much all that was in **it**$_i$ as a Claims Rep...(SSA)

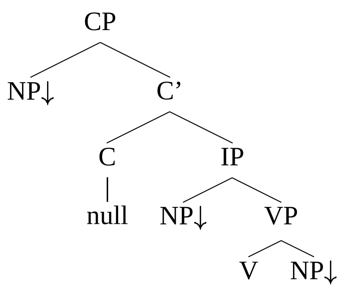
(18) (Being Unix, it comes with Gnu Emacs out of the box,)
Latex there are also a few options. (Email: T.M.)

Here, we have a coreferential gap in (15), a coreferential pronoun in (16), a coreferential pronoun in an NP-complement island in (17), and finally one with neither a coreferential NP nor a gap in (18). On first glance then, it

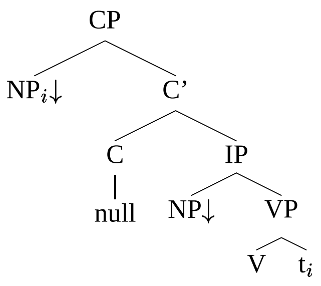
4. See Ferreira (2000) for a discussion of the simultaneous activation of multiple trees in sentence production.
5. The SSA marked examples are from transcribed oral histories at http://www.ssa.gov/history/orallist.html.

appears that speakers do have the option of both of the following trees for extraction from full clauses on analogy with the options they have for relatives clauses:

(19)



(20)



References

Ferreira, Fernanda. 2000. Syntax in Language Production: An Approach Using Tree-adjoining Grammars. In L. Wheeldon, editor, *Aspects of Language Production*. MIT Press, Cambridge, MA.

Frank, Robert. 2002. *Phrase structure composition and syntactic dependencies*. Cambridge, MA: MIT Press.

Frank, Robert and William Badecker. 2001. Modeling syntactic encoding with incremental tree-adjoining grammar. In *Proceedings of the 14th Annual CUNY Conference on Human Sentence Processing*, University of Pennsylvania.

Keenan, Edward. 1985. Relative Clauses. In Timothy Shopen, editor, *Language Typology and Syntactic Description, Vol. II*. Cambridge University Press.

Keenan, Edward and Bernard Comrie. 1977. Noun Phrase Accessibility and Universal Grammar. *Linguistic Inquiry*, 8(1):63–99.

Kroch, Anthony. 1981. On the role of resumptive pronouns in amnestying island constraint violations. In *Papers from the 17th regional meeting of the Chicago Linguistic Society*, pages 125–35, Chicago. Chicago Linguistic Society.

Kuno, Susumo. 1976. Subject, theme and speaker's empathy: A reexamination of relativization phenomena. In Charles Li, editor, *Subject and topic*. Academic Press, New York, pages 417–44.

Prince, E. F. 1999. How not to mark topics: 'Topicalization' in English and Yiddish. University of Pennsylvania.

Prince, Ellen. 1990. Syntax and discourse: a look at resumptive pronouns. In Kira Hall, Jean-Pierre Koenig, Michael Meacham, Sondra Reinman and Laurel Sutton, editors, *Proceedings of the Sixteenth Annual Meeting of the Berkeley Linguistics Society*. Berkeley Linguistics Society, Berkeley, CA, pages 482–97.

Statistical Morphological Tagging and Parsing of Korean with an LTAG Grammar

Anoop Sarkar and Chung-hye Han
University of Pennsylvania *Simon Fraser University*

1. Introduction

This paper describes a lexicalized tree adjoining grammar (LTAG) based parsing system for Korean which combines corpus-based morphological analysis and tagging with a statistical parser. Part of the challenge of statistical parsing for Korean comes from the fact that Korean has free word order and a complex morphological system. The parser uses an LTAG grammar which is automatically extracted using LexTract (Xia *et al.*, 2000) from the Penn Korean TreeBank (Han *et al.*, 2002). The morphological tagger/analyzer is also trained on the TreeBank. The tagger/analyzer obtained the correctly disambiguated morphological analysis of words with 95.78/95.39% precision/recall when tested on a test set of 3,717 previously unseen words. The parser obtained an accuracy of 75.7% when tested on the same test set (of 425 sentences). These performance results are better than an existing off-the-shelf Korean morphological analyzer and parser run on the same data.

In section 2, we introduce the Korean TreeBank and we discuss how an LTAG grammar for Korean was extracted from this TreeBank. Also, we discuss how the derivation trees extracted from the TreeBank are used in the training of the statistical parser. Section 3 presents the overall approach of the morphological tagger/analyzer that we use in the parser. A detailed discussion about the parser is presented in section 4. This section also presents the method we used to combine the morphological information into the statistical LTAG parser. We also provide the experimental evaluation of the statistical parser on unseen test data in section 4.

2. Automatically Extracted LTAG Grammar for Korean

In this section we describe the Penn Korean TreeBank and the nature of the extracted LTAG grammar from this TreeBank.

2.1. Korean TreeBank

The LTAG grammar we use in the parser is extracted using LexTract (Xia *et al.*, 2000) from the Penn Korean TreeBank. The derivation trees obtained by using LexTract on the Treebank are used to train the statistical parser. The TreeBank has 54,366 words and 5,078 sentences. The annotation consists of a phrase structure analysis for each sentence, with head/phrase level tags as well as function tags (e.g., -SBJ, -OBJ) and empty category tags for traces (*T*) and dropped arguments (*pro*). Each word is morphologically analyzed, where the lemma and the inflections are identified. The lemma is tagged with a part-of-speech (POS) tag (e.g., NNC: noun, NPN: pronoun, VV: verb, VX: auxiliary verb), and the inflections are tagged with inflectional tags (e.g., PCA: case, EAU: inflection on verbs followed by an auxiliary verb, EPF: tense, EFN: sentence type). Example TreeBank trees are given in Figure 1. The figure on the left is an example of a bracketed structure for a simple declarative with canonical subject-object-verb order. The figure on the right is an example with a displaced constituent. In this example, the object NP '권한을' appears before the subject, while its canonical position is after the subject. The sentences used to illustrate bracketing structures in Figure 1 are romanized, glossed and translated in the following examples:

(1) a.

Cey-ka	kwanchuk	sahang-ul	pokoha-yess-supnita.
I-Nom	observation	item-Acc	report-Past-Decl

'I reported the overvation items.'

b.

Kwenhan-ul	nwukwu-ka	kaci-ko	iss-ci?
authority-Acc	who-Nom	have-AuxConnective	be-Int

'Who has the authority?'

* We would like to thank Fei Xia for the use her LexTract package without which we could not have started this research. We are also indebted to Martha Palmer, Aravind Joshi and David Chiang for their helpful suggestions.

```
(S (NP-SBJ 제/NPN+가/PCA)
   (VP (NP-OBJ 관측/NNC
               사항/NNC+을/PCA)
       보고하/VV+었/EPF+습니다/EFN)
   ./SFN)
```

```
(S (NP-OBJ-1 권한/NNC+을/PCA)
   (S (NP-SBJ 누구/NPN+가/PCA)
      (VP (VP (NP-OBJ *T*-1)
              가지/VV+고/EAU)
          있/VX+지/EFN))
   ?/SFN)
```

Figure 1: Example TreeBank Trees

2.2. LTAG formalism

LTAGs are based on the Tree Adjoining Grammar formalism developed by Joshi and his colleagues (Joshi, Levy and Takahashi, 1975; Joshi and Schabes, 1997). The primitive elements of an LTAG are elementary trees which are of two types: initial trees (α trees) and auxiliary trees (β trees). Initial trees are minimal linguistic structures that contain no recursion. They include trees containing the phrasal structures of simple sentences, NPs and so forth. Auxiliary trees represent recursive structures, which are adjuncts to basic structures, such as adverbials, adjectivals and so forth. Auxiliary trees have unique leaf node, called the *foot* node ($*$), which has the same syntactic category as the root node. Each elementary tree is associated with a lexical item (anchor) and encapsulates all arguments of the anchor, possessing an extended domain of locality. Elementary trees are combined by two operations: substitution and adjunction. In the substitution operation, a node marked for substitution ($\downarrow$) in an elementary tree is replaced by another elementary tree whose root category is the same as the substitution-marked node. In an adjunction operation, an auxiliary tree is inserted into an initial tree. The root and the foot nodes of the auxiliary tree must match the node label at which the auxiliary tree adjoins. The combination of elementary trees produces two structures: derived and derivation trees. Derived trees correspond to phrase structure representation and derivation trees are a record of the history of the combination process.

2.3. Extracted LTAG grammar

We use LexTract (Xia *et al.*, 2000) to convert the phrase structure trees of the Korean TreeBank into LTAG derivation trees. Each node in these derivation trees is an elementary tree extracted from the Korean TreeBank by LexTract. The elementary trees of the LTAG Korean grammar are exactly the set of elementary trees used in the derivation trees obtained using LexTract. For example, the elementary trees extracted from the TreeBank bracketed structures in Figure 1 are given in Figure 2. The entire extracted grammar contains 632 elementary tree template types and 13,941 lexicalized elementary tree types (Xia *et al.*, 2001).

As mentioned earlier, in addition to the elementary trees, LexTract produces derived and derivation trees for each TreeBank tree. For instance, for the second TreeBank tree in Figure 1, 권한 and 누구 trees are each substituted into 가지 tree, and 있 tree is adjoined onto the VP node of 가지 tree. This produces the derived and derivation trees in Figure 3.

3. TreeBank-trained Morphological Tagger/Analyzer

Korean is an agglutinative language with a very productive inflectional system. This means that for any NLP application on Korean to be successful, some amount of morphological analysis is necessary. Without it, the development of a statistical based parser would not be feasible due to the sparse data problem bound to exist in the training data.

To avoid this problem in our parsing system, we use a morphological tagger/analyzer. This tagger/analyzer also performs statistical disambiguation and it was trained on 91% of the Korean TreeBank. The tagger/analyzer takes raw text as intput and returns a lemmatized disambiguated output in which for each word, the lemma is labeled with a POS tag and the inflections are labeled with inflectional tags. This system is based on a simple statistical model combined with a corpus-driven rule-based approach, comprising a trigram-based tagging component and a morphological rule application component.

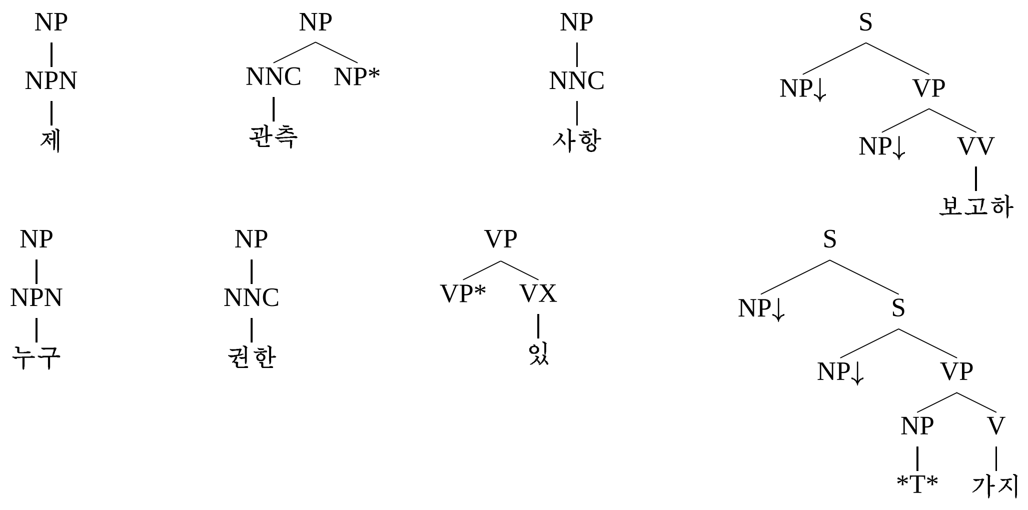
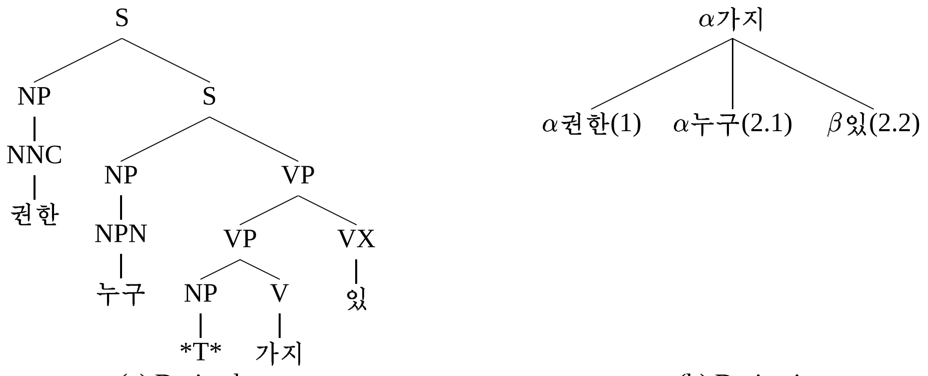


Figure 2: Some examples of Extracted Elementary Trees



(a) Derived tree (b) Derivation tree

Figure 3: Extracted Derived and Derivation Trees

The tagger/analyzer follows several sequential steps to label the raw text with POS and inflectional tags. After tokenization (mostly applied to punctuations), all morphological contractions in the input string are uncontracted (STRING CONVERSION). The known words are then tagged with tag sequences of the form POS + *inflectional-tag* (e.g., NNC+PCA, VV+EPF+EFN) extracted from the TreeBank, and unknown words are tagged with NNC (common noun) tag, NNC being the most frequent tag for unknown words (MORPH TAGGING). Tags for unknown words are then updated using inflectional templates extracted from the TreeBank (UPDATE TAGGING). And then using the inflection dictionary and stem dictionary extracted from the TreeBank, the lemma and the inflections are identified, splitting the inflected form of the word into its constituent stem and affixes (LEMMA/INFLECTION IDENTIFICATION), creating the final output. This process is summarized in Figure 4. The proposed approach to morphological analysis is different from other approaches in that the tagging phase precedes the morphological analysis phase. This allows morphological analysis to be done deterministically through using the information obtained from tagging. An example input to the tagger/analyzer and the final output are shown in Figure 5.

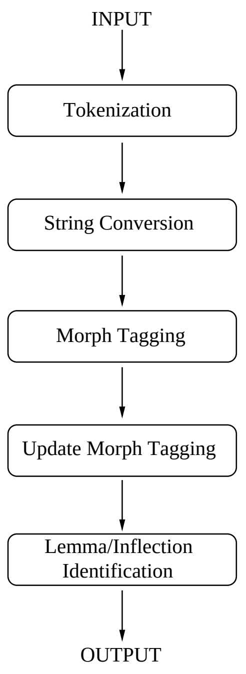


Figure 4: Overview of the Tagger/Analyzer

```
Input:
    제가 관측 사항을 보고했습니다 .

Output:
    제/NPN+가/PCA 관측/NNC 사항/NNC+을/PCA 보고하/VV+었/EPF+습니다 ./SFN
```

Figure 5: Input and output from the morphological tagging phase

The performance of the morphological analyzer/tagger has been evaluated on the 9% of the Treebank. The test set consists of 3,717 word tokens and 425 sentences. Both precision and recall were computed by comparing the morpheme/tag pairs in the test file and the gold file. The precision corresponds to the percentage of morpheme/tag pairs in the gold file that match the morpheme/tag pairs in the test file. And the recall corresponds to the percentage of morpheme/tag pairs in the test file that match the morpheme/tag pairs in the gold file. This approach yielded a precision/recall score of 95.79/95.39%.

An off-the-shelf morphological analyzer/tagger (Yoon *et al.*, 1999) was tested on the same test set. This system is reported to have obtained 94.7% tagging accuracy on a test set drawn from the same corpus as it was trained on. For the sake of fair comparison, the output of the off-the-shelf tagger was converted to look as close as possible

	precision/recall (%)
Treebank trained	95.78/95.39
Off-the-Shelf	29.42/31.25

Table 1: Evaluation of the Morphological Analyzer/Tagger

to the Treebank trained analyzer/tagger output, including the tagset. However, not all tokenization mismatches in the off-the-shelf system could be resolved. The results (in Table 1) show, not surprisingly, that better performance on test data from a particular domain is obtained by training on annotated data from the same domain. Even so, the results from another system on the same data provide at least a baseline performance to compare against our results.

4. Statistical LTAG Parser

The use of lexical information plays a prominent role in statistical parsing models for English. In this section, we discuss how to extend a statistical parser that relies on bigrams of lexical dependencies to a morphologically complex language like Korean. While these types of parsers have to deal with sparse data problems, this problem is exacerbated in the case of Korean due to the fact that several base-forms of words can appear with a wide array of morphological affixes. This problem is addressed by incorporating the morphological tagger/analyzer described above, which significantly improves performance.

Apart from the use of a specialized morphological tagger/analyzer for Korean, our methods are language independent and have been tested in previous work on the WSJ Penn English TreeBank (Marcus, Santorini and Marcinkiewicz, 1993). As described in above, we use Lextract to convert the TreeBank (the same method is used for both the English and the Korean TreeBanks) into a parser derivation tree for each sentence. The statistical parsing model is then trained using these derivation trees.

4.1. Probability Models

The statistical parser uses three probabilistic models: one model for picking a tree as the start of a derivation; and two models for the probability of one tree substituting or adjoining into another tree. Each of these models can be trained directly using maximum likelihood estimation from the Lextract output. The probabilistic models of substitution and adjunction provide a natural domain to describe the dependencies between pairs of words in a sentence.

(Resnik, 1992) provided some early motivation for a stochastic version of Tree Adjoining Grammars and gave a formal definition of stochastic TAG. Simultaneously, (Schabes, 1992) also provided an identical stochastic version of TAG and also extended the Inside-Outside algorithm for CFGs (Lari and Young, 1990) to stochastic TAGs. (Schabes, 1992) also performed experiments to show that a stochastic TAG can be learnt from the ATIS corpus.

A stochastic LTAG derivation proceeds as follows (Schabes, 1992; Resnik, 1992). An initial tree is selected with probability P_i and subsequent substitutions are performed with probability P_s and adjunctions are performed with probability P_a.

For each τ that can be valid start of a derivation:

$$\sum_{\tau} P_i(\tau) = 1$$

Each subsequent substitution or adjunction occurs independently. For possible substitutions defined by the grammar:

$$\sum_{\alpha} P_s(\tau, \eta \to \alpha) = 1$$

where, α is substituting into node η in tree τ. For possible adjunctions in the grammar there is an additional factor which is required for the probability to be well-formed:

$$P_a(\tau, \eta \rightarrow \textsc{na}) + \sum_{\beta} P_a(\tau, \eta \rightarrow \beta) = 1$$

where, β is adjoining into node η in tree τ, and $P_a(\tau, \eta \rightarrow \textsc{na})$ is the probability that there is no adjunction (NA) at node η in τ.

Each LTAG derivation $\mathcal{D}$ is built starting from some initial tree α. Let us consider the probability of a derivation $\mathcal{D}$ which was constructed using p substitutions and q adjunctions and r internal nodes which had no adjunction. If we assume that each elementary tree is lexicalized (*anchored*) by exactly one word, then the length of the sentence $n = p + q + 1$. In fact, in the experiments we report on in this paper, each elementary tree has exactly one lexical item as an anchor.

$$\begin{aligned} & Pr(\mathcal{D}, S = w_0 \ldots w_n) = \\ & \quad P_i(\alpha, w_i) \times \prod_p P_s(\tau, \eta, w \rightarrow \tau', w') \times \\ & \quad \prod_q P_a(\tau, \eta, w \rightarrow \tau', w') \times \\ & \quad \prod_r P_a(\tau, \eta, w \rightarrow \textsc{na}) \end{aligned} \tag{2}$$

This derivation $\mathcal{D}$ can be drawn graphically as a tree where each node in this derivation tree is an elementary tree in the original LTAG (this is the standard notion of a TAG derivation tree).

P_i and P_s can be written as the following lexicalized conditional probabilities which can be estimated from the training data.

$$\begin{aligned} P_i(\tau) &= P_i(\alpha, w \mid TOP) \\ P_s(\tau, \eta \rightarrow \alpha) &= P_s(\alpha, w' \mid \tau, \eta, w) \\ P_a(\tau, \eta \rightarrow \beta) &= P_a(\beta, w' \mid \tau, \eta, w) \end{aligned}$$

Events for each of these probability models can be directly read from the LexTract output. Using maximum likelihood estimation we convert these events into the above parameter values in our statistical parser.

For further details about decomposing these probabilities further to generalize over particular lexical items and to make parsing and decoding easier see (Chiang, 2000). (Chiang, 2000) also has details about the standard use of prior probabilities in statistical parsing for pruning which we use in our implementation.

The probability of a sentence S computed using this model is the sum of all the possible derivations of the sentence.

$$P(S) = \sum_D Pr(D, S)$$

A generative model can be defined instead of a conditional probability to obtain the best derivation $\mathcal{D}_{\text{BEST}}$ given a sentence S. The value for (3) is computed using the Equation 2.

$$\begin{aligned} \mathcal{D}_{\text{BEST}} &= \underset{\mathcal{D}}{\text{arg max}} \ \Pr(\mathcal{D} \mid S) \\ &= \underset{\mathcal{D}}{\text{arg max}} \ \frac{\Pr(\mathcal{D}, S)}{\Pr(S)} \\ &= \underset{\mathcal{D}}{\text{arg max}} \ \Pr(\mathcal{D}, S) \end{aligned} \tag{3}$$

The particular definition of a stochastic TAG is by no means the only way of defining a probabilistic grammar formalism with TAG. There have been some variants from the standard model that have been published since the original stochastic TAG papers.

For example, the restriction of one adjunction per node could be dropped and a new variant of standard TAG can be defined which permits arbitrary number of modifications per node. This variant was first introduced by (Schabes and Shieber, 1992; Schabes and Shieber, 1994). Tree Insertion Grammar (Schabes and Waters, 1995) is a variant of TAG where the adjoining operation is restricted in a certain way and this restricted operation is named insertion. TIGs are weakly equivalent to CFGs but they can produce structural descriptions that are not obtainable by any CFG.

A stochastic version of insertion (Schabes and Waters, 1996) was defined in the context of Tree Insertion Grammar (TIG). In this model, multiple trees can be adjoined to the left and to the right of each node with the following probabilities:

$$P_{la}(\tau, \eta \rightarrow \textrm{NA}_l) + \sum_{\tau'} P_{la}(\tau, \eta \rightarrow \tau') = 1$$

$$P_{ra}(\tau, \eta \rightarrow \textrm{NA}_r) + \sum_{\tau'} P_{ra}(\tau, \eta \rightarrow \tau') = 1$$

In our parser, we allow multiple adjunctions at a node and also we exploit TIG style probabilities P_{la} and P_{ra}. This was done so that the output easily convertible to the earlier dependency style parser that was used in the project (with which we compare performance in our evaluation).

There are many other probability measures that can be used with TAG and its variants. One can easily go beyond the bi-lexical probabilities that have been the main focus in this chapter to probabilities that invoke greater amounts of structural or lexical context. (Carroll and Weir, 1997), for example, gives some additional probability models one might consider useful when using TAGs.

An example output from the statistical parser is shown in Figure 6. In the parser (and in the Lextract output), each elementary tree is anchored by exactly one lexical item. The gloss and translation for the example in Figure 6 is given in the following example:

(4)	Motun	hochwul	tayho-nun	mayil	24	si-ey	pakkwui-key
	every	call	sign	everyday	24	hour-at	switch-AuxConnect
	toy-ciyo.						
	be-Decl						

‘Every call sign is switched at midnight everyday.’

Index	Word	POS tag (morph)	Elem Tree	Anchor Label	Node Address	Subst/Adjoin into (Index)
0	모든	DAN	βNP*=1	anchor	root	2
1	호출	NNC	βNP*=1	anchor	root	2
2	대호+는	NNC+PAU	αNP=0	anchor	0	6
3	매일	ADV	βVP*=25	anchor	1	6
4	24	NNU	βNP*=1	anchor	0	5
5	시+에	NNX+PAD	βVP*=17	anchor	1	6
6	바뀌+게	VV+ECS	αS-NPs=23	anchor	-	TOP
7	되+지요	VX+EFN	βVP*=13	anchor	1	6
8	.	SFN	-	-	-	-

Figure 6: Example derivation of a sentence reported by the statistical parser

4.2. Incorporating the Morphological Information into the Parser

In this section we describe how the morphological tagger (described earlier) is incorporated into the smoothing of the probability models that are used in the statistical parser.

The smoothing using the morphological tagger is handled as follows. The statistical parser has various probability models associated with it. One of the most crucial models is the one that decides on parser actions by using statistics collected on pairs of words. For example, P_a is the probability of combination of two trees anchored by

	On training data	On test data
Current Work	97.58	75.7
(Yoon, Kim and Song, 1997)	–	52.29/51.95 P/R

Table 2: Parser evaluation results

words w and w' via adjunction of those trees. Here w and w' are the inflected forms of the word, while p and p' are selected elements of the disambiguated morphological analysis of the inflected forms taken from the output of the morphological tagger described above. Based on the part of speech, we might want to select different components of the morphological analysis. For example, for nouns we might want to pick the final element which usually corresponds to the case marking, while for verbs we might want to pick the stem which is the first element of the analysis. We have the flexibility in the parser to choose which element should be picked. The best results were obtained when we chose the stem. That is, for the results reported here we always pick the first element of the morphological analysis for each inflected form.

$$Pr(t', p', w' \mid \eta, t, w, p)$$

We decompose this conditional probability into the following components:

$$\begin{aligned} &Pr(t', p', w' \mid \eta, t, w, p) = \\ &\quad Pr(t' \mid \eta, t, w, p) \times \\ &\quad Pr(p' \mid t', \eta, t, w, p) \times \\ &\quad Pr(w' \mid p', t', \eta, t, w, p) \end{aligned}$$

For each of the equations above, we use a backoff model which is used to handle sparse data problems. We compute a backoff model as follows. Let e_1 stand for the original lexicalized model and e_2 be the backoff level which only uses the output of the morphological tagger described above:

$$\begin{aligned} Pr_{e_1} &= Pr(t' \mid \eta, t, w, p) \\ Pr_{e_2} &= Pr(t' \mid \eta, t, p) \end{aligned}$$

The backoff model is computed as follows:

$$\lambda(c) \times Pr_{e_1} + (1 - \lambda(c)) \times Pr_{e_2}$$

where $\lambda(c) = \frac{c}{(c+D)}$. $c = \mathit{count}(r)$ is the total count for each conditional probability model P, where $P(l \mid r)$. D is the *diversity* of Pr_{e_1}. *diversity* is defined as the number of distinct counts for Pr_{e_1}. Note that this backoff model is implemented for each lexicalized model used in the statistical parser.

4.3. Experimental Results

In order to evaluate the statistical parser (combined with the tagger as described above), our parser was trained on the derivations and corresponding LTAG grammar extracted from 91% of the TreeBank (4653 sentences). The parser was then tested on 9% of the TreeBank which was reserved as unseen data (425 sentences). Note that the parser did not have access to any grammar information from LexTract or lexicon information which was taken from the unseen data.

For comparison, another parser (Yoon, Kim and Song, 1997) was tested on the same test set. For the sake of fair comparison, the output of this parser was converted to look as close as possible to our output. Even so, the number of node labels did not match, due to the difference in tokenization schemes for certain lexical elements such as copulas and auxiliary verbs. We thus report precision/recall in the comparison. We report word-to-word dependency accuracy compared with the gold standard for our parser. The evaluation results are summarized in Table 2. Not surprisingly, the results show that better performance on test data from a particular domain is obtained by training on annotated data from the same domain. The results from another parser on the same data provide a baseline performance to compare against our results. Also, the performance achieved on our test set is competetive with the state-of-the-art English statistical parsers when trained on similar amounts of data.

5. Conclusion

Our work is significant in that it is the first LTAG-based parsing system for Korean. We have shown that LTAG-based statistical parsing is feasible for a language with free word order and complex morphology. Our system has been successfully incorporated into a Korean/English machine translation system as source language analysis component (Han *et al.*, 2000; Palmer *et al.*, 2002). The LTAG parser produces a single-best analysis of the input Korean sentence. We showed that the tagger/analyzer described in this paper obtained the correctly disambiguated morphological analysis of words with 95.78/95.39% precision/recall when tested on a test set of 3,717 previously unseen words. The statistical parser described in this paper obtained an accuracy of 75.7% when tested on the same test set (of 425 sentences). These performance results are better than an existing off-the-shelf Korean morphological analyzer and parser run on the same data.

References

Carroll, J. and D. Weir. 1997. Encoding Frequency Information in Lexicalized Grammars. In *Proc. 5th Int'l Workshop on Parsing Technologies IWPT-97*, Cambridge, Mass.

Chiang, David. 2000. Statistical Parsing with an Automatically-Extracted Tree Adjoining Grammar. In *Proc. of ACL-2000*.

Han, Chung-hye, Na-Rae Han, Eon-Suk Ko, Heejong Yi and Martha Palmer. 2002. Penn Korean Treebank: Development and Evaluation. In *Proceedings of the 16th Pacific Asian Conference on Language and Computation*. Korean Society for Language and Information.

Han, Chung-hye, Benoit Lavoie, Martha Palmer, Owen Rambow, Richard Kittredge, Tanya Korelsky, Nari Kim and Myunghee Kim. 2000. Handling Structural Divergences and Recovering Dropped Arguments in a Korean/English Machine Translation System. In John S. White, editor, *Envisioning Machine Translation in the Information Future*. Springer-Verlag, pages 40–53.

Joshi, Aravind and Yves Schabes. 1997. Tree Adjoining Grammars. In A. Salomma and G. Rosenberg, editors, *Handbook of Formal Languages and Automata*. Springer-Verlag, Heidelberg.

Joshi, Aravind K., L. Levy and M. Takahashi. 1975. Tree Adjunct Grammars. *Journal of Computer and System Sciences*.

Lari, K. and S. J. Young. 1990. The estimation of stochastic context-free grammars using the Inside-Outside algorithm. *Computer Speech and Language*, 4:35–56.

Marcus, Mitch, Beatrice Santorini and M. Marcinkiewicz. 1993. Building a large annotated corpus of English. *Computational Linguistics*, 19(2):313–330.

Palmer, Martha, Chung-hye Han, Anoop Sarkar and Ann Bies. 2002. Integrating Korean analysis components in a modular Korean/English machine translation system. Ms. University of Pennsylvania and Simon Fraser University.

Resnik, P. 1992. Probabilistic Tree-Adjoining Grammars as a framework for statistical natural language processing. In *Proc. of COLING '92*, pages 418–424, Nantes, France.

Schabes, Y. 1992. Stochastic Lexicalized Tree-Adjoining Grammars. In *Proc. of COLING '92*, pages 426–432, Nantes, France.

Schabes, Y. and S. Shieber. 1992. An Alternative Conception of Tree-Adjoining Derivation. In *Proceedings of the* 20^{th} *Meeting of the Association for Computational Linguistics*.

Schabes, Y. and R. Waters. 1996. Stochastic Lexcalized Tree-Insertion Grammar. In H. Bunt and M. Tomita, editors, *Recent Advances in Parsing Technology*. Kluwer, pages 281–294.

Schabes, Yves and Stuart Shieber. 1994. An Alternative Conception of Tree-Adjoining Derivation. *Computational Linguistics*, 20(1):91–124.

Schabes, Yves and Richard Waters. 1995. Tree Insertion Grammar: A Cubic-Time, Parsable Formalism that Lexicalizes Context-Free Grammar without Changing the Trees Produced. *Computational Linguistics*, 21(4):479–513.

Xia, Fei, Chung-hye Han, Martha Palmer and Aravind Joshi. 2001. Comparing Lexicalized Treebank Grammars Extracted from Chinese, Korean, and English Corpora. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*.

Xia, Fei, Martha Palmer, and Aravind Joshi. 2000. A Uniform Method of Grammar Extraction and Its Applications. In *Proceedings of the EMNLP 2000*.

Yoon, J., S. Kim and M. Song. 1997. New Parsing Method using a Global Association Table. In *Proceedings of IWPT-97*.

Yoon, Juntae, C. Lee, S. Kim and M. Song. 1999. Morphological Analyzer of Yonsei Univ., Morany: Morphological Analysis based on Large Lexical Database Extracted from Corpus. In *Proceedings of Korean Language Information Processing*. Written in Korean.

Notes on the Complexity of Complex Heads in a Minimalist Grammar

Jens Michaelis
Universität Potsdam

Abstract

The type of a *minimalist grammar (MG)* introduced in Stabler (1997) provides a simple algebraic formalization of the perspectives as they arise from Chomsky (1995b) within the linguistic framework of transformational grammar. As known (cf. Michaelis 2001a; 2001b; Harkema, 2001), this MG–type defines the same class of derivable string languages as, e.g., *linear context–free (string) rewriting systems (LCFRSs)* (Vijay–Shanker, Weir and Joshi, 1987; Weir, 1988). In this paper we show that, in terms of weak equivalence, the subclass of MGs which allow *(overt) head movement* but no *phrasal movement* in the sense of Stabler (1997), constitutes a proper subclass of *linear indexed grammars (LIGs).* and thus *tree adjoining grammars (TAGs).* We also examine the "inner hierarchic complexity" of this embedding in some more detail by looking at the subclasses canonically resulting from a differentiation between left adjunction of the moved head to the attracting one, and right adjunction of this kind. Furthermore, we show that adding the possibility of phrasal movement by allowing just one "indistinguishable" licensee to trigger such movement has no effect on the weak generative capacity of the corresponding MG–subclasses. On the other hand however, MGs which do not employ head movement but whose licensee set consists of at most two elements, are shown to derive, a.o., languages not derivable by any LIG. In this sense our results contribute to sheding some light on the complexity as it arises from the interplay of two different structural transformation types whose common existence is often argued to be linguistically motivated.

1. Introduction

The type of a *minimalist grammar (MG)* introduced in Stabler (1997) provides a simple algebraic formalization of the perspectives as they arise from Chomsky (1995b) within the linguistic framework of a principles and parameter–approach to transformational grammar. As known (cf. Michaelis 2001a; 2001b; Harkema, 2001), this MG–type constitutes a *mildly context–sensitive formalism* in the sense that it defines the same class of derivable string languages as *linear context–free (string) rewriting systems (LCFRSs)* (Vijay–Shanker, Weir and Joshi, 1987; Weir, 1988).[1] In particular, the MG–definition permits a type of *(overt) head movement* which rather directly reflects the derivation mode of *(successive) head(–to–head) adjunction*—which, in the minimalist approach, takes place due to the necessity of feature checking—(successively) creating more complex heads (cf. Figure 1). Nev-

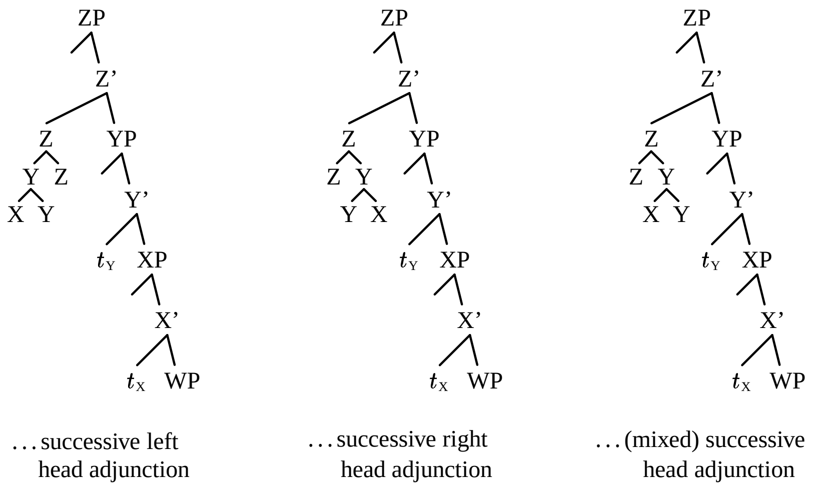

ZP
Z'
Z
YP
Y
Z
X
Y
Y'
t_Y
XP
X'
t_X
WP
. . . successive left head adjunction
. . . successive right head adjunction
. . . (mixed) successive head adjunction

Figure 1: Complex head "Z" [2] resulting from . . .

ertheless, there is a further notable property of MGs which—in connection with Michaelis (2001a)—follows from Harkema (2001) as well as Michaelis (2001b): each MG can be transformed into a weakly equivalent MG that

* This work has been funded by DFG–grant STA 519/1–1.

1. Hence, MGs as defined in Stabler (1997) join to a series of weakly equivalent formalism classes among which, beside LCFRSs, there is, e.g., the class of set–local *multicomponent tree adjoining grammars (MCTAGs)* (cf. Weir, 1988). For a list of some further of such classes of generating devices, beside MCTAGs, see e.g. Rambow and Satta (1999).
2. In terms of an X-Bar representation.

neither employs head movement nor *covert (phrasal) movement* as allowed by the general MG–definition.[3] In fact it is this MG–subtype which, e.g., is covered in terms of the succinct MG–reformulation in Stabler and Keenan (2000) (reducing MGs to their "bare essentials"), and which the MG–recognizer in Harkema (2000) (working in polynomial time depending on the length of the input string) is defined for. Moreover, the "MG–internal" equivalence result can be seen as providing some technical support to more recent linguistic work which, in particular, tries to completely dispense with any type of movement different from overt phrasal movement (e.g. Koopman and Szabolcsi, 2000; Mahajan, 2000).

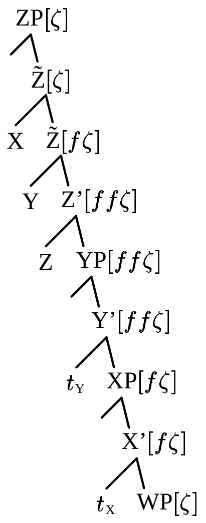


Figure 2: Complex head "Z" resulting from successive left head adjunction as representable in an RLIG.

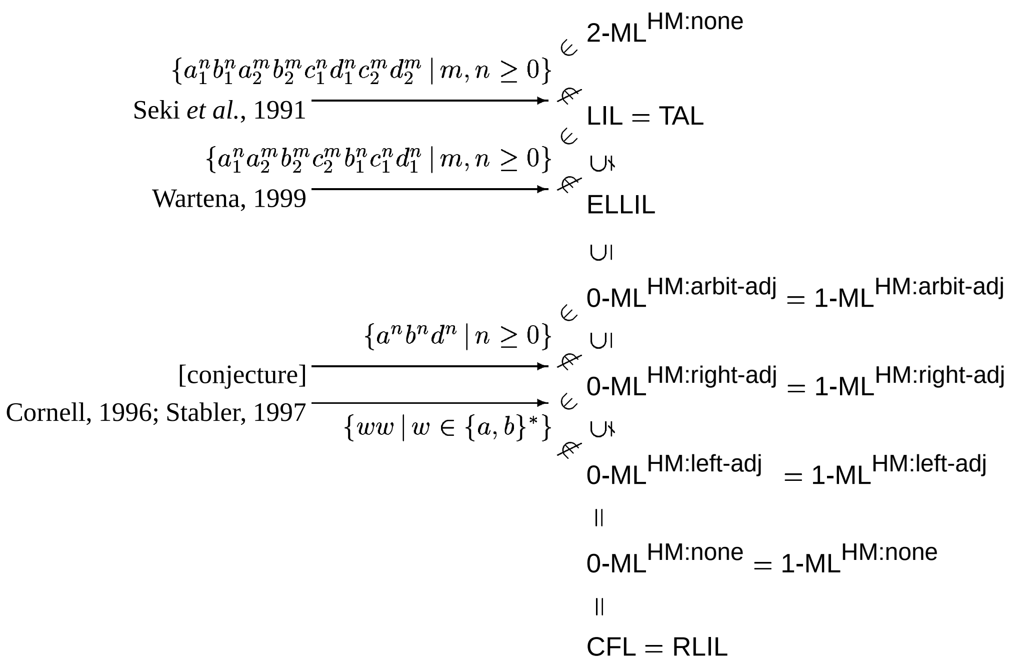
Many linguists working within the transformational tradition, however, believe head movement to be indispensable for an adequate explanation of natural language syntax.[4] How the kind of head movement originally allowed in an MG can be (re)integrated into the succinct MG–definition is shown in Stabler (2001). As indicated in there, the recognition complexity w.r.t. such an MG and an input string increases—adapting the methods of Harkema (2000)—at most as much as in the case when adding two new distinct *licensees*, i.e. two new distinct features potentially triggering phrasal movement, to the grammar.

Concentrating on questions concerning the increase of generative complexity, we show in this paper that, in terms of derivable string languages, the subclass of MGs allowing head movement but no phrasal movement in the sense of Stabler (1997), constitutes a proper subclass of LIGs, and thus TAGs. This is done by embedding MGs weakly equivalently into *extended left LIGs (ELLIGs)* in the sense of Michaelis and Wartena (1999).[5] Examining the "inner hierarchic complexity" of this embedding in some more detail, it can be shown that MGs which allow only left head adjunction define the same class of derivable languages as RLIGs,[6] and thus *context free grammars (CFGs)* (cf. Michaelis and Wartena, 1999). MGs which allow only right head adjunction derive more languages then CFGs, and MGs allowing right as well as left head adjunction seem to provide a further proper extension. Furthermore, adding the possibility of phrasal movement by allowing the MG's licensee set to consist of at most one feature[7] has no effect on the weak generative capacity of the considered MG–subclasses. On the other hand, it can be shown that MGs which do not employ head movement but whose licensee set consists of at most two elements, derive, a.o., languages not derivable by any LIG.[8]

The presented results are of interest in at least two respects: first, they contribute in a more general sense to one of the central issues mathematical approaches to linguistics are concerned with, namely, how much strong generative capacity can be squeezed out of a formalism without increasing the weak generative power.[9] Second, since the presented results provide a first narrow step towards an answer to the question of how the specific types of head movement and phrasal movement as defined in MGs are related to each other in terms of generative capacity, they may not only be a promising starting point, when seeking for a lower upper bound on the parsing complexity of MGs, but also shed some light on the structural complexity as it arises from the interplay of two different structural transformation types whose common existence is often argued to be linguistically motivated.

For illustrative purposes we demonstrate how MGs allowing head movement but no phrasal movement can be weakly equivalently embedded into a subclass of TAGs, instead of LIGs, which in its turn is weakly equivalent

3. The only movement possibly used is *overt phrasal movement*.
4. Even current accounts which argue in favour of overt vs. covert movement do not completely exclude overt head movement (e.g. Kayne 1998; 1999).
5. In Michaelis and Wartena (1999), ELLIGs were defined as the formal counterpart of *extended right LIGs (ERLIGs)*. An ELLIG (respectively, ERLIG) is an LIG, G, such that for each nonterminating production r, the distinguished symbol on the righthand side (rhs) is the leftmost (respectively, rightmost) nonterminal, i.e., r is of the form $A[\zeta\ldots] \to wB[\eta\ldots]\alpha$ (respectively, $r = A[\zeta\ldots] \to \alpha B[\eta\ldots]w$), where w is a string of terminals. Thus, applying such an r to a corresponding object $A[\zeta\theta]$, the stack associated with the nonterminal A is passed on to the leftmost (respectively, rightmost) nonterminal child replacing ζ, the upper part of the stack, by η. If, in addition, for each such nonterminating rule r, no terminal symbol appears to the left (respectively, right) of the distinguished nonterminal symbol of the rhs, i.e., if w is always the empty string, then G is simply referred to as an *LLIG (respectively, RLIG)*.
6. A corresponding weakly equivalent RLIG can be defined such that it represents complex heads, created by successive head adjunction, as indicated in Figure 2.
7. Thus, at most one "indistinguishable" type of phrasal movement is available.
8. A schematic overview is given in Figure 3.
9. See e.g. Joshi (2000) and references therein for a more recent discussion.



2-ML^HM:none
$\{a_1^n b_1^n a_2^m b_2^m c_1^n d_1^n c_2^m d_2^m \mid m,n \geq 0\}$
Seki *et al.*, 1991
LIL = TAL
$\{a_1^n a_2^m b_2^m c_2^m b_1^n c_1^n d_1^n \mid m,n \geq 0\}$
Wartena, 1999
ELLIL
0-ML^HM:arbit-adj = 1-ML^HM:arbit-adj
$\{a^n b^n d^n \mid n \geq 0\}$
[conjecture]
0-ML^HM:right-adj = 1-ML^HM:right-adj
Cornell, 1996; Stabler, 1997
$\{ww \mid w \in \{a,b\}^*\}$
0-ML^HM:left-adj = 1-ML^HM:left-adj
0-ML^HM:none = 1-ML^HM:none
CFL = RLIL

Figure 3: Schematic overview of our results.[10,11]

to ELLIGs (Section 3). Largely skipping formal details afterwards, we subsequently emphasize the crucial points concerning the hierarchy of the corresponding MG–subclasses resulting from the different types of head movement as available in the MG–formalism (Section 3.1–3.4). Then, we turn to simple phrasal movement as allowed by the MG–definition (Section 4) and, finally, present an example of an MG which does not employ any head movement, but phrasal movement "slightly beyond" the simple type, thereby deriving a language not derivable by any LIG (Section 5). But first, since the reader might be less familiar with MGs, we briefly introduce them in their aspects relevant here.

2. Minimalist Grammars

An MG is a formal device which specifies a countable set of *expressions (over $\mathcal{S} \cup \mathcal{P} \cup \mathcal{I}$)*,[12] i.e., a countable set of finite, binary (ordered) trees each equipped with a leaf–labeling function assigning a string from $\mathcal{S}^*\mathcal{P}^*\mathcal{I}^*$ to each leaf, and with an additional binary relation, the asymmetric relation of *(immediate) projection*, defined on the set of pairs of siblings (cf. Figure 4).

A *maximal projection* within such an expression τ is a subtree of τ which is either identical to τ, or its root is projected over by its root's sibling. The *specifiers*, the *complement* and the *head* of (a maximal projection in) τ are determined in the canonical way by means of the projection relation (cf. Figure 5). τ is *complete* if its head–label is in $\{\mathsf{c}\}\mathcal{P}^*\mathcal{I}^*$, and each other leaf–label in $\mathcal{P}^*\mathcal{I}^*$. The *phonetic yield* of τ is the string which results from concatenating the leaf–labels in "left–to–right–manner" ignoring all instances of non–phonetic features.

The base of an MG, G, is formed by a *lexicon* (a finite set of simple expressions, i.e. single node trees in the above sense, also called *heads*) and two structure building functions: *merge* (combining two trees) and *move*

10. Here, for $n \geq 0$ and $x \in \{\text{none}, \text{left-adj}, \text{right-adj}, \text{arbit-adj}\}$, $n\text{-ML}^{\text{HM}:x}$ denotes the class of all languages derivable by any MG whose licensee set consists of at most n elements, and which permits only head(–to–head) adjunction of the type indicated by x.

11. Note that Wartena (1999) actually provides a formal proof of the fact that the language $\{a_1^n b_1^n c_1^n a_2^m b_2^m c_2^m d_1^n \mid m,n \geq 0\}$, although derivable by some LIG, is not derivable by any ERLIG. For reasons of symmetry however, it becomes immediately clear from the corresponding proof details that the language $\{a_1^n a_2^m b_2^m c_2^m b_1^n c_1^n d_1^n \mid m,n \geq 0\}$, although derivable by some LIG, is not derivable by any ELLIG.

12. $\mathcal{S}$, $\mathcal{P}$ and $\mathcal{I}$ are assumed to be pairwise disjoint sets, namely, a set of *syntactic*, *phonetic* and *interpretable features*, respectively. $\mathcal{S}$ is partitioned into *basic categories*, *selectors*, *licensees*, and *licensors*. There is at least the basic category c.

13. Here, "<" (respectively, ">") as "label" of a non–leaf node means "my left (respectively, right) child projects over my right (respectively, left) child."

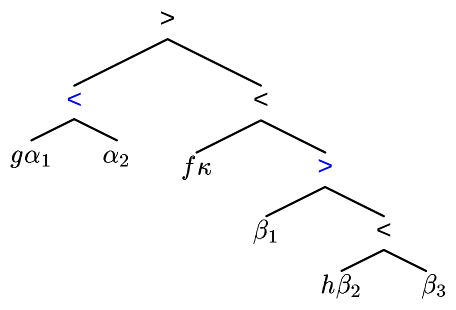


Figure 4: A typical (minimalist) expression.[13]

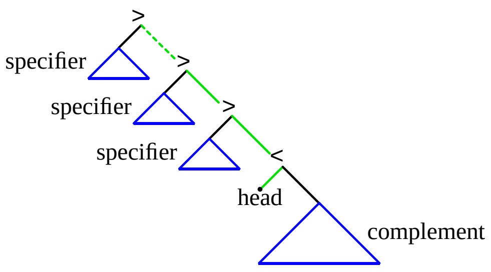
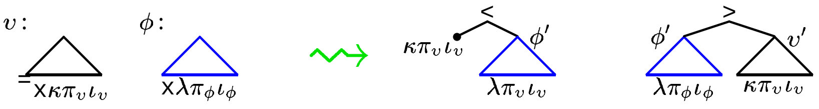


Figure 5: The typical (minimalist) expression structure.

(transforming a single tree). Both functions build structure by canceling two particular matching instances of syntactic features within the leaf–labels of the trees to which they are applied. The closure of the lexicon under these two functions is the set of trees specified by G. The *(string) language* derivable by G is a particular subset of $\mathcal{P}^*$, namely, the set of the phonetic yields of the complete expressions within this closure.

The function *merge* is applicable to $\langle \upsilon, \phi \rangle$, a pair of expressions, if ϕ's head–label starts with an instance of some basic category x, and υ's head–label with an instance of $\mathsf{=x}$, the corresponding *weak* selector of x. Depending on whether υ is simple or not, ϕ is selected as the complement or the highest specifier, respectively. Within the resulting tree, *merge*(υ, ϕ), the corresponding instances of $\mathsf{=x}$ and x are cancelled (cf. Figure 6). In



$\mathsf{=x}\kappa\pi_\upsilon\iota_\upsilon$
$\mathsf{x}\lambda\pi_\phi\iota_\phi$

$\kappa\pi_\upsilon\iota_\upsilon$

$\lambda\pi_\upsilon\iota_\upsilon$

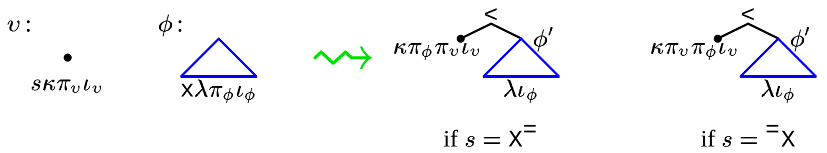
$\lambda\pi_\phi\iota_\phi$
$\kappa\pi_\upsilon\iota_\upsilon$

Figure 6: *merge*(υ, ϕ) — weak selection.

case υ is a head, its label may likewise start with an instance of a corresponding *strong* selector of x, $\mathsf{=X}$ or $\mathsf{X=}$, both additionally triggering *(overt) head movement*, i.e., *merge*(υ, ϕ) is defined as before, but in addition π_ϕ, the string of phonetic head–features of the selected ϕ, is incorporated into the label of the selecting head υ, either immediately to the right (triggered by $\mathsf{=X}$) or immediately to the left (triggered by $\mathsf{X=}$) of π_υ, the string of phonetic features within υ's (unique) label (cf. Figure 7).[14,15]

The function *move* is applicable to an expression υ, if there is exactly one maximal projection ϕ in υ whose head–label starts with in instance of some licensee $\mathsf{-x}$ such that υ's head–label starts with an instance of a cor-

14. In the minimalist approach suggested in Chomsky (1995b) the merge–operator applies freely, and head movement is a separate step following a corresponding application of this operator. As noted by Chomsky (1995a, p. 327), in a strong sense this can be seen as a violation of the "extension condition" on structure building functions. Stabler (1998, p. 78, fn. 5) argues that that the implementation of head movement within MGs not only avoids such a violation, but "it [also] explains the coincidence of the selection and head movement configurations." Note also that the implementation of head movement is in accordance with the *head movement constraint*, demanding that a moving head can never pass over the closest c–commanding head. To put it differently, whenever we are concerned with a case of successive head movement, i.e. recursive adjunction of a (complex) head to a higher head, it obeys *strict cyclicity*. The way in which MGs reflect the "usual" linguistic notion of head adjunction arising from head movement is made explicit in Stabler (1998).



υ :
$s\kappa\pi_\upsilon\iota_\upsilon$
ϕ :
$\mathsf{x}\lambda\pi_\phi\iota_\phi$
<
$\kappa\pi_\phi\pi_\upsilon\iota_\upsilon$
ϕ'
$\lambda\iota_\phi$
if $s = \mathsf{x}^=$
<
$\kappa\pi_\upsilon\pi_\phi\iota_\upsilon$
ϕ'
$\lambda\iota_\phi$
if $s = {}^=\mathsf{X}$

Figure 7: *merge*(υ, ϕ) — strong selection.

responding *strong* licensor +X or *weak* licensor +x triggering *overt* or *covert phrasal movement*, respectively.[16] If υ's head–label starts with a strong licensor then, within the resulting tree *move*(υ), ϕ is moved into the new created, highest specifier position, while the triggering instances of +X and -x are cancelled, and the "original" position of ϕ's root becomes a single node labeled with the empty string (cf. Figure 8). If υ's head–label starts with

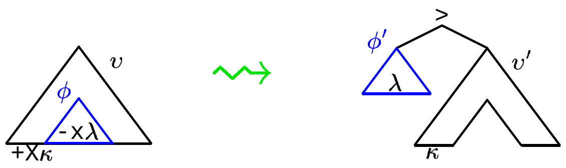


Figure 8: *move*(υ) — overt phrasal movement.

a weak licensor then, within the resulting tree *move*(υ), the triggering instance of +x is cancelled, while a copy of ϕ in which the triggering instance of -x as well as all instances of phonetic features are cancelled, is moved into the new created, highest specifier position, and while another copy of ϕ in which all instances of non–phonetic features are cancelled is "left behind."[17]

3. MG–Head Movement in Terms of TAGs

Let G be an MG which allows head movement but no phrasal movement.[18] A nonterminal in our weakly equivalent TAG, G', is either the start symbol, S, or a pair $\langle \mathsf{y}, t\rangle$ with y being a basic category from G, and with $t \in \{\mathtt{weak}, \mathtt{strong}\}$, where weak and strong are two new, distinct symbols.

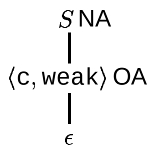


Figure 9: The unique initial tree of G'.

There is a single initial tree (cf. Figure 9), and for each lexical MG–item α, there are two elementary auxiliary trees depending on the form of the (unique) label of α: we generally distinguish the cases $\mathsf{y}\pi\iota$ (cf. Figure 10) and $s{}^=\mathsf{x}_1 \cdots {}^=\mathsf{x}_n\mathsf{y}\pi\iota$, s being any selector, ${}^=\mathsf{x}_1, \dots, {}^=\mathsf{x}_n$ weak selectors for an $n \geq 0$, y a basic category, $\pi \in \mathcal{P}^*$, and $\iota \in \mathcal{I}^*$. The latter case divides into three subcases depending on whether s is of the form ${}^=\mathsf{x}$, $\mathsf{X}^=$, or ${}^=\mathsf{X}$ (cf. Figure 11–13). Hence, G' only uses auxiliary elementary trees which may be called *extended right auxiliary*, i.e., auxiliary trees in which the foot node is the leftmost nonterminal node on the frontier, and all interior nodes left of the spine are marked for null adjunction.[19,20]

16. The uniqueness of ϕ provides us with a strict version of the *shortest move constraint (SMC)*.
17. For more details on the definition of *merge* and *move* see Stabler (1997). Particular examples of MGs are given below in Section 3–5.
18. That is, G does not employ the function *move* to derive any expression from some instances of the lexical items. Therefore, we may assume that no (label of any) lexical item contains an instance of some licensee or licensor feature.
19. This TAG–subtype may also be seen as a straightforward extension of a particular subtype of a *tree insertion grammar* (Schabes and Waters, 1995).
20. With the intend of simplifying our presentation, G' also fits in with the "classical" TAG–definition allowing selective adjunction, but not substitution (see e.g. Vijay–Shanker and Weir, 1994).

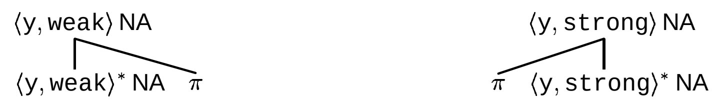


Figure 10: Elementary auxiliary trees of G' resulting from the lexical MG–item $\mathtt{y}\pi\iota$.

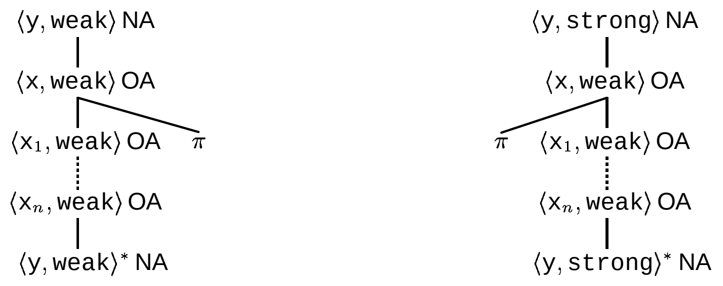


Figure 11: Elementary auxiliary trees of G' resulting from the lexical MG–item $^{=}\mathtt{x}^{=}\mathtt{x}_1 \cdots {}^{=}\mathtt{x}_n\mathtt{y}\pi\iota$.

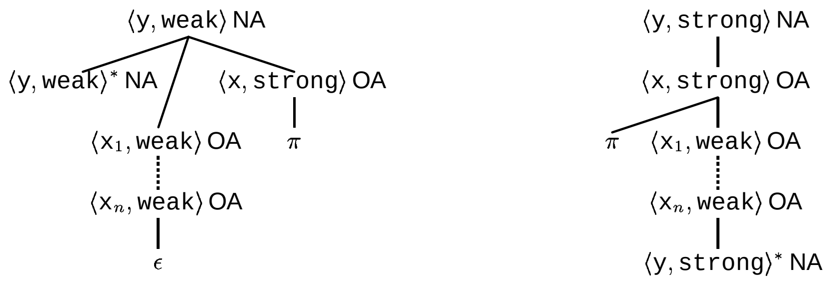


Figure 12: Elementary auxiliary trees of G' resulting from the lexical MG–item $\mathtt{X}^{==}\mathtt{x}_1 \cdots {}^{=}\mathtt{x}_n\mathtt{y}\pi\iota$.

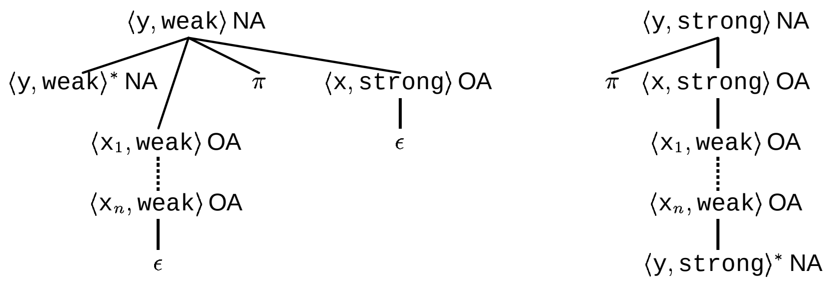


Figure 13: Elementary auxiliary trees of G' resulting from the lexical MG–item $^{=}\mathtt{X}^{=}\mathtt{x}_1 \cdots {}^{=}\mathtt{x}_n\mathtt{y}\pi\iota$.

G' simulates the MG–derivation of an expression τ whose head–label starts with a basic category y by "reversing the top–down order," i.e., the complement becomes the highest constituent, and the specifiers are successively attached top–down in the sense indicated in Figure 14. Such a derivation, indeed, is simulated by G' exactly twice in the two outline ways. Vice versa, each TAG–derivable auxiliary tree T which does not permit (further) adjunction to any of his nodes, corresponds to an MG–derivable expression τ whose head–label starts with a basic category y, in exactly one of the two outlined ways. Thus—ignoring the label of T's foot node—T's yield either equals τ's phonetic yield, or this is true up to the fact that the substring of τ's phonetic yield contributed by τ's head, $\textit{yield}_{\mathcal{P}}(\textit{head}_{\tau})$, is "shifted" to the front within T's yield. If the latter, it is just $\textit{yield}_{\mathcal{P}}(\textit{head}_{\tau})$ which constitutes T's yield left of its spine, and in MG–terms, T is connected with the expectation that the represented τ is inevitable selected strongly in a further derivation step (expressed by the second component of the label of T's root node, and thus foot node). Otherwise, T's yield left of its spine is empty, and the represented τ is connected with the demand of being selected weakly in a further derivation step (again coded by means of the second component of the label of T's root/foot node).

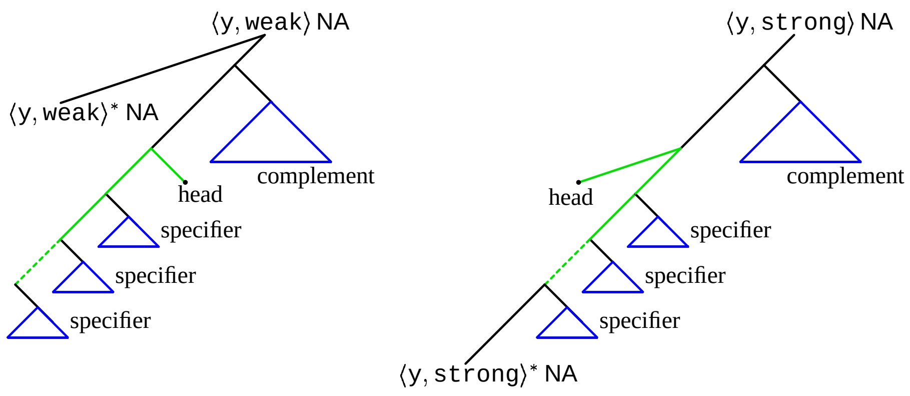


Figure 14: The MG–expression structure simulated by G'.

3.1. Null Head Adjunction

As an immediate consequence of our construction we observe that in case G does not use any strong selectors (i.e., no head movement takes place deriving an expression belonging to the closure of the lexicon of G), only (strictly) right auxiliary trees are effectively used in order to derive a tree in the weakly equivalent TAG G'. Thus, in this case, we are in fact concerned with a particular type of *tree insertion grammar* in the sense of Schabes and Waters (1995) additionally allowing adjunction constraints in the sense of the usual TAG–definition. Since adding the possibility of such constraints to the TIG–type does not increase the weak generative power, and since TIGs are known to constitute a weakly context–free formalism, this yields another proof of the well–known fact that MGs which do neither employ head movement nor phrasal movement only derive context–free languages (CFLs).[21]

3.2. Left Head Adjunction

As mentioned in the introduction, it can be shown that MGs which—beside the simple merge–operation—permit only left head adjunction triggered by a corresponding strong selection feature, do only derive CFLs as well. Skipping any formal details here, we just mention that, as far as a complex head of a corresponding MG is concerned, the dependencies between the (phonetic yields of the) single lexical heads incorporated into the (phonetic yield of the) complex head and their respective "traces" are nested. This allows us to use a single stack in order to "correctly redefine" the concept of successive left head adjunction within the weakly equivalent RLIG.[22]

3.3. Right Head Adjunction

The crucial difference between successive right head adjunction and successive left head adjunction is constituted by the fact that—within a complex head created by the former derivation mode—the dependencies between the (phonetic yields of the) single lexical heads incorporated into the (phonetic yield of the) complex head and their respective "traces" are cross–serial. This kind of dependencies can be made "visible" by means of a respective specifier being attached right beyond each "trace," and containing some particular phonetic material; e.g., a copy of the lexical phonetic material of the head by which the specifier is selected as in the case of the MG G_{ww} deriving the copy language $\{ww \mid w \in \{a,b\}^*\}$, and consisting of the following 9 lexical items:[23]

$\texttt{=C =X}_a\ \texttt{X}\ a$	$\texttt{X}_a\ a$	$\texttt{=X}_a\ \texttt{X}\ a$	$\texttt{=X C}$	$\texttt{C}$
$\texttt{=C =Y}_b\ \texttt{Y}\ b$	$\texttt{Y}_b\ b$	$\texttt{=Y}_b\ \texttt{Y}\ b$	$\texttt{=Y C}$	

21. Vice versa, it is not hard to verify that each CFG is weakly equivalent to some MG of this kind. This can be proven rather straightforwardly, e.g., by starting with a CFG in Chomsky normal form.
22. Note that the type of RLIG needed does use the stack in only the following "normalized" way: once an element has been popped from the stack, the stack has to be emptied before a new element can be pushed onto the stack. This, of course, is just a reflex of the successive cyclicity by which a complex complex head is created.
23. Since lexical items are always simple expression, we will usually identify each such item with its head–label. Note further that, referring to Cornell (1996), the example MG G_{ww} is also given in Stabler (1997).

3.4. Arbitrary Head Adjunction

An MG which derives the non–CFL $\{a^n b^n d^n \,|\, n \geq 0\}$ by means of mixed successive head adjunction is the MG $G_{a^n b^n d^n}$ from below. We see that, at the same time, while $G_{a^n b^n d^n}$ derives cross–serial dependencies between a's and d's by means of successive right head adjunction analogously to the way exploited by G_{ww}, $G_{a^n b^n d^n}$ additionally derives nested dependencies between a's and b'S as well as between b's and d's. Since these additional nested dependencies are derived by "stepwise intervening" left head adjunction this suggests that a language like $a^n b^n d^n$ is not derivable by an MG which uses only right head adjunction. The MG $G_{a^n b^n d^n}$ consists of the following 6 lexical items:

$\mathsf{X}^{=}\,{}^{=}\mathsf{z}\;\mathsf{y}\;b \qquad \mathsf{x}\;a \qquad \mathsf{z}\;d \qquad {}^{=}\mathsf{Y}\;\mathsf{x}\;a \qquad \mathsf{Y}^{=}\;\mathsf{c} \qquad \mathsf{c}$

4. Simple Phrasal Movement

Assume G_{MG} to be an MG such that each selection feature that occurs within the label of some lexical entry is weak, and such that the set of licensees is a singleton set. Thus, G_{MG} does not allow any kind of head movement, but an "indistinguishable" type of phrasal movement. Again, we will skip formal details, when arguing that the language derivable by G_{MG} is a CFL, and briefly sketch the crucial point here.

Suppose that an expression ϕ is selected by another expression υ, yielding an expression $\tau = \textit{merge}(\upsilon, \phi)$ such that the head–label of ϕ', the subtree of τ resulting from ϕ,[24] starts with an (unchecked) licensee instance. We could additionally "store" this information in the head–label of τ with the intend of preventing τ from being merged with another expression such that the resulting tree would contain two different maximal projections with an unchecked licensee instance. More generally, we could additionally "store" the head–label of ϕ within the head–label of τ; and within the head–label of each expression subsequently derived from τ we could not only store the head–label of ϕ, but also the number of the still unchecked licensee instances introduced by this head–label as long as there is still such an unchecked licensee instance. This would enable us to postpone the "actual insertion of ϕ" until it has reached its final landing site.

The last considerations, indeed, provide us rather rather straightforwardly with a method of constructing a weakly equivalent CFG for G_{MG}. This, at least, is true if we take into account only expressions being derivable from the lexicon by means of *merge* and *move*, and serving to derive a complete: first, it should be mentioned that there is only a finite number of possibilities for an extended head–labeling in the outlined way.[25] But the crucial reason why such a construction becomes finally possible is that each expression τ derivable from the lexicon of G_{MG}, and serving to derive a complete expression contains at most one maximal projection with an unchecked instance of some licensee feature starting the head-label, since the cardinality of the licensee set is 1.[26] That is to say, whenever we predict, in virtual terms of our weakly equivalent CFG, a specifier position to be the landing site of some maximal projection τ_1, we do not have to worry about the possibility that τ_1 in its turn contains a "trace" which has arisen from extracting some maximal projection τ_1' out of τ_1. Such a configuration cannot appear under any circumstances.

Let us also note here that—in a second step—it is possible to directly combine each of the converting methods presented in Section 3 with the just mentioned one in order to prove that the weak generative capacity of none of the considered subclasses of MGs allowing different types of head movement is increased if, additionally, the set of licensees is allowed to contain a single element.

5. Beyond simple phrasal movement

We conclude by emphasizing that, on the other hand, phrasal movement in the sense of the MG–definition which arises from the interaction of two different licensees already permits us to derive languages not derivable by any TAG. As an example of a corresponding MG we finally present the MG $G_{\text{NON-LIL}}$ deriving the language $\{a_1^n b_1^n a_2^m b_2^m c_1^n d_1^n c_2^m d_2^m \,|\, m, n \geq 0\}$, and consisting of the following 17 lexical items:

24. That is, ϕ' is either the complement or the highest specifier of τ.
25. Recall that the MG lexicon is finite, that each label of a lexical head is a finite string of features, and that an MG builds structure exclusively by canceling particular feature instances of these labels after an lexical head has been introduced in the course of a derivation.
26. Recall that, due to the definition of *move*, the implementation of the shortest move constraint (SMC) within MGs allows at most one such maximal projection for each different licensee in order to let an expression occur in the derive a complete expression.

$\mathtt{x}_{b_1}\ b_1$		$^{=}\mathtt{x}_{a_1}\ \mathtt{x}_{d_1}\ d_1$	$^{=}\mathtt{x}_{d_1}\ \mathtt{y}_1\ c_1$	$\mathtt{y}_1\ \mathtt{-l}_1$
$^{=}\mathtt{x}_{c_1}\ \mathtt{+L}_1\ \mathtt{x}_{b_1}\ b_1$	$^{=}\mathtt{x}_{b_1}\ \mathtt{x}_{a_1}\ \mathtt{-l}_1\ a_1$	$^{=}\mathtt{x}_{a_1}\ \mathtt{+L}_2\ \mathtt{x}_{d_1}\ d_1$	$^{=}\mathtt{x}_{d_1}\ \mathtt{x}_{c_1}\ \mathtt{-l}_2\ c_1$	
$\mathtt{x}_{b_2}\ b_2$		$^{=}\mathtt{x}_{a_2}\ \mathtt{x}_{d_2}\ d_2$	$^{=}\mathtt{x}_{d_2}\ \mathtt{y}_2\ c_2$	$\mathtt{y}_2\ \mathtt{-l}_2$
$^{=}\mathtt{x}_{c_2}\ \mathtt{+L}_2\ \mathtt{x}_{b_2}\ b_2$	$^{=}\mathtt{x}_{b_2}\ \mathtt{x}_{a_2}\ \mathtt{-l}_2\ a_2$	$^{=}\mathtt{x}_{a_2}\ \mathtt{+L}_1\ \mathtt{x}_{d_2}\ d_2$	$^{=}\mathtt{x}_{d_2}\ \mathtt{x}_{c_2}\ \mathtt{-l}_1\ c_2$	
				$^{=}\mathtt{y}_2\ ^{=}\mathtt{y}_1\ \mathtt{+L}_2\ \mathtt{+L}_1\ \mathtt{c}$

References

Chomsky, Noam. 1995a. Categories and Transformations. In Chomsky, 1995b, pages 219–394.

Chomsky, Noam. 1995b. *The Minimalist Program*. Cambridge, MA: MIT Press.

Cornell, Thomas L.˙ 1996. A Minimalist Grammar for Deriving the Copy Language. Report No. 79, Working papers of the SFB 340, Tübingen University, Tübingen. Available at `http://www.sfs.nphil.uni-tuebingen.de/sfb/reports/`.

de Groote, Philippe, Glyn Morrill and Christian Retoré, editors. 2001. *Logical Aspects of Computational Linguistics (LACL ’01)*, Lecture Notes in Artificial Intelligence Vol. 2099. Berlin, Heidelberg: Springer.

Harkema, Henk. 2000. A Recognizer for Minimalist Grammars. In *Proceedings of the Sixth International Workshop on Parsing Technologies (IWPT 2000)*, Trento, pages 111–122.

Harkema, Henk. 2001. A Characterization of Minimalist Languages. In de Groote, Morrill and Retoré, 2001, pages 193–211.

Joshi, Aravind K.˙ 2000. Relationship Between Strong and Weak Generative Power of Formal Systems. In *Proceedings of the 5th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+5)*, Paris, pages 107–114.

Kayne, Richard S.˙ 1998. Overt vs. covert movement. *Syntax*, 1:128–191.

Kayne, Richard S.˙ 1999. Prepositional complementizers as attractors. *Probus*, 11:39–73.

Koopman, Hilda and Anna Szabolcsi. 2000. *Verbal Complexes*. Cambridge, MA: MIT Press.

Mahajan, Anoop. 2000. Eliminating Head Movement. In *GLOW Newsletter #44*, pages 44–45. Abstract of the talk held at *the 23rd Generative Linguistics in the Old World Conference (GLOW 2000)*, Vitoria–Gasteiz/Bilbao.

Michaelis, Jens. 2001a. Derivational Minimalism is Mildly Context–Sensitive. In M. Moortgat, editor, *Logical Aspects of Computational Linguistics (LACL ’98)*, Lecture Notes in Artificial Intelligence Vol. 2014. Springer, Berlin, Heidelberg, pages 179–198.

Michaelis, Jens. 2001b. Transforming Linear Context–Free Rewriting Systems into Minimalist Grammars. In de Groote *et al.* (2001), pages 228–244. Also available at `http://www.ling.uni-potsdam.de/~michael/papers.html`.

Michaelis, Jens and Christian Wartena. 1999. LIGs with Reduced Derivation Sets. In Gosse Bouma, Geert–Jan M. Kruijff, Erhard Hinrichs and Richard T. Oehrle, editors, *Constraints and Resources in Natural Language Syntax and Semantics*. CSLI Publications, Stanford, CA, pages 263–279.

Rambow, Owen and Giorgio Satta. 1999. Independent Parallelism in Finite Copying Parallel Rewriting Systems. *Theoretical Computer Science*, 223:87–120.

Schabes, Yves and Richard C. Waters. 1995. Tree Insertion Grammar: A Cubic–time Parsable Formalism that Lexicalizes Context–Free Grammar Without Changing the Trees Produced. *Computational Linguistics*, 21:479–513.

Seki, Hiroyuki, Takashi Matsumura, Mamoru Fujii and Tadao Kasami. 1991. On multiple context–free grammars. *Theoretical Computer Science*, 88:191–229.

Stabler, Edward P.˙ 1997. Derivational Minimalism. In C. Retoré, editor, *Logical Aspects of Computational Linguistics (LACL ’96)*, Lecture Notes in Artificial Intelligence Vol. 1328. Springer, Berlin, Heidelberg, pages 68–95.

Stabler, Edward P.˙ 1998. Acquiring Languages with Movement. *Syntax*, 1:72–97.

Stabler, Edward P.˙ 2001. Recognizing Head Movement. In de Groote *et al.* (2001), pages 245–260. Draft available at `http://www.humnet.ucla.edu/humnet/linguistics/people/stabler/`.

Stabler, Edward P. and Edward L. Keenan. 2000. Structural Similarity. In *Algebraic Methods in Language Processing.* Proceedings of the 16th Twente Workshop on Language Technology (TWLT 16) joint with the 2nd AMAST Workshop on Language Processing, Iowa City, IA, pages 251–265.

Vijay–Shanker, K. and David J. Weir. 1994. The equivalence of four extensions of context–free grammars. *Mathematical Systems Theory*, 27:511–546.

Vijay–Shanker, K. David J. Weir and Aravind K. Joshi. 1987. Characterizing Structural Descriptions Produced By Various Grammatical Formalisms. In *25th Annual Meeting of the Association for Computational Linguistics (ACL ’87)*, Stanford, CA, pages 104–111. ACL.

Wartena, Christian. 1999. *Storage Structures and Conditions on Movement in Natural Language Syntax*. Ph.D. thesis, Potsdam University, Potsdam. Available at `http://www.ling.uni-potsdam.de/~wartena/`.

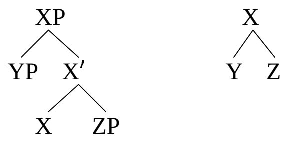
Weir, David J.˙ 1988. *Characterizing Mildly Context–Sensitive Grammar Formalisms*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA.

Learning Mirror Theory

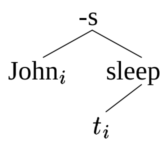
Gregory M. Kobele, Travis Collier, Charles Taylor, Edward P. Stabler
University of California, Los Angeles

1. Mirror Theory

Mirror Theory is a syntactic framework developed in (Brody, 1997), where it is offered as a consequence of eliminating purported redundancies in Chomsky's minimalism (Chomsky, 1995). A fundamental feature of Mirror Theory is its requirement that the syntactic head-complement relation mirror certain morphological relations (such as constituency). This requirement constrains the types of syntactic structures that can express a given phrase; the morphological constituency of the phrase determines part of the syntactic constituency, thereby ruling out other, weakly equivalent, alternatives. A less fundamental, but superficially very noticeable feature is the elimination of phrasal projection. Thus the X-bar structure on the left becomes the mirror theoretic structure on the right:



(Brody, 1997) calls this systematic collapse of X, X' and XP nodes 'telescope'. Every node may now have phonetic content, and children are identified as specifiers or complements depending on their direction of branching; left-daughters are specifiers and right-daughters are complements (previously, specifiers were children of XP, and complements were children of X'). Furthermore, the complement relation is a "word-forming" relation, where according to the "mirroring" relation, the phonetic content of each head follows the phonetic content of its complement. For example, MTGs can generate trees like the following, which given the "mirror" relation between morphology and syntax, is pronounced *John sleep -s*:



1.1. Trees

A mirror theoretic tree (MTT) can be viewed as a standard binary branching tree together with two functions; one, a function f from branches to a two element set $\{right, left\}$, the other, a function g from nodes to a two element set $\{strong, weak\}$. If a is the parent of a', then a' is a specifier (or left child) of a if $f(\langle a, a'\rangle) = left$, and a complement (or right child) of a otherwise. Formally, we represent a MTT as a particular kind of tree domain:

Definition 1
A MTT $\tau = \langle T, S\rangle$ where $T, S \subseteq \{0,1\}^*$ such that

1. $S \subseteq T$
2. T is prefix closed (if $x \in T$ and $x = yz$ then $y \in T$)

Domination corresponds to the initial substring relation with x dominating y iff x is an initial substring of y. The greatest node dominating both x and y, $x \wedge y$, is their longest common initial substring. The function g from nodes to $\{strong, weak\}$ is the characteristic function of the set S:

$$g(t) = \begin{cases} strong, \text{ if } t \in S \\ weak, \text{ if } t \notin S \end{cases}$$

© 2002 Gregory M. Kobele, Travis Collier, Charles Taylor, Edward P. Stabler. *Proceedings of the Sixth International Workshop on Tree Adjoining Grammar and Related Frameworks (TAG+6)*, pp. 66–73. Universitá di Venezia.

From Definition 1 we define f from branches to $\{right, left\}$ as follows. A child is a left child if it ends in a '1', and it is a right child if it ends in a '0'.

$$f(\langle x, xn\rangle) = \begin{cases} left, \text{ if } n = 1 \\ right, \text{ if } n = 0 \end{cases}$$

As even internal nodes may have phonetic content, the standard definitions of the yield of a tree will not suffice. We want a node to be spellt out after its left subtree, and before its right subtree. We define a total order $\prec\!\!\prec$ on the nodes of T, such that $x \prec\!\!\prec y$ whenever x is visited before y in an in-order tree traversal of T. Thus $x \prec\!\!\prec y$ holds between x and y just in case one of the following is true:

1. $y \lhd^* x$ and x is in the left subtree of y
2. $x \lhd^* y$ and y is in the right subtree of x
3. $x \prec\!\!\prec x \wedge y$ and $x \wedge y \prec\!\!\prec y$

This gives us a strict SPEC - HEAD - COMP ordering. But wait. There's more. The partitioning of branches into left branches and right branches is not just to determine a reletive precedence with respect to a parent. Right branches are different from left branches in kind; a maximal sequence of right branches is what Brody (1997) calls a morphological word (MW), and a morphological word has a special status with respect to spellout - all the nodes in a MW are spellt out as a single unit. The relation $\prec\!\!\prec$ determines the relative ordering of MWs at spellout.

We define a morphological word to be a block in (the partition induced by) the equivalence relation $\approx$ defined on T in the following manner:

$$x \approx y \text{ iff } y \in x0^* \vee x \in y0^*$$

Two nodes are equated by this relation just in case one is the complement of (...the complement of) the other. As trees are binary branching, the immediate domination relation totally orders each MW. With each MW B, we associate an element $p(B) \in B$ and call $p(B)$ the spellout position of B.[1] Given two MWs B, B', every $y \in B$ is spellt out before any $z \in B'$ iff $p(B) \prec\!\!\prec p(B')$. At this point the nodes in each MW must be spellt out in a contiguous manner (as $\prec\!\!\prec$ totally orders T), but nothing has been said about the relative order in which they are spellt out. In keeping with (Brody, 1997) (but see (Kobele, forthcoming) for alternatives), we adapt Brody's mirror principle (whence '*Mirror Theory*') to our terminology:

(1) *The Mirror Principle*
if x is the complement of y then y is spellt out before x.

Thus, each MW is spellt out in 'reverse domination order' (x is spellt out before y iff $y \lhd^+ x$).

1.2. Mirror Theoretic Grammars

A formal treatment of mirror theory inspired by Minimalist Grammars (Stabler, 1997) is given in (Kobele, forthcoming), where an empirically grounded restriction of the formalism therein is shown to be weakly equivalent to MCTAGs (Joshi, 1987).[2]A mirror theoretic expression is defined to be a mirror theoretic tree along with a labeling function from the nodes of the tree to a set of labels. A label consists of a phonetic part (which is opaque to the syntax) and a finite sequence of syntactic features. A mirror theoretic grammar consists of a finite lexicon of 'basic' expressions, together with two structure building operations, $merge$ and $move$, which build expressions from others either by adjoining structures, or by displacing sub-parts of structures. Each operation is feature driven, and 'checks' features (and thus a derived expression will have fewer features than the sum total of the features of the expressions (tokens) used to derive it). The expressions generated by the grammar are those in the closure of the lexicon under the structure building functions. A complete expression is one all of whose features have been checked, save for the category feature of the root, and the string language at a particular category is simply the yields of the complete expressions of that category.

1. The element so picked out, $p(B)$, is defined in (Brody, 1997) to be the 'deepest' node, if no nodes in B are strong. If some nodes in B are strong, then $p(B)$ is the 'highest' one of the strong nodes. In other words, if $S \cap B \neq \emptyset$, then $p(B) = x$, where $\forall y \in S \cap B\ x \lhd^* y$. If $S \cap B = \emptyset$, then $p(B) = x$ such that $\forall y \in B\ y \lhd^* x$

2. Because of the "mirroring" and the relative flexibility in where MWs get spellt out in relation to the other material, even the movement-free subset of the framework defines a proper superset of the context free languages. See (Michaelis, this volume) for a discussion closely related to this issue.

Definition 2
A MTG $G = \langle \Sigma, Syn, Lex, \{merge, move\} \rangle$, where

1. Σ is a non-empty set (the pronounced elements)
2. Syn is the disjoint union of the following sets (the syntactic features):
   (a) $base$, a non-empty finite set.
   (b) $cselect = \{=b | b \in base\}$
   (c) $sselect = \{b= | b \in base\}$
   (d) $licensees = \{-b | b \in base\}$
   (e) $licensors = \{+b | b \in base\}$

   An expression is a pair $\langle \langle T, S \rangle, \mu \rangle$, where $\langle T, S \rangle$ is a MTT, and $\mu : T \to \Sigma^* \times Syn^*$ is the labelling function.
3. Lex is a finite set of expressions $\langle \langle T, S \rangle, \mu \rangle$, such that[3]
   (a) $|T| = 1$, and
   (b) $\mu : T \to \Sigma^* \times cselect^? \; (sselect + licensors)^? \; base \; licensees^*$

   The shape of the lexical labels is partly determined by the nature of MTTs (and the particular generating functions we have).[4]The precategory sequence (the features before the $base$ category) allows for up to one complement (cselection features) and up to one specifier (sselection or licensor features). Each lexical item has a category (a $base$ feature), and no more than one, as nodes in any tree have only at most one parent. There are no restrictions as to the number of licensee features - movement is ad libitum.

merge $merge$ is a function from pairs of expressions to single expressions. We divide the presentation of the function definition into two cases according to whether the merged item is merged into the specifier ($smerge$) or the complement ($cmerge$) position.

SMERGE $smerge$ is defined on the pair of expressions $\langle \langle T_1, S_1 \rangle, \mu_1 \rangle, \langle \langle T_2, S_2 \rangle, \mu_2 \rangle$ iff all of the following obtain:
- the root of T_1 has an available specifier position ($1 \notin T_1$)
- the first syntactic feature of the root of T_1 is $b=$, and
- the first syntactic feature of the root of T_2 is b

In this case, $smerge$ is defined on the pair, and it maps to the expression $\langle \langle T, S \rangle, \mu \rangle$, where
- $T = T_1 \cup 1T_2$
- $S = S_1 \cup 1S_2$
- the label of the root of T is gotten from the label of the root of T_1 by deleting the first syntactic feature
- the label of the left child of T is gotten from the label of the root of T_2 by deleting the first syntactic feature
- otherwise, for $x \in T_1$, $\mu(x) = \mu_1(x)$, and for $x \in T_2$, $\mu(1x) = \mu_2(x)$

CMERGE $cmerge$ is defined on the pair of expressions $\langle \langle T_1, S_1 \rangle, \mu_1 \rangle, \langle \langle T_2, S_2 \rangle, \mu_2 \rangle$ iff all of the following obtain:
- the root of T_1 has an available complement position ($0 \notin T_1$)
- the first syntactic feature of the root of T_1 is $=b$, and
- the first syntactic feature of the root of T_2 is b

In this case, $cmerge$ is defined on the pair, and it maps to the expression $\langle \langle T, S \rangle, \mu \rangle$, where
- $T = T_1 \cup 0T_2$
- $S = S_1 \cup 0S_2$
- the label of the root of T is gotten from the label of the root of T_1 by deleting the first syntactic feature
- the label of the right child of T is gotten from the label of the root of T_2 by deleting the first syntactic feature
- otherwise, for $x \in T_1$, $\mu(x) = \mu_1(x)$, and for $x \in T_2$, $\mu(0x) = \mu_2(x)$

3. $\Phi^?$ should be read as 'one or zero tokens of Φ'.
4. Only partly, as there is no functional reason that sselection features cannot precede cselection features. Doing so makes no difference (other than further complicating the description of a lexical label).

move $move$ is a function from expressions to expressions. $move(\langle\langle T_1, S_1\rangle, \mu_1\rangle)$ is defined whenever the following conditions obtain:

- the root of T_1 has an available specifier position
- the first syntactic feature of the root of T_1 is +b, and
- there is *exactly one* node $n \in T_1$ such that the first syntactic feature in n is -b, and, moreover, n cannot be in the same MW as the root of T_1

If the above conditions obtain, then $move(\langle\langle T_1, S_1\rangle, \mu_1\rangle)$ is defined, and is equal to $\langle\langle T, S\rangle, \mu\rangle$, which is the result of moving the subtree rooted in the least node in the MW containing n, to the specifier position of the root. Note that since n is not in the same MW as the root, we must have that $n = x10^i$ for some $i \in \mathbb{N}$. The subtree we are to move is $\langle\langle T_2, S_2\rangle, \mu_1 \upharpoonright T_2\rangle$, where $T_2 = \{y|x1y \in T_1\}$, and $S_2 = \{y|x1y \in S_1\}$. Then

- $T = 1T_2 \cup (T_1 - x1T_2)$
- $S = 1S_2 \cup (S_1 - x1S_2)$
- the label of the root of T is gotten from the label of the root of T_1 by removing the first syntactic feature
- the label of $10^i \in T$ is gotten from the label of $n = x10^i \in T_1$ by removing the first syntactic feature
- otherwise, for $z \in 1T_2$, $\mu(z) = \mu_1(xz)$, and for $z \in T - (1T_2)$, $\mu(z) = \mu_1(z)$

Given a MTG $G = \langle\Sigma, Syn, Lex, \{merge, move\}\rangle$, $L(G)$ denotes the closure of Lex under the structure building functions $merge$ and $move$. An expression is complete just in case the only node that has syntactic features is the root, and it has only a base feature. The string language of G at a category $b \in base$ ($L_b(G)$) is the set of the yields of the complete expressions whose root's unique syntactic feature is b. The mirror theoretic languages (MTLs) are the string languages of an MTG G at a category b, for some $G \in MTG$ and $b \in base_G$.

1.3. Derivations

An expression $e \in L(G)$ might have been built up in several ways from the generating functions. A derivation tree for an expression is a record of one possible sequence of steps taken to derive the expression in question from lexical items. Given a MTG G, we denote by $\Gamma(G)$ the set of all derivation trees for each expression in the language of G. $eval : \Gamma(G) \to L(G)$ is the map which takes each derivation of an expression to the expression it derives. We define $\Gamma(G)$ and $eval : \Gamma(G) \to L(G)$ by mutual recursion:

1. for each $\ell \in Lex_G$, $\ell \in \Gamma(G)$, and $eval(\ell) = \ell$
2. for $\gamma, \gamma' \in \Gamma(G)$, if $merge(eval(\gamma), eval(\gamma'))$ is defined, then $\gamma'' = \langle\gamma, \gamma'\rangle \in \Gamma(G)$ and $eval(\gamma'') = merge(eval(\gamma), eval(\gamma'))$
3. for $\gamma \in \Gamma(G)$, if $move(eval(\gamma))$ is defined, then $\gamma' = \langle\gamma\rangle \in \Gamma(G)$ and $eval(\gamma') = move(eval(\gamma))$

These structures are called derivation *trees* because it is simple to give them a (standard) tree-interpretation:

1. $\ell \in Lex_G$ denotes the tree with one node, labelled ℓ, and no branches.
2. $\langle\gamma, \gamma'\rangle \in \Gamma(G)$ denotes the tree with root labelled $\langle\gamma, \gamma'\rangle$, whose left child is the tree denoted by γ, and whose right child is the tree denoted by γ'
3. $\langle\gamma\rangle \in \Gamma(G)$ denotes the tree with root labelled $\langle\gamma\rangle$, and whose only child is the tree denoted by γ

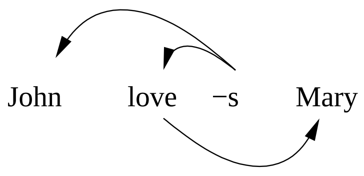
The sequence of lexical items used in a derivation γ is the yield of the tree denoted by γ, and, as shown in (Hale and Stabler, 2001), no two distinct derivations use the same sequence of lexical items.

2. Learning

Adapting a technique familiar from (Kanazawa, 1998) and others, we show that if the lexical ambiguity in target grammars is restricted, this can provide a basis for generalization from a finite sample. We describe an algorithm that identifies the class of languages generated by the rigid mirror theoretic grammars (rMTG) (grammars in which every lexical item has a unique string component) in the limit from any text of "dependency structures."

2.1. Dependancy Structures

Dependency structures show relations among the lexical items in a sentence. Information about these relations is, at least in many cases, plausibly available to the language learner (surface order, morphological decomposition and affixation (Baroni, 2000; Goldsmith, 2001) and selection relations (Siskind, 1996)). For example, imagine that upon hearing "John loves Mary" the language learner is able to infer these relations (Here, 's' is marked as a suffix (by the dash preceding it), and the arcs indicate that the source selected the target at some point in the derivation):



A dependency structure (henceforth: 'd-structure') is a tuple $\langle V, E, S, \mu, < \rangle$, where $\langle V, E \rangle$ is a directed multi-graph (i.e. $E \subseteq V \times V$ is a multi-set), $\mu : V \rightarrow \Sigma^*$ is a labeling function from vertices to phonetic strings, $S \subseteq V$ is a distinguished subset (of suffixes), and $<$ is a total ordering on V (the surface order). Intuitively, the vertices correspond to the lexical items used in a derivation, and there is one edge between two vertices for every pair of features, one from each of the two lexical items, such that the one checks the other in the course of the derivation. Formally, a d-structure d is 'for' a derivation γ just in case:[5]

1. for $s = \langle s_1, \ldots, s_n \rangle$ the sequence of lexical items used in γ, there is a sequence $v = \langle v_1, \ldots, v_n \rangle$ which enumerates without repetition the elements of V, and for $1 \leq i \leq n$, $\mu(v_i)$ is the string component of (the label of the lexical expression) s_i

2. there is a bijection from edges in E to non-leaf nodes in the derivation tree denoted by γ (or equivalently, to occurances of left brackets in $\gamma \ldots$) such that if edge $\langle v_i, v_j \rangle$ is mapped to γ', then γ' checks a syntactic feature in s_i against a syntactic feature in s_j

3. $v_i < v_j$ iff the phonetic features from s_i precede the phonetic features from s_j at spellout

$d(G) = \{d | \exists \gamma \in \Gamma(G)\ d$ is for $\gamma\}$ is the d-structure language of the MTG G.

Given a d-structure $\langle V, E, S, \mu, < \rangle$, we define the following notions which we will use in the description of the learning algorithm:

- vEv' just in case there is an edge from v to v'. We write vE^+v' in case there is a finite sequence of vertices $v_1, \ldots, v_{n+1}$ such that v_iEv_{i+1}, $v = v_1$, and $v' = v_{n+1}$.

- $<_E$ is a partial ordering of E such that $a <_E a'$ iff $a = \langle v, v' \rangle$, $a' = \langle v'', v' \rangle$ and $v''E^+v$

- $v <^1 v'$ (v immediately precedes v') iff $v < v'$ and no vertex follows v and precedes v'

- an arc $\langle v, v' \rangle$ is a $cmerge$ arc iff vEv', $v \in S$, and $v' <^1 v$

- an arc $\langle v, v' \rangle$ is a $move$ arc iff vEv', and $\exists v''\ vE^+v'' \& v''Ev'$

- an arc $\langle v, v' \rangle$ is a $smerge$ arc iff vEv', and it is neither a $cmerge$ arc nor a $move$ arc

- a vertex v is the surface specifier of a vertex v' iff $v'Ev$, $\neg\exists v''\ v''E^+v' \& v''Ev$, and $\langle v', v \rangle$ is not a $cmerge$ arc

- a vertex v has been shown weak iff there is a sequence $a_1, \ldots, a_n$ of $cmerge$ arcs such that $a_1 = \langle v, t_1 \rangle$, $a_i = \langle t_{i-1}, t_i \rangle$, and there is some v' such that $v' < v$ and v' is the surface specifier of t_n. Intuitively, v has been shown weak just in case it is pronounced after some specifier it mediately dominates.

5. There could be more than one derivation a d-structure is 'for' in any given MTG.

2.2. Learning

We work within the learning paradigm established in (Gold, 1967). There, a learner is a function from finite sequences of sentences to grammars for languages. When the learner is presented with a sequence of sentences, she makes a guess as to the language that these sentences are from (in the form of a grammar). A learner *converges* on an infinite sequence of sentences s just in case there is some finite i such that for all $j > i$, $\phi(s_1, \ldots, s_i)$ is (some variant of) the learner's guess on the sequence of the first i sentences in s is the same as her guess on the sequence of the first j sentences in s, namely, G. She *identifies* a language L (a set of sentences) *in the limit* iff on every infinite sequence enumerating the sentences of L she converges to some grammar G for L (possibly different grammars for different sequences). A learner ϕ identifies a class of languages $\mathcal{L}$ in the limit iff ϕ identifies every $L \in \mathcal{L}$ in the limit. The question we address here is whether the class of rigid d-structure languages ($d(rMTG) = \{d(G)|G \in rMTG\}$) is identifiable in the limit. Our result that this class is indeed indentifiable in the limit relies on a result by Angluin (1980) which shows that a class of languages $\mathcal{L}$ is identifiable in the limit iff for every $L \in \mathcal{L}$ there is a finite subset $D_L \subseteq L$ such that no other $L' \in \mathcal{L}$ can both contain D_L and be properly contained by L. We describe the construction of such a set for each $L \in d(rMTG)$, and show that it has these properties.

First we introduce some concepts that will help us in this section. A substitution is any total function θ over the set of base features that fixes the start category. A grammar G is an instance of a grammar G' ($G' \sqsubseteq G$) iff there is some substitution θ such that for each lexical expression ℓ in G, the result of applying the substitution to every feature in the label of ℓ (where θ 'commutes' with the complex features (e.g. $\theta(+b) = +(\theta(b))$) is some lexical item $\theta(\ell)$ in G'. G and G' are alphabetic variants of one another ($G \square G'$) iff they are variants of each other. A grammar G is *reduced* in the sense of (Kanazawa, 1998) iff there is no G' such that $d(G') = d(G)$ and $G' \sqsubset G$. One way to think of this is to read the $\sqsubseteq$ relation as 'makes more category distinctions than' (in the sense of '$=aa$' makes fewer category distinctions than '$=ab$'). Then a grammar is reduced iff you can't make more category distinctions and still derive the same language. For example, in a reduced grammar no element $b \in base$ occurs as both a selector/base feature ($=b$, $b=$, b) and as a licensee/or feature ($+b$, $-b$). This is because movement and selection features never select the same feature. Thus, one can 'rename' all occurances of the selection features with distinct names without changing the expresivity of the grammar. In the remainder of this paper we will be focussing on reduced rigid mirror theoretic grammars (rrMTGs). This change of perspective serves to simplify discussion, and does not alter the class of languages to be learned.

The idea behind the construction of the sets D_L is to constrain as much as possible the grammars capable of generating supersets of D_L. As MTGs differ only in their lexical inventories, we do this by putting information about the lexicon of a grammar that generates L into D_L. Given a dependency structure d of a derivation γ in which lexical item ℓ occurs, we can reconstruct not only which types of features ℓ has ($sselect$, $cselect$, $licensor$, ...), but also the order in which they occur (so if the syntactic features of ℓ are $b=\ c\ -d\ -a$, we can determine that ℓ begins with a sselect feature of some sort, followed by some base feature, followed by two licensee features of some kind). To be able to determine *which* features of the given type ℓ has, we need to also add information about what other features each feature of ℓ can check/be checked by.

We associate with each rrMTG G a finite set $D_G \subseteq d(G)$, such that

1. for each lexical item ℓ in G, D_G contains a d-structure containing ℓ, if one exists
2. for each weak lexical item ℓ in G, D_G contains a d-structure which witnesses ℓ's weakness (a d-structure in which ℓ is shown weak), if one exists
3. for each selector or licensor feature $x(f)$ on every lexical item ℓ in G, for every feature f on each additional lexical item ℓ', D_G contains a d-structure of a derivation in which the feature $x(f)$ on ℓ checks f on ℓ', if one exists

Now we quickly outline a proof that the rigid MTGs are in fact learnable.

Lemma 1 Let $G, G' \in rrMTG$ such that $D_G \subseteq d(G') \subseteq d(G)$. Then the lexicons of G and G' are identical up to renaming of the syntactic features modulo the strength of their lexical items.

Proof Sketch: By the first clause in the definition of D_G, $Lex_{G'}$ and Lex_G have the same lexical items with respect to the string component, and the sequence of syntactic types, as $d(G') \supseteq D_G$, and $d(G) \supseteq d(G') \supseteq D_{G'}$. By the third clause in the definition of D_G, every feature of a lexical item has an example in D_G of every feature it can combine with in the course of a derivation. As $D_G \subseteq d(G')$, $d(G')$ must at least give the same feature to those elements which can combine with one another, and as $d(G') \subset d(G)$, $d(G')$ must not unify category features more than $d(G)$. As $d(G)$ is reduced, it does not unify syntactic categories beyond what is recorded in D_G. □

Lemma 2 Given $G, G' \in rrMTG$ such that the lexicons of G and G' differ only in the strength they assign to the lexical item ℓ, if ℓ is not shown to be weak, then $d(G) = d(G')$.

Proof Sketch: By changing the strength of a node to strong, one ensures only that it is not pronounced after any surface specifiers of nodes further down in its morphological word, in any derivation. But as ℓ is never shown to be weak, there is no derivation γ where the surface specifier of a node in ℓ's MW which ℓ properly dominates, precedes ℓ at spellout. □

Theorem 1 Let $G \in rMTG$. For any $G' \in rMTG$, if $D_G \subseteq d(G')$, then $d(G')$ is not a proper subset of $d(G)$.

Proof Sketch: Let $D_G \subseteq d(G')$, and assume $d(G') \subseteq d(G)$. We show $d(G') = d(G)$. As $D_G \subseteq d(G') \subseteq d(G)$, the lexicons of G and G' are identical save possibly for the strength of their lexical items (Lemma 1). Let $\ell \in Lex_G$ and $\ell' \in Lex_{G'}$ be identical except perhaps for their strength. We show that neither ℓ nor ℓ' can be shown weak independantly of the other, and thus the lexicons of G and G' agree on any lexical items that are shown weak. The conclusion then follows from Lemma 2. If ℓ were shown weak in G, then some d-structure would be a witness to it in $D_G \subseteq d(G')$, whereby ℓ' would be shown weak as well in $d(G')$. If ℓ' were shown weak in G', then, as by hypothesis $d(G') \subseteq d(G)$, ℓ would also be shown weak in G. □

Corollary 1 The class of rigid mirror theoretic languages is identifiable in the limit from texts of dependency structures.

On a finite sequence $t = \langle d_1, \ldots, d_i \rangle$ of d-structures, our algorithm first constructs a 'general form' grammar, $GF(t)$, assigning to each phonetic string in each dependency structure a unique syntactic category sequence. This grammar is not normally rigid, and does not generalize (i.e. the language it guesses contains exactly the sentences it has seen). We then unify lexical items in $GF(t)$ to get a reduced rigid grammar, $RG(t)$.

The idea behind the learning algorithm is that, given a d-structure d, we can almost exactly reconstruct the derivation γ that d is of - we can not normally determine the strength of lexical items used in γ. Our learner, when determining the strength of a lexical expression, assumes it to be strong unless there is evidence to the contrary. During the unification process, if we have two lexical items which differ only in whether they are strong, we unify them as though they were both weak (as all weak features are data-driven).

Input: a sequence of d-structures $t = \langle d_1, \ldots, d_i \rangle$ of some rigid MTG G

11. let T be the base feature of the root of each of $d_1, \ldots, d_i$
2. for each non-root node we construct the base + post-base feature sequences, with one feature per incoming arc, as follows:
   (a) with the head of the least incoming arc in we associate the feature f, where f is a new, unique base feature
   (b) with the head of each subsequent incoming arc, in turn, we associate -f, where f is a new, unique base feature
3. we then construct the pre-base sequence, associating with the tail of each outgoing arc one feature as follows:
   (a) if the head of the arc is associated with a feature -f, we associate the feature +f with the tail
   (b) if the head of the arc is associated with a feature f, then
   - if the tail of the arc is a suffix, and if the head of the arc is ordered immediately before the tail by $<$, the arc is a $cmerge$ arc, and we associate the feature =f with its tail
   - otherwise, the arc is an $smerge$ arc, and associate the feature f= with its tail

2. Collect the lexical categories from $d_1, \ldots, d_i$, making a lexical item weak if it has been shown so, and strong otherwise to get an MTG $GF(t)$

3. Unify the categories assigned to each vocabulary element to get a reduced rigid MTG, $RG(t)$, resolving strength conflicts in favour of weak features

References

Angluin, Dana. 1980. Inductive inference of formal languages from positive data. *Information and Control*, 45:117–135.

Baroni, Marco. 2000. Distributional Cues in Morpheme Discovery: A Computational Model and Empirical Evidence. UCLA, dissertation.

Brody, Michael. 1997. Mirror Theory. ms. University College London.

Chomsky, Noam. 1995. *The Minimalist Program*. Cambridge, Massachusetts: MIT Press.

Gold, E. Mark. 1967. Language identification in the limit. *Information and Control*, 10:447–474.

Goldsmith, John. 2001. Unsupervised Learning of the Morphology of a Natural Language. *Computational Linguistics*, 27:153–198.

Hale, John and Edward P. Stabler. 2001. Notes on Unique Readability. ms. UCLA.

Joshi, Aravind K. 1987. An Introduction to Tree Adjoining Grammars. In A. Manaster-Ramer, editor, *Mathematics of Language*. John Benjamins, Amsterdam.

Kanazawa, Makoto. 1998. *Learnable Classes of Categorial Grammars*. Stanford University.: CSLI Publications.

Kobele, Gregory M. forthcoming. Formalizing Mirror Theory. UCLA.

Michaelis, Jens. 2002. Notes on the complexity of complex heads in a minimalist grammar. In *Proceedings of the Sixth International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+6)*, Venezia.

Siskind, Jeffrey M. 1996. A Computational Study of Cross-Situational Techniques for Learning Word-to-Meaning Mappings. *Cognition*, 61:39–91.

Stabler, Edward P. 1997. Derivational minimalism. In Christian Retoré, editor, *Logical Aspects of Computational Linguistics*. Springer-Verlag (Lecture Notes in Computer Science 1328), NY, pages 68–95.

Defining a Lexicalized Context-Free Grammar for a Subdomain of Portuguese Language

Cinthyan Renata Sachs C. de Barbosa
Electronic Engineering and Computation Program of the Aeronautics Technological Institute (ITA) and Computer Sciences
Davidson Cury
Informatic Institute of the Federal University of Espírito Santo (UFES), Brazil.
José Mauro Volkmer de Castilho (*in memorian*)
Informatic Technology Institute of the Federal University of Rio Grande do Sul (UFRGS), Brazil.
Celso de Renna e Souza
Computer Science Division of the Aeronautics Technological Institute (ITA), Brazil.

1. Introduction

According to [1], an emphasis must be given to grammar representations for describing and generating the sentences that make up a given language and an emphasis on processing and computation that demonstrate that the grammar for a language has important implications as to how it can be processed. Some observations by [2] and [3], say that a general-purpose grammars should be linguistically plausive, both to be extensible and to take advantage of work in computational linguistics. Linguistic theories have become much more computationally oriented recently, and some large, general-purpose grammars are now available [4]. But the grammars have to be sufficiently robust to cope gracefully with sentence fragments and ill-formed input. It is also very important in the database context to have a good treatment of proper names, domain-specific identifiers, abbreviations and other items which cannot be part of a general-purpose lexicon [2].

A NLP system's success depends on its knowledge of the application domain, namely the relative completeness of the natural language model it encapsulates and the appropriateness or efficiency of its algorithms [5]. In the specific case of the NLs, the study of context-free languages (CFL) has been of special interest because they permit a simple representation of the syntax, adequate for formal structuring, as for computational analysis [6]. Context-free grammars (CFGs) have been a well accepted framework for computational linguistics for a long time [7], [8]. The recognizer algorithms and generator algorithms that implement CFLs are relatively simple and have a good efficiency. In addition, according to [9], it was the CFGs with some restrictions that had greater progress in the description of NL.

According to [7], lexicalization is important, from a computational perspective, because, other things being equal, lexicalized grammars can often be parsed much more efficiently than non-lexicalized ones. In [10] it is affirmed that this can be done associating each elementary structure in a grammar with a lexical item (terminal symbol in the context of formal grammars). A type of lexicalization is Greibach Normal Form (GNF) for Context-Free Grammar (CFG). In contrast to GNF, that is regarded as a kind of *weak* lexicalization, for not preserving structure of the original, Lexicalized Grammar can be considered as a *stronger* version of GNF, in the sense that the structures are preserved and not just the string sets (weak generative capacity). Lexicalization is of interest from a linguistic perspective, because most current linguistic theories give lexical accounts of a number of phenomena that used to be considered purely syntactic [7]. The information put in the lexicon is thereby increased in both amount and complexity [10]. According to [10], some of the linguistic formalisms illustrating the increased use of lexical information are, lexical rules in Lexical-Functional Grammar (LFG), Generalized Phrase Structure Grammar (GPSG), Head-Driven Phrase Structure Grammar (HPSG), Combinatory Categorial Grammar, Karttunen's version of Categorial Grammar, some versions of Chomsky's Government Binding Theory (GB theory), and Lexicon-Grammars.

Every method for lexicalized CFGs in the strong sense defined has required context-sensitive operations. As a result, every method for lexicalizing CFGs has shared with LTAG the unfortunable feature that lexicalization leads to dramatically decreased rather than increased computational performance [7]. According to [8], although LTAG is considered to be interesting, it is executed at the cost of decreased efficiency, $O(n^6)$-time in the worst case [11], [12], [13]. As a result [7], there are no computational advantages in lexicalizing a CFG using LTAG because of the speed up due to the grammar becoming lexicalized being swamped by the dramatic increase in fundamental worst case cost. So, we will focus on Lexicalized Context-Free Grammars (LCFG), a class of grammars originally introduced in [14]. They are attractive because they combine the elegance of LTAG and the efficiency of CFG [14] and, it is also possible through an algorithm [8] to show how the $O(n^3)$-time worst case complexity can be achieved for LCFG (n is the length of sentence). Through this formalism, constructions of CFGs for the definition of the real constructions of an interface, will be dealt with.

In this work the medical area was chosen, with emphasis on *a portuguese grammar generating a medical language for radiological queries* over a set of radiographs of a Ewing sarcom case. The treated language subset includes active and passive voices, relative and interrogative clauses, their combinations and pronouns. The language generated by this grammar was validated from knowledge obtained in the FLAMA (*Tools and Authoring Language for Modeling Process*). This framework had as objective a set of tools and an authoring language that constitutes the framework of a highly specialized architecture for authoring activities [15], [16]. FLAMA was based on an archetypical reference RUI environment (*Representation for Understanding Images*) for the learning of radiology [17]. The access to knowledge contained in the knowledge base will be done through radiological language which *grammar* is the focus of investigation through this work. We are developing a correlated work dealing with parsing techniques for our grammar.

2. Radiological Grammar

Gs:
1) S’ Comp S / Comp S_q / Comp PP_q
2) S_q Pro S
3) S NP (M) VP / VP / ProReto VP
4) S_r Rel S
5) NP (Det) (Mod) N (Mod) (W)
6) W (W) Mod
7) NP (Det) (Mod) Ncom (Mod) (W)
8) NP NP Conec NP (X)
9) X (X) Conec NP
10) NP NP S_r
11) Det (Predet) Detbase (Pósdet)
12) Mod AP / PP
13) N N S_r
14) Ncom Ncom S_r
15) VP (Neg) V_{tr} (Intens) NP (SP_A) (Y) /
(Neg) V_{tr} OblAt (Intens) (SP_A) (Y)
16) Y (Y) PP
17) VP (Neg) V_{tr} (Intens) SP_C (SP_A) (Y)
18) VP (Neg) V_{tr} (proap) (Intens) NP SP_C (SP_A) (Y) /
(Neg) V_{tr} (proap) (Intens) SP_C NP (SP_A) (Y) /
(Neg) V_{tr} OblAt (Intens) NP (SP_A) (Y) /
(Neg) OblAt V_{tr} (Intens) NP (SP_A) (Y) /
(Neg) V_{tr} RefCli (Intens) (SP_A) (Y) /
(Neg) V_{tr} (Intens) NP SP_R (SP_A) (Y) /
(Neg) V_{tr} (Intens) SP_R NP (SP_A) (Y)
19) VP (Neg) V_{tr} (Intens) SP_C SP_C (SP_A) (Y)
20) VP (Neg) V_{intr} (Intens) (SP_A) (Y)
21) SP_C P NP / P OblTon
22) SP_R P RefObl
23) VP (Neg) V_E AP (Y)
24) VP (Neg) V_E NP (Y)
25) VP (Neg) V_E PP (Y)
26) VP (Neg) V_E V_{par} (NP) (por NP) (PP)
27) VP (Neg) V_{estar} V_{ger} (Y)
28) V_{tr} (Neg) V_{tr} S_r
29) Comp $\underline{+}wh$
30) PP P NP
31) PP_q P_q Pro
32) PP Adv
33) AP (Intens) (SP_A) Adj (SP_C)
34) AP (Intens) (SP_A) Adjcom (SP_C)
35) AP AP Conec AP (Z)
36) Z (Z) Conec AP
37) Adj Adj S_r
38) Adjcom Adjcom S_r

FIGURE 2.1 - Surface Grammar

A set of rules that identify references to the various kinds of diagnosis and symptoms are needed. Through Portuguese Transformational Grammar (GT) [3], [18], [19] it was possible to work the proposed domain for the following kinds of constructions: **grammatical groups** as sentence, noun phrase (e.g. “*the exam*”), verb phrase (e.g. “*has some differential diagnosis*”), adjective phrase (e.g. “*the osteomyelitis* <u>*is chronic*</u>”), prepositional phrase (e.g. “*the osseous texture is* <u>*of diffuse form*</u>”), adverb phrase, restrictive clause (e.g. “*the parameters* <u>*that were analyzed led to osteomyelitis*</u>”), **yes/no sentences** (e.g. “*The femur is reduced?*”), ***wh*-sentences** (e.g. “*What is the patient age?*”), **alternative sentences** in the usual form (not cleaved) (e.g. “*appear lithic or blastic lesion?*"), **sentences of solicitation of explanation** (e.g. “*Why* S?”), **existential S** (e.g. “*Some region was examined?*”), **S in the active voice** (e.g. “*The lesion compromise some region?*”), **S in the passive voice** (e.g. “*Some region is compromised by lesion?*”), **cleaved S** (e.g. “*Is the exam that confirmed the diagnosis?*”). A grammar was proposed (fig.2.1) which foresees the treatment of expressions of *specific vocabulary* (words, expressions and medical jargon [20]) utilized in radiology. A meticulous study of TG and of Transformation Rules for portuguese [21] was necessary for constructing a *surface grammar* (in TG the rewriting rules generate the *deep structure* of the sentence) that treats several syntactical aspects of radiological language. Next the relationship between the surface grammar and LCFG, will be shown.

3 Lexicalization of LCFG

In this section, we propose to extend the domain of locality of Radiological CFG in order to make lexical items appear local to the production rules. The domain of locality of a CFG is extended by using a tree rewriting system that uses substitution and a restricted form of adjunction. So, the syntactical relations described in the FCG presented in [21] are needed also in LCFGs. However, these relations are expressed by trees (*initial* and *auxiliary* trees) and operations above. In [22], it is affirmed that, due to formal properties of the adjunction, the formalism utilized becomes more powerful than FCGs. Not every grammar is in a lexicalized form. Given a grammar *G* stated in a formalism, we will try to find another grammar G_{lex} (not necessarily stated in the same formalism) that generates the same language and also the same tree set as *G* and for which the lexicalized property holds. We refer to this process as *lexicalization* of a grammar. We say that a formalism *F* can be lexicalized by another formalism *F'*, if for any finitely ambiguous grammar *G* in *F* there is a grammar *G'* in *F'* such that *G'* is a lexicalized grammar and such that *G* and *G'* generate the same tree set (and a fortiori the same language) [10]. Taking our *surface grammar* [21], [23] it is possible to map LCFG, permitting in the last representation an improvement in the computational performance. According to [7], the presence of lexical item as its leftmost non-empty constituent in LCFG facilitates efficient left to right parsing. The main reason motivate the construction of an CFG (fig.2.1) is due to number of initial and auxiliary trees in LCFG can be, in the worst case, much greater than the number of production rules in a FCG (in this case the surface grammar). The number of elementary trees in G_{lex} is related to the number of acyclic and minimal cycle paths in LG (*lexicalization graph*). More details in [7]. By using the *theorem* exhibited in [8], it is possible to transform the surface grammar into a LCFG, by means of a special artifice in the treatment of terminals, obtaining consequently, as in the surface grammar, valid constructions for the radiological language.

In linguistic context, according to [24], the nodes on the frontier in LCFG are *preterminal* lexical category symbols such as N (Noun), V (Verb), etc. For denoting these categories, the artifice utilized in this work will be the insertion of *pseudoterminals* in the surface grammar so that the theorem can be applied. The pseudoterminals, denoted by detbase, n, v, etc. (all denoted in small letter) will be in the productions of the preterminals as in N□ n, V □ v, etc. After applying the theorem the previous preterminals and these pseudoterminals will be marked with □, as seen in fig.3.1, indicating that its pseudoterminal child node, will have to activate a search in a dictionary for verifying if it is possible to use, in this point, the token from the input string. If possible, the pseudoterminal will be substituted by the terminal at issue. The lexical items, utilized in the radiological language, are in a dictionary, which is exhibited in [21].

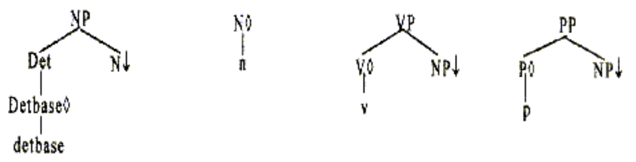


FIGURE 3.1 - Examples of trees before lexicalization

The principal unit of syntactic information associated with a LCFG entry is a tree structure in which the tree nodes are labeled with syntactic categories and feature information and there is at least one leaf node labeled with a *lexical* category (such lexical leaf nodes are known as *anchors*). Thus, the *lexicon* consists of a finite set of structure each associated with an anchor. The structures defined by the lexicon are called *elementary structures*. Structures built by combination of others are called *derived structures* [10]. With respect to a LCFG lexicon, this can be defined as the LTAG lexicon, for example, where this consists of a set of trees each one associated with one or more lexical items [14]. These elementary trees can be viewed as elementary clauses (including their transformational variants) in which the lexical items participate. The trees are combined by *substitution* and a restricted form of *adjunction* that is context-free in nature.

Fig.3.2 exhibits some of the constructions LCFG utilized in radiological domain. In grammars as LCFG, and others of the TAG family, the linguistic unit is this elementary tree, which corresponds to a minimal predicative structure. According to [22], these structures are syntactic and semantic units at the same time. They are gathered in tree families which encode the different lexical and syntactic rules which may apply to them [25]. When trees are combined by substitution or adjunction, the corresponding semantic representations are combined. In [24] focus that the elementary trees are the appropriate domains for characterizing certain dependencies (e.g., subcategorization and filler-gap dependencies). That is, an elementary tree localizes agreement dependencies, filler-gap dependencies and predicate-argument dependencies and also serves as a complex description of the anchor.

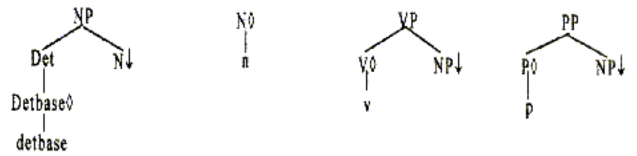


FIGURE 3.1 - Examples of trees before lexicalization

The principal unit of syntactic information associated with a LCFG entry is a tree structure in which the tree nodes are labeled with syntactic categories and feature information and there is at least one leaf node labeled with a *lexical* category (such lexical leaf nodes are known as *anchors*). Thus, the *lexicon* consists of a finite set of structure each associated with an anchor. The structures defined by the lexicon are called *elementary structures*. Structures built by combination of others are called *derived structures* [10]. With respect to a LCFG lexicon, this can be defined as the LTAG lexicon, for example, where this consists of a set of trees each one associated with one or more lexical items [14]. These elementary trees can be viewed as elementary clauses (including their transformational variants) in which the lexical items participate. The trees are combined by *substitution* and a restricted form of *adjunction* that is context-free in nature.

Fig.3.2 exhibits some of the constructions LCFG utilized in radiological domain. In grammars as LCFG, and others of the TAG family, the linguistic unit is this elementary tree, which corresponds to a minimal predicative structure. According to [22], these structures are syntactic and semantic units at the same time. They are gathered in tree families which encode the different lexical and syntactic rules which may apply to them [25]. When trees are combined by substitution or adjunction, the corresponding semantic representations are combined. In [24] focus that the elementary trees are the appropriate domains for characterizing certain dependencies (e.g., subcategorization and filler-gap dependencies). That is, an elementary tree localizes agreement dependencies, filler-gap dependencies and predicate-argument dependencies and also serves as a complex description of the anchor.

The sentences of radiological language will be obtained utilizing these structures and the lexical items contained in dictionary presented in [21]. Some of these sentences are exhibited in fig.3.3 which utilize elementary trees a13, a50, a26 e a38, and lexical items a, tíbia, tem, um, contorno, totalmente, regular, which are necessary to preterminals Detbase, N, Vtr, Detbase, N, Intens, Adj respectively. The trees that describe portuguese syntactic structures are grouped in families, specially according to criteria of verbal regency and transitivity [26]. Hierarchical representations of portuguese LCFG have been proposed, e.g., defining tree families [21]. A tree family contains the different possible trees for a given canonical subcategorization (or predicate-argument structure).

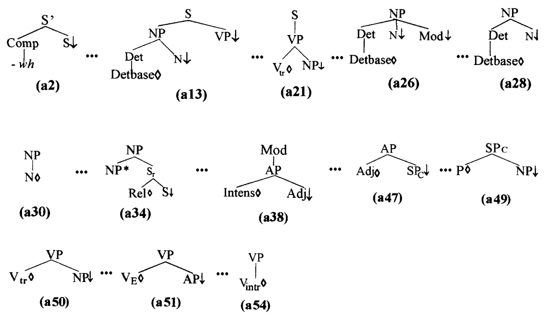


FIGURE 3.2 - Elementary structures utilized in radiological domain

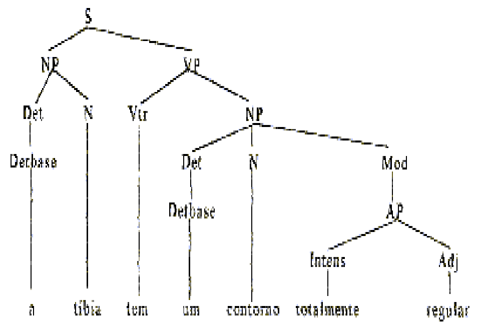


FIGURE 3.3 - Substitution in *a tíbia tem um contorno totalmente regular*

4. Conclusions

In this paper, we have presented some novel applications of LCFG. We have illustrated a radiological grammar, based on the LCFG formalism. A new approach to grammar in natural language to description of portuguese utilized in Natural Language Interfaces to Database (NLIDBs) was applied. The formalism TAG family was used, which is a tree-generating system rather than a string generating system. The set of trees derived from this family constitutes the object language. To describe these structures of portuguese, the conventional CFGs gave support in the definition of the syntactic structures of portuguese for LCFG. A meticulous study about context-free rules of phrase structure of TG was done. These rules of TG that foresee various linguistic problems of the NLIDBs [27] were defined for the deep structures. Transformation rules defined for the portuguese were described in [21] for attainment of surface structures. These rules were mapped for a surface grammar, which was taken as entry in the utilization of theorem presented in [8] that has as output an LCFG. LCFG is a formalism integrating lexicon and grammar. It has both <u>linguistic advantages</u> (e.g. elegant handling of unbounded dependencies and idioms [28]) and computational advantages, particularly due to lexicalization ([29]). The lexicalization in the LCFG showed to be interesting not only from the linguistic perspective, as such formal interest, because it is systematically associated with a lexical anchor. These structures specify extended domains of locality over which constraints can be stated. In [30] emphasizes the importance of key-concept of *extended domain of locality*, which is capable of allowing the dependency information and phrase structure information to be represented in one structure, that is, an elementary tree. This feature, according to [30], allows the phrase structure parse tree to be represented in terms of dependency information. The advantage of the extended domain of locality, according to [31], is that many relationships that must be mediated via grammar rules in other formalisms can be stated directly in the lexical entries.

The linguistic and mathematical advantages of lexicalized formalisms are useful in practical applications. Currently, most large scale NLP systems adopt CFG-like grammars, that have to face the problem of syntactic ambiguity, because the constraints expressed on the syntactic categories are too general to limit the huge hypothesis space generated by wide-coverage grammars [32]. So, LCFG favored the syntactical analysis of NL for the portuguese, because a broader domain of locality than usual phrase structure rules. This allows us to state, for example, subcategorization imposed on an element by another that does not directly dominate it, e.g., between the verb and the determiner of its first complement. Other linguistic advantage is that the adjunction permits the recursion in the composition of trees. The lexicalization notion in the LCFG is also linguistically very significant by preserving not only the string sets (weak generative capacity) but also the structures, i.e., *strong generative capacity*. The *syntactic criteria* defined by LCFG restrict the number of possible structures for a sentence, simplifying the semantic analysis. With relation to *concordance criteria*, these can be taken outside grammar, as proposed by [33] for not causing exponential explosion of the rules. In this work the concordance criteria are similar to *top* and *bottom* features [34] that are unified in LTAG.

An important fact, undoubtedly, is that the LCFGs do not require more computational recourses than CFGs. The greatest computational advantage [7] is that the parsing of an LCFG (obtained by CFG through lexicalization) is significantly faster than the one of the CFG. Although the string sets generated by LCFG are the same as those generated by CFG, LCFG is capable of generating more complex sets of trees than CFGs [7]. The fact that LCFG lexicalizes CFG is significant, because every other method for lexicalizing CFG without changing the trees derived require context-sensitive operations [10] and therefore dramatically increases worst case processing time. Context-sensitive operations (e.g., mildly context-sensitive formalisms [11]) entail much larger computational costs for parsing and recognition than CFGs. In particular, the fastest known LTAG parser require $O(n^6)$ - time in the worst case [12] in contrast to $O(n^3)$ for CFG. Since LCFG is a restricted case of TAG, standard $O(n^6)$-time TAG parsers [11], [12], [13] can be used for parsing LCFG. Although they require $O(n^6)$-time for TAG parsing, they can be made very easily to require at most $O(n^4)$-time for LCFG. This bound is still too high since we know that LCFGs generate only context-free languages. Then, recognizer and left to right parsing algorithm which requires $O(n^3)$-time in the worst case for LCFG has been proposed [14], [7], [21] to process a sentence of length n. Since the attractive aspects of LTAGs come at some computational cost, LCFG provides an efficient alternative which does not sacrifice the elegance of the LTAG analyses and which may be useful in different areas of computational linguistics.

Very restricted natural language does not work particularly well as a straight substitute for a traditional formal query language. A broader system covering a larger part of the Portuguese grammar and medical vocabulary is currently under development. The grammar has to be sufficiently robust to cope gracefully with sentence fragments and ill-formed input, words, expressions and medical jargon utilized in medicine. The grammar described in this paper has the possibility of to be inserted in broader studies which generate the portuguese language and not only the radiological language ones. Portuguese language interfaces for another applications as Operational Systems [35], Expert Systems [36], [37], [38], Intelligent Tutor Systems, etc. [15], [16], [21], [39] can use these grammar. Stochastic extensions [40] can also be incorporated to grammar for making it possible to capture both distributional and hierarchical information about portuguese words.

References

[1] KRULEE, G. K. *Computer Processing of Natural Language*. Englewood Cliffs, NJ: Prentice-Hall, 1991. 456p.

[2] COPESTAKE, A.; JONES, K. S. Natural Language Interfaces to Databases. *The Knowledge Engineering Review*, [S.l.], v.5, n.4, p.225-249, 1990.

[3] BARBOSA, C. R. S. C. de. Interfaces em Linguagem Natural para Banco de Dados. Porto Alegre: CPGCC da UFRGS, 1997. 165p. Trabalho Individual n.640.

[4] GROVER, C. et. al. The Alvey Natural Language Tools Grammar (second release). Cambridge: Computer Laboratory, University of Cambridge, 1989. (Technical Report 162).

[5] MANARIS, B. Z.; SLATOR, B. M. Interactive Natural Language Processing: Building on Success. IEEE Computer, New York, v.29, n.7, p.28-32, July 1996.

[6] MENEZES, P. F. B. Linguagens Formais e Autômatos. Porto Alegre: Instituto de Informática da UFRGS: Sagra Luzzatto Editores, 1998. 165p.

[7] SCHABES, Y.; WATERS, R. C. Lexicalized Context-Free Grammar: A Cubic-Time Parsable, Lexicalized Normal Form For Context-Free Grammar That Preserves Tree Structure. Broadway, Cambridge: Mitsubishi Electric Research Laboratories, June 1993. 30p. (Technical Report 93-04).

[8] SCHABES, Y.; WATERS, R. C. Lexicalized Context-Free Grammars. In: ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, 31., 1993, Ohio, USA. Proceedings... Ohio: [s.n.], 1993. p.121-129.

[9] RICH, E.; KNIGHT, K. Inteligência Artificial. São Paulo: Makron Books, 1993. 722p.

[10] JOSHI, A. K.; SCHABES, Y. Tree-Adjoining Grammars and Lexicalized Grammars. In: EUROPEAN SUMMER SCHOOL IN LOGIC, LANGUAGE AND INFORMATION, 4., 1992, Colchester, U.K. Proceedings... Colchester: University of Essex, 1992. p.1-23.

[11] VIJAY-SHANKER, K. A Study of Tree Adjoining Grammars. Philadelphia, USA: Department of Computer Science, University of Pennsylvania, 1987. PhD Thesis.

[12] LANG, B. The Systematic Constructions of Earley Parsers: Application to the Production of O(n6) Earley Parsers for Tree Adjoining Grammars. In: INTERNATIONAL WORKSHOP ON TREE ADJOINING GRAMMARS, 1., 1990, Dagstuhl Castle, FRG. Proceedings... Castle: [s.n.], 1990.

[13] SCHABES, Y. The Valid Prefix Property and Left to Right Parsing of Tree-adjoining Grammar. In: INTERNATIONAL WORKSHOP ON PARSING TECHNOLOGIES, 2., 1991, Cancun, Mexico. Proceedings... Cancun: [s.n.], 1991.

[14] SCHABES, Y. Lexicalized Context-Free Grammars. Broadway, Cambridge: Mitsubishi Electric Research Laboratories, Jan. 1993. 16p. (Technical Report TR 93-01).

[15] CURY, D. FLAMA: Ferramentas e Linguagem de Autoria para a Modelagem da Aprendizagem. São José dos Campos: CPG da Engenharia Eletrônica e Computação do Instituto Tecnológico da Aeronáutica, 1996. 151p. PhD Thesis.

[16] CURY, D.; OMAR, N.; DIRENE, A. I. Modelos Baseados em Estereótipos e Oráculos para a Aprendizagem de Conceitos Visuais. Revista Brasileira de Informática na Educação, n.02, p.43-53, Apr. 1998.

[17] DIRENE, A. I. Methodology and Tools for Designing Concept Tutoring Systems. Sussex: School of Cognitive and Computing Sciences, University of Sussex, 1993. PhD Thesis.

[18] CHOMSKY, N. Aspects of the Theory of Syntax. Cambridge, MA: MIT Press, 1965.

[19] ZORZO, A. F. Gramática Transformacional com Atributos. Porto Alegre: CPGCC da UFRGS, 1994. 99p. Master Thesis.

[20] BAUD, R. H.; RASSINOUX, A. M.; SCHERRER, J. R. Natural Language Processing and Knowledge Representation of Medical Texts. Methods of Information in Medicine, [S.l.], v.31, p.117-125, 1992.

[21] BARBOSA, C. R. S. C. de. Gramática para Consultas Radiológicas em Língua Portuguesa. Porto Alegre: CPGCC da UFRGS, 1998. 143p. Master Thesis.

[22] ABEILLÉ, A. et al. Non Compositional Discontinous Constituents in Tree Adjoining Grammar. In: EUROPEAN SUMMER SCHOOL IN LOGIC, LANGUAGE AND INFORMATION, 4., 1992, Colchester, U.K. Proceedings... Colchester: University of Essex, 1992. p.1-20.

[23] BARBOSA, C. R. S. C. de.; CASTILHO, J. M. V. de. Gramática Livre de Contexto Lexicalizada para a Análise Sintática da Língua Portuguesa - Uma Experiência na Geração de Consultas de uma Interface em Linguagem Natural para Banco de Dados. In: ENCONTRO PARA O PROCESSAMENTO COMPUTACIONAL DA LÍNGUA PORTUGUESA ESCRITA E FALADA, 5., 2000, Atibaia, SP. Anais... Atibaia: ICMC/USP, 2000. 193p. p.155-164.

[24] KROCK, A. Analysing Extraposition in a Tree Adjoining Grammar. In: EUROPEAN SUMMER SCHOOL IN LOGIC, LANGUAGE AND INFORMATION, 4., 1992, Colchester, U.K. Proceedings... Colchester: University of Essex, 1992. p.107-149.

[25] ABEILLÉ, A. et al. A Lexicalized Tree Adjoining Grammar for English. Philadelphia, USA: Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA, 1990. (Technical Report MS-CIS-90-24).

[26] JOSHI, A.; VIJAY-SHANKER, K.; WEIR, D. The Convergence of Mildly Context-Sensitive Grammatical Formalisms. In: SELLS P.; SHIEBER, S.; WASOW, T. (Eds.). Foundational Issues in Natural Language Processing. Cambridge, MA: MIT Press, 1991. p.31-81.

[27] ANDROUTSOPOULOS, I.; RITCHIE, G. D.; THANISCH, P. Natural Language Interfaces to Databases - An Introduction. Journal of Natural Language Engineering, Cambridge, p.1-50, 1994.

[28] CANDITO, M.H. A Principle-based Hierarchial Representation of LTAGs. In: COLING'96., 1996, Copenhagen, Proceedings... Copenhagen, [s.n.], 1996. p.194-199.

[29] SCHABES, Y.; ABEILLÉ, A.; JOSHI, A. K. Parsing Strategies with 'Lexicalized' Grammars: Application to Tree Adjoining Grammars. In: INTERNATIONAL CONFERENCE ON COMPUTATIONAL LINGUISTICS, 12., 1988, Budapeste, Hungria. Proceedings... Budapeste: ACL, 1988. 843p. p.578-583.

[30] JOSHI, A. K.; SRINIVAS, B. Using Parsed Corpora for Circumventing Parsing. In: WERMTER, S.; RILOFF, E.; SCHELER, G. (Eds.). Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language Processing. Berlin: Springer-Verlag, 1996. p.413-424.

[31] POPOWICH, F. Lexical Characterization of Local Dependencies with Tree Unification Grammar. Burnaby, Canada: School of Computing Science, Simon Fraser University, Sep. 1993. 36p. (Technical Report CMPT TR 93-13).

[32] BARBERO, C.; LOMBARDO, V. Wide-coverage Lexicalized Grammars. [S.l.:s.n.], 1996.

[33] MONTEIRO, S. L. Um Subconjunto Regular do Português. Rio de Janeiro: Laboratório Nacional de Computação Científica, 1988. 43p. (Relatórios de Pesquisa e Desenvolvimento ISSN 0101 6113).

[34] XTAG RESEARCH GROUP. A Lexicalized Tree Adjoining Grammar for English. Pennsylvania: Institute for Research in Cognitive Science, University of Pennsylvania, 1995. 155p. (Technical Report IRCS 95-03).

[35] MANARIS, B. Z.; DOMINICK, W.D. NALIGE: a User Interface Management System for the Development of Natural Language Interfaces. International Journal of Man-Machine Studies, [S.l.], v.38, n.6, p.891-892, June 1993.

[36] BARBOSA, C. R. S. C. de. Sistema Especialista Pedagógico. Rio Claro: DEMAC-UNESP. 1992. 120p. Trabalho de Conclusão.

[37] OLIVEIRA, C. A. de. Interface em Linguagem Natural para Sistemas Especialistas. In: 2o SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL. 20-22 nov. 1985, São José dos Campos. Anais... São José dos Campos: 1985. p.173-177.

[38] DREW, C. F. A Natural Language Interface for Expert System Help Desks. In: CONFERENCE ON MANAGING EXPERT SYSTEM PROGRAMS AND PROJECTS, 1990, Bethesdas, MD. Proceedings... Bethesdas: 1990. p.209-215.

[39] BARBOSA, C. R. S. C. de.; OMAR, N. Detecção de Erros Durante um Diálogo Tutorial. In: WORKSHOP DE COMPUTAÇÃO, 3., 2000, São José dos Campos, SP. Anais... São José dos Campos: CTA/ITA, 2000. 164p. p.1-8.

[40] SCHABES, Y.; WATERS, R. C. Stochastic Lexicalized Context-Free Grammars. Broadway, Cambridge: Mitsubishi Electric Research Laboratories, July 1993. 12p. (Technical Report 93-12).

Practical, Template–Based Natural Language Generation with TAG

Tilman Becker
DFKI GmbH

1. Introduction

This paper describes a TAG–based (Joshi and Schabes, 1997), template–based approach to Natural Language Generation. It extends the idea of lexicalization to whole phrases, similar in style to the representation of idioms in a TAG grammar. In addition to this, there is a second type of templates: partial derivations.

The first phase in the generator, driven by planning rules, produces a derivation tree which is then used in a straightforward realizer to create the derived tree. This tree is then the basis for the input to a Concept–To–Speech synthesis component (Schweitzer, Braunschweiler and Morais, 2002).

There are two basic methods for constructing these two kinds of templates. The first and preferred one is based on an existing grammar where the templates represent shortcuts that would be used for reasons as simplicity, efficiency, or because the existing grammar does not contain an interface to the representation language, e.g., semantics, that is used as the generator‘s input. The second method which is used so far in the SMARTKOM project (Wahlster, Reithinger and Blocher, 2001) is necessary when no suitable grammar exists. It allows for an approach similar to rapid prototyping: only the templates that are needed are specified and the templates are initially kept as large as possible. Only in the following development steps, the templates are made smaller and smaller, eventually themselves becoming a grammar.

The following sections briefly present the SMARTKOM project and the generator architecture. Then the use of fully specified templates is discussed in this context, including their use for concept–to–speech synthesis. Finally we present the current work, including some tools that are under development.

2. The SmartKom Project

SMARTKOM is a multi-modal, dialogue system currently being developed at several academic and industrial partners (see `www.smartkom.org`). User input modalities are speech, including prosody, various gestures,[1] and an interpretation of the user‘s facial expression. The system output is a combination of graphical presentations and an animated, talking agent, Smartakus, plus various external effects such as controlling a VCR, sending e-mail, querying databases etc.

The key idea behind the system is to develop a kernel system which can be used within several scenarios and applications. Currently there exist three such scenarios – *public, home*, and *mobile* – with a number of applications such as EPG (electronic programming guide), scanning and sending documents, route finding, etc. They all are different in their external appearance but share most of the basic processing techniques. The system depicted in Figure 1 is the “public scenario” system. Within this scenario, an intelligent kiosk is developed with which one is able to access the various applications. Development is mainly for German, but some applications are also ported to English.

2.1. Architecture

Figure 2 shows the overall architecture of the generation system which follows a straightforward classical pipeline design. On the interfaces to input and output, XSLT stylesheet transformations adapt the various formats, while internally there are two important sub–components: Preplan, a simple planning engine that maps from the presentation goals to derivation trees and a TAG component that implements TAG trees with their operations, including full feature unification.

Preplan is a top–down rule expanding planner, implemented in Java. Rules have associated constraints that can refer to knowledge bases that are constructed from the input to the generator. The planner then matches parts of this input to select appropriate templates (i.e., partial derivation trees) and fills them with data from the input.

1. analyzed through an infrared camera, thus obliterating the need for a touch screen.

Figure 1: Multi-modal interaction with the system.

e.g., the name of the user is taken from the input and inserted into an existing tree for a name, overwriting an uninstantiated lexical item in this tree.

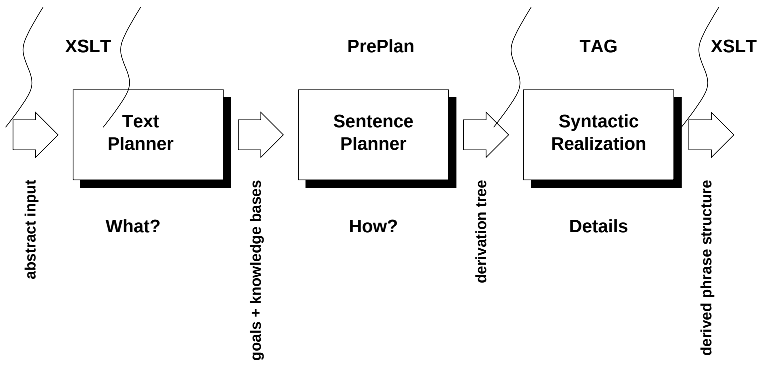


Figure 2: Classical pipeline architecture of the generator.

3. Fully Specified Templates

The design decision to use template–based generation in SMARTKOM, is driven by practical considerations: First, generator output was needed as early as possible in the project. Since there are a number of non–overlapping, but well–defined applications in SMARTKOM, output for a new application has to be created in a short time, suggesting the use of templates. On the other hand, simple string concatenation is not sufficient. E.g., for integrating Concept–to–Speech information, especially in the way the synthesis component of SMARTKOM is designed(Schweitzer, Braunschweiler and Morais, 2002), calls for an elaborate syntactic representation, i.e., phrase structure trees with features, to guide the decisions on prosodic boundaries. At least since (Reiter, 1995) (also see (Becker and Busemann, 1999)), the use of templates and "deep representations" is not seen as a contradiction. Picking up on this idea, the generation component in SMARTKOM is based on fully lexicalized generation (Becker, 1998), packing whole parts of a sentence together into one *fully specified template*, representing them not as a string but rather as a partial TAG derivation tree. See also figure 3. All nodes in the TAG trees carry top and bottom feature structures, see (Vijay-Shanker, 1987), which also contain discourse information as needed for concept–to–speech synthesis. The current setup uses fully inflected forms, but this is due to be changed soon and the changes will be minimal. A call to a morphology module will get inflectional information from the feature structures and stem information from the leaf as, e.g., in (Becker, 1998).

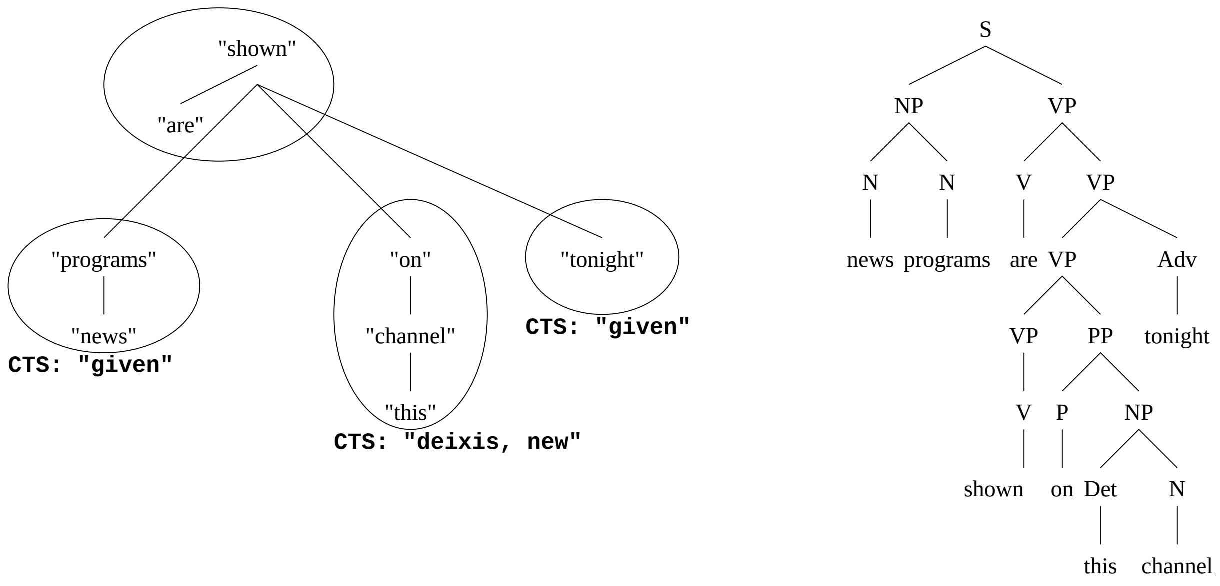


Figure 3: Derivation tree with CTS (Concept–to–Speech) markup. Each ellipse is a fully specified template. The sentence–planning process combines such templates to a complete derivation tree.

3.1. Specifying only Relevant Representations

With this approach to specify intermediate levels of representation (which commonly only exist inside the NLG module), the question remains whether *all* levels of representation have to be specified. Clearly, this is desirable, but not necessary in SMARTKOM. Thus only the level of dependency and phrase structure are represented fully. Dependency is necessary to guide the top–down generation approach, phrase structure is necessary for (Concept–to–)speech synthesis. However, there is nothing preventing the later inclusion of other levels of representation, e.g., adding a level of semantic representation (e.g., (Joshi and Vijay-Shanker, 1999)) which might be used by a sentence aggregation component.

3.2. Dependency and Speech–Markup

Specifying templates on the level of derivation trees rather than on derived trees or even strings has several advantages. In the context of Concept–to–Speech synthesis, it is necessary to add markup to parts of the string. This can be done easily by adding the information to the corresponding node in the derivation tree from where it is percolated and automatically distributed to the corresponding parts of the utterance when constructing the derived tree and the string from the derivation tree. Such markup relates to parts of the output that have to (de-)emphasized, parts that refer to objects in the graphical presentation and must be coordinated with a pointing gesture.

Figure 3 shows a derivation tree with speech-relevant markup on some nodes. Besides mere convenience in the markup[2] the additional power of TAG allows the distribution of semantically connected markup to discontinuous parts in the final string. Since formal complexity is a very different issue in generation than in parsing, we are open to the use of extensions from the standard TAG formalism as in (Rambow, 1994) or (Gerdes and Kahane, 2001) which might be necessary for German.

3.3. Tools and Current Work

Currently we have editors available for the planning rules and the TAG-tree templates. Both build on XML

2. E.g., XML-style opening and closing parentheses can be integrated into the trees and thus are realized by a single marked node vs. the situation in a classical context–free based string-expanding template generator, where opening and closing elements have to be denoted independently–a typical source for errors.

representations of the knowledge bases and present them in an easily accessible format: a directory structure[3] as known from the Windows Explorer for the set of trees and a graphical tree editor for the TAG-trees.[4]

Current work is centered around adding templates for new applications and has shown that managing a large set of templates can be problematic. Eventually we hope to switch to the first development method as mentioned in the introduction: We plan to extend the rule editor with a TAG-parser. To add a new template to the generator, the user will then type in an example sentence, have it parsed, select the correct parse, mark (delete) the variable parts, keeping the fixed part and add the remainder of the rule. Thus rules can be created without ever writing trees by hand.[5]

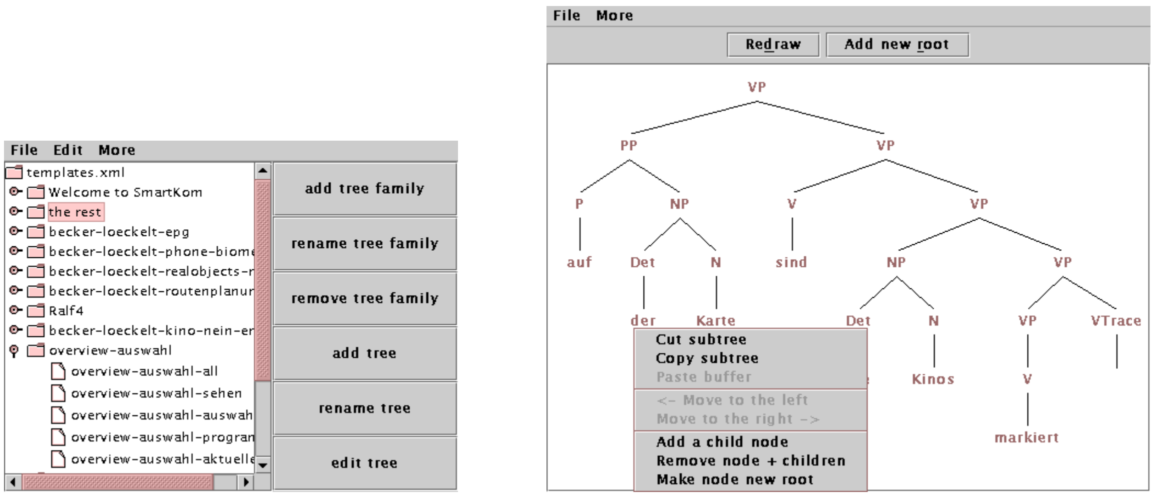


Figure 4: The grammar and tree editor tools.

References

Becker, Tilman. 1998. Fully lexicalized head-driven syntactic generation. In *Proceedings of the Ninth International Workshop on Natural Language Generation*, Niagara-on-the-Lake, Ontario, Canada, August.

Becker, Tilman and Stephan Busemann, editors. 1999. *May I Speak Freely? Between Templates and Free Choice in Natural Language Generation.*, Bonn, September. Workshop at the 23rd German Annual Conference for Artificial Intelligence (KI '99).

Gerdes, Kim and Sylvain Kahane. 2001. Word order in German: A formal dependency grammar using a topological hierarchy. In *Proc. of ACL 2001*, Toulouse, France.

Joshi, A. and Y. Schabes, 1997. "Tree–Adjoining Grammars". In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, pages 69–124. volume 3. Berlin, New York: Springer.

Joshi, A. K. and K. Vijay-Shanker. 1999. Compositional Semantics for Lexicalized Tree-Adjoining Grammars. In *Proc. of the 3rd International Workshop on Computational Semantics*, Utrecht, The Netherlands, January.

Rambow, Owen. 1994. *Formal and Computational Models for Natural Language Syntax.* Ph.D. thesis, University of Pennsylvania, Philadelphia. Available as Technical Report 94-08 from the Institute for Research in Cognitive Science (IRCS).

Reiter, Ehud. 1995. NLG vs. templates.

Schweitzer, Antje, Norbert Braunschweiler and Edmilson Morais. 2002. Prosody Generation in the SmartKom Project. In Bernard Bel and Isabel Marlien, editors, *Proceedings of Speech Prosody 2002*, pages 639–642, Aix-en-Provence, France.

Vijay-Shanker, K. 1987. *A Study of Tree Adjoining Grammars*. Ph.D. thesis, Department of Computer and Information Science, University of Pennsylvania.

Wahlster, Wolfgang, Norbert Reithinger and Anselm Blocher. 2001. SmartKom: Multimodal Communication with a Life-Like Character. In *Proceedings of Eurospeech 2001*, Aalborg, Denmark, September.

3. implemented with JTree
4. A transformation of the XTAG grammar into the XML format is in the works, the main problem is parsing and translating the feature equations. Eventually the feature macros need to be incorporated into the editor, too.
5. Which will also avoid inconsistencies in the set of templates.

Relative Clause Attachment and Anaphora: A Case for Short Binding

Rodolfo Delmonte
Ca' Garzoni-Moro, San Marco 3417, Università "Ca Foscari", 30124 - VENEZIA
E-mail: delmont@unive.it

Abstract

Relative clause attachment may be triggered by binding requirements imposed by a short anaphor contained within the relative clause itself: in case more than one possible attachment site is available in the previous structure, and the relative clause itself is extraposed, a conflict may arise as to the appropriate s/c-structure which is licenced by grammatical constraints but fails when the binding module tries to satisfy the short anaphora local search for a bindee.

1 Introduction

It is usually the case that anaphoric and pronominal binding take place after the structure building phase has been successfully completed. In this sense, c-structure and f-structure in the LFG framework - or s-structure in the chomskian one - are a prerequisite for the carrying out of binding processes. In addition, they only interact in a feeding relation since binding would not be possibly activated without a complete structure to search, and there is no possible reversal of interaction, from Binding back into s/c-structure level seen that they belong to two separate Modules of the Grammar. As such they contribute to each separate level of representation with separate rules, principles and constraints which need to be satisfied within each Module in order for the structure to be licensed for the following one.

However we show that anaphoric binding requirements may cause the parser to fail because the structure is inadequate. We propose a solution to this conflict by anticipating, for anaphors only the though, the agreement matching operations between binder and bindee and leaving the coindexation to the following module.

In a final section we discuss data from syntactic Treebanks of English – the Penn Treebank – and Italian, the Italian Treebank and the Venice Treebank.

1.1 Positive and Negative Constraints

Anaphoric and Pronominal Binding are usually treated as if they were one single grammatical phenomenon, even though the properties of the linguistic elements involved are quite different, as the subdivision of Binding Principles clearly shows. However, it is a fact, that the grammatical nature of a pronoun - be it an anaphor (short or long one), or a free pronoun - is never taken into account when searching for the antecedent. The anaphoric module of the grammar takes for granted the fact that both the structure associated to the anaphor/pronoun, the grammatical function - at f-structure level in LFG - and the functional features are consistent, coherent and respondent to the Grammaticality constraints stipulated in each grammatical theory. It is the structural level that guarantees consistency, not the Anaphoric/Pronominal Binding Module, which has the only task to add antecedent-pronoun/anaphor indices in the structure, to be used by the semantic modules.

We chose a couple of examples which represent the theoretical query to be solved, given a certain architecture of linguistic theories, which may differ in the way in which they reach a surface representation into syntactic constituents of the input string, but all converge into the need to keep the anaphoric module separate from the structure building process. The examples are in English but may be easily replicated in other languages:

(1) The doctor called in the son of the pretty nurse who hurt herself

(2) The doctor called in the son of the pretty nurse who hurt himself

In the second example we have the extraposition of the relative clause, a phenomenon very common in English but also in Italian and other languages. The related structures theoretically produced, could be the following ones:

```
(1)a s[np[The doctor],
       ibar[called in],
         vp[np[the son,
                pp[of, np[the pretty nurse,
                        cp[who, s[pro, ibar[hurt],
                                        vp[sn[herself]]]]]]]]
(2)a. s[np[The doctor],
            ibar[called in],
              vp[np[the son,
                    pp[of, np[the pretty nurse]],
                 cp[who, s[pro, ibar[hurt],
                           vp[sn[himself]]]]]]]
```

If this is the correct input to the Binding Module, it is not the case that 2a. will be generated by a parser of English without special provisions. The structure produced in both cases will be 1a. seen that it is perfectly grammatical, at least before the binding module is applied to the structure and agreement takes place locally, as required by the nature of the short anaphor. It is only at that moment that a failure in the Binding Module warns the parser that something wrong has happened in the previous structure building process. However, as the respective f-structures show, the only output available is the one represented by 2b, which wrongly attaches the relative clause to the closest NP adjacent linearly to the relative pronoun:

```
2b. s[np[The doctor],
ibar[called in],
vp[np[the son,
   pp[of, np[the pretty nurse,
            cp[who, s[pro, ibar[hurt],
                           vp[sn[himself]]]]]]]]
```

The reason why the structure is passed to the Binding Module with the wrong attachment is now clear: there is no grammatical constraint that prevents the attachment to take place. The arguments of the governing predicate HURT are correctly expressed and are both coherent and consistent with the information carried out **by the lexical form. At the same time the Syntactic Binding has taken place again correctly by allowing the** empty "pro" in SUBJect position of the relative adjunct to be "syntactically controlled" by the relative pronoun, which is the TOPic binder, in turn syntactically controlled by the governing head noun, the NURSE. There is no violation of agreement, nor of lexical information, nor any other constraint that can be made to apply at this level of analysis in order to tell the parser that a new structure has to be produced.

2 Parsing Strategies and Preferences

In order for a parser to achieve psychological reality it should satisfy requirements coming simultaneously from three different fields/areas: psycholinguistic plausibility, computational efficiency in implementation, grammatical constraints. Principles underlying the parser architectures should not belong exclusively to one or the other field disregarding issues which might explain the human processor behaviour. Principles are bestowed psychological reality in performance whenever they may be safely tested, on a statistically relevant sample of individuals. So we annect a lot of importance to the fact that the parser actually behaves like what is expected with human processors. In this case and only in this case we say that the principles are predictive and that the parser we implemented is actually relevant for a theory of parsing.

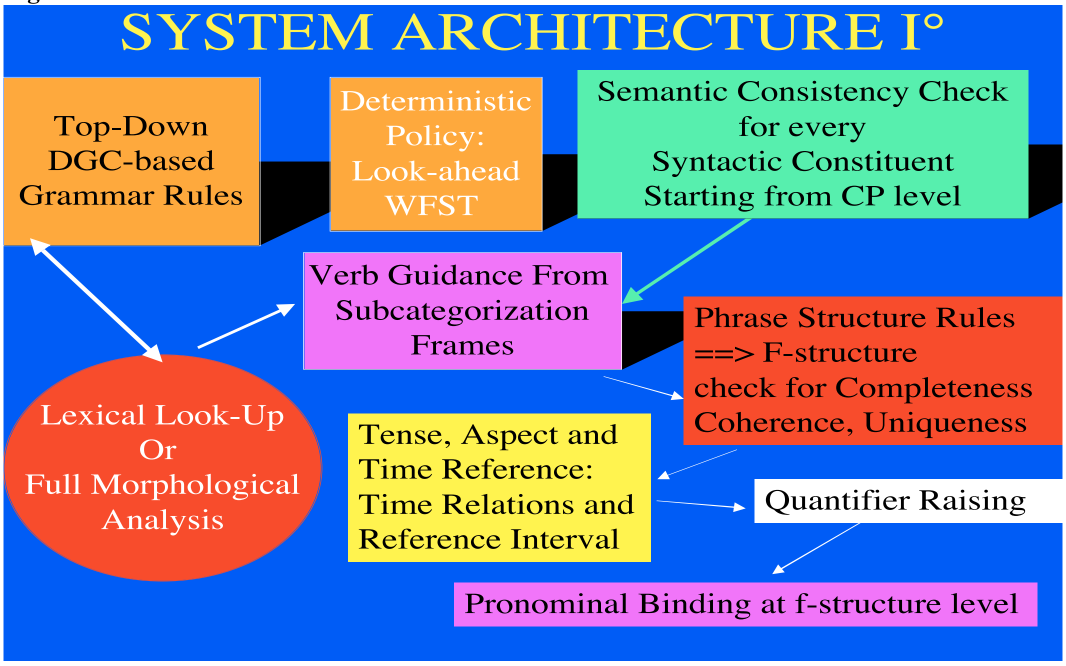
Among contemporary syntactic parsing theories, the garden-path theory of sentence comprehension proposed by Frazier(1987a, b), Clifton & Ferreira (1989) among others, is the one that most closely represents our point of view. It works on the basis of a serial syntactic analyser, which is top-down, depth-first - i.e. it works on a single analysis hypothesis, as opposed to other theories which take all possible syntactic analysis in parallel and feed them to the semantic processor.

Differently from what is asserted by global or full paths approaches (see Schubert, 1984), we believe that decisions on structural ambiguity should be reached as soon as possible rather than deferred to a later level of representation. In particular, Schubert assumes "...a full paths approach in which not only complete phrases but also all incomplete phrases are fully integrated into (overlaid) parse trees dominating all of the text seen so far. Thus features and partial logical translations can be propagated and checked for consistency as early as possible, and alternatives chosen or discarded on the basis of all of the available information(ibid., 249)." And further on in the same paper, he proposes a system of numerical 'potentials' as a way of implementing preference trade-offs. " These potentials (or levels of activation) are assigned to nodes as a function of their syntactic/semantic/pragmatic structure and the preferred structures are those which lead to a globally high potential. Other important approaches are represented by Hindle et al., 1993, who attempt to solve the problem of attachment ambiguity in statistical terms. The important contribution they made, which was not possible in the '80s, is constituted by the data on attachment typologies derived from syntactically annotated corpora.

Our parser copes with ambiguity while at the same time allowing for psychological coherence. Parser architecture is presented in Fig.1 below. The structures produced by the parser take only different processing time to allow for backtracking to take place within the main parser body: but then the right attachment is achieved and the complete structure is produced with the right binding.

We implemented two simple enough mechanisms in order to cope with the problem of nondeterminism and backtracking. At bootstrapping we have a preparsing phase where we do lexical lookup and we look for morphological information: at this level of analysis of all input tokenized words, we create a stack of pairs input wordform - set of preterminal categories, where preterminal categories are a proper subset of all lexical categories which are actually contained in our lexicon. The idea is simply to prevent attempting the construction of a major constituent unless the first entry symbol is well qualified. When consuming any input wordform, we remove the corresponding pair on top of stack.

Fig.1 Deterministic Parser Architecture with Lookahead and WFST


SYSTEM ARCHITECTURE I°
Top-Down DGC-based Grammar Rules
Deterministic Policy: Look-ahead WFST
Semantic Consistency Check for every Syntactic Constituent Starting from CP level
Verb Guidance From Subcategorization Frames
Phrase Structure Rules ==> F-structure check for Completeness Coherence, Uniqueness
Lexical Look-Up Or Full Morphological Analysis
Tense, Aspect and Time Reference: Time Relations and Reference Interval
Quantifier Raising
Pronominal Binding at f-structure level

In order to cope with the problem of recoverability of already built parses we built a more subtle mechanism that relies on Kay's basic ideas when conceiving his Chart(see Kay, 1980). Differently from Kay, however, we are only interested in a highly restricted topdown depthfirst parser which is optimized so as to incorporate all linguistically motivated predictable moves. An already parsed RC is deposited in a table lookup accessible from higher levels of analysis and consumed if needed. To implement this mechanism in our DCG parser, we assert the contents of the RC structure in a table lookup storage which is then accessed whenever there is an attempt on the part of the parser to build up a RC. In order to match the input string with the content of the store phrase, we implemented a WellFormed Substring Table(WFST) as suggested by Kay(1980).

Now consider the way in which a WFST copes with the problem of parsing ambiguous structure in his chart. It builds up a table of well-formed substrings or terms which are partial constituents indexed by a locus, a number corresponding to their starting position in the sentence and a length, which corresponds to the number of terminal symbols represented in a term. For our purposes, two terms are equivalent in case they have the same locus and the same length. In this way, the parser would consume each word in the input string against the stored term, rather than against a newly built constituent. In fact, this would fit and suit completely the requirement of the parsing process which rather than looking for lexical information associated to each word in the input string, only needs to consume the input words against a preparsed well-formed syntactic constituent.

However, in order for a parser to show coherent psychological behaviour it should show “garden path” effects while simulating a condition of failure to parser at propositional level (see Pritchett). Full paths parsers, and in general all bottom-up chart-like parsers will not show any of the garden-paths effects simply because failure is prevented from taking place by the fact that all possible parses are always available and can be retrieved by the parser itself. The question that is posed by our two examples will not however be covered by a full-paths parser seen that there is no principled reason in the grammar to prefer one structure over the other. Failure only takes place in the pronominal binding module which is usually a separate module of the parser.

3 Short Anaphora

The parser we use has shown the effect of "garden path", in that it has gone into a loop with the unwanted result of "freezing" the computer, due to data overflow. In other words, as soon as the Binding Module tries to process the f-structure received as input, seen that short anaphora requires binding to take place within a local domain, f-command - the corresponding c-command in functional terms, applied to grammatical functions and a graph structure - will impose the same level of containment for both the pronoun and the antecedent. And seen that the only antecedent available is the empty SUBJect which has functional features inherited by means of syntactic control from the governing relative pronoun, the agreement match is attempted, and a failure ensues systematically.

As a result of a failure at the Binding Level, a call to the structural level is issued which attempts to build the structure another time. But seen that no failure has taken place at this level of analysis, the result will be the same as the previous one. And this process will go on indefinitely, seen that the two modules obey different Principles and satisfy them separately.

We will now put forward a theoretical proposal regarding exclusively short anaphors, thus disregarding long anaphors and reciprocals in particular or "proprio" in Italian, which call for a different treatment. The proposal we will make is very simple:
"short anaphora must be checked for agreement with their available binder already at the level of satisfaction of grammatical principles, before the structure is licensed"

This requirement is not introduced by the need to improve on the implementation side of the parser, but responds to theoretical principles inherent in the formulation of the Binding Principles. Short anaphora not only obey positive constraints, as opposed to the other pronominals, they also carry a locality requirement which is equivalent to the same domain in which Grammaticality Principles apply, such as the ones expressed in LFG - Uniqueness, Completeness, Consistency. At each "propositional" level, corresponding to a simple f-structure and roughly to a Complete Functional Complex in GB terms(see Chomksy 1986, 169), all arguments of the governing predicate must be checked for completeness - they must all be present at functional level, even if they may be lexically empty; they must be coherent, only those included in the corresponding lexical form must be present; each functional attribute must be assigned to a unique functional value. And in our case no violation is detectable seen that the attributes belonging to the empty "pro" SUBJect are unique even though they are not appropriate to bind the short anaphor OBJect of the same predicate HURT. However there is no indication in the grammar that they should be checked for agreement at this level of analysis.

By anticipating the working of the Binding Module, we assume that Short Anaphors belong partly to the Grammar level and partly to the Binding level: they belong to the grammar level since they require and can to be licensed at sentence or propositional level withouth their f-features being in agreement with their antecedent and binder. Besides, they belong to the binding level where agreement takes place and coindexation follows, in case of success.

As to cases in which the anaphor is contained within a NP in SUBJect position of a sentential complement, the search for the antecedent is suspended not being available locally and no agreement match can be performed. This will not apply to anaphors contained within the NP of the OBJect seen that the antecedent is available.

A failure in the Anaphoric Module will simply cause the Parser to backtrack but the structure produced will not change seen that the failure has taken place in a separate module. Of course, the alternative is using a single unification mechanism that takes context-free rules with all possible alternatives, builds a tentative structure than unifies functional features, and in case of failure tries another possible structure. However this perspective is not only computationally inefficient, it is basically psychologically unfeasible: there will be no principled reason to tell Garden Path sentences apart from the rest seen that all sentences can be adjusted within the parser, sooner or later. Also processing time is not controllable seen that the parser will produce all possible structures anyway and there is no way to control the unification mechanism in a principled manner. On the contrary, in a parser like ours, the order of the rules is controlled strictly, and also the way to produce backtracking is controlled, seen that the parser has a lookahead mechanism that tells the parser which rule to access or not at a given choice point.

Going back to our couple of examples of the Extraposed Relative Clause containing a Short Anaphor, the question would be to prevent Failure since we do not want Constituent Structure Building to be dependent upon the Binding of the Short Anaphor. The only way out of this predicament is that of anticipating in Sentence Grammar some of the Agreement Checking Operations as proposed above. So the Parser would be able to backtrack while in the Grammar and to produce the attachment of the Relative Clause at the right place, in the higher NP headed by the masculine N, “the son”. The important result would be that of maintaining the integrity of Syntax as a separate Module which is responsible in “toto” of the processing of constituent structures. The remaining Modules of the Grammar would be fully consistent and would use the information made available in a feeding relation, so that interpretation will follow swiftly.

To integrate this suggestion coming from Implementation problems, into the theoretical Framework of LFG or other similar theories we simply need to integrate GRAMMATICALITY PRINCIPLES as they have been stipulated so far, to be consisting of:

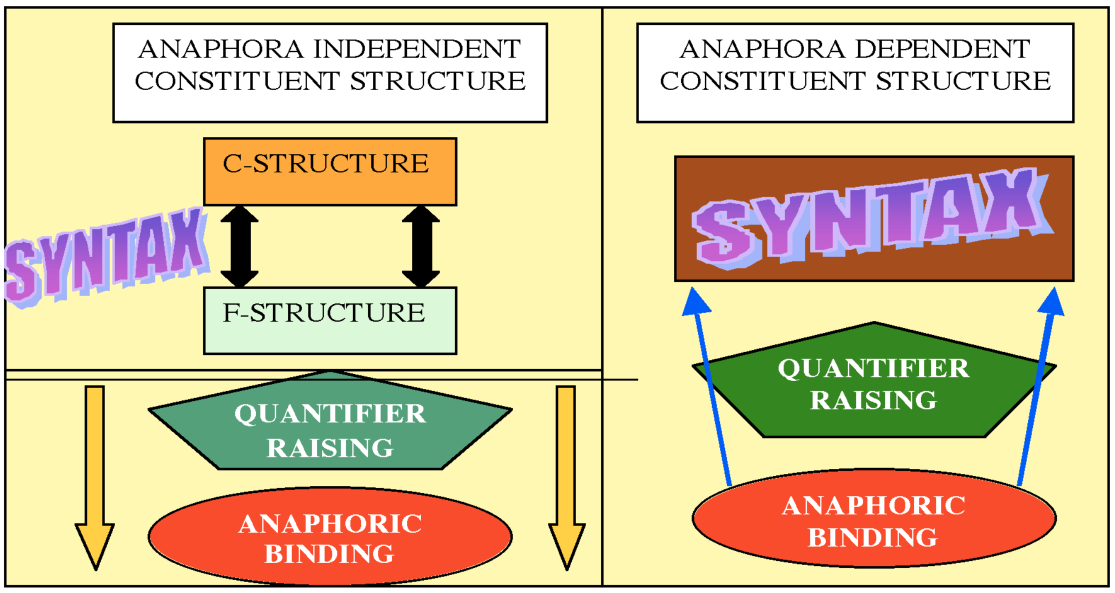
- UNIQUENESS; COHERENCE; COMPLETENESS

with the additional restriction:

- BOUND ANAPHORA AGREEMENT

i.e. short anaphors should be checked before leaving sentence grammar, for agreement with their antecedents iff available in their Minimal Nucleus. In particular, seen that in our framework Quantifier Raising is performed before Anaphoric Binding and will produce new arcs in the graph to represent the scope of quantifiers, this will also undergo failure in order to try a new analysis. This is both time-consuming and unrealistic. A simpler way to solve this problem is to introduce Short Binding as has been defined above. In this way we split Bound Anaphors and make them obey the same principles of Sentence Grammar to which they belong in all respect. In Fig.2 below we show how anaphoric binding and grammatical principles interact. In Fig.2b Anaphoric Binding interacts with Syntax thus causing a failure to take place which cannot be recovered seen that there are other intervening parsing modules. In Fig.2a, on the contrary we postulate the separation between the output of the syntax to be fully autonomous from QR and AB, thus resulting in a more efficient and psychologically viable simulation.

Fig.2a Anaphora Independent Syntactic Parsing **Fig.2b Anaphora Dependent Syntactic Parsing**



4 Experimental Results from Treebanks

We decided to look at corpus data derived from available treebanks in order to ascertain whether the phonemon we are modeling is actually present in real texts. We also wanted to verify whether the RC extraposition was subject to variation from one language to another. We searched in the available treebanks, PennTreebank for English with 1,000,000 tokens, and the Treebank of Italian we are currently working in for syntactic constituenty XML annotation as well as the Venice Treebank made up of approximately the same number of tokens for a total of 300,000 tokens.
We considered only relative clause with morphologically expressed complementizer, thus disregarding all reduced relative clauses. As to the distinction between extraposed vs. non-extraposed we simply looked at the number of brackets – only one - intervening between the constituent label introducing the relative clause in PennTreebank, which is the following (SBAR (WH, and none in the VeniceTreebank. For all remaining cases we counted an extraposed RC.

We tabulated the results in the Table 1. below where we see that Italian is a language much richer on Relative Clauses than American English. In particular the amount of relative clauses in the Italian Venice Treebank is 3 times that of the PT. Yet more interesting seems the ratio of Head Adjacent vs. Non Head

Adjacent RCs: we see that here, whereas Italian has 1 potentially ambiguous RC every 4 RCs, PT has 1 every 6. We can thus conclude that Italian is much more ambiguous to be parsed than English as far as relative clause attachment is concerned.

Table 1. Treebank Derived Structural Relations for Relative Clauses

	Total No. Tokens	Total No. Sentences	Total No. Rel.Cls.	Head Adjacent	Non Head Adjacent
PENN Treebank	**1,000,000**	**44808**	**11559**	**8906 = 77.05%**	**2653 = 22.9%**
SUSANNE Corpus	**130,000**	**7912**	**1380**	**1089 = 78.9%**	**291 = 21.1%**
VENICE Treebank	**300,000**	**11108**	**5155**	**3867 = 75%**	**1288 = 25%**

However, the most interesting fact is constituted by the proportion of relative clauses in relation to the total number of sentences: Generic American English in the Susanne Corpus, counts 1 relative clause every 5/6 sentences; Specialized American English in the PT, goes up to 1 relative clause every 4 sentences. Italian raises the proportion to one relative clause every 2 sentences. Data reported by J.Fodor are in favour of a Head Adjacent use of RC in English, being a language governed by a phonologically related strategy of Minimal Attachment which tends to prevent RC Extraposition. This state of affairs would have RC production in English more restricted than in languages like Italian, which allow for multiple syntactic binders, both adjacent and non-adjacent ones. Data reported in Table 2. seem to support this hypothesis.

Table 2. Treebank Derived Structural Relations for Relative Clauses

	Total No. Tokens	Total No. Sentences	Total No. Rel.Cls.	Complex Relative Clauses	Ratio Rel.Cls. / Sentences	Ratio Rel.Cls /Tot.Tokens
PENN Treebank	**1,000,000**	**44808**	**11559**	**2724**	**25.8%**	**1.16%**
SUSANNE Corpus	**150,000**	**7912**	**1380**	**106**	**17.44%**	**0.92%**
VENICE Treebank	**300,000**	**11108**	**5155**	**-**	**46.40%**	**1.72%**

References

Clifton C., & F. Ferreira(1989), Ambiguity in Context, in G.Altman(ed), Language and Cognitive Processes, op.cit., 77-104.

Delmonte R., D.Bianchi(1991), Binding Pronominals with an LFG Parser, Proceeding of the Second International Workshop on Parsing Technologies, Cancun(Messico), ACL 1991, pp. 59-72.

Delmonte R.(2000), Generating and Parsing Clitics with GETARUN, Proc. CLIN'99, Utrech, pp.13-27.

Delmonte R.(2000),(to appear), Parsing Preferences And Linguistic Strategies, Proc.Workshop Communicating Agents, IKP, Bonn, pp.15.

Delmonte R.(2000), Parsing with GETARUN, Proc.TALN2000, 7° confèrence annuel sur le TALN,Lausanne, pp.133-146.

Fodor J.(2002), Psycholinguistics cannot escape prosody, Invited Talk, SpeechProsody2002, Aix-en-Provence.

Frazier L.(1987a), Sentence processing, in M.Coltheart(ed), Attention and Performance XII, Hillsdale, N.J., Lawrence Elbaum.

D.Hindle & M.Roth(1993), Structural Ambiguity and Lexical Relations, Computational Linguistics 19, 1, 103-120.

Schubert L.K.(1984), On Parsing Preferences, Proc. of COLING, 247-250.

Kay Martin(1980), Algorithm Schemata and Data Structures in Syntactic Processing, CSL-80-12, Xerox Corporation, Palo Alto Research Center.

Pritchett B.L.(1992), Grammatical Competence and Parsing Performance, The University of Chicago Press, Chicago.

A Left Corner Parser for Tree Adjoining Grammars

Víctor J. Díaz[†], Vicente Carrillo[†], and Miguel A. Alonso[‡]
[†]*Universidad de Sevilla and* [‡]*Universidade da Coruña*

1. Introduction

Tabular parsers can be defined as deduction systems where formulas, called items, are sets of complete or incomplete constituents (Sikkel, 1997; Shieber, Schabes and Pereira, 1995). Formally, given an input string $w = a_1 \dots a_n$ with $n \geq 0$ and a grammar G, a parser $\mathbb{P}$ is a tuple $(\mathcal{I}, \mathcal{H}, \mathcal{D})$ where $\mathcal{I}$ is a set of items, $\mathcal{H}$ is a set of hypothesis ($[a_i, i-1, i]$ with $1 \leq i \leq n$) that encodes the input string, and $\mathcal{D}$ is a set of deduction steps that determines how items are combined in order to deduce new items. The deductive approach allows us to establish relations between two parsers in a formal way. One of the most interesting relations between parsers are *filters* because they can be used to improve the performance of tabular parsers in practical cases. The application of a filter to a parser yields a new parser which performs less deductions or contracts sequences of deductions to single deduction steps.

One well-known example of a filter is the relation between Earley and Left Corner (LC) parsers for Context-Free Grammars (CFGs). A LC parser reduces the number of items deduced by Earley's parser using the left corner relation. Given a CFG, the left corner of a non-terminal symbol A is the terminal or non-terminal symbol X if and only if there exists a production $A \to X\nu$ in the grammar, where ν is a sequence of symbols. In the case of $A \to \varepsilon$, we consider ε as the left corner of A. The notion of the left corner relation allow us to rule out the prediction performed on X by an Earley's parser.

Most tabular parsers for Tree Adjoining Grammars (TAGs) are extensions of well-known tabular parser for CFGs. For example, we can cite a number of tabular parsers for TAGs defined on the basis of the Earley's algorithm (Alonso Pardo *et al.*, 1999; Lang, 1990; Joshi and Schabes, 1997; Nederhof, 1999). Although, several approaches have been described to improve the performance of TAGs parsers, most of them based on restrictions in the formalism (Schabes and Waters, 1995) or compilation into finite-state automata (Evans and Weir, 1998), to the best of our knowledge, no attempt has been made to improve the practical performance of Earley-based parsers for TAGs by introducing the left-corner relation.

2. Notation

Let $\mathcal{G} = (V_N, V_T, S, \boldsymbol{I}, \boldsymbol{A})$ be a TAG, where V_N and V_T are the alphabets of non-terminal and terminal symbols, respectively, $S \in V_N$ is the axiom, and $\boldsymbol{I}$ and $\boldsymbol{A}$ are the set of initial and auxiliary trees, respectively. We refer to the root of an elementary tree γ as $\mathbf{R}^\gamma$ and to the foot of an auxiliary tree β as $\mathbf{F}^\beta$. The set $\mathrm{adj}(M^\gamma)$ includes every auxiliary tree that may be adjoined at node M^γ. We use a dummy symbol $\mathbf{nil} \notin \boldsymbol{A}$ for denoting adjoining constraints. If adjunction is not mandatory at M^γ, then $\mathbf{nil} \in \mathrm{adj}(M^\gamma)$. If adjunction is forbidden at M^γ, then $\mathrm{adj}(M^\gamma) = \{\mathbf{nil}\}$. We say M^γ is an adjunction node if there exists an auxiliary tree β which can be adjoined at that node.

Although TAGs are tree-rewriting systems, we can translate every elementary tree γ into a set of productions $\mathcal{P}(\gamma)$. This notation will be useful when defining the set of items for TAGs parsers since dotted productions can be introduced for representing partial parse trees. We define a production $N^\gamma \to N_1^\gamma \dots N_g^\gamma$ for every node N^γ and its ordered g children $N_1^\gamma \dots N_g^\gamma$ in an elementary tree. We refer to the set of productions related to an elementary tree γ as $\mathcal{P}(\gamma)$. For technical reasons, we consider additional productions $\top \to \mathbf{R}^\alpha$, $\top \to \mathbf{R}^\beta$ and $\mathbf{F}^\beta \to \bot$ for each initial tree α and each auxiliary tree β. No auxiliary tree can be adjoined at the two fresh nodes (top) $\top$ and (bottom) $\bot$.

[*] Supported in part by Plan Nacional de Investigación Científica, Desarrollo e Innovación Tecnológica (Grant TIC2000-0370-C02-01), Ministerio de Ciencia y Tecnología (Grant HP2001-0044) and Xunta de Galicia (Grant PGIDT01PXI10506PN).

3. An Earley-based Parser for TAGs

To get a better understanding of our proposal, we first overview the Earley-based parser $\mathbb{P}_{\mathrm{E}}$ for TAGs defined in (Joshi and Schabes, 1997; Alonso Pardo *et al.*, 1999). This parser does not guarantee the valid prefix property.

Given the input string $w = a_1 \ldots a_n$ with $n \geq 0$ and a TAG grammar $\mathcal{G}$, the items in the deductive system $\mathbb{P}_{\mathrm{E}}$ will be of the form:

$$[N^\gamma \rightarrow \delta \bullet \nu, i, j \mid p, q]$$

where $N^\gamma \rightarrow \delta\nu \in \mathcal{P}(\gamma)$ and i, j, p, q are indices related to positions in the input string. The intended meaning of the indices $0 \leq i \leq j$ is that δ spans the substring $a_{i+1} \ldots a_j$. When a node in δ dominates the foot of γ, the values of p and q are known and the substring spanned by the foot is $a_{p+1} \ldots a_q$ with $i \leq p \leq q \leq j$.

As we said, we consider the input string $a_1 \ldots a_n$ is encoded with a set of hypothesis $[a_i, i-1, i]$. Furthermore, being ε the empty word, we assume that $[\varepsilon, i, i]$ trivially holds for each $0 \leq i \leq n$.

We will now introduce the set of deduction steps $\mathcal{D}_{\mathrm{E}}$ for $\mathbb{P}_{\mathrm{E}}$:

$$\mathcal{D}_{\mathrm{E}} = \mathcal{D}_{\mathrm{E}}^{\mathrm{Ini}} \cup \mathcal{D}_{\mathrm{E}}^{\mathrm{Sc}} \cup \mathcal{D}_{\mathrm{E}}^{\mathrm{Pred}} \cup \mathcal{D}_{\mathrm{E}}^{\mathrm{Comp}} \cup \mathcal{D}_{\mathrm{E}}^{\mathrm{AdjPred}} \cup \mathcal{D}_{\mathrm{E}}^{\mathrm{FootPred}} \cup \mathcal{D}_{\mathrm{E}}^{\mathrm{FootComp}} \cup \mathcal{D}_{\mathrm{E}}^{\mathrm{AdjComp}}$$

The recognition process starts by predicting every initial tree:

$$\mathcal{D}_{\mathrm{E}}^{\mathrm{Ini}} = \frac{}{[\top \rightarrow \bullet \mathbf{R}^\alpha, 0, 0 \mid -, -]} \quad \alpha \in \boldsymbol{I}$$

Scanner deduction steps can be applied when the recognition reaches a node V^γ whose label is the empty string or a terminal symbol which matches the current symbol in the input string:

$$\mathcal{D}_{\mathrm{E}}^{\mathrm{Sc}} = \frac{\begin{array}{c}[N^\gamma \rightarrow \delta \bullet V^\gamma \nu, i, j, p, q],\\ [\mathrm{label}(V^\gamma), j, j + |\mathrm{label}(V^\gamma)|]\end{array}}{[N^\gamma \rightarrow \delta V^\gamma \bullet \nu, i, j + |\mathrm{label}(V^\gamma)|, p, q]}$$

where $|w|$ denotes the length of the w word.

The more important deduction steps in the Earley parser for CFGs are those corresponding to predictions and completions. In the case of TAGs, we have three kinds of predictions with their associated completion deduction steps: subtree, adjunction and foot.

- Subtree prediction: This deduction step is similar to predictions in Earley's parser for CFGs. Whenever there is no mandatory adjunction on a node M^γ located in a tree γ, i.e. $\mathbf{nil} \in \mathrm{adj}(M^\gamma)$, we can continue the top-down recognition of the subtree rooted with M^γ:

$$\mathcal{D}_{\mathrm{E}}^{\mathrm{Pred}} = \frac{[N^\gamma \rightarrow \delta \bullet M^\gamma \nu, i, j \mid p, q]}{[M^\gamma \rightarrow \bullet v, j, j \mid -, -]} \quad \mathbf{nil} \in \mathrm{adj}(M^\gamma)$$

- Subtree completion: Once the the subtree rooted with M^γ has been completely recognized, we must continue the bottom-up recognition of the elementary tree γ:

$$\mathcal{D}_{\mathrm{E}}^{\mathrm{Comp}} = \frac{\begin{array}{c}[N^\gamma \rightarrow \delta \bullet M^\gamma \nu, i, j \mid p, q],\\ [M^\gamma \rightarrow v \bullet, j, k \mid p', q']\end{array}}{[N^\gamma \rightarrow \delta M^\gamma \bullet \nu, i, k \mid p \cup p', q \cup q']} \quad \mathbf{nil} \in \mathrm{adj}(M^\gamma)$$

  where $p \cup q = p$ if $q = -$, $p \cup q = q$ if $p = -$, being undefined in other case.

- Adjunction prediction: Let β be an auxiliary tree that can be adjoined on a node M^γ, i.e. $\beta \in \mathrm{adj}(M^\gamma)$. When the recognition of γ reaches M^γ, a new instance of the auxiliary tree β must be predicted:

$$\mathcal{D}_{\mathrm{E}}^{\mathrm{AdjPred}} = \frac{[N^\gamma \rightarrow \delta \bullet M^\gamma \nu, i, j \mid p, q]}{[\top \rightarrow \bullet \mathbf{R}^\beta, j, j \mid -, -]} \quad \beta \in \mathrm{adj}(M^\gamma)$$

- Foot prediction: Considering that $\beta \in \mathrm{adj}(M^\gamma)$, when the recognition of an auxiliary tree β has reached its foot, we must start the recognition of the subtree excised by the adjunction[1]:

$$\mathcal{D}_{\mathrm{E}}^{\mathtt{FootPred}} = \frac{[\mathbf{F}^\beta \to \bullet\bot, k, k \mid -,-]}{[M^\gamma \to \bullet\delta, k, k \mid -,-]} \;\; \beta \in \mathrm{adj}(M^\gamma)$$

- Foot completion: Once the recognition of the excised subtree rooted with M^γ is exhausted we must continue with the recognition of the auxiliary tree β that has been adjoined:

$$\mathcal{D}_{\mathrm{E}}^{\mathtt{FootComp}} = \frac{\begin{array}{c}[M^\gamma \to \delta\bullet, k, l \mid p, q],\\ [\mathbf{F}^\beta \to \bullet\bot, k, k \mid -,-]\end{array}}{[\mathbf{F}^\beta \to \bot\bullet, k, l \mid k, l]} \;\; \beta \in \mathrm{adj}(M^\gamma)$$

- Adjunction completion: Once the recognition of the auxiliary tree β is exhausted, we must continue the recognition of the tree γ where the adjunction was performed:

$$\mathcal{D}_{\mathrm{E}}^{\mathtt{AdjComp}} = \frac{\begin{array}{c}[\top \to \mathbf{R}^\beta\bullet, j, m \mid k, l],\\ [M^\gamma \to \upsilon\bullet, k, l \mid p, q],\\ [N^\gamma \to \delta \bullet M^\gamma\nu, i, j \mid p', q']\end{array}}{[N^\gamma \to \delta M^\gamma \bullet \nu, i, m \mid p \cup p', q \cup q']} \;\; \beta \in \mathrm{adj}(M^\gamma)$$

The input string $a_1 \dots a_n$ belongs to the language defined by the grammar if and only if for some $\alpha \in \boldsymbol{I}$ is obtained a final item:

$$[\top \to \mathbf{R}^\alpha\bullet, 0, n \mid -,-]$$

4. A Left Corner Parser for TAGs

In order to extend the left corner parser for CFGs to the case of TAGs, we need to define the left corner relation on elementary trees, taking into account that we can not miss any admissible adjunction during the recognition. Therefore, an item

$$[N^\gamma \to \delta \bullet M^\gamma\nu, i, j \mid p, q]$$

must be deduced if there exists an auxiliary tree that can be attached to M^γ, even when δ is empty.

Given an elementary tree γ, we say that M^γ is a left corner of N^γ, denoted $N^\gamma >_\ell M^\gamma$, if and only if $N^\gamma \to M^\gamma\mu \in P(\gamma)$ and M^γ is a node with a null adjoining constraint. As usual, we will denote with $>_\ell^*$ the reflexive and transitive closure of the left corner relation.

Informally, left corner relation for TAGs goes down on nodes of elementary trees starting on a node labeled with a non-terminal symbol and ending on an adjunction node, i.e, nodes where an adjunction can be performed. When there not exists such adjunction node, the left corner relation can also end in a $\bot$ node or a node whose label is a terminal symbol or the empty word ε. As it is the case in CFGs parser, the left corner relation for TAGs only depends on the grammar, and it can be computed and stored before applying the parser.

We will go to the definition of the left corner parser $\mathbb{P}_{\mathrm{LC}}$ for TAGs. The set of items and hypothesis for $\mathbb{P}_{\mathrm{LC}}$ is the same as $\mathbb{P}_{\mathrm{E}}$. Left corner relation is applied only in the case of predictive deduction steps. Therefore, while $\mathcal{D}_{\mathrm{E}}^{\mathtt{Ini}}$, $\mathcal{D}_{\mathrm{E}}^{\mathtt{Sc}}$, $\mathcal{D}_{\mathrm{E}}^{\mathtt{Comp}}$,$\mathcal{D}_{\mathrm{E}}^{\mathtt{FootComp}}$ and $\mathcal{D}_{\mathrm{E}}^{\mathtt{AdjComp}}$ remains the same in the left corner parser, we must replace the following: $\mathcal{D}_{\mathrm{E}}^{\mathtt{Pred}}$, $\mathcal{D}_{\mathrm{E}}^{\mathtt{AdjPred}}$ and $\mathcal{D}_{\mathrm{E}}^{\mathtt{FootPred}}$.

4.1. Filtering Subtree Predictions

We now introduce the following deduction steps ($\mathcal{D}_{\mathrm{LC}}^{\mathtt{PredLC}}$, $\mathcal{D}_{\mathrm{LC}}^{\mathtt{Pred}'}$, $\mathcal{D}_{\mathrm{LC}}^{\mathtt{CompLC}}$) replacing $\mathcal{D}_{\mathrm{E}}^{\mathtt{Pred}}$. Given a subtree rooted with M^γ where no adjunction is mandatory, these new steps filter subtree predictions applied on nodes that are left corners of M^γ.

1. The valid prefix property is not fulfilled due to $\mathcal{D}_{\mathrm{E}}^{\mathtt{FootPred}}$ since every subtree rooted with a node M^γ where $\beta \in \mathrm{adj}(M^\gamma)$ is introduced in the recognition.

- In the case that $M^\gamma >_\ell^* O^\gamma$ and the left-most daughter of O^γ is labeled with a terminal symbol or ε, we can go down on the tree directly to that node:

$$\mathcal{D}_{\mathrm{LC}}^{\mathrm{PredLC}} = \frac{\begin{array}{c}[N^\gamma \to \delta \bullet M^\gamma \nu, i, j, p, q],\\ [\mathrm{label}(V^\gamma), j, j + |\mathrm{label}(V^\gamma)|]\end{array}}{[O^\gamma \to V^\gamma \bullet \upsilon, j, j + |\mathrm{label}(V^\gamma)|, -, -]}$$

- In the case that $M^\gamma >_\ell^* O^\gamma$ and O^γ is a node labeled with a non terminal symbol whose left-most daughter P^γ is an adjunction node, we will stop at that node:

$$\mathcal{D}_{\mathrm{LC}}^{\mathrm{Pred}'} = \frac{[N^\gamma \to \delta \bullet M^\gamma \nu, i, j, p, q]}{[O^\gamma \to \bullet P^\gamma \upsilon, j, j, -, -]}$$

- In the bottom-up traversal we should go up on those nodes O^γ and Q^γ that are left corners of M^γ:

$$\mathcal{D}_{\mathrm{LC}}^{\mathrm{CompLC}} = \frac{\begin{array}{c}[N^\gamma \to \delta \bullet M^\gamma \nu, i, j, p, q],\\ [O^\gamma \to \omega \bullet, j, k, p', q']\end{array}}{[Q^\gamma \to O^\gamma \bullet \upsilon, j, k, p', q']}$$

4.2. Filtering Adjunction Predictions

We now explain the set of deduction steps ($\mathcal{D}_{\mathrm{LC}}^{\mathrm{AdjPredLC}}$, $\mathcal{D}_{\mathrm{LC}}^{\mathrm{AdjPred}'}$ and $\mathcal{D}_{\mathrm{LC}}^{\mathrm{AdjCompLC}}$) replacing $\mathcal{D}_{\mathrm{E}}^{\mathrm{AdjPred}}$. Let M^γ be a node in an elementary tree γ where the auxiliary tree β can be adjoined. These new deduction steps filter predictions on those nodes that are left corners of the top node of β.

- When $\top >_\ell^* O^\beta$ and the left-most daughter of O^β is a node V^β labeled with a terminal symbol or ε, we will apply:

$$\mathcal{D}_{\mathrm{LC}}^{\mathrm{AdjPredLC}} = \frac{\begin{array}{c}[N^\gamma \to \delta \bullet M^\gamma \nu, i, j, p, q],\\ [\mathrm{label}(V^\beta), j, j + |\mathrm{label}(V^\beta)|]\end{array}}{[O^\beta \to V^\beta \bullet \upsilon, j, j + |\mathrm{label}(V^\beta)|, -, -]}$$

- When $\top >_\ell^* O^\beta$ and O^β dominates on the left a node P^β such that either P^β is an adjunction node or P^β is the $\perp$ node of β, we will apply:

$$\mathcal{D}_{\mathrm{LC}}^{\mathrm{AdjPred}'} = \frac{[N^\gamma \to \delta \bullet M^\gamma \nu, i, j, p, q]}{[O^\beta \to \bullet P^\beta \upsilon, j, j, -, -]}$$

- During the bottom-up recognition we must go up on the tree through the nodes O^β that are left corners of the top node of β:

$$\mathcal{D}_{\mathrm{LC}}^{\mathrm{AdjCompLC}} = \frac{\begin{array}{c}[N^\gamma \to \delta \bullet M^\gamma \nu, i, j, p, q],\\ [O^\beta \to \omega \bullet, j, k, p', q']\end{array}}{[Q^\beta \to O^\beta \bullet \upsilon, j, k, p', q']}$$

4.3. Filtering Foot Predictions

We now show the set of deduction steps ($\mathcal{D}_{\mathrm{LC}}^{\mathrm{FootPredLC}}$, $\mathcal{D}_{\mathrm{LC}}^{\mathrm{FootPred}'}$ and $\mathcal{D}_{\mathrm{LC}}^{\mathrm{FootCompLC}}$) replacing $\mathcal{D}_{\mathrm{E}}^{\mathrm{FootPred}}$. Let M^γ be a node in an elementary tree γ such that β can be adjoined. Suppose the recognition has reached a node E^β where it is not mandatory to perform an adjunction and that $E^\beta >_\ell^* \perp$. These new deduction steps filter predictions on nodes belonging to the auxiliary tree β and to the elementary tree γ where the adjunction is performed:

- When $M^\gamma >_\ell^* O^\gamma$ and the left-most daughter of O^γ is a node V^γ labeled with a terminal symbol or ε we will apply:

$$\mathcal{D}_{\mathrm{LC}}^{\mathrm{FootPredLC}} = \frac{\begin{array}{c}[D^\beta \to \delta \bullet E^\beta \nu, j, k, -, -],\\ [\mathrm{label}(V^\gamma), k, k + |\mathrm{label}(V^\gamma)|]\end{array}}{[O^\gamma \to V^\gamma \bullet \upsilon, k, k + |\mathrm{label}(V^\gamma)|, -, -]}$$

Sentence	Time Reduction	Items Reduction
Srini bought a book	-3%	-44%
Srini bought Beth a book	-5%	-47%
Srini bought a book at the bookstore	-6%	-46%
he put the book on the table	-8%	-44%
$*$ he put the book	-13%	-42%
the sun melted the ice	-11%	-48%
the ice melted	-14%	-46%
Elmo borrowed a book	-7%	-45%
$*$ a book borrowed	+5%	-41%
he hopes Muriel wins	-12%	-49%
he hopes that Muriel wins	-16%	-49%
the man who Muriel likes bought a book	+6%	-42%
the man that Muriel likes bought a book	-2%	-44%
the music should have been being played for the president	-28%	-56%
Clove caught a frisbee	-4%	-45%
who caught a frisbee	-7%	-45%
what did Clove catch	-13%	-49%
the aardvark smells terrible	-4%	-46%
the emu thinks that the aardvark smells terrible	-12%	-48%
who does the emu think smells terrible	-14%	-49%
who did the elephant think the panda heard the emu said smells terrible	-14%	-49%
Herbert is angry	-24%	-53%
Herbert is angry and furious	-21%	-54%
Herbert is more livid than angry	-25%	-51%
Herbert is more livid and furious than angry	-18%	-50%

Table 1: Results of the experiment based on XTAG

- In the case $M^\gamma >_\ell^* O^\gamma$ and the left-most daughter of O^γ is a node P^γ such that either P^γ is an adjunction node or P^γ is the $\perp$ node of γ, prediction is stopped at P^γ:

$$\mathcal{D}_{\mathrm{LC}}^{\mathrm{FootPred}'} = \frac{[D^\beta \rightarrow \delta \bullet E^\beta \nu, j, k, -, -]}{[O^\gamma \rightarrow \bullet P^\gamma \upsilon, k, k, -, -]}$$

- During the bottom-up recognition we must go up on the tree through the nodes O^γ that are left corners of M^γ, the node where the adjunction was performed, guaranteeing we do not go up beyond M^γ itself, i.e. $M^\gamma \neq O^\gamma$:

$$\mathcal{D}_{\mathrm{LC}}^{\mathrm{FootCompLC}} = \frac{\begin{array}{c}[D^\beta \rightarrow \delta \bullet E^\beta \nu, j, k, -, -],\\ [O^\gamma \rightarrow \omega \bullet, k, l, p, q]\end{array}}{[Q^\gamma \rightarrow O^\gamma \bullet \upsilon, k, l, p, q]}$$

5. Experimental results

The time complexity of the algorithm with respect to the length n of the input string is $O(n^6)$ for both parsers. The improvement in the performance of Left Corner parsers comes from the reduction in the size of the chart (the set of deduced items). It is clear that this reduction depends on the grammar and the input string considered. We have made a preliminary study where we have tested and compared the behavior of the LC parser and the Earley-based parser explained before.

We have incorporated both parsers into a naive implementation in Prolog of the deductive parsing machine presented in (Shieber, Schabes and Pereira, 1995). We have taken a subset of the XTAG grammar (XTAG Research Group, 2001), consisting of 27 elementary trees that cover a variety of English constructions: relative clauses, auxiliary verbs, unbounded dependencies, extraction, etc. In order to eliminate the time spent by unification, we have not considered the feature structures of elementary trees. Instead, we have simulated the features using local constraints. Every sentence has been parsed without previous filtering of elementary trees. Table 1 includes the reduction ratio with respect to the parsing time in seconds and with respect to the chart size. Briefly, we can remark that LC parser shows on average a time reduction of 11% and a chart size reduction of 50%.

6. Conclusion

We have defined a new parser for TAG that is an extension of the Left Corner parser for Context Free Grammars. The new parser can be view as a filter on an Earley-based parser for TAGs where the number of predictions is reduced due to the generalized left corner relation that we have established on the nodes of elementary trees. The worst-case complexity with respect to space and time is the standard one for TAG parsing, but preliminary experiments have shown a better performance than classical Earley-based parsers for TAG. Finally, as further work, we are investigating the conditions the parser should satisfy in order to guarantee the valid prefix property.

References

Alonso Pardo, Miguel A., David Cabrero Souto, Eric de la Clergerie and Manuel Vilares Ferro. 1999. Tabular Algorithms for TAG Parsing. In *Proc. of EACL'99, Ninth Conference of the European Chapter of the Association for Computational Linguistics*, pages 150–157, Bergen, Norway, June. ACL.

Evans, Roger and David Weir. 1998. A structure-sharing parser for lexicalized grammars. In *COLING-ACL'98, 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Proceedings of the Conference*, volume I, pages 372–378, Montreal, Quebec, Canada, August. ACL.

Joshi, Aravind K. and Yves Schabes. 1997. Tree-Adjoining Grammars. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages. Vol 3: Beyond Words*. Springer-Verlag, Berlin/Heidelberg/New York, chapter 2, pages 69–123.

Lang, Bernard. 1990. The systematic construction of Earley parsers: Application to the production of $\mathcal{O}(n^6)$ Earley parsers for tree adjoining grammars. In *Proc. of the 1st International Workshop on Tree Adjoining Grammars*, August.

Nederhof, Mark-Jan. 1999. The Computational Complexity of the Correct-Prefix Property for TAGs. *Computational Linguistics*, 25(3):345–360.

Schabes, Yves and Richard C. Waters. 1995. Tree Insertion Grammar: A Cubic-Time Parsable Formalism That Lexicalizes Context-Free Grammar Without Changing the Trees Produced. *Computational Linguistics*, 21(4):479–513, December. Also as Technical Report TR-94-13, June 1994, Mitsubishi Electric Research Laboratories, Cambridge, MA, USA.

Shieber, Stuart M., Yves Schabes and Fernando C. N. Pereira. 1995. Principles and Implementation of Deductive Parsing. *Journal of Logic Programming*, 24(1–2):3–36, July-August.

Sikkel, Klaas. 1997. *Parsing Schemata — A Framework for Specification and Analysis of Parsing Algorithms*. Texts in Theoretical Computer Science — An EATCS Series. Berlin/Heidelberg/New York: Springer-Verlag.

XTAG Research Group. 2001. *A Lexicalized Tree Adjoining Grammar for English*. Technical Report IRCS-01-03, University of Pennsylvania, USA.

Context-Free Parsing of a Tree Adjoining Grammar Using Finite-State Machines

Alexis Nasr*, Owen Rambow†, John Chen‡, and Srinivas Bangalore‡
* *Université Paris 7,* † *University of Pennsylvania,* ‡ *AT&T Labs – Research*
nasr@linguist.jussieu.fr, rambow@unagi.cis.upenn.edu, jchen@research.att.com, srini@research.att.com

1. Introduction: A Dependency-Only Parser

In this paper, we describe work in progress that originated with the goal of creating a dependency parser which does not use phrase structure, but does use an explicit generative (and of course lexicalized) encoding of the grammar.[1] The motivation for this goal is threefold.

- First, we believe that the phrase structure used by many (if not most) linguistic theories is useful only in deriving the syntactic behavior of lexical heads (or classes of lexical heads) from more general principles(which is of course the goal of syntactic theory); once derived, the syntactic behavior can be expressed in simpler terms that can be easier to manipulate in a computational setting.[2] Furthermore, applications need as output from a parser something close to dependency (typically, lexical predicate-argument structure), but not phrase-structure.
- Second, we prefer an explicit encoding of a grammar in an inspectable declarative syntactic formalism over an implicit encoding in a computational framework. While we believe that explicit grammars have various advantages, such as the ability to test different syntactic hypotheses, or the possibility of hand-crafting grammars for limited domains (as is commonly done in commercial speech-based systems), we do not attempt to justify this preference further.
- Third, we believe that generative formalisms have certain advantages over constraint-based formalisms, in particular computational advantages. A large body of research on parsing in generative formalisms can be reused for different formalisms if they share certain basic properties, and we exploit this fact.

The formalism we used is based on that presented in (Kahane et al., 1998), but in the first phase we have omitted the non-projectivity, leaving us with a simple generative algorithm which allows us to lexicalize a context-free grammar by allowing regular expressions in the right-hand side.[3] We call this formalism GDG,or Generative Dependency Grammar. Its parsing algorithm, naturally, expresses the regular expressions as finite-state machines (FSMs), and the chart parser records FSMs and the state they are in during the parse.

It is well known that the derivation tree of a Tree Adjoining Grammar (TAG) is a dependency tree if the grammar is lexicalized (though not always the linguistically most plausible one), and as a result, there are many parallels between a dependency analysis and a TAG analysis if we abstract from the phrase structure of the latter (Rambow and Joshi, 1997). In fact, a dependency parse can be seen as a direct parse of a TAG derivation tree. And furthermore, we can derive a grammar in our lexicalized GDG formalism from a TAG grammar in a relatively straightforward manner. This has advantages in grammar reuse. Given the close relation between dependency and TAG, our approach can be seen as an exercise in applying the FSM-based parsing techniques of Evans and Weir (1997) and (1998) to the Tree Insertion Grammars (TIG) of Schabes and Waters (1995), though the different origin of the approach (dependency parsing vs. TAG parsing) translates to differences in the way the FSMs are used to parse.

In this overview, we introduce GDG and its parsing algorithm in Section 2, and describe how we compile a TAG to a GDG in Section 3. We discuss the empirical adequacy and speed of the parser in Section 4, and conclude with a discussion of how our projected work relates to other current work in parsing. We postpone a discussion of related work in formalizing dependency to another publication.

1. We would like to thank David Chiang, Anoop Sarkar, and three insightfully critical anonymous reviewers for their comments, explanations, and suggestions.
2. We leave open whether this point also has cognitive relevance.
3. In this we follow a suggestion made by Abney (1996), essentially extending the formalism of (Gaifman, 1965).

2. GDG and Parsing GDG

An Extended Context-Free Grammar (or ECFG for short) is like a CFG, except that the right-hand side is a regular expression over the terminal and nonterminal symbols of the grammar.[4] At each step in a derivation, we first choose a rewrite rule (as we do in CFG), and then we choose a string of terminal and nonterminal symbols which is in the language denoted by the regular expression associated with the rule. This string is then treated like the right-hand side of a CFG rewrite rule. A Generative Dependency Grammar (GDG) is a lexicalized ECFG. For our formalism, this means that the regular expression in a production is such that each string in its denoted language contains at least one terminal symbol. When we use a GDG for linguistic description, its left-hand side nonterminal will be interpreted as the lexical category of the lexical item and will represent its maximal projection. The right-hand side symbols represent the active valency of the lexical head of the rule, i.e., the categories that must or may (depending on whether the symbol is optional) be dependents on this lexical head for it to be the root of a well-formed subtree. Passive valency (a representation of where and to what other heads a lexical head may attach itself) is not explicitly encoded.

The parsing algorithm is a simple extension of the CKY algorithm for CFG. The difference is in the use of finite state machines in the items in the chart to represent the right-hand sides of the rules of the ECFG. A rule with category C as its left-hand side will give rise to a finite state machine which we call a **C-FSM**; its final states mark the completed recognition of a constituent of label C. Let T be the parse table for input sentence W and GDG G such that $T_{i,j}$ contains (M, q) iff M is a C-FSM, q is one of the final states of M, and we have a derivation C of substring $w_i \cdots w_j$ from C in G.

Initialization: We start by adding, for each i, $1 \leq i \leq n$, w_i to $T_{i,i}$.

Completion: If $T_{i,j}$ contains either the input symbol w or an item (M, q) such that q is a final state of M, and M is a C-FSM, then add to $T_{i,j}$ all (M', q') such that M' is a rule-FSM which transitions from a start state to state q' on input w or C.

Scanning: If (M_1, q_1) is in $T_{i,k}$, and $T_{k+1,j}$ contains either the input symbol w or the item (M_2, q_2) where q_2 is a final state and M_2 is a C-FSM, then add (M_1, q) to $T_{i,j}$ (if not already present) if M_1 transitions from q_1 to q on either w or C.

Note that because we are using a dependency grammar (or, in TAG terms, parsing the derivation tree directly), each scanning step corresponds to one attachment of a lexical head to another (or, in TAG terms, to an adjunction or a substitution). At the end of the parsing process, a packed parse forest has been built. The nonterminal nodes are labeled with pairs (M, q) where M is an rule FSM and q a state of this FSM.

Obtaining the dependency trees from the packed parse forest is performed in two stages. In a first stage, a forest of binary syntagmatic trees is obtained from the packed forest and in a second stage, each syntagmatic tree is transformed into a dependency tree.

3. Compilation of a TAG into GDG

In fact, we do not derive a formal GDG from a TAG but instead directly compile a set of FSMs which are used by the parser (though we consider the distinction irrelevant, as FSMs and regular expressions are easily inter-convertible). To derive a set of FSMs from a TAG, we do a depth-first traversal of each elementary tree in the grammar (but excluding the root and foot nodes of adjunct auxiliary trees) to obtain a sequence of nonterminal nodes. Each node becomes two states of the FSM, one state representing the node on the downward traversal on the left side (the **left node state**), the other representing the state on the upward traversal, on the right side (the **right node state**). For leaf nodes, the two states immediately follow one another. The states are linearly connected with ϵ-transitions, with the left node state of the root node the start state, and its right node state the final state (except for predicative auxiliary trees – see below). To each non-leaf state, we add one self loop transition for each tree in the grammar that can adjoin at that state from the specified direction (i.e., for a state representing a node on the downward traversal, the auxiliary tree must adjoin from the left), labeled with the tree name. For each pair of adjacent states representing a substitution node, we add transitions between them labeled with the names of the trees that can substitute there. For the lexical head, we add a transition on that head. For footnodes of predicative

4. ECFG has been around informally since the sixties (e.g., the Backus-Naur form); for formalizations see (Madsen and Kristensen, 1976) or Albert et al. (1999).

auxiliary trees which are left auxiliary trees (in the sense of Schabes and Waters (1995), i.e., all nonempty frontier nodes are to the left of the footnode), we take the left node state as the final state. There are no other types of leaf nodes since we do not traverse the passive valency structure of adjunct auxiliary tees. Note that the treatment of footnodes makes it impossible to deal with trees that have terminal, substitution or active adjunction nodes on both sides of a footnode. It is this situation (iterated, of course) that makes TAG formally more powerful than CFG; in linguistic uses, it is very rare.[5] The result of this phase of the conversion is a set of FSMs, one per elementary tree of the grammar, whose transitions refer to other FSMs. We give a sample grammar and the result of converting it to FSMs in Figure 1.

The construction treats a TAG as if were a TIG (or, put differently, it coerces a TAG to be a TIG): during the traversal, both terminal nodes and nonterminal (i.e., substitution) nodes between the footnode and the root node are ignored (because the traversal stops at the footnode), thus imposing the constraint that the trees may not be wrapping trees and that no further adjunction may occur to the right of the spine in a left auxiliary tree. Furthermore, by modeling adjunction as a loop transition, we adopt the definition of adjunction of Schabes and Shieber (1994), as does TIG. Chiang (2000) also parses with an automatically extracted TIG, but unlike our approach, he uses standard TAG/TIG parsing techniques. Rogers (1994) proposes a different context-free variant, "regular-form TAG". The set of regular-form TAGs is a superset of the set of TIGs, and our construction cannot capture the added expressive power of regular-form TAG.

As mentioned above, this approach is very similar to that of Evans and Weir (1997). One important difference is that they model TAG, while we model TIG. Another difference is that they use FSMs to encode the sequence of actions that need to be taken during a standard TAG parse (i.e., of the derived tree), while we encode the active valency of the lexical head in the FSM. A result, in retrieving the derivation tree, each item in the parse tree corresponds to an attachment of one word to another, and there are fewer items. Furthermore, our FSMs are built left-to-right, while Evans and Weir only explore FSMs constructed bottom-up from the lexical anchor of the tree. As a result, we can perform a strict left-to-right parse, which is not straightforwardly possible in TAG parsing using FSMs.

4. Adequacy of GDG and Initial Run-Time Results

To investigate the adequacy of a context-free parser for English, as well as the speed of the parser, we use an automatically extracted grammar called "Bob" (Chen, 2001). Bob has been extracted from Sections 02-21 of the Penn Treebank. Parameters of extraction have been set so that the tree frames of the resulting grammar have a "moderate" domain of locality, and preserve many but not all of the empty elements that are found in the Penn Treebank (typically those cases where empty elements are found in the XTAG grammar, such as PRO subjects). Bob consists of 4909 tree frames. We tested our parser with Bob on 1,814 sentences of Section 00 of the PTB with an average sentence length of 21.2 words (excluding pure punctuation, i.e., punctuation which does not play a syntactic role such as conjunction or apposition). We evaluate performance using **accuracy**, the ratio of the number of dependency arcs which are correctly found (same head and daughter nodes) in the best parse for each sentence to the number of arcs in the entire test corpus. We also report the percentage of sentences for which we find the correct analysis (along with many others, of course).

To show that GDG is adequate for parsing English (an empirical question), we use the correct supertag associated with each input word and evaluate the performance of the parser. We expect only those sentences which do not have a context-free analysis not to have any analysis at all. This is indeed the case: there is no case of non-projectivity in the test corpus. Note that since we analyze matrix verbs as depending on their embedded verb, following the TAG analysis, long-distance *wh*-movement is not in fact non-projective or us. However, the punctuation mark associated with the matrix verb does cause non-projectivity, but since we are disregarding true punctuation, this does not affect our result.

The average run-time for each sentence is 56ms (parsing using the correct supertag for each word, and no other supertag), which to our knowledge is significantly quicker than existing full-fledged TAG parsers.[6] We show the

5. Our construction cannot handle Dutch cross-serial dependencies (not surprisingly), but it can convert the TAG analysis of *wh*-movement in English and similar languages.

6. Note that the extracted grammar does not have any features, so no feature unification is performed during parsing. Agreement phenomena can be enforced by using extended label sets, at the cost of increasing the size of the grammar. (This is a parameter in the extraction algorithm.) Of course, features in TAG are always bounded in size anyway, and hence always equivalent to an extended label set.

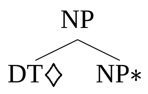
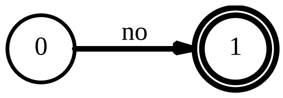
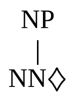
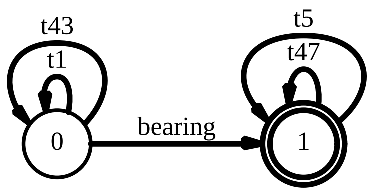
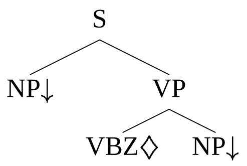
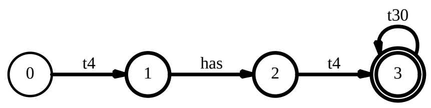
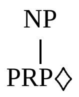
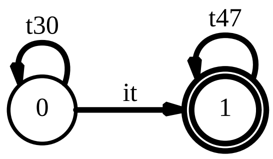
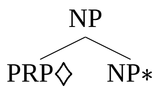
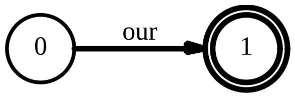
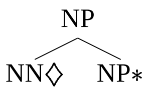
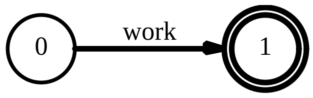
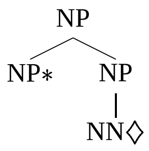
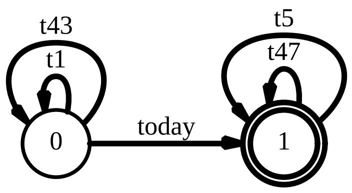
Name	Tree	FSM
t1[no]		
t4[bearing] t4[force]		
t31[has]		
t34[it]		
t36[our]		
t43[work]		
t47[today]		

Figure 1: Bob subgrammar for sentence *It has no bearing on our workforce today* and set of FSMs that corresponds to that subgrammar

dependence of parse time on sentence length in Figure 2. As is known both theoretically and empirically (Sarkar, 1998), lexical ambiguity has a drastic effect on parse time. We have not yet run experiments to show the relation (and, as mentioned above, this is where we hope to profit from using FSMs).

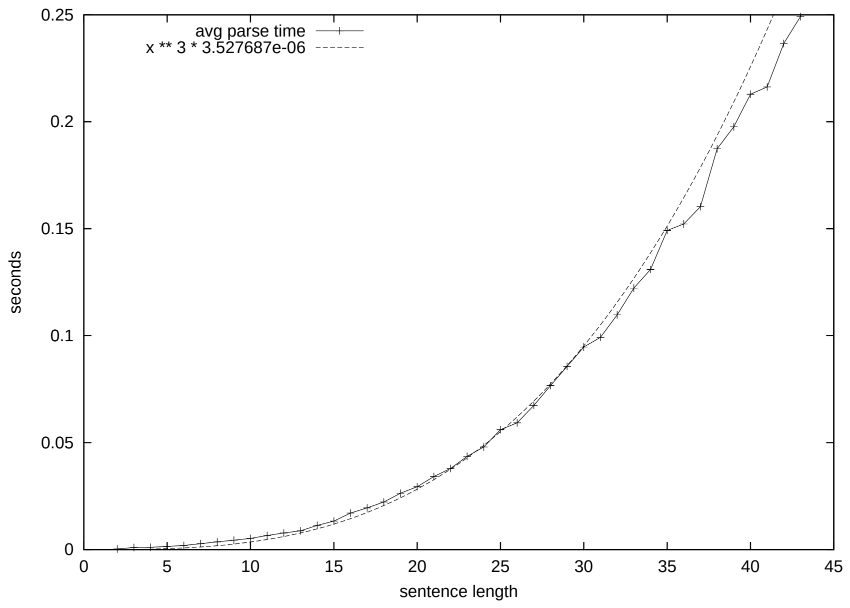


Figure 2: Average analysis time (in ms) against sentence length

5. Parsing Using Lexicalized Grammars

Recently, parsers using bilexical stochastic models which are derived from phrase-structure corpora with the aid of "head percolation tables" have been successfully developed (Magerman, 1995; Collins, 1997)). In some sense, these approaches use an unlexicalized formalism (CFG) and implicitly lexicalize it.[7] The use of the same tool (head percolation tables), often the same tables, in automatically extracting TAG grammars raises the question whether a stochastic TAG parser or TIG parser (such as (Chiang, 2000)) using bilexical stochastic models can ever outperform the implicitly lexicalized approaches, since they use the same kind of data, though packaged differently (David Chiang, personal communication). We suggest that the interest in pursuing parsing using explicitly lexicalized grammar formalisms such as TAG, TIG, or GDG lies not in a simple replication of the results obtained previously using head percolation. Rather, the lexicalized formalisms allow for a different approach, which is not possible with the implicitly lexicalized approaches: namely gathering as much syntactic information as possible about the syntactic properties of the words of the input sentence using very fast algorithms prior to (or interleaved with, but separate from) parsing.[8] The best-known such approach is supertagging (Bangalore and Joshi, 1999). While supertagging has been explored in the past (Bangalore and Joshi, 1999; Chen, 2001), we are not aware of a systematic investigation into the relation between the quality of supertagging, the use of n-best supertagging, parse quality, and parse time (but see initial results in Chen et al. (2002)). We intend to perform such investigations using the framework we have developed, and hope that the use of an explicit lexicalized grammar formalism can be shown to be useful precisely because it allows us to use other disambiguation techniques which the implicitly lexicalized formalisms do not support.

7. Of course, this work can also be interpreted as a rediscovery of the core insight of TAG.
8. Some parsers make use of prior part-of-speech parsing, which provides some very shallow syntactic information.

References

Abney, Steven (1996). A grammar of projections. Unpublished manuscript, Universität Tübingen.

Albert, Jürgen; Guammarresi, Dora; and Wood, Derick (1999). Extended context-free grammars and normal form algorithms. In Champarnaud, Jean-Marc; Maurel, Denis; and Ziadi, Djelloul, editors, *Automata Implementations: Third International Workshop on Implementing Automata (WIA'98)*, volume 1660 of *LNCS*, pages 1–12. Springer Verlag.

Bangalore, Srinivas and Joshi, Aravind (1999). Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2):237–266.

Chen, John (2001). *Towards Efficient Statistical Parsing Using Lexicalized Grammatical Information*. PhD thesis, University of Delaware.

Chen, John; Bangalore, Srinivas; Collins, Michael; and Rambow, Owen (2002). Reranking an n-gram SuperTagger. In *Proceedings of the Sixth International Workshop on Tree Adjoining Grammar and Related Formalisms (TAG+6)*, Venice, Italy.

Chiang, David (2000). Statistical parsing with an automatically-extracted tree adjoining grammar. In *38th Meeting of the Association for Computational Linguistics (ACL'00)*, pages 456–463, Hong Kong, China.

Collins, Michael (1997). Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, Madrid, Spain.

E. Black et al. (1991). A procedure for quantitatively comparing the syntactic coverage of english grammars. In *Proceedings of the February 1991 DARPA Speech and Natural Language Workshop*. DARPA.

Evans, Roger and Weir, David (1997). Automaton-based parsing for lexicalized grammars. In *5th International Workshop on Parsing Technologies (IWPT97)*, pages 66–76.

Evans, Roger and Weir, David (1998). A structure-sharing parser for lexicalized grammars. In *36th Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL'98)*, pages 372–378, Montréal, Canada.

Gaifman, Haim (1965). Dependency systems and phrase-structure systems. *Information and Control*, 8:304–337.

Kahane, Sylvain; Nasr, Alexis; and Rambow, Owen (1998). Pseudo-projectivity: A polynomially parsable non-projective dependency grammar. In *36th Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL'98)*, pages 646–652, Montréal, Canada.

Madsen, O.L. and Kristensen, B.B. (1976). *LR*-parsing of extended context-free grammars. *Acta Informatica*, 7:61–73.

Magerman, David (1995). Statistical decision-tree models for parsing. In *33rd Meeting of the Association for Computational Linguistics (ACL'95)*.

Rambow, Owen and Joshi, Aravind (1997). A formal look at dependency grammars and phrase-structure grammars, with special consideration of word-order phenomena. In Wanner, Leo, editor, *Recent Trends in Meaning-Text Theory*, pages 167–190. John Benjamins, Amsterdam and Philadelphia.

Rogers, James (1994). Capturing cfls with Tree Adjoining Grammars. In *32nd Meeting of the Association for Computational Linguistics (ACL'94)*. ACL.

Sarkar, Anoop (1998). Conditions on Consistency of Probabilistic Tree Adjoining Grammars. In *36th Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL'98)*, pages 1164–1170, Montreal.

Schabes, Yves and Shieber, Stuart (1994). An alternative conception of tree-adjoining derivation. *Computational Linguistics*, 1(20):91–124.

Schabes, Yves and Waters, Richard C. (1995). Tree Insertion Grammar: A cubic-time, parsable formalism that lexicalizes Context-Free Grammar without changing the trees produced. *Computational Linguistics*, 21(4):479–514.

How to prevent adjoining in TAGs
and its impact on the Average Case Complexity

Jens Woch
University of Koblenz, Germany

The introduction of the adjoining operation to context-free grammars comes at high costs: The worst case time complexity of (Earley, 1968) is $O(n^3)$, whereas Tree Adjoining Grammars have $O(n^6)$ ((Schabes, 1990)). Thus, avoiding adjoining as far as possible seems to be a good idea for reducing costs (e.g. (Kempen and Harbusch, 1998)). Tree Insertion Grammars (TIGs, (Schabes and Waters, 1995)) address this problem more radically by restricting the adjoining operation of TAGs such that it is no context-sensitive operation anymore. The result is $O(n^3)$ worst case parseability which stems from TIG's context-freeness. However, to preserve TAG's mildly context-sensitiveness the adjoining operation must not be restricted in any way. Another solution would be simply to call the adjoining operation less frequently: The production of items directly depends on the fashion of the underlying grammar and often adjoining is used to make the grammar more comprehensible or more adequate to the linguistic phenomenon even if there would be simpler representations as, for instance, left- or right recursion.

This abstract (1st) sketches a way of reducing item generation through grammar transformation by using Schema-TAGs (S-TAGs, as introduced by (Weir, 1987), where tree sets are enumerated by regular expressions) which in contrast to TIGs keeps weak equivalence and performs better than factoring as proposed by (Harbusch, Widmann and Woch, 1998), and (2nd) provides a proof of the average case time complexity of S-TAGs based on the proposed transformation.

In the following, adressing of nodes occurs in the fashion of (Gorn, 1967), i.e. each node of a tree gets a unique number – beginning with zero – which preserves the structure of the tree. For example, 1.2 points to the second daughter of the first daughter of a root node, and in grammar G_1 of Fig. 2, $(A_2, 2)$ identifies the foot node of A_2. The regular expressions of S-TAGs are defined as "schema descriptors": Let g be a Gorn number, then

- $|g|$ is a schema descriptor.
- If μ and ν are schema descriptors, then $\mu + \nu$, describing the alternative between μ and ν, is a schema descriptor.
- If μ and ν are schema descriptors, then $\mu.\nu$, describing the concatenation of μ and ν, is a schema descriptor.
- If μ is a schema descriptor, so are (μ) (bracketing), μ^* (arbitrary iteration) and $\mu^{(0|n)}, n \in \mathbb{N}$ (n times iteration) schema descriptors.
- If $|g|$ is a schema descriptor, so is $|n - n.g|, n \in \mathbb{N}$ a schema descriptor (the via g addressed subtree is cut out from the via n addressed (sub)tree).

Reduction of item production by factoring

A first idea of circumventing adjoining is to avoid it by reducing generation of items, on which *Predict* can be applied. Thus, the idea is to aggregate common substructures appropriately, i.e. to condense the grammar in order to get rid of redundancies (Harbusch, Widmann and Woch, 1998).

In general, factoring is applied to the schema descriptors (the regular expressions) of the S-TAG and can be done by applying, e.g., the following rules:

(1) $\alpha.\gamma^{(l|k)}.\gamma.\beta = \alpha.\gamma.\gamma^{(l|k)}.\beta = \alpha.\gamma^{(l+1|k+1)}.\beta$
(2) $\alpha.(\gamma.\delta_1).\beta + \cdots + \alpha.(\gamma.\delta_m).\beta = \alpha.\gamma.(\delta_1 + \cdots + \delta_m).\beta$
(3) $\alpha.\gamma.\beta + \alpha.\beta = \alpha.\gamma^{(0|1)}.\beta$
(4) $\alpha.(\gamma + \gamma).\beta = \alpha.\gamma.\beta$

where $\alpha, \beta, \gamma, \delta_1, ..., \delta_m$ are arbitrary complex schema descriptors. Rule (1) saves nothing, since the amount of predictions remains the same, as well as the amount of alternatives $k - l$. However, applying it reversely, in combination with rule (4) in case of alternatives it is probably possible to factor γ up to l times, such that $\gamma^{(0|k-l)}$ remains as infix, which is not reduce-able any more. The factoring of rule (2) not only saves $m - 1$ predictions of γ, but also of α and β. Thus, rule (2) saves $3(m - 1)$ predictions. Finally, rule (3) reduces the number of predictions of α and β down to 1, while rule (4) divides the number of predictions caused by γ by 2.

Generally, the above rules save $\sum_{i=1}^{|N|} c|ADJ(\eta_i)|$ predictions and $\sum_{i=1}^{|N|} m_i c|ADJ(\eta_i)|$ completions with m_i being the number of occurrences of node $\eta_i \in N$, which can be factored out c times. Obviously, each circumvented prediction saves the whole cycle of left and right prediction, resp. completion. Additionally, and much more important, it reduces the amount of hypothesis checking needed within the operations, as expressed by the factor m_i of the above sum.

Factoring, however, is not able to effectively prevent (unnecessary) adjoinings, it simply reduces redundancies. This strategy only pays back if direct processing mechanisms are applied, who do not explicitely iterate alternatives of schemata as instances of trees (cf. (Harbusch and Woch, 2000)).

Preventing adjoining (at least, partially)

A more promising approach to circumvent adjoining is to prevent it. The use of null- and selective constraints, which restrict adjoining at the node, at which they are attached, perform this very good, but the application of constraints always should be linguistically motivated. However, adjoining is often used for modelling simple left, resp. right recursion of the context-free form $S \to \alpha S$, resp. $S \to S\alpha$. Auxiliary trees, whose terminal nodes are unexceptional either to the left or to the right of the root-to-foot spine are modelling exactly this behaviour. Transforming those trees do not compromise the underlying linguistic concept. Those trees can be merged with the help of schema descriptors into the trees, in which they should adjoin and after merging can completely be removed from the auxiliary tree set. Hence their prediction is effectively prevented.

Definition 1 (Simplified S-TAG (S^2-TAG)) *Let the 6-tuple $G = (N,T,S,I,A,\Sigma)$ be a Schema-TAG.*

- *The tree schema $\beta \in A$ is called right recursive, if and only if for each tree $t_{|\beta}$ holds, that the foot node of t is the rightmost node of the frontier of t.*
- *The tree schema $\beta \in A$ is called left recursive, if and only if for each tree $t_{|\beta}$ holds, that the foot node of t is the leftmost node of the frontier of t.*

G is called Simplified S-TAG, or S^2-TAG, if for each $\beta \in A$ holds, that β is neither left nor right recursive.

Theorem 1 (L(S^2-TAG) = L(S-TAG)) *Let G be an S-TAG and G' be an S^2-TAG. Then $L(G) = L(G')$.*

This is shown by transforming auxiliary left and right recursive tree schemata into equivalent schema descriptor representations and *vice versa*. Def. 1 states, that an auxiliary schema β is right/left recursive, if and only if all $t_{|\beta}$ are right/left recursive. With other words, for the decision whether a schema is right/left recursive, its schema descriptor has to be investigated. To make things easier, this aspect is not mentioned in the following discussion, but should be kept in mind.

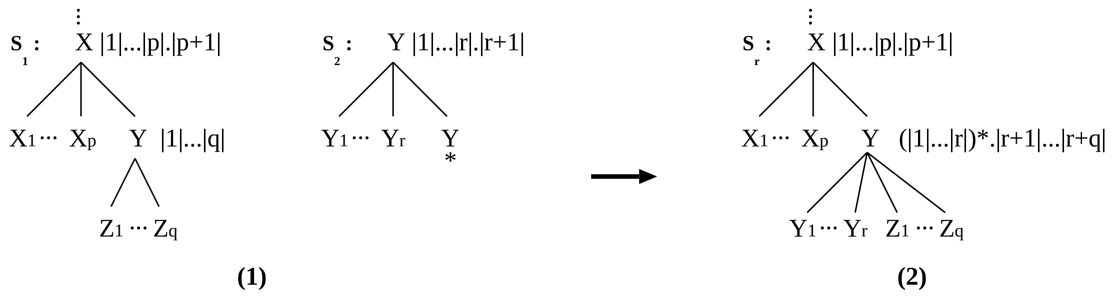


Figure 1: Avoiding (right-recursive) adjoining by iteration

L(S-TAG) $\subseteq$ L(S^2-TAG):

Case 1: Right recursion

To be applied to each $\beta \in ADJ(\eta), \eta \in N$, where β is right recursive. Fig. 1 shows the elimination of an auxiliary schema, which effectively performs right recursion at node (S_1,Y)[1]. Without loss of generality, the descriptors

1. It is not possible to give correct absolute Gorn addresses for nodes of S_1 in that example, because it is not known, whether X is the root node or not. Hence, for the time being, the Gorn number is replaced with the node label, e.g. (S_1,Y), which also should be non-ambiguous in Fig. 1.

shown in Fig. 1 are simple enumerations of the respective children nodes, in order to make the relocation arithmetics clear, but they could be arbitrary complex, as long as the position of the foot node does not change.

Given that S_2 is adjoinable at (S_1,Y), the substructures of $(S_2,0)$ (without foot node), namely Y_1 ... Y_r, are copied over as children of (S_1,Y) to the left of its own children Z_1 ... Z_q, giving the sibling's sequence $Y_1...Y_rZ_1...Z_q$. Then the descriptor $(|1|\dots|r|)^*$, which licenses the arbitrary repetition of S_2's children Y_1 ... Y_r, is added as prefix to the descriptor of (S_1,Y). After that, the Gorn numbers $|1|\dots|q|$, which referred to the now right-shifted nodes Z_1 ... Z_q, are updated to $|r|\dots|r+q|$. Apply this to any node Y of grammar G, except for foot nodes, because they would be destroyed by attaching children to it[2]. See Fig. 2 for an example how the right-recursive auxiliary tree A_1 is eliminated through the transformation process.

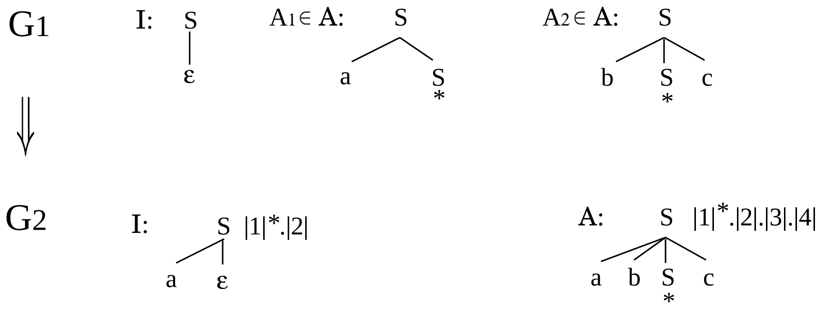


Figure 2: $L(G_1) = \{w|w \in \{a,b\}^*c^n, |b| = |c|\} = L(G_2) = \{w|w \in (a^*b)^n a^* c^n\}$

However, things are getting more complicated, if $|ADJ(\eta)| > 1$, i.e. there exist more than one auxiliary schema β with root node η: The prefixes, which iterate the substructures of the auxiliary schemata have to be combined as alternatives: Having the descriptors $\sigma_i, \dots, \sigma_j$ from the root node of auxiliary schemata $\beta_i, \dots, \beta_j$ would result in descriptor $\sigma = (\sigma_i' + \dots + \sigma_j')^*.\sigma_r'$. The respective Gorn numbers $\{|1|, \dots, |\beta_{im_i}|\} \in \sigma_i, \dots, \{|1|, \dots, |\beta_{jm_j}|\} \in \sigma_j$, would be relocated to

$$\begin{array}{rcl} \{|1|,\dots,|\beta_{im_i}|\} & \in & \sigma_i' \\ & \vdots & \\ \{|\beta_{im_i} + \dots + \beta_{j-1m_{j-1}} + 1|, \dots, |\beta_{im_i} + \dots + \beta_{jm_j}|\} & \in & \sigma_j' \\ \{|\beta_{im_i} + \dots + \beta_{jm_j} + 1|, \dots, |\beta_{im_i} + \dots + \beta_{jm_j} + q|\} & \in & \sigma_r' \end{array}$$

This relocation reveals an important characteristics of the transformation process: Either the transformation is done simultaneously, or later transformations lead to unhealthy grammars: If, e.g., after transforming $\beta_i, \dots, \beta_j$ the auxiliary tree β_{j+1} has to be transformed, then the resulting descriptor σ falsely would be $(\sigma_{j+1})^*.(\sigma_i + \dots + \sigma_j)^*.\sigma_r$ instead of $(\sigma_i + \dots + \sigma_j + \sigma_{j+1})^*.\sigma_r$.

Case 2: Left recursion
Analogue.

Case 3: Both recursions
To be applied to each $\beta \in ADJ(\eta), \eta \in N$ holds, that, where β is either right or left recursive. This case is performed like left and right recursion, but simultaneously: Let $\sigma_i, \dots, \sigma_j$ be schema descriptors from the root node of left recursive auxiliary schemata $\beta_i, \dots, \beta_j$, and $\sigma_k, \dots, \sigma_l$ schema descriptors from the root node of right recursive auxiliary schemata $\beta_k, \dots, \beta_l$, each without the Gorn number for their foot nodes. Let σ_r be the original descriptor of the node, in which $\beta_i, \dots, \beta_l$ may adjoin. Then the resulting schema descriptor is $(\sigma_i' + \dots + \sigma_j')^*.\sigma_r'.(\sigma_k' + \dots + \sigma_l')^*$ with the Gorn number relocations are performed as described above.

2. This suffices, because the children are attached as well to the node, in which adjoining occurs.

After applying case 1,2 or both for each node $\eta \in N, ADJ(\eta)$ is either empty, or $\exists\beta \in ADJ(\eta)$ with β is neither left nor right recursive. See Fig. 3 for an example how the right-recursive auxiliary tree A_1 and the left-recursive auxiliary tree A_2 or eliminated simultaneously through the transformation process.

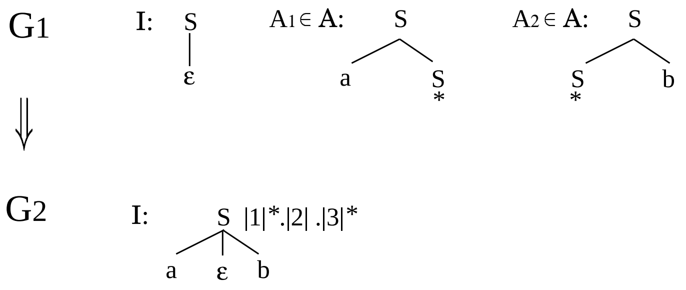


Figure 3: G_2 is the result of transforming G_1

L(S^2-TAG) $\subseteq$ L(S-TAG):
Trivial, since each S^2-TAG is an S-TAG per definition. ■

Gentle application of rules (1)–(4) of page 102 to σ might be advisable. As a side note, transforming the grammar $G = \{S \Rightarrow \gamma,\ S \Rightarrow S\beta,\ S \Rightarrow \alpha S\}$ simultaneously is not the same as transforming $G' = \{S \Rightarrow \gamma,\ S \Rightarrow \alpha S\beta\}$ (which is not allowed, anyway). The former grammar G expresses $\alpha^n\gamma\beta^m$, whereas G' expresses $\alpha^n\gamma\beta^n$, but fatally, after transformation the resulting descriptors would be identical for G and G': Something according to $\alpha^*\gamma\beta^*$.

Whilst the method of transforming whenever possible auxiliary schemata greatly reduces the costs as shown below, there are two aspects to consider:

- Rewriting of auxiliary tree schemata raises the redundancies of the grammar, such that maintainability suffers.
- Furthermore, as side effect of the last point: The principle of locality of TAGs may get lost, where (groups of) auxiliary schemata are completely replaced with that method. Thus, the intended linguistic concept of that (group of) auxiliary schemata is scattered throughout the grammar.

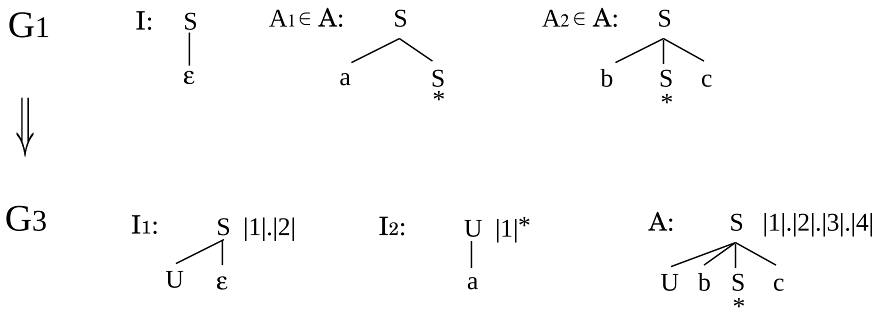


Figure 4: Transformation of G_1 of Fig. 2 with substitution

A solution to get rid of those drawbacks is to introduce unique substitution tree schemata for each transformed auxiliary schema as shown in Fig. 4. The modularisation effect of substitution remedies the redundancies similar to adjoining and preserves the locality of simple left or right auxiliary schemata.

Theorem 2 (Average Case of S^2-TAGs) *The average case for S^2-TAGs is $O(n^4)$.*

The assumption here is that in S^2-TAGs adjoining rarely happens, since the most common, i.e. pure left or right auxiliary schemata are not existent. Those non-transformable schemata, which are neither left nor right recursive, however, are very seldom even in large grammars. The XTAG grammar ((Doran *et al.*, 1994)), for example, consists of more than 1000 trees and only about three non-transformable ones. Therefore, the average case assumption is that there are barely any adjoining operations at all.

According to (Schabes, 1990), *Right Completion* and *Move Dot Up* are by far the most expensive operations: Their worst case complexity is $O(n^5)$ and this is the reason for the overall worst case complexity of $O(n^6)$. Assuming that there are no adjoinings, the *left/right prediction* and *completion* operations *(LP1, LC, RP1, RC)* as well as *move dot up (MU2.2)* (the dotted node subsumes a foot node) do not apply to any state. The latter ommission reduces the complexity of *MU* to the complexity of *MU2.1*(the dotted node does not subsume a foot node), which is $O(n^3)$. Following from the worst case behaviour of S-TAGs, *MU* is still the most expensive operation. Therefore, the algorithm takes at most $O(n^3)$ time to process a given state set S_i. Having at most n state sets, the overall complexity for S^2-TAGs under above assumption is $O(n^4)$. As a side note, Schabes' algorithm performs worse than Earley for complete context-free TALs.

Further investigations

Whilst $L(G_1) = L(G_2) = \{w|w = a^n b^m\}$ (cf. Fig. 3), attaching indices to the terminals in the order of their prediction makes clear that the number of hypotheses varies heavily between G_1 and G_2: *aabb* can be predicted by G_2 in the sequence $a_1a_2b_3b_4$ only. G_1, however, is able to predict it as any sequence of $\{a_ia_jb_kb_l|ijkl \in PERM(1234)\}$. The consequence is the prediction of all possible sequences, which leads to heavily interconnected derivation graphs. Applying null constraints at $(A_1, 2)$ and $(A_2, 1)$ would help to reduce the number of predictions by the half, because the sequence of a and b would be determined to $a_j \ldots a_i$, $b_k \ldots b_l$ with $i < j$ and $k < l$, but nevertheless the prediction order of a's and b's would be mixed up. The transformed grammar G_2, however, only permits one derivation, and that is linear from left to right. Thus, the total number of items produced by G_2 is a fraction of that of grammar G_1, as depicted in Fig. 5 and 6, in which the relative growth of time (TR), number of items (IR) and number of operations performed (OR) are shown. In G_1, e.g. for $n = 20$ approx. 5500 times more operations are performed than for $n = 2$. The time behaviour of G_2 is linear, and that is not surprising, since the only production rule left in G_2 is regular. In G_2, e.g. for $n = 20$ only about 7 times more operations are performed than for $n = 2$. The open question is whether this is an artifact of the example's simplicity, or whether this behaviour can be expected and to what degree in larger grammars after transforming them into S^2-TAGs.

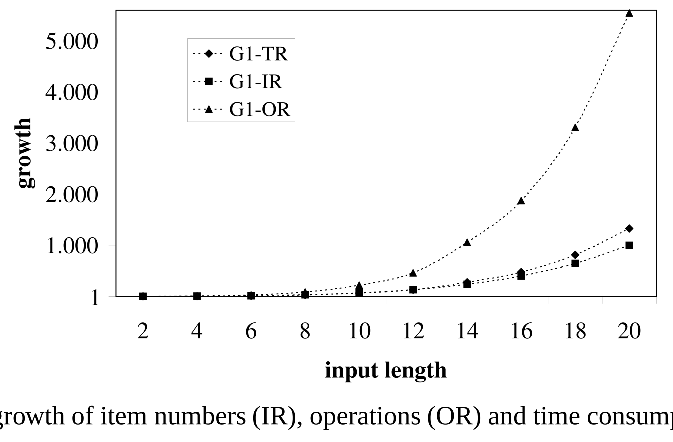


Figure 5: Relative growth of item numbers (IR), operations (OR) and time consumption (TR) for G_1

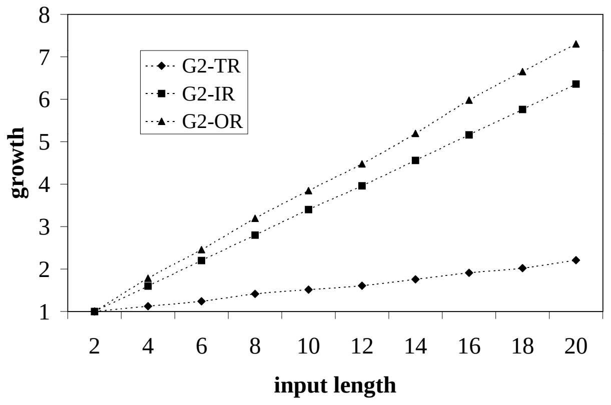


Figure 6: Relative growth of item numbers (IR), operations (OR) and time consumption (TR) for G_2

References

Doran, Christine, Dania Egedi, Beth Ann Hockey, Bangalore Srinivas and Martin Zaidel. 1994. XTAG System — A Wide Coverage Grammar for English. In Makoto Nagao, editor, *Procs. of the 15th International Conference on Computational Linguistics (COLING)*, volume 2, pages 922–928, Kyoto, Japan, August 23–28.

Earley, Jay. 1968. *An Efficient Context–Free Parsing Algorithm*. Ph.D. thesis, Carnegie-Mellon University, Pittburgh, PA, USA.

Gorn, Saul. 1967. Explicit definitions and linguistic dominoes. In John Hart and Satoru Takasu, editors, *Systems and Computer Science*. University of Toronto Press, Toronto, Canada, pages 77–115.

Harbusch, Karin, Friedbert Widmann and Jens Woch. 1998. Towards a Workbench for Schema-TAGs. In Anne Abeillé, Tilman Becker, Owen Rambow, Giorgio Satta and K. Vijay-Shanker, editors, *Procs. of the 4th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+4)*, pages 56–61, University of Pennsylvania, Philadelphia, PA, USA, August 1–2. Institute for Research in Cognitive Science. IRCS-Report 98–12.

Harbusch, Karin and Jens Woch. 2000. Direct Parsing of Schema–TAGs. In Harry C. Bunt, editor, *Procs. of the 6th International Workshop on Parsing Technologies (IWPT)*, pages 305–306, Trento, Italy, February 23–27. Institute for Scientific and Technological Research.

Kempen, Gerard and Karin Harbusch. 1998. Tree Adjoining Grammars without Adjoining? The Case of Scrambling in German. In Anne Abeillé, Tilman Becker, Owen Rambow, Giorgio Satta and K. Vijay-Shanker, editors, *Procs. of the 4th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+4)*, pages 80–83, University of Pennsylvania, Philadelphia, PA, USA, August 1–2. Institute for Research in Cognitive Science. IRCS-Report 98–12.

Schabes, Yves. 1990. *Mathematical and Computational Aspects of Lexicalized Grammars*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, USA.

Schabes, Yves and Richard C. Waters. 1995. Tree Insertion Grammar: Cubic-Time, Parsable Formalism that Lexicalizes Context-Free Grammar without Changing the Trees Produced. *Computational Linguistics*, 21(4):479–513.

Weir, David. 1987. Characterising Mildly Context–Sensitive Grammar Formalisms. PhD. proposal, University of Pennsylvania, Philadelphia, USA, 1987.

Quantification Over Situation Variables in LTAG: Some Constraints

Maribel Romero
University of Pennsylvania

1. Introduction

Some natural language expressions –namely, determiners like *every, some, most*, etc.— introduce quantification over individuals (or, in other words, they express relations between sets of individuals). For example, the truth conditions of a sentence like (1a) are represented in Predicate Logic (PrL) by binding the occurrences of the individual variable *x* with the quantifier ∀, as in (1b).[1]

(1) a. Every professor run the marathon.
b. $\forall x$ [professor(x) $\rightarrow$ run-the-marathon(x)]

In a similar way, it has been argued that certain expressions introduce quantification over possible worlds or possible situations (Lewis 1973, among many others). The set of possible worlds includes the actual world --where the individuals are the way they actually are-- and any other logically possible world --where individuals may have different properties from the ones they have in actuality. In this paper, I assume a Situation Semantic framework (Barwise-Perry 1983, Kratzer 1989, von Fintel 1994) and do not quantify over entire possible worlds, but over *parts* of possible worlds, i.e., over possible situations.[2] In (2a), e.g., the speaker predicates the property of being in the Bahamas by 5pm of Jorge in all the (actual or non-actual relevant) situations *s* where all my actual obligations are fulfilled. This is informally represented in (2b). The variable s_0 stands for the situation at which the sentence as a whole is evaluated (the actual situation), and the variable *s* ranges over the possible (actual or non-actual) situations considered.

(2) a. Jorge has to be in the Bahamas by 5pm.
b. [[(2a)]](s_0) = 1 iff $\forall$s [all my actual obligations in s_0 are fulfilled in s $\rightarrow$ in(jorge, bahamas, 5pm, s)]

The present paper has two goals.

The first one is to implement in LTAG the semantics of some natural language expressions that quantify over possible situations. In particular, I will propose lexical entries for the modal auxiliary *must*, for the intensional adverb *probably* and for the adverb of quantification *sometimes*.[3] The proposal will model insights from the philosophical and linguistic literature (Stalnaker 1968, Lewis 1973, Cresswell 1990, Kratzer 1979) into the LTAG quantificational schemata developed in Kallmeyer-Joshi (2001).

The second, more important aim is to account, within LTAG, for a certain constraint on binding of situation variables discussed in Percus (2000) (see also Musan 1995 for related observations on time variables). In an nutshell, this binding constraint requires the following. Whereas predicates within (simple) Noun Phrases (NPs) can be evaluated with respect to a non-local situation binder, the situation variable of the main predicate in the Verb Phrase (VP) has to be bound by the closest c-commanding situation operator. It will be argued that this constraint follows automatically if, using the LTAG denotations proposed in this paper, the compositional semantics is computed on the derivation tree rather than on the derived tree.

The paper is organized as follows. In section 2, I will briefly present Kallmeyer-Joshi's (2001) proposal for quantifiers over individuals and extend it to quantifiers over situations. Semantic denotations for *must*, *probably* and *sometimes* will be spelled out in LTAG. Section 3 introduces the core data on the behavior of NPs versus

1. (1b) is a simplified version. Once we add situation variables, the denotation of (1a) for a given evaluation situation s_0 is represented as in (i). [[.]] is the interpretation function from linguistic expressions to their intensions.
(i) [[(1a)]](s_0) = 1 iff $\forall$x [professor(x,s_0) $\rightarrow$ run-the-marathon(x,s_0)]
2. The choice of situations instead of worlds is orthogonal to the arguments in this paper. Situations are best fitted to account for adverbs of quantification (*always, sometimes, often*, etc.) involving indefinites and donkey-anaphora (see von Fintel 1994).
3. The denotation of other members of each category can, of course, be easily modeled after the proposed entries (e.g., for modals like *can*, *might*, *would, should*; intensional adverbs like *necessarily*, *possibly*, *perhaps*, *unlikely*; and adverbs of quantification like *always, usually*, *often, rarely*.)

VPs with respect to situation binding. Section 4 derives the asymmetric behavior of NPs and VPs with respect to situation binding. We will consider a simple case with one situation adverb. It will be shown that the freedom of the NP situation variable and the locality of the VP situation variable follow in a straightforward way if we apply the compositional semantics to the derivation tree instead of the derived tree. A more complex case will be briefly considered, where two situation adverbs are at issue. Section 5 concludes.

2. Quantification over possible worlds in LTAG

A sentence α denotes a proposition, $[[\alpha]]$, i.e., a function from possible worlds to the truth values in $\{0,1\}$. A way to encode the denoted proposition $[[\alpha]]$ is to equate with a Ty2 formula the conditions under which $[[\alpha]]$ applied to the actual situation s_0 will yield 1. This is done for a simple example in (3), and its LTAG correspondent is given next to it. Note that, in a Ty2 formula (Gallin 1975), all logical predicates (formal translations of nouns, adjectives and verbs) include a situation argument, represented using the variables s_0 (by convention, the actual situation, the evaluation situation of the sentence as a whole), and s, s', s'', etc.

(3) $[[$*Pat visits Kate*$]](s_0) = 1$ iff visit(p,k,s_0)

l_0: visit(x,y,s_0) pat(x) kate(y)
arg: -

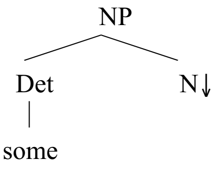
Let us introduce quantificational determiners into this Ty2 intensional framework. Kallmeyer-Joshi propose that the contribution of a quantificational determiner consists of two parts: on the one hand, the quantificational NP adds an argument to the predicate-argument structure of the sentence; on the other, it introduces a scopal element, a logical operator which takes scope over (at least) that predicate-argument structure. For each quantificational determiner, these two components are separated into two trees, as exemplified in (4)-(5): the predicate-argument component substitutes into the appropriate argument slot in the verb's tree, and the scopal component adjoins to the root node of the verb tree.

(4) Some / A:

β_1 S*

l_1: some(x,h_1,h_2) $r_1 \leq h_2$
arg: r_1

α_1



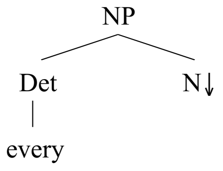
l_2: $p_2(x,s')$ $l_2 \leq h_1$
arg: $<p_2, 01>$

(5) Every:

β_2 S*

l_3: every(y,h_3,h_4) $r_2 \leq h_4$
arg: r_2

α_2



l_4: $p_4(y,s'')$ $l_4 \leq h_3$
arg: $<p_4, 01>$

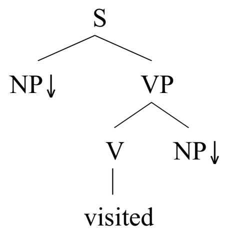
An example sentence is given in (6). The remaining basic trees and their LTAG denotations are spelled out in (7)-(9). Following the derivation tree in (10), these denotations compose to yield the final denotation in (11). Note that the relative scope of two quantifiers *every* and *some* remains underspecified in this output denotation.

The two possible scopal readings are determined by the disambiguation functions δ_1 and δ_2 in (12).[4]

(6) A student visited every club.

(7) Visited:

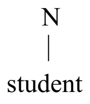
α_0



l_o: visit(x,y,s''')
arg: <x, 00>, <y,011>

(8) Student:

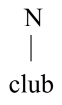
α_3



q_1: student
arg: -

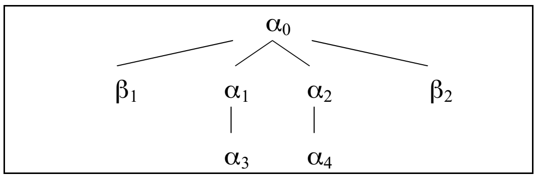
(9) Club:

α_4



q_2: club
arg: -

(10) Derivation tree:



(11) Output denotation:

l_1: some(x,h_1,h_2), l_3: every(y,h_3,h_4), l_o: visit(x,y,s'''), l_2: (q_1: student)(x,s'), l_4: (q_2: club)(y,s'') $l_o \leq h_2$, $l_2 \leq h_1$, $l_0 \leq h_4$, $l_4 \leq h_3$
arg: -

(12) Scope disambiguation functions:

a. δ_1 for the scope *some >> every* = { $\langle h_1,l_2\rangle$, $\langle h_2,l_3\rangle$,$\langle h_3,l_4\rangle$,$\langle h_4,l_0\rangle$}

b. δ_2 for the scope *every >> some* = { $\langle h_1,l_2\rangle$, $\langle h_2,l_0\rangle$,$\langle h_3,l_4\rangle$,$\langle h_4,l_1\rangle$}

The semantic representations above are faithful to Kallmeyer-Joshi's proposal expect for the fact that we have introduced a situation variable for each predicate, namely *s'*, *s''*, *s'''*. In this case, since there is no situation binder in the sentence, all three situations variables are identified with the evaluation situation s_0. The result of this variable identification is spelled out in (13).

(13) [[*A student visited every club*]](s_0) = 1 iff $\exists$x [student(x,s_0) $\wedge$ $\forall$y [club(y,s_0) $\rightarrow$ visit(x,y,s_0)]] or
$\forall$y [club(y,s_0) $\rightarrow$ $\exists$x [student(x,s_0) $\wedge$ visit(x,y,s_0)]]

l_1: some(x,h_1,h_2), l_3: every(y,h_3,h_4), l_o: visit(x,y,s_0), l_2: (q_1: student)(x,s_0), l_4: (q_2: club)(y,s_0) $l_o \leq h_2$, $l_2 \leq h_1$, $l_0 \leq h_4$, $l_4 \leq h_3$
arg: -

4. I will ignore the semantic contribution of tense throughout the paper.

Let us now extend Kallmeyer-Joshi's quantification procedure, already intensionalized, to quantification over situations. In the same way that quantificational NPs introduce quantification over individuals, it has been argued that modal auxiliaries, intensional adverbs and adverbs of quantification quantify over possible situations (Kratzer 1979, 1989, von Fintel 1994). A small difference between the two quantification procedures concerns the so called "restrictor". The restrictor subformula for NPs –e.g., club(y,s_0) in (13)-- originates from the noun inside it, whereas the restrictor for modals is (mostly) contextually given. For example, (14) can be understood as quantifying over deontic situations (roughly, situations where all our actual obligations are fulfilled) or over epistemic situations (roughly, situations such that, as far as the speaker knows, could be the actual situation s_0). These two readings are encoded in the Ty2 formulae (14a)-(14b) by placing the 2-place predicates *Deo* and *Epi* --defined in (15)-- in the restrictor of the situation quantifier.

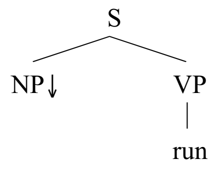
(14) John must run.
a. $\forall$s' [Deo(s',s$_0$) $\rightarrow$ run(j, s')]
b. $\forall$s' [Epi(s',s$_0$) $\rightarrow$ run(j, s')]

(15) a. Deo(s",s') =1 iff s" is a situation accessible from s' such that all the obligations in s' are fulfilled in s".
b. Epi(s",s') =1 iff s" is a situation accessible from s' such that, for all the speaker knows, s' could be s".

Following Kallmeyer-Joshi's quantification schemata, I propose to implement quantification over situations in LTAG as follows. The semantic contribution of the tree for *run* and the tree for *John* is given in (16)-(17). The double semantic contribution of *must* under its deontic reading is provided in (18). Note that *must* carries universal quantificational force, that is, the scopal part of *must* includes the quantifier every, this time applied to a situation variable *s'*. Other items like *may* or *perhaps*, expressing existential force, would yield a comparable double semantic value. Furthermore, note that, whereas *every* in (5) needs to look for its restrictor in its 01 address, I have spelled out the restrictor of *must* –contextually provided—directly in the predicate-argument part of *must* itself, for simplicity reasons.

(16) Run:

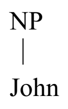
α_1



l_0: run(x,s')
arg: <x, 00>

(17) John:

α_2



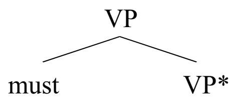
john(x)
arg: -

(18) Must:

β_1 S*

l_1: every(s''',h_1,h_2) $r_1 \leq h_2$
arg: r$_1$

β_2

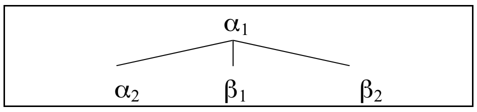


l_2: Deo(s''',s'') $l_2 \leq h_1$
arg: <s"', 01>

The reader may have noticed a further difference between the predicate-argument value of *must* in (18) and the predicate-argument value of *every* in (5): *must*'s β_2 has an address for one of its internal variables (namely *s"'*, marked in boldface), but *every*'s α_2 does not. This difference stems from the fact that the predicate-argument part of *every* will be substituted into the verb's tree, whereas the predicate-argument part of *must* will be adjoined to the verb's tree. Now, recall that the difference between a substituted element and an adjoined element yields the following effect in (L)TAG. The to-be-adjoined element can identify one or more of the variables within its semantic value with the variables (of the appropriate type) provided at a given address in its tree. But the to-be-substituted element does not have any address in its tree for the tree it is substituting into. Hence, there is no way it can force any such variable identification. For the case at issue, this means that the predicate-argument part of *every* in (5) cannot force identification of *y* or *s"* with any variable in the tree for *visit*. The predicate-argument part of *must* in (18), instead, is allowed to identify *s'''* with a variable provided at the 01 address, that is, with the situation variable provided by the tree of *run*. This difference, as we will see, will play a crucial role to capture the data forthcoming in section 3.

The denotations in (16)-(18) are combined following the derivation tree in (19). The result is (20).

(19) Derivation tree for (14):



(20) [[*John must run*]](s_0) = 1 iff
$\forall$s' [Deo(s',s_0) $\rightarrow$ run(j, s')]

l_1: every(s''',h_1,h_2) l_2: Deo(s''',s_0), l_0: run(x,s'''), john(x) $l_2 \leq h_1$, $l_0 \leq h_2$
arg: -

Finally, before turning to the data in section 3, let me introduce the double semantic value of two situation adverbs that will be needed later in the paper: *probably* and *sometimes*. The restrictor predicate for *sometimes* is described in (21).

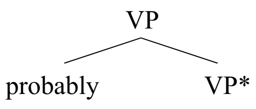
(21) Part(s'',s') =1 iff the situation s'' is part of the situation s'.

(22) Probably:

β_1 S*

l_1: most(s''',h_1,h_2) $r_1 \leq h_2$
arg: r_1

β_2



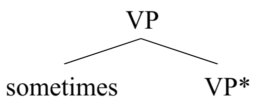
l_2: Epi(s''',s'') $l_2 \leq h_1$
arg: <s“‘, 01>

(23) Sometimes:

β_3 S*

l_1: every(s''',h_1,h_2) $r_1 \leq h_2$
arg: r_1

β_4



l_2: Part(s''',s'') $l_2 \leq h_1$
arg: <s“‘, 01>

3. An asymmetry with situation variables: NPs can be transparent or opaque, but VPs must be opaque

Take the sentence in (24) under the reading where *every* has scope inside the *if*-clause. Farkas (1997) –among others-- notes that there is still an ambiguity, rooted on the situation variable that we assign to the complex predicate *poor child*: we may be talking about the set of poor children in the hypothetical situations *s'* introduced by the conditional (opaque reading), or, interestingly, we may interpret it as the set of poor children in the actual situation s_0 (transparent reading). The transparent reading is particularly salient in (25), since (25)'s opaque reading yields a contradiction in the hypothetical situations and, hence, is pragmatically odd.

(24) If you fed every poor child, I would be happy.
a. Opaque NP: In every situation s' accessible to s_0: if you fed in s' all the **poor children in s'**, I am happy in s'.

b. Transparent NP: In every situation s' accessible to s_0: if you fed in s' all the people who are **poor children in the actual situation s_0**, I am happy in s'.

(25) If every poor child was very rich instead, I would be happy

a. # Opaque NP: In every situation s' accessible to s_0: if all the **poor children in s'** are very rich in s', I am happy in s'.

b. Transparent NP: In every situation s' accessible to s_0: if all the people who are **poor children in the actual situation s_0** are very rich in s', I am happy in s'.

The same ambiguity obtains in simpler sentences with modals and intensional adverbs. Take (26)-(27) under the reading where the indefinite determiner scopes under the (deontic) modal *must* (or *should*). Still, the complex predicate *poor child* can be interpreted with respect to the deontic situations *s'* (opaque reading) or with respect to the actual situation s_0 (transparent reading). Again, the transparent reading is particularly clear in (27), since the opaque reading is pragmatically out.

(26) A poor person must / should be fed.

a. Opaque NP: In every situation s' accessible to s_0 where all our obligations in s_0 are fulfilled: a **poor person in s'** in fed in s'.

b. Transparent NP: In every situation s' accessible to s_0 where all our obligations in s_0 are fulfilled: a person who is a **poor person in the actual situation s_0** is fed in s'.

(27) A poor person must / should be rich.

a. # Opaque NP: In every situation s' accessible to s_0 where all our obligations in s_0 are fulfilled: a **poor person in s'** is rich in s'.

b. Transparent NP: In every situation s' accessible to s_0 where all our obligations in s_0 are fulfilled: a person who is a **poor person in the actual situation s_0** is rich in s'.

Percus (2000) adds the interesting observation that a transparent reading of the main predicate in the VP is impossible. That is, even if we give the subject NP scope under the relevant intensional operator and we interpret it opaquely (so that the intensional operator binds at least one situation variable and it does not yield vacuous quantification), the VP predicate cannot be interpreted as transparent:

(24) If every poor child was fed, I would be happy.

c. * Transparent VP (and opaque NP): In every situation s' accessible to s_0: if all the poor children in s' **are fed in s_0**, I am happy in s'.

(25) If every poor child was very rich instead, I would be happy

c. * Transparent VP (and opaque NP): In every situation s' accessible to s_0: if all the poor children in s' **are rich in s_0**, I am happy in s'.

(26) Some poor person must / should be fed.

c. * Transparent VP (and opaque NP): In every situation s' accessible to s_0 where all our obligations in s_0 are fulfilled: a poor person in s' in **fed in s_0**.

(27) Some poor person must / should be rich.

c. * Transparent VP (and opaque NP): In every situation s' accessible to s_0 where all our obligations in s_0 are fulfilled: a poor person in s' in **rich in s_0**.

The readings (24c-27c) are simply impossible. Let us take sentence (26) and judge it in the scenario Σ_{26}. The sentence is judged false. But the reading (26c) is true in this scenario. Since a sentence with reading (26c) should be judged true in the all scenarios that make the reading (26c) is true, and since (26) is not judged true in one such scenario, (26) lacks the reading (26c).

(28) Scenario Σ_{26} for (26) :

In the actual situation s_0, Pat, Lucy, Miguel and nobody else are fed. Our obligation (as evil witches and wizards) is to make at least one of them (any of them) poor. There are no further obligations in s_0. In particular, there is no obligation to feed anybody.

The same reasoning applies to (27) and the scenario Σ_{27}. Sentence (27) is false in Σ_{27}, whereas the reading (27c) is true in Σ_{27}. Hence, sentence (27) lacks reading (27c).

(29) Scenario Σ_{27} for (27) :
In the actual situation s_0, Pat, Lucy, Miguel and nobody else are rich. Our obligation (as evil witches and wizards) is to make at least one of them (any of them) poor. There are no further obligations in s_0. In particular, there is no obligation to make anybody rich.

I leave examples (24)-(25) and the construction of the relevant scenarios as an exercise for the reader. However, before concluding this section, let me illustrate the asymmetry between the situation variables in NPs and VPs with adverbs of quantification as well. The following example, from Percus (2000), has a transparent NP reading ((30a)), but it lacks a transparent VP reading ((30b)). The sentence is judged true in scenario Σ_{30a} –a scenario that makes the transparent NP, opaque VP reading true-- but false in scenario Σ_{30b} –a scenario where the opaque NP, transparent VP reading is true.

(30) The winner sometimes lost.
a. Transparent NP, opaque VP reading: In some (relevant) situations s' that are part of s_0: the **winner in s_0** lost in s'.
b. * Opaque NP, transparent VP reading: In some (relevant) situations s' that are part of s_0: the winner in s' **lost in s_0**.

(31) Scenario Σ_{30} for (30):
We are in a situation s_0 that contains a game among five participants. The game is such that there is exactly one winner of the game and exactly one loser of the game. The other three participants neither win nor lose (e.g., if the winner receives money and the loser pays, the other three participants neither receive nor pay money). The game consists of fifteen rounds (each of which can be considered a natural sub-situation s' of s_0). Each round has exactly one winner and exactly one looser, and, as before, the other three participants of each round neither win nor lose. The winner of the game is the person that wins more rounds, and the loser of the game is the person who loses more rounds. (In case of tie, the relevant participants play until there is no tie.).
a. Σ_{30a}: This time, in situation s_0, Sue, the winner of the game, lost rounds 2 and 3, whereas Mario, the loser of the game, won no round at all.
b. Σ_{30b}: This time, in situation s_0, Sue, the winner of the game, lost no round at all, whereas Mario, the loser of the game, won rounds 6 and 9.

In sum, the question we need to answer is the following: Why is the main predicate in a VP necessarily opaque with respect to the immediate situation operator, whereas predicates embedded in an NP can be interpreted as opaque or transparent?

4. Capturing the asymmetry in LTAG semantics

To rephrase the question in LTAG terms, take the semantic representation in (32). Why is there a choice between l_4: poor-person(x, $\boldsymbol{s_0}$) and l_4: poor-person(x, $\boldsymbol{s'}$), whereas only the opaque situation option l_2: rich(x, $\boldsymbol{s'}$) is available?.

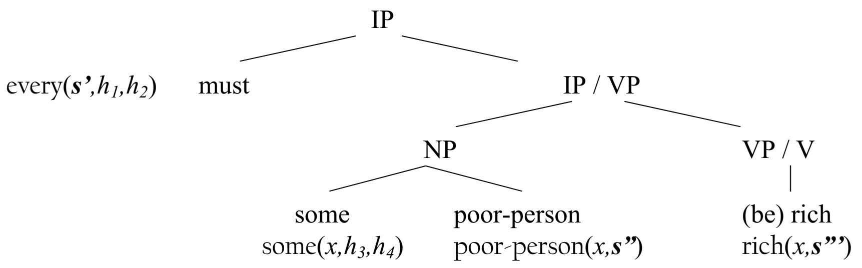
(32) [[*Some poor person must be rich*]](s_0) = 1
iff ∀s' [Deo(s',s_0) → ∃x [poor-person(x, $\mathbf{s_0}$/**s'**) ∧ rich(x,**s'**)]]

δ (for *must*>>*every)* =
{<h_4,h_2>, <h_2,l_3>, <h_1,l_1>, <h_3,l_4>}

l_0: every(s',h_1,h_2) l_3: some(x,h_3,h_4) l_1: Deo(s',s_0), l_2: rich(x, $\boldsymbol{s'}$), l_4: poor-child(x, $\boldsymbol{s_0}$ / $\boldsymbol{s'}$) $l_1 \leq h_1$, $l_2 \leq h_2$, $l_4 \leq h_3$, $l_2 \leq h_4$
arg: -

The question is particularly puzzling for grammars where the compositional semantics is performed on the derived tree. Take a GB Logical Form tree or an LTAG derived tree where the modal *must* takes scope over the determiner *some*. We have assumed, as proven in Gallin (1975), that the expressive power needed to generate intensional readings in natural language amounts to a Ty2 formal language where we have direct quantification over situation (or world) variables. Furthermore, following Percus (2000), every predicate in a sentence is in principle allotted its own situation variable. That yields, roughly, the syntactico-semantic representation in (33). Note that, in the derived tree, the situation operator [[*must*]] combines with the denotation of its sister as a whole. Why should [[*must*]], then, make a distinction between NP situation variables and verbal situation variables if they are all equally available within its sister's denotation? How could we possibly account for the fact that [[*must*]] will necessarily bind *s'''* in rich(*x,s'''*) and will only optionally bind *s''* in poor-person(*x,s''*)?[5]

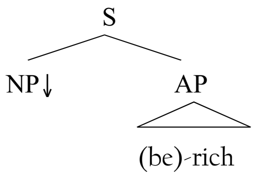
(33) Some poor person must be rich.



If, instead, we perform the semantic computation on the LTAG derivation tree using the proposed denotations, the asymmetry between NPs and main predicates follows straightforwardly from the way the derivation proceeds. Take the denotations below and the derivation tree in (38):

(34) (Be) rich:

α_1



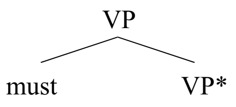
l_0: rich(*x,s''*)
arg: <x, 00>

(35) Must

β_1 S*

l_1: every(s',h_1,h_2) $r_1 \leq h_2$
arg: r_1

β_2



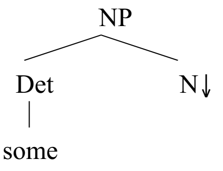
l_2: Deo(*s',s)* $l_2 \leq h_1$
arg: <s', 01>

(36) Some / A:

β_3 S*

l_3: some($x,h_3,,h_4$) $r_2 \leq h_4$
arg: r_2

α_2

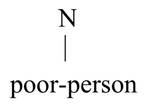


l_4: $p_2(x,s'')$ $l_4 \leq h_3$
arg: <p_2, 01>

5. A reviewer suggested the possibility that situation variables are not indices generated as sister of predicates, but indices introduced by the determiner in the NP. This way, NPs would have a free situation variable that may be optionally bound higher up, whereas the main predicate in a VP would not have a free situation variable at any point (e.g. [[*(be) rich*]] would be λxλs.rich(x,s)). However, being introduced by a determiner is neither a necessary nor a sufficient condition for an NP to be optionally transparent. First, bare plurals can have a transparent reading, as Kratzer's (i) illustratres. Second, NPs with a determiner acting as main predicates in copular sentences cannot be transparent: (ii) lacks the VP transparent reading as much as (25) and (27).
(i) Sue wanted to put belladonna berries in the salad because she mistook them for raspberries.
(ii) If some poor child was the richest child instead, I would be happy.
In fact, Percus (2000), who assumes a GB Logical Form derived tree as the input to the semantics, does not capture the binding asymmetry in the semantics. He proposes, instead, a syntactic constraint on LF, basically a Binding Theory for situation variables.

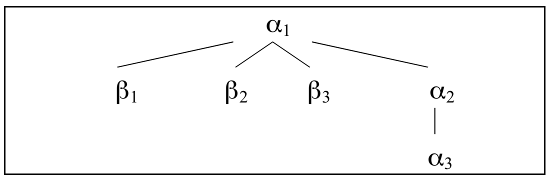
(37) Poor-person:

α_3



q_1: poor-person
arg: -

(38) Derivation tree for *Some poor people must be rich*:



In this derivation tree, β_2 (the semantic part of *must* in charge of identifying the variable *s'* with another situation variable at the address 01) applies to α_1, the main predicate's denotation, and it ensures that the two situation variables are identified. β_2 never applies to the denotation α_2 of the NP *some poor people*, thus it cannot enforce variable identification with it. In fact, given that we follow the derivation tree and not the derived tree, β_2 does not even apply to a semantic object that includes the contribution of α_2. That is, β_2 only finds α_1 at the 01 address, and not α_1 composed with α_2. Hence, the obligatory situation variable identification encoded in the denotation β_2 cannot choose a variable from α_2, but only from the main predicate's denotation α_1 found at the 01 address.

This way, the main predicate in *Some poor people must be rich*, namely *rich*, is necessarily opaque with respect to *must*, whereas no such constraint can be imposed for the predicate *poor-people* buried in the NP. The NP is, hence, free to be interpreted as transparent or as opaque with respect to the modal. If it is transparent, we take the option of identifying its situation variable with the actual situation s_0; if it is opaque, we identify it with the situation variable *s'* introduced by the modal. This derives the existing readings of the sentence represented in (32), and it successfully excludes the non-existing ones.

To conclude this section, I will briefly consider a more complicated case involving two situation quantifiers, about which I will tentatively make some speculations. Recall example (30), repeated below with the added intensional adverb *probably*. Besides the readings discussed in section 3, Percus (2000:214ff) notes that, in examples with two situation operators in a c-command relation, the c-commanding one necessarily binds the second situation variable of the immediately c-commanded one. In our example (39), this means that the second situation variable introduced by *sometimes* cannot be identified with the actual situation s_0, but it has to be identified with the situation variable that *probably* quantifies over, namely *s'*. This is shown in (40): the second situation variable in l_3:Part(*s''*,***s'***) (in boldface) has to be locally bound by the quantifier most introduced by *sometimes*.[6]

(39) The winner probably sometimes lost.

(40) [[*The winner probably sometimes lost*]](s_0) = 1 iff
MOST s' [Epi(s',s_0)]
[∃s'' [Part(s'',**s'**) ∧ ιy.winner(y, s_0/s'/#s'')=x ∧ lost(x,s'')]]

l_0: most(s',h_1,h_2) l_2: some(s'',h_3,h_4) l_1: Epi(s',s_0), l_3: Part(s'',$\boldsymbol{s'}$), l_4: lost(x, s'') l_5: ιy.winner($y, s_0/s'/\#s''$)=x $l_1 \leq h_1$, $l_4 \leq h_2$, $l_3 \leq h_3$, $l_4 \leq h_4$
arg: -

Using the denotations of *probably* and *sometimes* provided in (22) and (23), this result can be easily achieved if the β_2 denotation of *probably* in (22) adjoins to the β_4 denotation of *sometimes* in (23), and β_4 adjoins to the tree for *lost*. This type of dependent adjunction is defended in Vijay-Shanker 1987. The

6. The NP *the winner* can now be evaluated with respect to any of the tree situations s_0, *s'* and *s''*, though the last one yields a pragmatically odd reading.

mandatory identification obtains straightforwardly. I leave to the reader the compositional semantic computation of this example.

A second possibility, presented in Schabes-Shieber (1991), consists of performing multi-adjunction of both adverbs at the same node of the main predicate's tree. If this syntactic approach is pursued, both the β_2 denotation of *probably* and the β_4 denotation of *sometimes* apply to the meaning of *lost*. The mandatory identification of variables encoded in β_2 and β_4 would then yield the wrong result in (41):

(41) MOST s" [Epi(s",s_0)]
[∃s" [Part(s",s_0/s") ∧ ιy.winner(y, s_0/#s")=x ∧ lost(x,s")]]

l_0: most(s'',h_1,h_2) l_2: some(s'',h_3,h_4) l_1: Epi(s'',s_0), l_3: Part(s'', s_0/s'), l_4: lost(x, s'') l_5: ιy.winner(y, s_0/#s'')=x $l_1 \leq h_1$, $l_4 \leq h_2$, $l_3 \leq h_3$, $l_4 \leq h_4$
arg: -

This is what the wrong result consists of: the variable identification instructions in β_2 and β_4 force both most and some to try to bind the same variable occurrence *s"*. This is not just an empirical wrong result, but an impossible task in Predicate Logic (PrL): one variable occurrence can only be bound by one quantifier. Hence, if this type of multi-adjunction is pursued, perhaps it is possible to ban this result on principled logical grounds, by appealing to a secondary variable identification procedure when the default one cannot be successfully implemented in PrL. I leave the issue open at this point.

5. Conclusions

We have seen that situation variables in NPs and in main predicates behave asymmetrically: NPs can be transparent and opaque with respect to the immediately c-commanding situation operator, whereas main predicates can only be opaque. Following Kallmeyer-Joshi's (2001) quantification procedure, I have proposed a double semantic value for the modal *must*, for the intensional adverb *probably* and for the adverb of quantification *sometimes*. The asymmetry between NPs and main predicates has been shown to follow if we apply the proposed denotations to the derivation tree.

References

Barwise, Jon and John Perry. 1983. *Situations and Attitudes*, Cambridge, Mass.: MIT Press.
Cresswell, M. 1990. *Entities and indices*, Dordrect: Kluwer.
von Fintel, Kai. 1994. *Restrictions on Quantifier Domains*, Amherst, Mass: GLSA.
Farkas, Donca. 1997. Evaluation indices and scope. In: A. Szabolcsi (ed.), *Ways of scope taking*, Drodrecht: Kluwer.
Gallin, Daniel. 1975. *Intensional and Higher-Order Modal Logic: with Application to Montague Semantics*, Oxford: North-Holland.
Kallmeyer, Laura and Aravind Joshi. 2001. Factoring Predicate Argument and Scope Semantics: Underspecified Semantics with LTAG. Penn- Univ. Paris 7 manuscript.
Joshi, Aravind. and K. Vijay-Shanker. 1999. Compositional semantics with LTAG: How much underspecification is necessary? In: Blunt H. C. and E.G.C. Thijsse, eds., *Proceedings of the third international workshop on computational semantics*, Tilburg.
Kratzer, Angelika. 1979. Conditional necessity and possibility. In: Baeuerle, Egli and von Stechow, eds., *Semantics from different points of view*, Berlin: Springer.
Kratzer, Angelika. 1989. An Investigation of the Lumps of Thought. Linguistics and Philosophy 12: 607-653.
Lewis, D. 1973. *Counterfactuals*, Oxford: Blackwell.
Musan, R. 1995. *On the temporal interpretation on noun phrases*, MIT Ph. D. diss.
Percus, O. 2000. Constraints on some other variables in syntax. Natural Language Semantics 8:173-229.
Schabes, Yves and Stuart M. Shieber. 1991. An Alternative Conception of Tree-Adjoining Derivation. Computational Linguistics 20.1: 91-124.
Stalnaker, R. 1968. A theory of conditionals. In: Rescher, ed., *Studies in Logical Theory*, Oxford: Blackwell.
Vijay-Shanker, K. 1987. *A Study of Tree Adjoining Grammars*, University of Pennsylvania Ph. D. dissertation.

One More Perspective on Semantic Relations in TAG

James Rogers
Earlham College, USA

1. Introduction

It has often been noted that the derivation trees of "standard" TAG grammars for natural languages (Group, 1998) resemble semantic dependency trees (Mel'ĉuk, 1988). More interesting, from a formal perspective, are the ways in which the derivation trees and the dependency trees diverge for certain problematic constructions. The desire to fix these cases has led to a variety of proposals for modifying or extending the TAG formalism ranging from Tree-Local Multi-component TAGs (TL-MCTAG) and Set-Local Multi-component TAGs (SL-MCTAG), through reconceptualization of the adjoining operation (Schabes and Shieber, 1994), through D-Tree Grammars (DTG) (Rambow, Vijay-Shanker and Weir, 1995) and Graph-driven Adjunction Grammar (GAG) (Candito and Kahane, 1998b), through reformalization of elementary trees in terms of C-Command rather than domination (Frank, Kulick and Vijay-Shanker, 2000), through the use of Meta-Grammars (Dras, 1999; Dras, Chiang and Schuler, To Appear), and through interpreting the semantic relations in a predicate-argument structure derived, but distinct, from the derivation trees (Joshi and Vijay-Shanker, 1999; Schuler, 1999). In this note, we look at some of these problematic constructions from yet another formal perspective—a mild extension of TAGs with well-constrained generative capacity which allows semantic dependency to be expressed as a relation orthogonal (in a quite literal sense) to constituency and linear precedence.

We should be clear at the outset, our focus is nearly exclusively formal—we explore ways of expressing these relationships in the extended formalism without pursuing their potential linguistic repercussions. Neither is our account exhaustive. There are problematic constructions that have not yet yielded to this approach. Our goal is to introduce the techniques involved, to explore their limits and, possibly, open up discussion of their linguistic consequences.

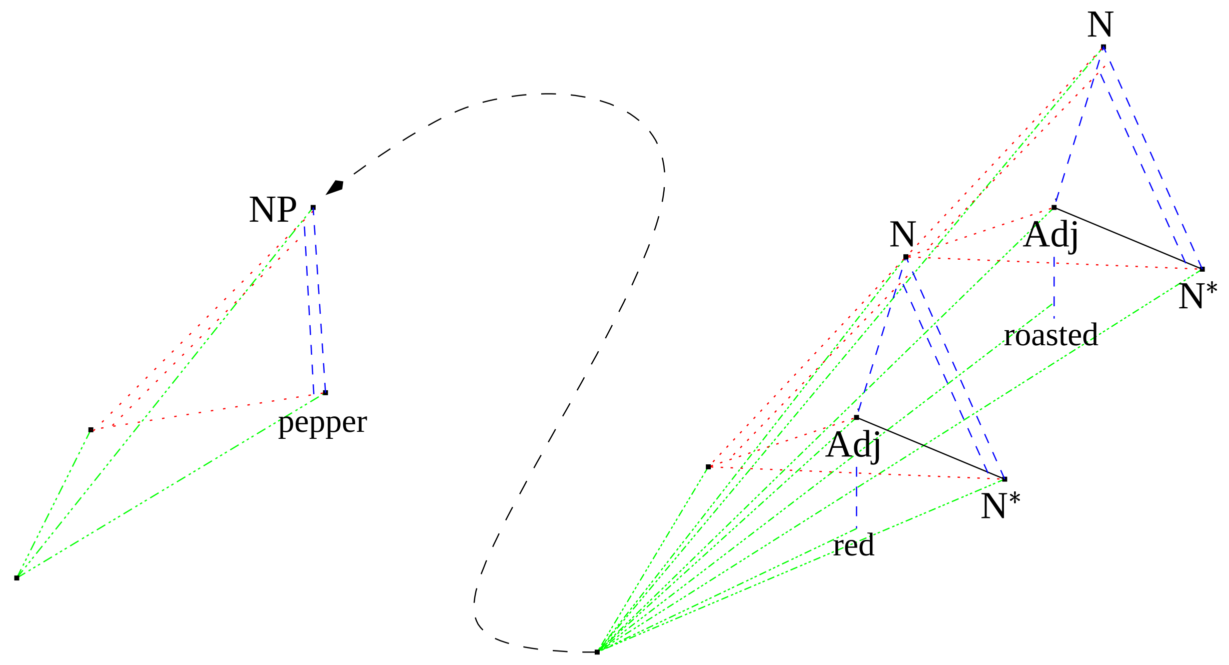
We will look five at examples from the literature:

1. roasted red peppers.
2. Does Gabriel seem to eat gnocchi?
3. Does Gabriel seem to be likely to eat gnocchi?
4. What does Mary think that John seems to eat?
5. That Paul has to stay surprised Mary.

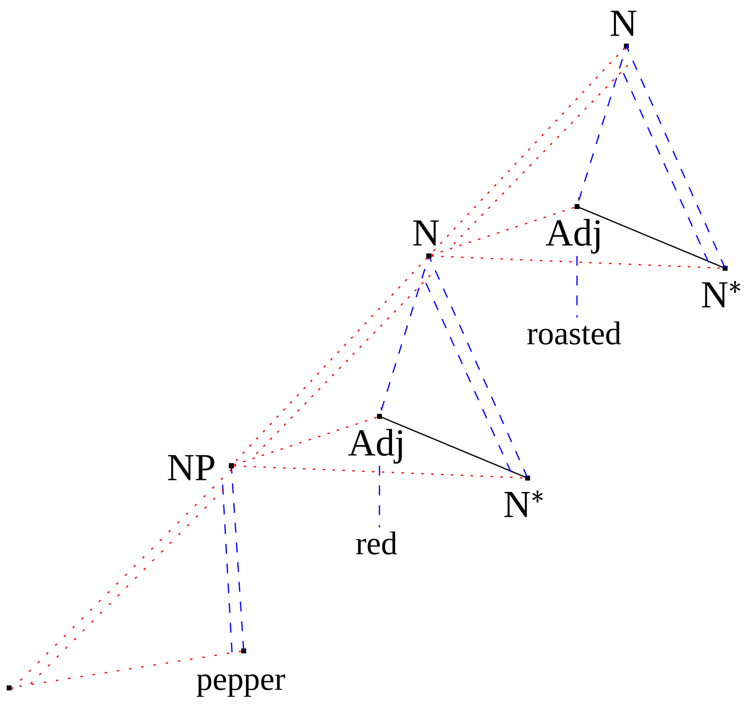
Again, our intent is not to offer novel linguistic analyses of these constructions—the analyses we five are taken from that literature. Rather, we offer slightly different formal interpretations of those analyses, ones that seem to work out without seriously distorting the original spirit of TAG.

2. Generalized Tree-Adjoining Grammar

The mechanism we employ is a form of higher-order adjunction obtained as a natural extension of the Generalized Tree-Adjoining Grammar of Rogers (Rogers, 1999). GTAG starts with an interpretation of adjunction as concatenation of local three dimensional tree-like structures in direct analogy with the concatenation of local (two dimensional) trees characteristic of CFGs. One consequence of this perspective is that, as with CFGs, the derivation structures and the derived structures are the same: they are, in essence, the structures one obtains by expanding the nodes in a standard TAG derivation tree to instances of the elementary trees they represent. A second consequence is that it is an easy step to extend these three dimensional grammars in a way that is analogous to the way that CFGs are extended to admit regular expressions on the rhs of productions in GPSG. Here we allow the yields of the local three dimensional trees to form any local set. Finally, the generalization from two dimensional grammars to three dimensional grammars suggests a natural generalization to higher dimensions. This leads to an infinite hierarchy of grammar types (Rogers, To Appear) which are equivalent, in weak generative power, to the grammars in Weir's Control Language Hierarchy (Weir, 1992).



Concatenation of four-dimensional elementary structures



Three-dimensional yield

Figure 1: roasted red pepper

We will use the four dimensional grammars, not because of their additional weak generative capacity, but, rather, because the fourth dimension allows us to encode semantic relations independently of the relations—linear precedence, domination, and the relation expressed in the derivation trees—already built into TAG. The fundamental operation of the four dimensional grammars (concatenation of local four dimensional trees) is equivalent to adjunction of three dimensional trees. Again at this level, we can allow the yields of the local four dimensional trees to form any local set of three dimensional trees.

3. Multiple Modifiers

The issue in (1) (Schabes and Shieber, 1994) is the multiple modifiers of pepper. In the standard TAG account *red* adjoins into *roasted* and the derived auxiliary tree then adjoins into *pepper*. The correct semantic relationship, however, is a direct relation between *pepper* and each of modifiers *red* and *roasted*. Schabes and Shieber (Schabes and Shieber, 1994) suggest relaxing the notion of adjunction for modifier trees to allow both *red* and *roasted* to adjoin into the same node of *pepper*. Rambow, Vijay-Shanker and Wier (Rambow, Vijay-Shanker and Weir, 1995) employ a similar mechanism called *sister adjunction*. In our approach (Fig. 1[1]), we retain the standard adjunction of the one modifier tree into the other, but we do this in the elementary structure: the concatenation of the three-dimensional structures representing the modifiers forms the three-dimensional yield of a single four-dimensional local tree. This, then, adjoins as a whole into the elementary structure for *pepper*. The result is a local relationship, in the fourth dimension, between each of the modifier trees and the noun tree.

4. Raising Verbs and Sub-Aux Inversion

The issue in (2) is the interaction of the raising verb and the subject-aux inversion. In standard TAG accounts *does* and *seems* cannot occur in the same elementary tree. This leads either to a violation of CETM (in practice) or to a TL-MCTAG account or, in DTG, to the use of subsertion to interleave two underspecified trees. Our account depends on reinterpreting the elementary structure of *eat* as a four dimensional structure in which the yield, in effect, represents the subject as being adjoined into the root of the $[_{\mathrm{VP}}\mathrm{eat}]$.[2] This allows the structure for *does seem* to adjoin in the fourth dimension at that root node, in effect simultaneously adjoining (in the third dimension) *does seem* into the root of $[_{\mathrm{VP}}\mathrm{eat}]$ and the subject into the root of $[_{\mathrm{VP}}\mathrm{seem}]$. Once again, we have a local semantic relationship between the two elementary structures in the fourth dimension. More importantly, *does* and *seems* inhabit the same elementary structure but, in effect, wrap around the subject in the three dimensional yield.

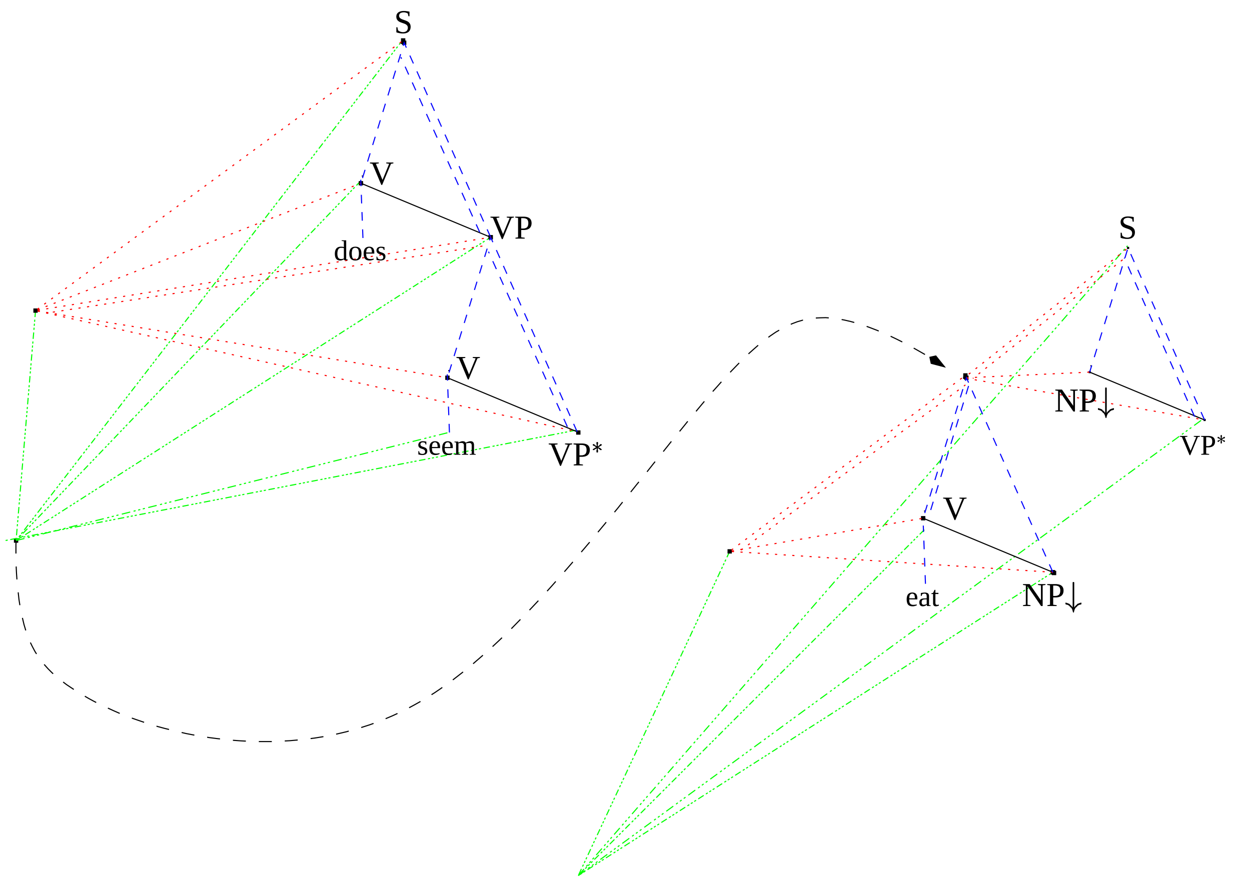
5. Multiple Raising Verbs

In example (3) (Frank, Kulick and Vijay-Shanker, 2000) the problem of the last example is exacerbated by multiply nested raising verbs. Frank, Kulick and Vijay-Shanker (2000) point out that, under the usual assumption that adjunction at the foot of a structure is prohibited, this sentence cannot be obtained from elementary structures that observe the CETM even by TL-MCTAG.

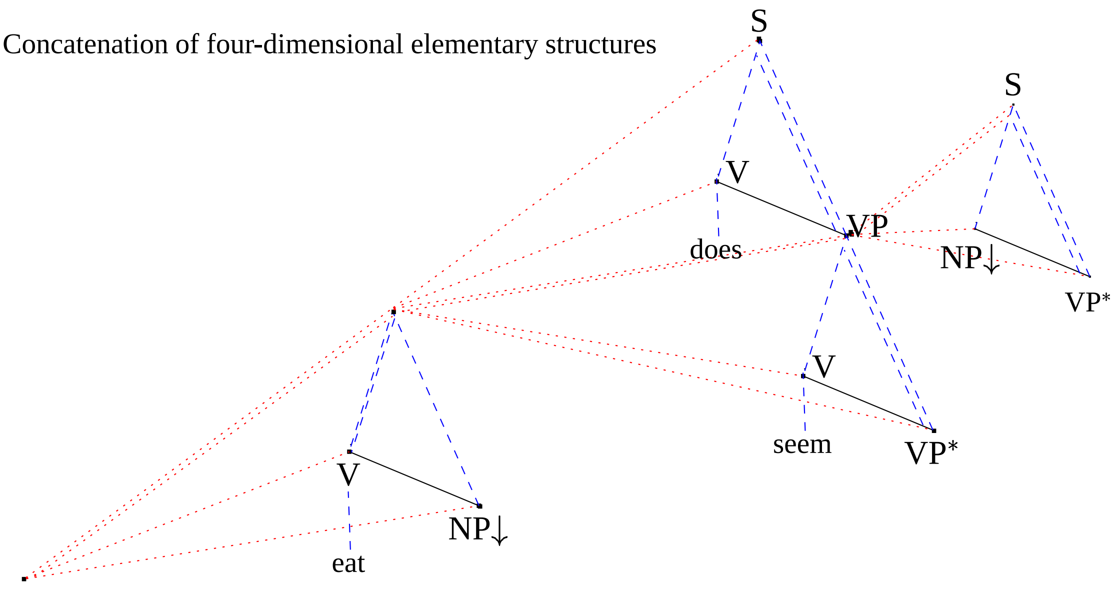
Given our interpretation of (2), the account of (3) follows with little additional complication. The *does seem* structure simply attaches to the *to be likely* structure which then attaches to the *eat* structure. Following the three-dimensional spines in taking the three-dimensional yield, the subject ends up attaching to the root of $[_{\mathrm{VP}}\mathrm{seem}]$. Following the two-dimensional spines in taking the the two-dimensional yield of the result, the composite [does Gabriel seem] effectively adjoins into the root of the $[_{\mathrm{VP}}\mathrm{to\ be\ likely}]$.

1. In interpreting the figures, the solid lines represent adjacency in the first dimension (string adjacency), the dashed lines represent adjacency in the second dimension (immediate domination), the dotted lines represent adjacency in the third dimension (the adjunction relationship) and the dash/dot-dot-dot lines represent adjacency in the fourth dimension. In addition, spines are marked, in each of the second and third dimensions, by doubling the lines. (Spines are trivial in the first dimension and irrelevant in the major dimension.) The significance of the spines is that the maximal point in the spine marks the *foot* of the structure. As in standard TAG, the foot is the point at which the structure dominated by the point of adjunction is attached in forming the $(n-1)$-dimensional yield of an n-dimensional structure.

2. Since there is no identification of the root and foot nodes with the node at which they adjoin we are free to label these nodes independently. We leave open the question of appropriate label for the root node of the *eat* structure. Note that the relaxation of the requirement that the root/foot/node of adjunction bear the same label is not new here—it was effectively abandoned at least by the introduction of FTAG. Formally, if we allow either features or adjunction constraints, the labeling constraint has no substantive effect on the generative capacity.

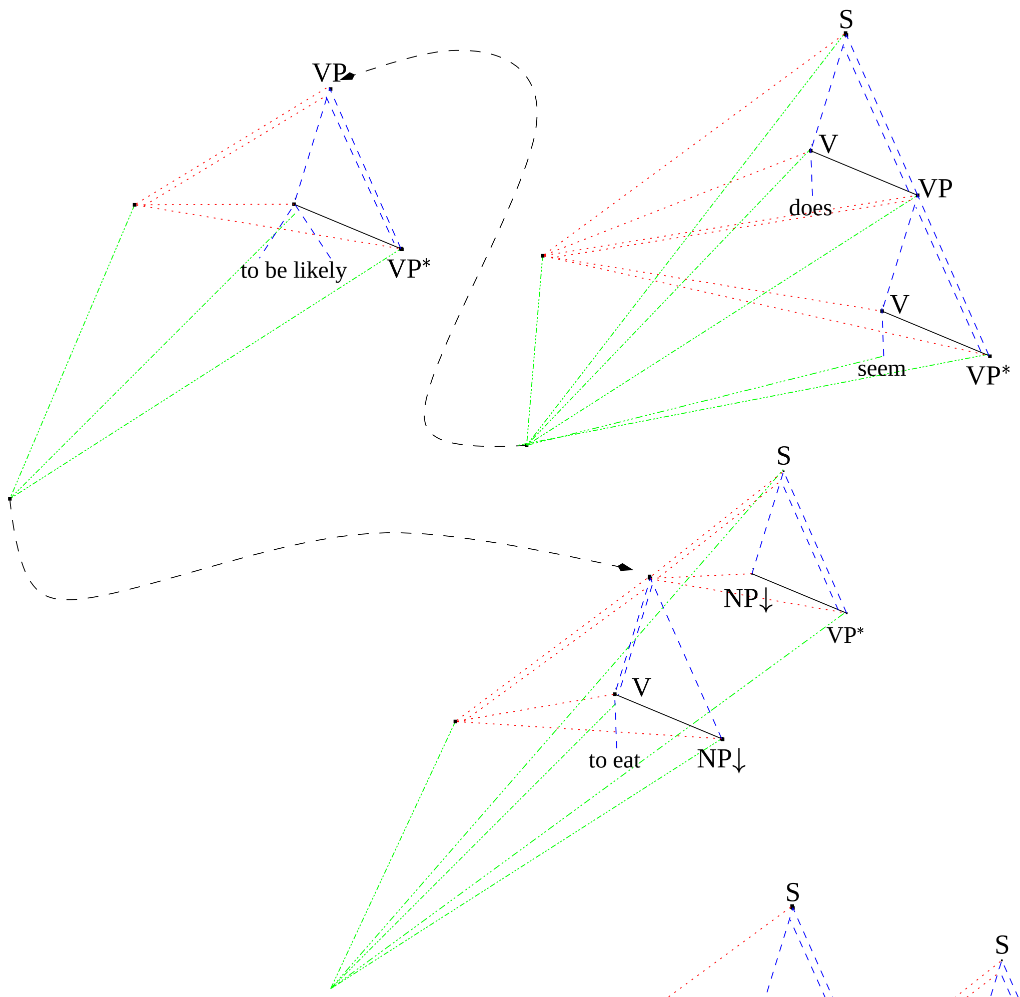


Concatenation of four-dimensional elementary structures

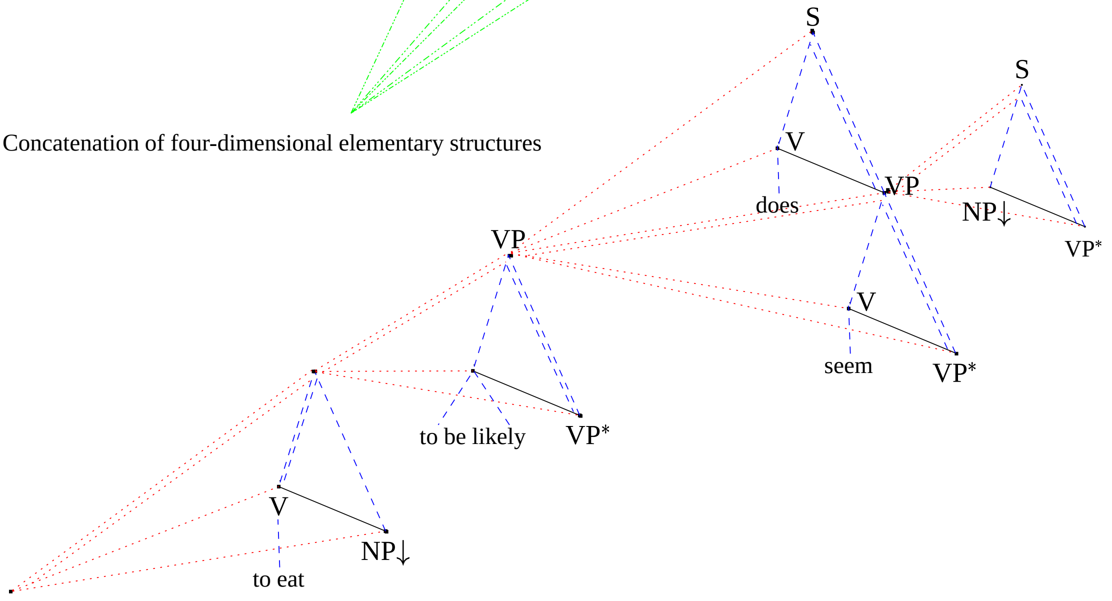


Three-dimensional yield

Figure 2: Does Gabriel seem to eat gnocchi



Concatenation of four-dimensional elementary structures



Three-dimensional yield

Figure 3: Does Gabriel seem to be likely to eat gnocchi

6. Interaction of Bridge and Raising Verbs

The fundamental issue in example (4 (Dras, Chiang and Schuler, To Appear) is, again, the inconsistency between the relations expressed in the derivation tree of the standard TAG account and the semantic relations between the constituent phrases. Standardly, the tree for *seems* adjoins into the tree for *eat* at the root of $[_{\mathrm{VP}}$to eat$]$ and the tree for *think* adjoins into the *eat* tree at the root of $[_{\mathrm{S}}$John to eat$]$. Consequently, there is no direct relationship between the *think* structure and the *seem* structure. Dras, Chiang and Schuler (To Appear) get the desired relationships (*think*—*seem*—*eat*) by using a two-level TAG in which the elementary trees of the first-level TAG are generated by a second-level TAG.

As we note below, this approach is pretty much equivalent to adopting a fourth dimension. Working within the current framework, this analysis turns out to be only slightly more complicated than the last. The interesting issue here is the complementizer *that* which belongs in the same initial structure as seem, but which needs to be separated from *seem* by the subject in the final two-dimensional structure. Here, again, we resort to attaching, in the third-dimension, the complementizer to the root of the [seem] substructure in much the same way that we attach the subject to the root of the VP in the main verb structure. With this configuration, the complementizer and *seem* are in the same elementary structure but can be wrapped around the *does Mary think* structure in the same way that *John* and *to eat* do.

Finally, note that, in this case, the root of the $[_{\mathrm{VP}}$think$]$ tree already has the *Mary* tree attached. Rather than pass the three-dimensional spine through that node, we pass it through the VP* node. Hence, *that* and *John* end up following the entire yield of *does Mary think* rather than being embedded in the middle of it.

7. Modifiers of Sentential Subjects

The final example (5) is from Candito and Kahane (1998a). It is also treated in Schuler (1999). The standard TAG account has *have to* adjoining into *stay* and the result substituting into *surprise*. But, adopting the semantic account that it is the fact that Paul *has to* stay that is surprising, we would expect to find direct semantic relations between *surprise* and *have to* and between *have to* and *stay*. To capture this, we can let the *stay* structure attach at the three-dimensional root of the *has to* structure and attach the *has to* structure at the S substitution node of the *surprise* structure. When these are collapsed to three dimensions, the *has to* structure ends up attached at the three-dimensional foot of the *stay* structure, which properly places it between *Paul* and *stay* in the two-dimensional yield.

One of the potentially attractive aspects of this formal interpretation is that it is equally possible to attach *has to* to *stay* (at the root of $[_{\mathrm{VP}}$stay$]$) from which one obtains a reading in which it is the fact that what Paul has to do is to stay that is surprising.

8. Conclusions and Relationship to Other Approaches

For each of the problematic constructions we started with, we have obtained structural configurations that express the intended semantic relationships directly while preserving the CETM in atomic, fully specified elementary structures. Moreover the semantic relations are consistently expressed in one particular structural relation, that of adjacency in the fourth dimension of our structures. While the notion of grammars over four dimensional objects may seem conceptually obscure, one can dispell much of that obscurity if one simply considers each dimension to represent a single type of relationship. Our structures have four dimensions solely because there are four types of relationship we need to express.

While each of the configurations we have proposed has been tailored to the needs of the construction we were attempting to capture, there does seem to be considerable regularity in the way the the elementary structures are arranged. Although we have made no attempt to motivate these structures on linguistic grounds, none of them seems completely beyond the pale. Whether the potential for overgeneration can be constrained in a principled way remains to be seen, but the fact that, at least in one case, a potential alternation reflects a corresponding ambiguity in the semantics is a little encouraging.

Formally, our approach is closest to that of Dras, Chiang and Schuller's use of Multi-Level TAGs (Dras, Chiang and Schuler, To Appear), in that both accounts employ mechanisms equivalent to the second level of Wier's Control Language Hierarchy. From our perspective, the structures generated by Multi-Level TAGs, in a sense, conflate relations of different type by encoding them with the same formal relation. This leads to structures that are quite complex and of less than ideal transparency. In clearly distinguishing the semantic relations from the syntactic

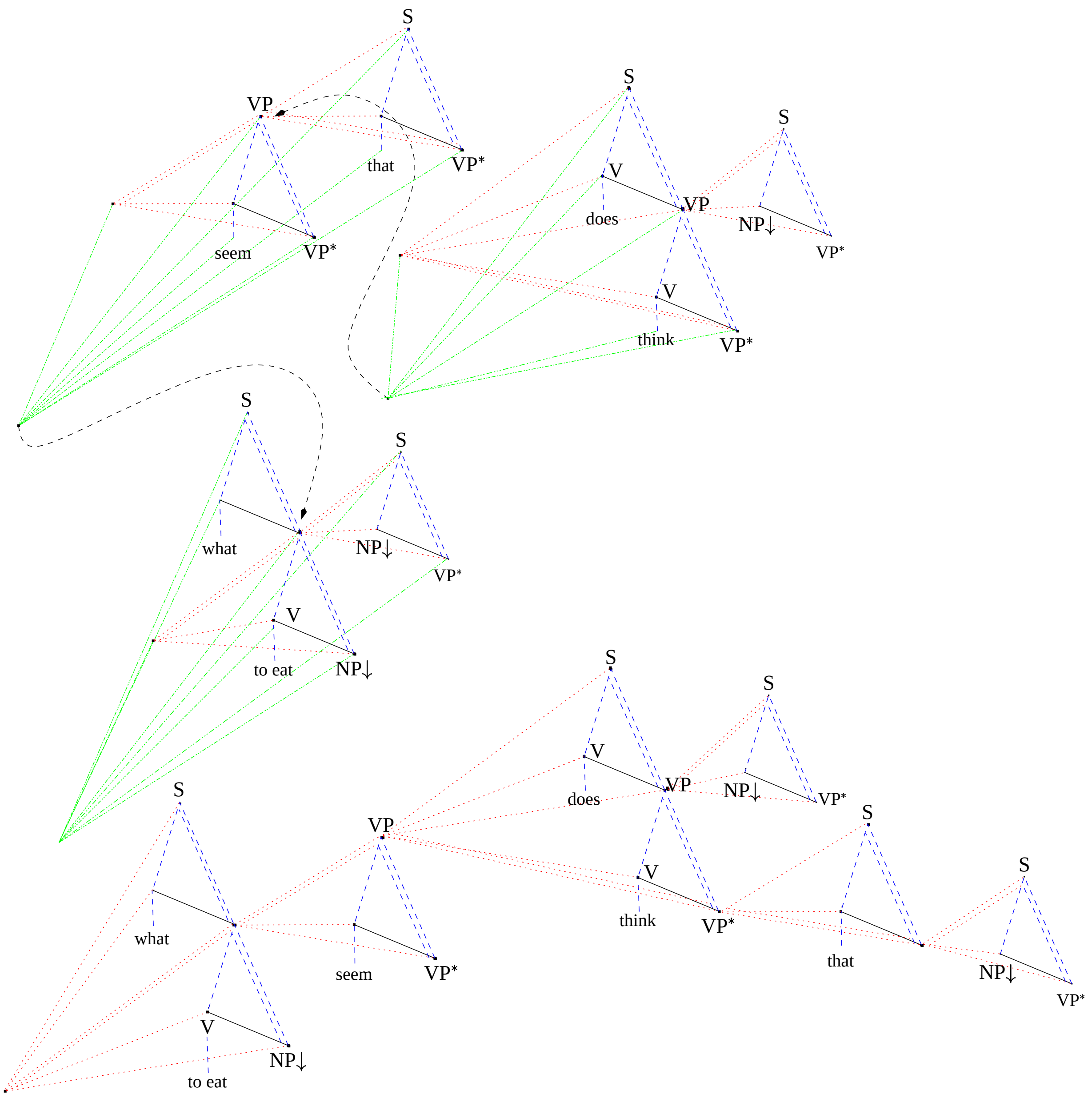


Figure 4: What does Mary think that John seems to eat

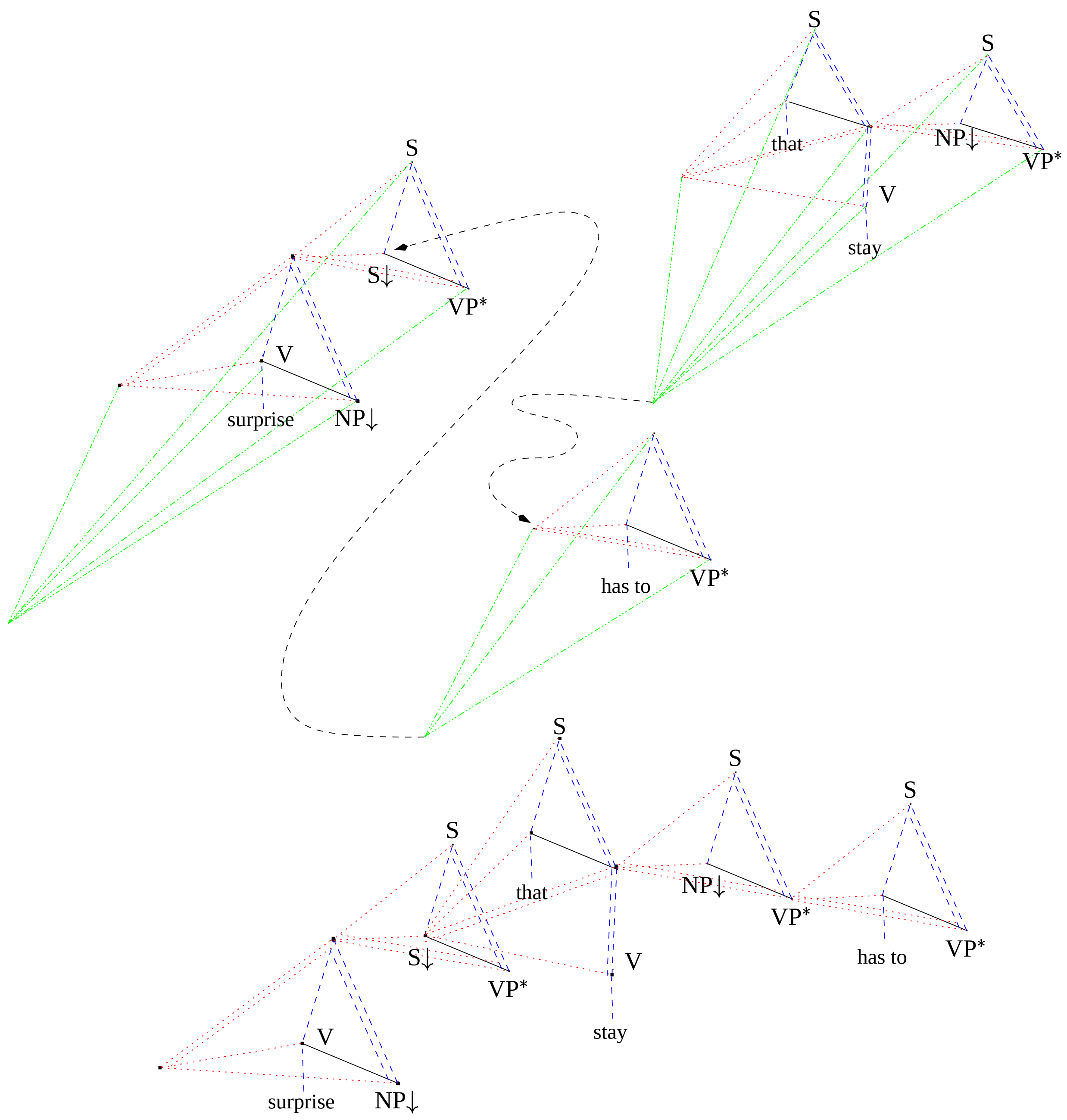


Figure 5: That Paul has to stay surprised Mary

ones, our approach is, perhaps, more closely related to that of Joshi and Vijay-Shanker (Joshi and Vijay-Shanker, 1999) in which semantic relations are not read directly off of the derivation trees but are rather expressed directly in semantic structures derived from those trees. In our approach, however, we preserve the original intuition that the structures generated by the syntactic analysis might directly express the semantic relationships.

References

Candito, Marie-Helene and Sylvain Kahane. 1998a. Can the TAG Derivation Tree Represent a Semantic Graph? An Answer in the Light of Meaning-Text Theory. In *Fourth International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+4)*, University of Pennsylvania.

Candito, Marie-Helene and Sylvain Kahane. 1998b. Defining DTG Derivations to get Semantic Graphs. In *Fourth International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+4)*, University of Pennsylvania.

Dras, Mark. 1999. A Meta-Level Grammar: Redefining Synchronous TAG for Translation and Paraphrase. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL'99)*, pages 80–87, Univ. of Maryland.

Dras, Mark, David Chiang and William Schuler. To Appear. A Multi-Level TAG Treatment of Dependency. *Journal of Language and Computation.*

Frank, Robert, Seth Kulick and K. Vijay-Shanker. 2000. Monotonic C-Command: A New Perspective on Tree Adjoining Grammar. *Grammars*, 3(2–3):151–173.

Group, The XTAG Research. 1998. A Lexicalized Tree Adjoining Grammar for English. Technical Report IRCS-98-18, Institute for Research in Cognitive Science.

Joshi, Aravind K. and K. Vijay-Shanker. 1999. Compositional Semantics with Lexicalized Tree-Adjoining Grammar (LTAG): How Much Underspecification is Necessary? In *Proceedings of the 2nd International Workshop on Computational Semantics*.

Mel'ĉuk, Igor A. 1988. *Dependency Syntax: Theory and Practice.* SUNY Series in Linguistics. Albany, NY: State University of New York Press.

Rambow, Owen, K. Vijay-Shanker and David Weir. 1995. D-Tree Grammars. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL'95)*, pages 151–158, Cambridge, MA.

Rogers, James. 1999. Generalized Tree-Adjoining Grammar. In *Sixth Meeting on Mathematics of Language*, pages 189–202.

Rogers, James. To Appear. wMSO Theories as Grammar Formalisms. *Theoretical Computer Science.*

Schabes, Yves and Stuart M. Shieber. 1994. An Alternative Conception of Tree-Adjoining Derivation. *Computational Linguistics*, 20(1):91–124.

Schuler, William. 1999. Preserving Dependencies in Synchronous Tree Adjoining Grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL'99)*, pages 88–95, Univ. of Maryland.

Weir, David J. 1992. A Geometric Hierarchy Beyond Context-Free Languages. *Theoretical Computer Science*, 104:235–261.

Using an Enriched TAG Derivation Structure as Basis for Semantics

Laura Kallmeyer
laura.kallmeyer@linguist.jussieu.fr
TALaNa-Lattice, University Paris 7, 2 place Jussieu, 75251 Paris Cedex 05, France

Abstract

Most of the proposals for semantics in the Tree Adjoining Grammar (TAG) framework suppose that the derivation tree serves as basis for semantics. However, in some cases the derivation tree does not provide the semantic links one needs. This paper concentrates on one of these cases, namely the analysis of quantifiers as adjuncts. The paper proposes to enrich the TAG derivation tree and use the resulting structure as basis for semantics. This allows to deal with quantifiers, even in PPs embedded into NPs, such that an adequate semantics with appropriate scope orders is obtained. The enriched derivation structure allows also to treat other cases that are problematic for the assumption that a TAG semantics can be based on the derivation tree.

1. TAG and the syntax-semantics interface

1.1. Lexicalized Tree Adjoining Grammars (LTAG)

A LTAG (Joshi and Schabes, 1997) consists of a finite set of trees (elementary trees) associated with lexical items and of composition operations of substitution (replacing a leaf with a new tree) and adjoining (replacing an internal node with a new tree). The elementary trees represent extended projections of lexical items and encapsulate syntactic/semantic arguments of the lexical anchor. They are minimal in the sense that all and only the arguments of the anchor are encapsulated, all recursion is factored away.

LTAG derivations are represented by derivation trees that record the history of how the elementary trees are put together. A derived tree is the result of carrying out the substitutions and adjoinings. For a sample derivation see the TAG analysis of (1) in Fig. 1. The numbers at the nodes in the derivation tree are the positions of the nodes where the trees are added: *John* is substituted for the node at position (1), *Mary* for the node at position (22) and *always* is adjoined to the node at position (2).

(1) John always loves Mary.

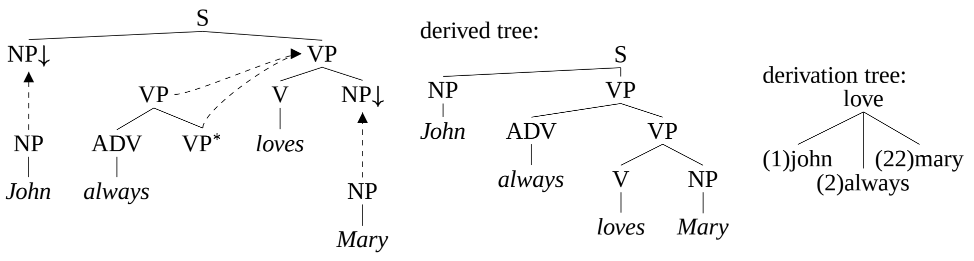


Figure 1: TAG derivation for (1)

1.2. Compositional semantics with LTAG

Because of the localization of the arguments of a lexical item within elementary trees TAG derivation trees express predicate argument dependencies. Therefore it is generally assumed that the proper way to define compositional semantics for LTAG is with respect to the derivation tree, rather than the derived tree (see e.g. Shieber and Schabes, 1990; Candito and Kahane, 1998; Joshi and Vijay-Shanker, 1999; Kallmeyer and Joshi 1999, 2002).

The overall idea is as follows. Each elementary tree is connected with a semantic representation. The way these semantic representations combine with each other depends on the derivation tree. Following Kallmeyer and Joshi (1999, 2002), in this paper, we will adopt 'flat' semantic representations as in, for example, Minimal Recursion Semantics MRS, (Copestake *et al.*, 1999). (2) shows the elementary semantic representations for (1).

(2)

l_1 : *love'*(x_1, x_2)	*john'*(x)	l_2 : *always'*(h_2)	*mary'*(y)
$h_1 \geq l_1$		$g_1 \geq l_2, h_2 \geq s_1$	
arg: $\langle x_1, (1) \rangle, \langle x_2, (22) \rangle$	arg: –	arg: g_1, s_1	arg: –

Roughly, a semantic representation consists of a conjunctively interpreted set of formulas (typed lambda-expressions), scope constraints and a set of argument variables. The formulas may contain labels and holes (metavariables for propositional labels). In the following, $l_1, l_2, \ldots$ are propositional labels, $h_1, h_2, \ldots$ are propositional holes, $s_1, s_2, \ldots$ are propositional argument variables (whose values must be propositional labels) and $g_1, g_2, \ldots$ are hole variables (special argument variables whose values must be holes). Argument variables may be linked to positions in the elementary tree, as it is the case for the variables of *love*.

The use of holes is motivated by the desire to generate underspecified representations (as in, e.g., Bos, 1995) for scope ambiguities. In the end, after having constructed a semantic representation with holes and labels, disambiguation is done which consists of finding bijections from holes to labels that respect the scope constraints and that are such that no label is below two labels that are siblings (e.g., this ensures that nothing can be in the restriction and the body of a quantifier at the same time). In the semantic representation for *love*, there is for example a hole h_1 above the label l_1 (indicated by the constraint $h_1 \geq l_1$). Between h_1 and l_1, other labels and holes might come in (introduced for example by quantifiers or adverbs) or, if this is not the case, l_1 will be assigned to h_1 in the disambiguation(s).

When combining semantic representations, values are assigned to argument variables and, roughly, the union of the semantic representations is built. The values for the argument variables of a certain (elementary) semantic representation must come from semantic representations that are linked to it in the derivation tree.

The linking of argument variables and syntactic positions restricts the possible values as follows: In a substitution derivation step at a position p, only argument variables linked to p get values. In an adjunction step, only argument variables that are not linked to any position can get values. In the case of a substitution, a new argument is inserted and therefore a value is assigned to an argument variable in the old semantic representation. However, in the case of an adjunction, a new modifier is applied and therefore a value is assigned to a variable in the semantic representation that is added. In this sense, in a substitution step, the variable assignment is downwards whereas in an adjunction step it is upwards.

The derivation tree in Fig. 1 indicates that the value of x_1 needs to come from the semantic representation of *John*, the one of x_2 from *Mary* and the values of g_1 and s_1 need to come from *love*. Consequently, $x_1 \to x, x_2 \to y, g_1 \to h_1$ and $s_1 \to l_1$. As a result we obtain the semantic representation shown in (3).

(3)

l_1 : *love'*(x, y), *john'*(x), *mary'*(y), l_2 : *always'*(h_2)
$h_1 \geq l_1, h_1 \geq l_2, h_2 \geq l_1$
arg: –

According to (3), $h_1 \geq l_2$, $l_2 > h_2$ (because h_2 appears inside a formula labelled l_2) and $h_2 \geq l_1$. Consequently $h_1 \neq l_1$ and therefore the only possible disambiguation is $h_1 \to l_2, h_2 \to l_1$. This leads to the semantics *john'*$(x) \wedge$ *mary'*$(y) \wedge$ *always'*(*love'*$(x, y))$.

1.3. Separating scope and predicate argument information

A central aspect of (Kallmeyer and Joshi 1999, 2002) is the idea that the contribution of a quantifier is separated into a scope and a predicate argument part. Accordingly, quantifiers have a set of two elementary trees and mulicomponent TAGs are used. An auxiliary tree consisting of a single node is linked to the scope part of the semantics of the quantifier, while an initial tree is linked to the predicate argument part. E.g., consider (4).

(4) every dog barks

Fig. 2 shows the syntactic analysis of (4) in this framework. The semantic representations corresponding to the four elementary trees are shown in (5).

(5)

l_1 : *bark'*(x_1)	l_2 : *every'*(x, h_2, h_3)	l_3 : $p_1(x)$	q_1 : *dog'*
$l_1 \leq h_1$	$s_1 \leq h_3$	$l_3 \leq h_2$	
arg: x_1	arg: s_1	arg: p_1	arg: –

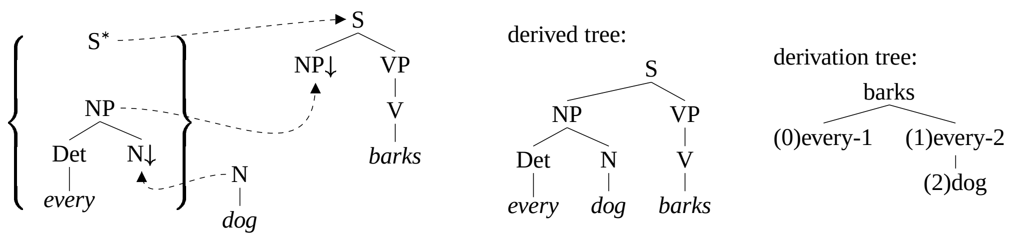


Figure 2: Syntactic analysis of (1)

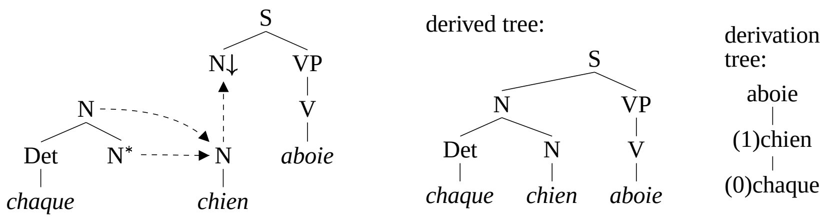


Figure 3: Syntactic analysis of (7)

The scope part of the quantifier (second representation in (5)) introduces a proposition containing the quantifier, its variable and two holes for its restrictive and nuclear scope. The proposition this semantic representation is applied to (variable s_1) is in the nuclear scope of the quantifier ($s_1 \leq h_3$). The predicate argument part (third representation in (5)) introduces a proposition $p_1(x)$ where p_1 will be the noun predicate *dog'*. This proposition is in the restrictive scope of the quantifier ($l_3 \leq h_2$). The values for the argument variables are $x_1 \rightarrow x, s_1 \rightarrow l_1, p_1 \rightarrow q_1$ which leads to (6). The only disambiguation is $h_1 \rightarrow l_2, h_2 \rightarrow l_3, h_3 \rightarrow l_1$ which leads to *every'*$(x,$ *dog'*$(x),$ *bark'*$(x))$.

(6)

l_1 : *bark'*(x), l_2 : *every'*(x, h_2, h_3), l_3 : *dog'*(x) $l_1 \leq h_1, l_1 \leq h_3, l_3 \leq h_2$
arg: –

To account for cases with more than one quantifier, a restricted use of multiple adjunctions is necessary.

2. Quantifiers as adjuncts

2.1. The problem

The approach of Kallmeyer and Joshi is problematic in cases where quantifiers are analyzed as adjuncts. Such an analysis however is proposed for English in (Hockey and Mateyak, 2000) and for French in (Abeillé, 1991). Abeillé's proposal for French is even adopted in the French TAG implemented at the University Paris 7 (Abeillé, Candito and Kinyon, 2000). In the following we will sketch the French quantifier analysis.

(7) chaque chien aboie

According to Abeillé (1991), in (7), the French translation of (4), the noun is first added to the verb by substitution, and then the quantifier is adjoined to the noun (see Fig. 3). In the derivation tree, there is no link (i.e., no edge) between the quantifier and the verb. However, the variable introduced by the quantifier is an argument of the verb, and, furthermore, the proposition introduced by the verb is part of the nuclear scope of the quantifier. (This is why l_1 was assigned to the argument variable s_1 in (5).) Therefore, for semantics, a link between the quantifier and the verb is needed.

The use of a second elementary tree for the scope part that is adjoined to the whole sentence would require non-local MCTAG since the quantifier is not adjoined to the verb but to the noun. Non-local MCTAG is much more powerful than TAG and a solution using TAG or a mildly context-sensitive TAG variant is preferable. Therefore I will not adopt the idea of (Kallmeyer and Joshi, 2002) to separate the contribution of a quantifier into two parts.

An advantage of not doing so is that multiple adjunctions are not necessary. Multiple adjunctions are problematic because in combination with multicomponent derivations, they extend the generative power of the grammar, and therefore their use needs to be restricted.

2.2. Enriching the derivation tree

The syntactic analysis of (7) shows that, if one wants to retain the semantic analysis given in (5) for quantifiers, the quantifier *chaque* in Fig. 3 needs to have access to both elementary semantic representations, the one of *aboie* and the one of *chien*. This means that the derivation tree is too restrictive, it does not provide the dependency structure one needs for semantics. On the other hand, the way syntactic elements are put together in a derivation reflects predicate-argument relations, i.e., seems still to provide the dependencies semantics should be based on. Therefore this paper proposes to keep the idea of using the derivation tree for semantics, but to enrich the derivation tree in order to obtain the semantic links one needs.

The basic intuition is as follows: In Fig. 3 for example, the quantifier is adjoined to the root of the initial tree for *chien*, and consequently in the derived tree it will be a direct neighbour not just of the elementary tree for *chien* but also of the one for *aboie* (see Fig. 4 where the elementary tree of the quantifier is marked by a triangle). Therefore there is a syntactic link between the quantifier and the verb and this should enable the quantifier to have semantic access to the verb. For this reason, in the case of an adjunction at a root node of some elementary γ, the adjoined tree is not only connected to γ but also to the tree to which γ was added in some previous derivation step. The enriched derivation structure used for semantics is called *e-derivation structure* for short. The e-derivation structure of (7) is shown in Fig. 4, the additional link, i.e., the one that does not be part of the derivtaion tree, is depicted as a dotted edge.

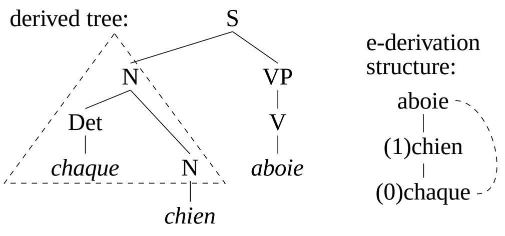


Figure 4: Derived tree and e-derivation structure of (7)

In a feature-structure based TAG (FTAG, (Vijay-Shanker and Joshi, 1988)), the additional connection between the qunatifier and the verb is even more obvious, since a unification of feature structures from all three trees (here *aboie*, *chaque* and *chien*) takes place. See Fig. 5 for a general FTAG derivation sequence of substitution and adjunction at the root of the tree that was added by substitution. In the derived tree, the root of the adjoined tree carries a top feature structure that results from unifying feature structures from all three trees involved in this derivation. In this sense, the links in a e-derivation structure reflect unifications of feature structures between elementary trees.

This feature-structure view is related to the question whether in the derived tree in Fig. 4, it is appropriate to consider the upper N node as being part of just the elementary tree of the quantifier, as done in standard TAG, instead of considering it as a node shared between the three elementary trees. I will not further pursue this question in this paper.

As mentioned above, trees that are neighbours of each other in the derived tree, should be related in the e-derivation structure. However, it is not just neighbourhood in the derived tree that determines whether two elementary trees are linked in the e-derivation structure. If this was the case, such a link (for example the one between *chien* and *aboie*) would be destroyed when adjoining at the root of the lower tree. Consequently, the e-derivation structure would not be monotonic with respect to derivation. Since semantics will be defined based on this structure, non-monotonicity is something one definitely wants to avoid.

Therefore I propose the following definition of e-derivation structure: all edges occurring in the derivation structure are also e-derivation links. Furthermore, two nodes labelled γ and β are linked if in the derivation tree there are nodes $\gamma', \beta_1, \dots, \beta_n$ such that γ' is daughter of γ, β_1 daughter of γ' with position 0 (adjunction at the root of γ'), β_{i+1} is daughter of β_i with position 0 ($1 \leq i < n$) and $\beta_n = \beta$. (The definition applies to the derivation

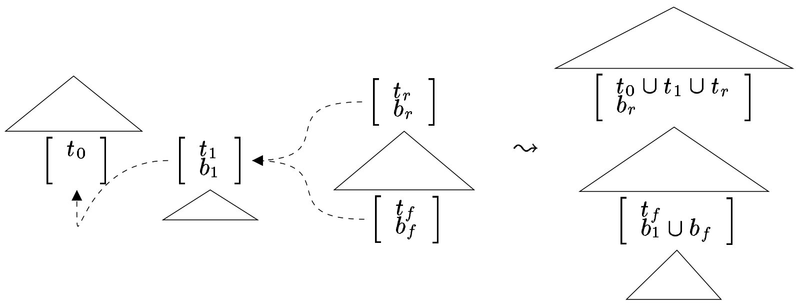


Figure 5: E-derivation structure reflects unifications of feature structures between elementary trees

of (7) with γ being the elementary tree of *aboie*, γ' being the elementary tree of *chien* and $\beta_1 = \beta_n = \beta$ being the one of *chaque*.) Fig. 6 sketches the general definition of the e-derivation structure.

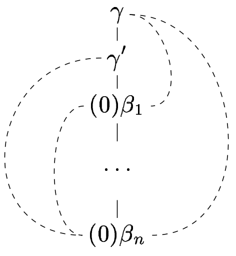


Figure 6: E-derivation structure

Two kinds of edges are distinguished in e-derivation structures: edges belonging also to the derivation tree are called *primary edges* whereas edges that do not belong to the derivation tree are called *secondary edges*. We will see that this is a useful distinction, since primary edges seem to be stronger with respect to semantics.

The e-derivation structure is a graph, not necessarily a tree. This is one of the differences compared to the meta-level derivation structure proposed in (Dras, Chiang and Schuler, 2000) that is also intended to represent a "more semantic" dependency structure than the original derivation tree.

It is important to emphasize that the e-derivation structure is a way of making information about shared nodes (or in an FTAG shared feature structures) explicit that is already present in the original derivation tree. In this sense a semantics based on the e-derivation structure is still a semantics based only on the derivation tree.

For the semantics of (7), the representations in (8) are used. These are more or less the same as in (Kallmeyer and Joshi, 2002), except that the quantifier contribution is not separated into two parts.

(8)

l_1 : *aboire'*(x_1) $l_1 \leq h_1$	l_2 : *chaque'*(x, h_2, h_3), $l_3 : p_1(x)$ $s_1 \leq h_3, l_3 \leq h_2$	q_1 : *chien'*
arg: $\langle x_1, (1) \rangle$	arg: s_1, p_1	arg: –

I will keep the idea that in case of a substitution, the variable assignment is downwards while in case of an adjunction it is upwards. An argument variable linked to a substitution position p receives its value from an elementary tree below or equal to the tree substituted at position p. A variable that is not linked to a substitution position and that belongs to a tree that is added by adjunction receives its value from some tree above the adjoined tree. Here, 'below' and 'above' refer to the derivation tree, i.e., the primary edges in the e-derivation structure.

According to the e-derivation structure of (7), the only possible assignment for the argument variables is $x_1 \to x, s_1 \to l_1, p_1 \to q_1$ which leads to (9).

(9)

l_1 : *aboire'*(x), l_2 : *chaque'*(x, h_2, h_3), l_3 : *chien'*(x) $l_1 \leq h_1, l_1 \leq h_3, l_3 \leq h_2$
arg: –

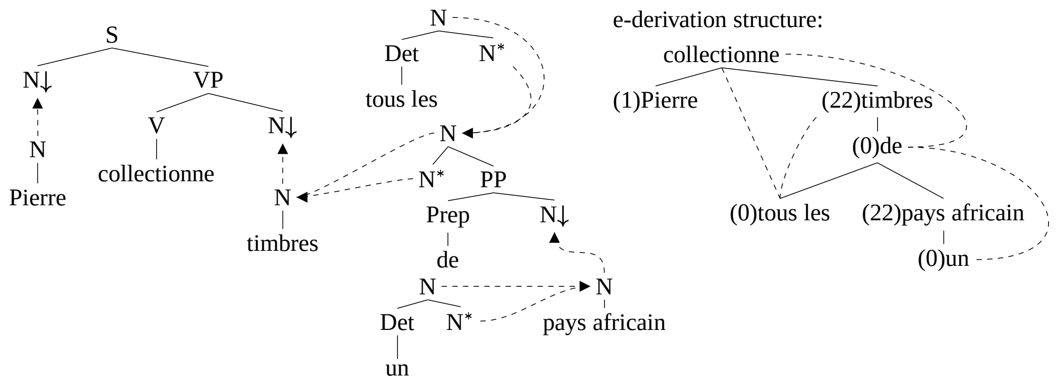


Figure 7: Syntactic analysis and e-derivation structure of (10)

3. Quantifiers embedded into NPs

One of the first questions to be considered is whether the analysis proposed above allows quantifiers embedded into NPs to have wide scope. Consider (10) and (11).

(10) Pierre collectionne tous les timbres d'un pays africain
'Pierre collects every stamp of an African country'

(11) Pierre connaît tous les détenteurs d'un prix
'Pierre knows every winner of a price'

Both sentences are ambiguous with respect to quantifier scope: the two meanings of (10) are that Pierre collects either 1) all the stamps coming from African countries (wide scope of *tous les timbres*) or 2) the stamps of a single specific African country (wide scope of *un pays africain*). The two meanings of (11) are that either 1) Pierre knows everybody who obtained some price (wide scope of *tous les détenteurs*) or 2) there is a specific price such that Pierre knows everybody who obtained that price (wide scope of *un prix*). An adequate semantic analysis should allow for both scope orders, if possible they should be represented in one underspecified semantic representation.

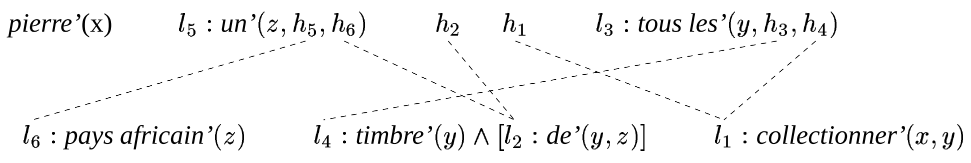
There is a crucial difference between (10) and (11). In (10) the embedded PP is not an argument and therefore, in the FTAG analysis, it is adjoined to the noun of the higher NP, whereas in (11) the PP *d'un prix* is an argument of the noun *détenteur* and therefore it is added by substitution.[1] In the following, I will consider in detail the two syntactic and semantic analyses.

3.1. PPs as noun adjuncts

For (10), the elementary trees and the way they are put together is shown in Fig. 7 (leaving aside the decomposition of *pays africain*). As it is traditionally done in TAG, I suppose NA conditions for foot nodes. The elementary trees shown in Fig. 7 allow a second analysis, namely adjoining *tous les* at the root of *timbres* and then adjoining *de* at the root of *tous les*. This would lead to another derived and another derivation tree but the e-derivation structure would be the same, except for the distinction between primary and secondary edges. However, the use of adequate features can block this second derivation. This is for example done in the French TAG Grammar (Abeillé, Candito and Kinyon, 2000). Therefore, in the following, I will only consider the analysis in Fig. 7.

The semantic analysis that I propose in the following is such that the scope of a quantifier is not restricted by something higher in the scope order and therefore, in the case of (10), *un* can rise and get scope over *tous les*. The semantic representations for the elementary trees in Fig. 7 are shown in Fig. 12. The ones for *collectionne, Pierre, timbres, tous les, pays africain* and *un* follow the proposals made in Section 2.2. The semantic representation

1. Interestingly, quantifiers in adjunct PPs seem to have a stronger preference for wide scope than quantifiers in argument PPs. I would like to pursue the issue of scope preferences in future work, but in this paper this is left aside.



pierre'(x)
l_5 : *un'*(z, h_5, h_6)
h_2
h_1
l_3 : *tous les'*(y, h_3, h_4)
l_6 : *pays africain'*(z)
l_4 : *timbre'*$(y) \wedge [l_2$: *de'*$(y, z)]$
l_1 : *collectionner'*(x, y)

Figure 8: Graphical representation of the scope constraints for (10)

of *de* is such that it takes a predicate p_1, in this case *timbre'*, and modifies it such that *timbre'* is replaced by $\lambda u[\textit{timbre'}(u) \wedge [l_2 : \textit{de'}(u, x_3)]]$. The propositional label l_2 is needed in order to make *de'*(u, x_3) accessable for quantifiers. When adding *un*, l_2 is assigned to s_2.

(12)

l_1 : *collectionner'*(x_1, x_2) $l_1 \leq h_1$ arg: $\langle x_1, 1\rangle, \langle x_2, 22\rangle$	*pierre'*(x) arg: –	q_1 : *timbre'* arg: –	$q_2 : \lambda u[p_1(u) \wedge [l_2$: *de'*$(u, x_3)]]$ $l_2 \leq h_2$ arg: $p_1, \langle x_3, 22\rangle$
l_3 : *tous les'*$(y, h_3, h_4), l_4 : p_2(y)$ $s_1 \leq h_4, l_4 \leq h_3$ arg: s_1, p_2	q_3 : *pays-africain'* arg: –	l_5 : *un'*$(z, h_5, h_6), l_6 : p_3(z)$ $s_2 \leq h_6, l_6 \leq h_5$ arg: s_2, p_3	

Taking the semantic representations from (12) and the e-derivation structure from Fig. 7, the semantic assignments when building a semantic representation of (10) are as follows:

For x_1, x_2 and x_3 free individual variables that are not argument variables must be found. The value of x_1 has to come from something (below the tree) substituted at position (1), therefore $x_1 \rightarrow x$. The value of x_2 has to come from some tree that is (below the tree) substituted at position (22) and that is linked to *collectionne*. Consequently, it has to come from *timbres*, *de* or *tous les*. The only possibility is $x_2 \rightarrow y$. For x_3, the value needs to come from one of the elementary trees added below the position (22) to the elementary tree of *de*, i.e. the value is taken from *pays africain* or *un*. Consequently, $x_3 \rightarrow z$.

For p_1, since it is not linked to a substitution position, one needs to find a unary predicate label or free variable in one of the elementary representations already present when adjoining the elementary tree of *de*, i.e., a value coming from *timbres* or *collectionne*. The only possibility is $p_1 \rightarrow q_1$. The values of s_2 and p_3 come from *de* or *pays africain*. Consequently $s_2 \rightarrow l_2$ and $p_3 \rightarrow q_2$ or $p_3 \rightarrow q_3$. With $s_2 \rightarrow l_2, p_3 \rightarrow q_2$ is not possible. Otherwise, l_2 would be in the restriction of *un'* ($h_5 \geq l_6 > l_2$) and in its body ($l_2 \leq h_6$). This is a contradiction to the definition of well-formed semantic representations. Therefore $p_3 \rightarrow q_3$.

The values for s_1 and p_2 must be taken from *de*, *timbres* or *collectionne*. For s_1 we get two possibilities, l_1 or l_2 and for p_2 possible values are q_1 or q_2.

For p_2, $p_2 \rightarrow q_2$ is preferable because $p_2 \rightarrow q_1$ would mean that we first have to do this assignment, i.e. to produce $l_4 : (q_1 : \textit{timbre'})(y)$ and then to perform $p_1 \rightarrow q_1$, i.e., to produce $l_4 : \textit{timbre'}(y) \wedge [l_2 : \textit{de'}(y, z)]$ because this last makes q_1 disappear (see the definition of semantic composition in (Kallmeyer and Joshi, 2002)). On the other hand, with $p_2 \rightarrow q_2$, the order in which the semantic representations are put together, does not matter. Therefore the last assignment is preferable and I propose to adopt the rule that, in case of two possibilities where just one corresponds to a primary edge in the e-derivation structure (in this case the one between *tous les* and *de*, i.e., $p_2 \rightarrow q_2$), this last one is chosen.

For s_1, $s_1 \rightarrow l_2$ is excluded because it would lead to a contradiction: it would lead to the constraints $l_2 \leq h_4$ and $l_4 \leq h_3$. With $l_2 \leq l_4$ (because the proposition labeled l_2 is embedded in the one labeled l_4, this gives $l_2 \leq h_4$ and $l_2 \leq h_3$, i.e. l_2 is in the restriction and the body of *tous les'*. Therefore $s_1 \rightarrow l_1$.

The semantic representation one obtains is (13):

(13)

l_1 : *collectionner'*(x, y), *pierre'*(x), l_3 : *tous les'*(y, h_3, h_4), $l_4 : (q_2 : \lambda u[\textit{timbre'}(u) \wedge [l_2 : \textit{de'}(u, z)]])(y), l_5 : \textit{un'}(z, h_5, h_6), l_6 : (q_3 : \textit{pays africain'})(z)$ $l_1 \leq h_1, l_2 \leq h_2\ l_1 \leq h_4, l_4 \leq h_3\ l_2 \leq h_6, l_6 \leq h_5$
arg: –

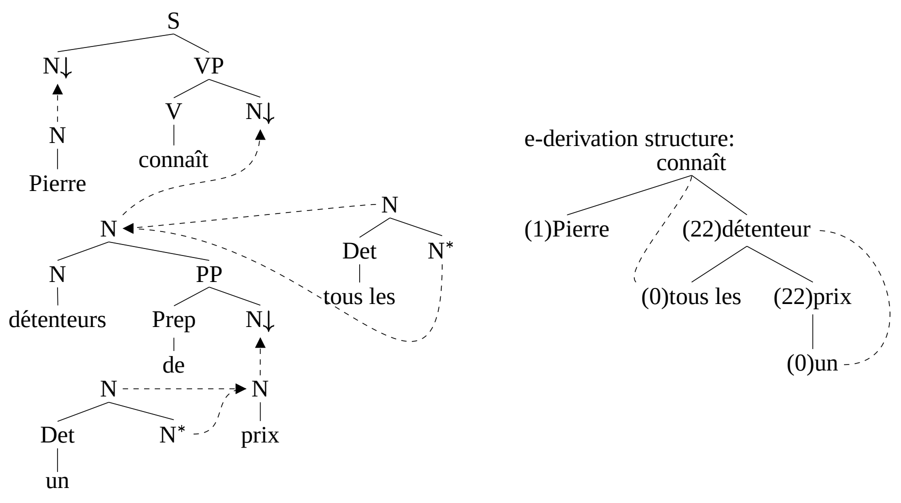


Figure 9: Syntactic analysis of (11)

The scope constraints in (13) are depicted in Fig. 8. As one can see, there is no constraint restricting the order of l_5 and l_3. The only restrictions we have are that, if *tous les* is in the scope of *un*, it must be in its body, and if *un* is in the scope of *tous les*, it must be in its restriction. This follows from the constraints involving l_2 and l_4.

(14) $\delta_1 : \left\{ \begin{array}{lll} h_1 \to l_3 & h_2 \to l_2 & h_3 \to l_5 \\ h_4 \to l_1 & h_5 \to l_6 & h_6 \to l_4 \end{array} \right.$, $\delta_2 : \left\{ \begin{array}{lll} h_1 \to l_5 & h_2 \to l_2 & h_3 \to l_4 \\ h_4 \to l_1 & h_5 \to l_6 & h_6 \to l_3 \end{array} \right.$

The two disambiguations corresponding to the two scope orders are shown in (14). δ_1 corresponds to wide scope of *tous les'* and δ_2 to wide scope of *un'*. These are the only two disambiguations for (13).

3.2. PPs as arguments of NPs

The case of the PP being an argument of the noun, as in (11), is actually the simpler case of the two constructions with PPs embedded into NPs. There is no extra elementary tree for the preposition. Instead the preposition is treated as semantically void and it is part of the elementary tree of the noun that selects for the PP, in this case *détenteur*. Furthermore, this elementary tree contains a substitution node for the embedded noun, its argument. The syntactic analysis and the e-derivation structure for (11) is shown in Fig. 9.

The semantic representations for (11), shown in (15), are more or less the same as for (10), except for the one for *détenteur*. This semantic representation gives the predicate used as argument of *tous les* and, at the same time, contains the proposition that is argument of *un*.

(15)

$\boxed{\begin{array}{l} l_1 : \textit{connaître'}(x_1, x_2) \\ l_1 \le h_1 \\ \hline \text{arg: } \langle x_1, 1\rangle, \langle x_2, 22\rangle \end{array}}$ $\boxed{\begin{array}{l} \textit{pierre'}(x) \\ \hline \text{arg: } - \end{array}}$ $\boxed{\begin{array}{l} q_1 : \lambda u[l_2 : \textit{détenteur-de'}(u, x_3)] \\ l_2 \le h_2 \\ \hline \text{arg: } \langle x_3, 22\rangle \end{array}}$

$\boxed{\begin{array}{l} l_3 : \textit{tous les'}(y, h_3, h_4), l_4 : p_1(y) \\ s_1 \le h_4, l_4 \le h_3 \\ \hline \text{arg: } s_1, p_1 \end{array}}$ $\boxed{\begin{array}{l} q_2 : \textit{prix'} \\ \hline \text{arg: } - \end{array}}$ $\boxed{\begin{array}{l} l_5 : \textit{un'}(z, h_5, h_6), l_6 : p_2(z) \\ s_2 \le h_6, l_6 \le h_5 \\ \hline \text{arg: } s_2, p_2 \end{array}}$

For the same reasons as in (10), we get the following assignments: $x_1 \to x$, $x_2 \to y$, $x_3 \to z$, $p_1 \to q_1$ and $s_1 \to l_1$. For s_2, $s_2 \to l_2$ is the only possibility, and for p_2, $p_2 \to q_2$ is chosen, it follows not only the primary edge but it is even the only possibility because $p_2 \to q_1$ would lead to $l_2 \le h_6$ and $l_2 \le h_5$ which contradicts the separation of restriction and body of a quantifier. The semantic representation in (16) is obtained for (11).

(16)

l_1 : *connaître'*(x, y), *pierre'*(x), l_3 : *tous les'*(y, h_3, h_4), l_4 : $(q_1 : \lambda u[l_2 :$ *détenteur-de'*$(u, z)]])(y)$ l_5 : *un'*(z, h_5, h_6), l_6 : $(q_2 :$ *prix'*$)(z)$ $l_1 \leq h_1, l_2 \leq h_2$ $l_1 \leq h_4, l_4 \leq h_3$ $l_2 \leq h_6, l_6 \leq h_5$
arg: –

(16) corresponds to the semantic representation of (10), in particular the constraints for quantifier scope are the same. Consequently, as in the case of (10), the two scope orders of the quantifiers are both possible.

As we have seen, the approach proposed in this paper correctly allows quantifiers in embedded PPs to take wide scope. Furthermore, for cases of scope ambiguities, it even allows to generate appropriate underspecified representations.

4. Unbounded dependencies in embedded interrogatives

The problem this paper concentrates on are quantifiers and their analysis in TAG. However, the enriching of the derivation tree proposed above is also useful for other problems one encounters when doing semantics with TAG. One often mentioned problem are unbounded dependencies in embedded interrogatives as in (17).

(17) Mary wondered who Peter thought John said Bill liked

An adequate semantics for (17) should have the following structure:

(18) *wonder'*$($*mary'*$,$ *who'*$(x,$ *think'*$($*peter'*$,$ *say'*$($*john'*$,$ *like'*$($*bill'*$, x)))))$.

The embedding of *think'*$(\ldots)$ into *who'*$(x, \ldots)$ is a scope relation while the other embeddings are predicate argument relations. Both should be part of an adequate semantic representation and I expect the structure underlying semantics to provide all the links necessary for scope and for the predicate argument structure. (In the case of scope ambiguities, scope can of course be partly unspecified.)

In order to obtain the relation between *think'* and *who'*, *think* must be connected either to *who* or to *like* (if the semantic representation of *like* contains a part that corresponds to its 'moved' wh-part). Fig. 10 shows the classical TAG analysis of (17), following (Kroch, 1987). The derivation tree does not contain the necessary links. The e-derivation structure however provides an additional link between *like* and *say*. Based on this structure it is possible to build an appropriate semantics for such cases.

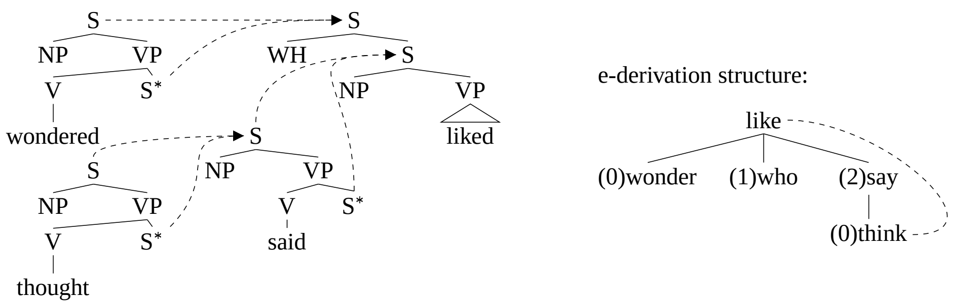


Figure 10: TAG derivation and e-derivation structure for (17)

5. Related work

An approach that also defines an additional structure related to the derivation tree in order to solve the problems one accounts with a semantics directly based on the derivation tree is (Dras, Chiang and Schuler, 2000). Dras et al. view the derivation tree as a tree derived by a meta-level TAG and the derivation trees provided by this second TAG are the structures they use for semantics. An obvious advantage of our approach is that it is less complex. Starting from the derivation tree it is easy to obtain the e-derivation structure. Furthermore, the e-derivation structure is a natural extension in the sense that it just makes things explicit that are already present in the derivation tree.

Frank and van Genabith (2001) propose to define TAG semantics based on the derived tree in order to solve the problems mentioned in this paper. However, they make use not only of the information available in the derived tree but also of information about how the elementary trees were put together, i.e., of the derivation tree. Compared to this, the advantage of the approach proposed here is that semantics is based only on the enriched derivation tree and does not need to go back and to use both, the derived tree and the derivation tree.

6. Conclusion

I have shown in this paper that, in spite of some mismatches between TAG derivation trees and dependency structures, it is possible to build a semantics in the TAG framework based on the derivation trees. The key idea is that I am enriching the derivation tree by making links between elementary trees explicit that are already present in the derivation and that can be read off the derivation tree. This enriched structure called e-derivation structure is used as basis for semantics. This approach allows to account for the semantics of quantifiers even if a syntactic analysis is assumed that treats quantifiers as noun adjuncts. I have shown that the semantics proposed here correctly allows quantifiers embedded into NPs to take wide scope. Furthermore, the e-derivation structure also allows to deal with other phenomena that are problematic for the assumption that derivation trees provide the right dependency structure to use for semantics, such as unbounded dependencies in embedded interrogatives.

References

Abeillé, Anne. 1991. *Une grammaire lexicalisée d'arbres adjoints pour le français: application à l'analyse automatique*. Ph.D. thesis, Université Paris 7.

Abeillé, Anne, Marie-Hélène Candito and Alexandra Kinyon. 2000. The current status of FTAG. In *Proceedings of TAG+5*, pages 11–18, Paris.

Bos, Johan. 1995. Predicate Logic Unplugged. In Paul Dekker and Martin Stokhof, editors, *Proceedings of the 10th Amsterdam Colloquium*, pages 133–142.

Candito, Marie-Hélène and Sylvain Kahane. 1998. Can the TAG Derivation Tree represent a Semantic Graph? An Answer in the Light of Meaning-Text Theory. In *Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks, IRCS Report 98–12*, pages 25–28, University of Pennsylvania, Philadelphia.

Copestake, Ann, Dan Flickinger, Ivan A. Sag and Carl Pollard. 1999. Minimal Recursion Semantics. An Introduction. Manuscript, Stanford University.

Dras, Mark, David Chiang and William Schuler. 2000. A Multi-Level TAG Approach to Dependency. In *Proceedings of the Workshop on Linguistic Theory and Grammar Implementation, ESSLLI 2000*, pages 33–46, Birmingham, August.

Frank, Anette and Josef van Genabith. 2001. GlueTag. Linear Logic based Semantics for LTAG – and what it teaches us about LFG and LTAG. In Miriam Butt and Fracy Holloway King, editors, *Proceedings of the LFG01 Conference*, Hong Kong.

Hockey, Beth Ann and Heather Mateyak. 2000. Determining Determiner Sequencing: A Syntactic Analysis for English. In Anne Abeillé and Owen Rambow, editors, *Tree Adjoining Grammars: Formalisms, Linguistic Analyses and Processing*. CSLI, pages 221–249.

Joshi, Aravind K. and Yves Schabes. 1997. Tree-Adjoning Grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*. Springer, Berlin, pages 69–123.

Joshi, Aravind K. and K. Vijay-Shanker. 1999. Compositional Semantics with Lexicalized Tree-Adjoining Grammar (LTAG): How Much Underspecification is Necessary? In H. C. Blunt and E. G. C. Thijsse, editors, *Proceedings ot the Third International Workshop on Computational Semantics (IWCS-3)*, pages 131–145, Tilburg.

Kallmeyer, Laura and Aravind K. Joshi. 1999. Factoring Predicate Argument and Scope Semantics: Underspecified Semantics with LTAG. In Paul Dekker, editor, *12th Amsterdam Colloquium. Proceedings*, pages 169–174, Amsterdam, December.

Kallmeyer, Laura and Aravind K. Joshi. 2002. Factoring Predicate Argument and Scope Semantics: Underspecified Semantics with LTAG. *Journal of Language and Computation*. To appear.

Kroch, Anthony S. 1987. Unbounded dependencies and subjacency in a Tree Adjoining Grammar. In A. Manaster-Ramer, editor, *Mathematics of Language*. John Benjamins, Amsterdam, pages 143–172.

Shieber, Stuart M. and Yves Schabes. 1990. Synchronous Tree-Adjoining Grammars. In *Proceedings of COLING*, pages 253–258.

Vijay-Shanker, K. and Aravind K. Joshi. 1988. Feature Structures Based Tree Adjoining Grammar. In *Proceedings of COLING*, pages 714–719, Budapest.

A Proof System for Tree Adjoining Grammars

Adi Palm
University of Passau

1. Introduction

Many TAG-based systems employ a particular tree adjoining grammar to generate the intended structures of the set of sentences they aim to describe. However, in most cases, the underlying set of elementary trees is more or less hand-made or maybe derived from a given tree data-base. We present a formal framework that allow to specify tree adjoining grammars by logical formulae. Based on this formalism we can check whether a given specification is TAG-consistent or whether a given TAG meets some particular properties. In addition, we sketch a method that generates a TAG from a given logical specification. As formal foundation, we employ a particular version of modal hybrid logic to specify the properties of T/D-trees. Such trees structurally combine a derived TAG-tree T and its associated derivation tree D. Finally, we sketch a labeled tableau calculus that constructs a set of tree automata representing the elementary trees of the specified TAG and a special tree automaton for the corresponding derivation trees.

In literature, we find some approaches specifying TAGs, or more generally, mildly context-sensitive grammar formalisms, that gradually vary in their underlying framework. Commonly, either starts with a logical description of recognizable sets of trees (Thatcher and Wright, 1968). However, they differ in their method of leaving the context-free paradigm. The approach mentioned in (Morawietz and Mönnich, 2001) and (Michaelis, Mönnich and Morawietz, 2000) uses a 'lifting' function that encodes a TAG into a regular tree grammar. In (Rogers, 1999) (and related works) we find a logical description of TAGs that is based on a 3-dimensional view of trees. The important issue of this approach is to combine the derived TAG-tree and its derivation tree to a single 3-dimensional structure.

Similarly, we also consider the derived TAG-tree and its derivation tree employ so-called T/D-trees. However we only associate the nodes of the derived tree with the corresponding node in the derivation tree. Consequently, all nodes of the same instance of an elementary tree refer to the same corresponding node in the derived tree. Therefore, we can specify structural properties of the derived TAG-tree and of the derivation tree at the same time. Using the links to the derivation tree, we can identify nodes in the TAG tree that belong to the same instance of some elementary tree. In contrast to the other approaches mentioned above which encode the TAG-tree into other kind of structures, we keep the original derived TAG tree as a structural unit. Consequently, we can directly access the nodes and the structural properties of the TAG tree without employing a particular projection function or any other special coding issues.

In essence, our formalism employs modal hybrid logic that combines the simplicity of modal logic and the expressivity of classical logic. The use of so-called nominals in hybrid logic offer explicit references to certain tree nodes which is (directly) possible in modal approaches. We introduce the hybrid language HL_{TAG} that specifies properties of the combined structure of derived TAG-trees and their derivation trees. Using this language we specify a number of TAG axioms which establish a notion of TAG-consistency. Further, we briefly illustrate a formalism that constructs a number of tree automata representing the underlying TAG for a given TAG-consistent HL_{TAG} formula.

2. A Hybrid Language for TAGs and their Derivations

Our formalism considers pairs of trees called T/D-trees as introduced in (Palm, 2000) where T represents a derived TAG-tree and D denotes the corresponding derivation tree. In general, a derived TAG tree $T = (t, V_t)$ is made up of a k_t-tree domain $t \subseteq \{1, \ldots, k_t\}^*$ for $k_t > 0$ and a labeling function $V_t\colon t \to Pow(\mathcal{P}_t)$ decorating tree nodes with a set of propositions of $\mathcal{P}_t$. The set of propositions $V_t(n)$ of some node n may be viewed as the label of n. Likewise, a derivation tree $D = (d, V_d)$ is made up of a k_d-tree domain $d \subseteq \{1, \ldots, k_d\}^*$ for some $k_d > 0$ and a labeling function $V_d\colon d \to Pow(\mathcal{P}_d)$. In addition, each T/D-tree includes the total linking function $\tau : t \to d$ that associates each node in the derived TAG tree T with the corresponding instance of its elementary tree in the derivation tree D.

* An extended version can be found at `http://www.phil.uni-passau.de/linguistik/palm/papers/`

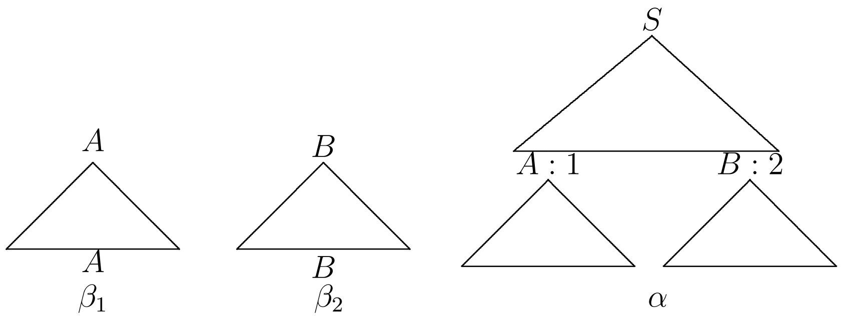


Figure 1: Sample TAG with the initial tree α and two auxiliary trees β_1 and β

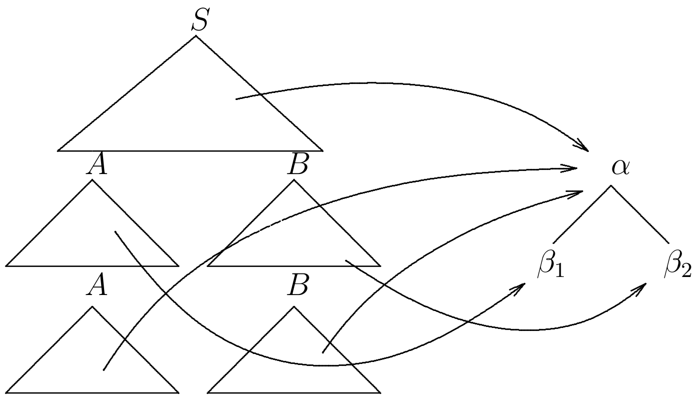


Figure 2: Resulting T/D-tree after adjoining β_1 and β_2 in α

The correspondence between either trees works straightforwardly. By the tree T we represented the derived TAG tree which results from an initial tree after adjoining and substituting auxiliary trees. By the derivation tree D we graphically represent these operations. Each children position of some node n in D represents a certain place of adjunction (or substitution) in the elementary tree represented by n. For instance, in Figure 1, the elementary tree α includes two nodes where we can adjoin another tree; we uniquely associate these nodes with the numbers 1 and 2, respectively . Now if we adjoin β_2 at the second node, this instance of β_2 in the derivation tree becomes the second child $n.2$ of the node n representing the corresponding instance α. Once we adjoined the tree β_1 at the first position, n obtains its first child $n.1$ representing this instance of β_1. Obviously, we associated each node of the corresponding instances of α, β_1 and β_2 in the derived TAG-tree with the nodes n, $n.1$ and $n.2$ in the derivation tree, respectively. Figure 2 shows the resulting T/D-tree. Note that for our formalism we assume that we can only adjoin at the inner nodes of an elementary tree, i.e. there is no adjunction at the root or at some leaves. This restriction ensures that the parent of the root and the children of the foot are nodes of the tree at which the adjunction took place.

For the formal foundation of our TAG-specification language we employ hybrid modal logic HL (Blackburn and Tzakova, 1998), (Blackburn and Tzakova, 1999), (Blackburn, 2000a; Blackburn, 2000b). This formalism extends modal (or temporal) logic with particular propositions called nominals which enable references to particular nodes (or terms) in a model. Further, there is an implemented tableau-based prover (Blackburn, Burchard and Walter, 2001) which is partially based on (Tzakova, 1999). Compared with classical logic we prefer modal and hybrid approaches since they allow more compact proofs and specifications.

In essence, we employ a modal logic on trees where the reflexive dominance relation denotes the modal reachability relation. We enhance this language by the next operator $\bigcirc_r$ referring to the r-th child of a node, by the link operator $\heartsuit$ referring to the associated node in the derivation tree. For the hybrid formulae we include the jump operator $i{:}\,\varphi$ and nominal propositions i with $i \in \mathcal{Nom}$ where $\mathcal{Nom}$ is an enumerable set of nominal symbols. Further, the language depends on the finite sets of constant propositions $\mathcal{P}_T$ and $\mathcal{P}_D$ and on the set of nominal symbols $\mathcal{N}om$. Altogether, we obtain the hybrid language *HL*$_{TAG}(\mathcal{P}_t, \mathcal{P}_d, \mathcal{N}om)$ which is defined as:

$$\varphi ::= p \,|\, i \,|\, i{:}\,\varphi \,|\, \neg\varphi \,|\, \varphi \wedge \varphi \,|\, \bigcirc_r\varphi \,|\, \Diamond\varphi \,|\, \heartsuit\varphi$$

where $1 \leq r \leq k$ (with $k = max\{k_t, k_d\}$), $p \in \mathcal{P}_t \cup \mathcal{P}_d$ denotes a propositional constant and $i \in \mathcal{N}_{T/D}$ a nominal. Further, we can define the operators $\vee$, $\rightarrow$, $\leftrightarrow$ and $\Box$ in the standard way. In addition, we define the *next*-operator referring to some child by $\bigcirc\varphi \equiv \bigcirc_1\varphi \vee \ldots \vee \bigcirc_k\varphi$ and its dual universal counterpart by $\circledast\varphi \equiv \neg\bigcirc\neg\varphi$.

For the semantics of hybrid logic, we consider, in general, Kripke-structures which are, for the case of *HL*$_{TAG}$, T/D-trees. Besides the structural information a T/D-tree associates each tree node of either tree with sets of constant propositions from $\mathcal{P}_T$ and $\mathcal{P}_D$, respectively. In addition, we require a nominal denotation function $g{:}\,\mathcal{N}om \rightarrow (t \cup d)$ evaluating the nominals. We interpret a given *HL*$_{TAG}(\mathcal{P}_t, \mathcal{P}_d, \mathcal{N}om)$ formula φ at some node $n \in \cup d$ of a tree T/D for a nominal denotation function $g{:}\,\mathcal{N}om \rightarrow t \cup d$ where g is only necessary for formulae including nominals. For the node n we assume that we know whether it is a member of t or d.

$$\begin{array}{lll}
T/D, n \models p & \text{iff} & n \in V_t(p) \cup V_d(p), \text{ for } p \in \mathcal{P}_T \cup \mathcal{P}_D \\
T/D, n \models \neg\varphi & \text{iff} & T/D, n \not\models \varphi \\
T/D, n \models \varphi \wedge \psi & \text{iff} & T/D, n \models \varphi \text{ and} \\
 & & T/D, n \models \psi \\
T/D, n \models \bigcirc_r\varphi & \text{iff} & T/D, n.r \models \varphi, 1 \leq r \leq k \text{ where } k = max\{k_t, k_d\} \\
T/D, n \models \Diamond\varphi & \text{iff} & T/D, n.a \models \varphi \text{ for some } a \in \{1, \ldots, k\}^* \text{ where } k = max\{k_t, k_d\} \\
T/D, n \models \heartsuit\varphi & \text{iff} & T/D, \tau(n) \models \varphi
\end{array}$$

A T/D-tree satisfies the formula φ if φ holds for the root of T. The link operator $\heartsuit$ is self-dual, i.e. $\heartsuit\varphi \equiv \neg\heartsuit\neg\varphi$. For the nominal expressions, we define the semantics as follows:

$$\begin{array}{lll}
T/D, n, g \models i & \text{iff} & g(n) = i \\
T/D, n, g \models i{:}\,\varphi & \text{iff} & T/D, n', g \models \varphi \\
 & & \text{and } g(n') = i
\end{array}$$

A nominal uniquely denotes a certain tree node where we do not explicitly distinguish the elements of T and D. The statement i is true if and only if the nominal i denotes the node under consideration. In contrast, in $i{:}\,\varphi$ we refer to the node denoted by i which does not depend on the node considered currently. We say a T/D-tree T/D satisfies the (nominal) formula φ at the node $n \in t \cup d$, written $T/D, n \models \varphi$, if there is a nominal denotation $g{:}\,\mathcal{N}om \rightarrow (t \cup d)$ such that $T/D, n, g \models \varphi$ is true. Similarly, T/D satisfies φ, written $T/D \models \varphi$ if there is a nominal denotation g such that $T/D, root_t, g \models \varphi$ where *root*$_t$ denotes the root of the derived TAG tree T. Hence $T/D \models \varphi \wedge \Box\varphi$ states that φ must apply to all nodes of the derived TAG tree, $T/D \models \heartsuit\varphi$ states that φ applies to the root of the derivation tree and $T/D \models \heartsuit(\varphi \wedge \Box\varphi)$ states that φ applies to all nodes of the derivation tree. Finally, a *HL*$_{TAG}$ formula φ is satisfiable if and only if there is a $T/D - tree$ and a nominal denotation g such that φ satisfies T/D by g.

Note that employing nominal propositions increases the expressivity of the former language. For instance, we can define the until-operator "until φ is true ψ must apply" or the unique existence operator $\Diamond_1\varphi$ which are not expressible in ordinary modal logic (Blackburn and Tzakova, 1999).

$$\begin{array}{rcl}
\textit{until}(\varphi, \psi) & \equiv & \Diamond(\varphi \wedge i) \wedge \Box(\Diamond i \rightarrow \psi) \\
\Diamond_1\varphi & \equiv & \Diamond(i \wedge \varphi) \wedge \Box(\varphi \rightarrow i)
\end{array}$$

In the first case we search a descendant node that satisfies φ and mark this node by the nominal i. Then each descendant node that dominates i is an intermediate node that must satisfy ψ. Similarly, we specify the unique existence operator. Again we search a descendant node that satisfies φ and employ the nominal i in order to identify this node. Now all descendants that meet φ must also meet i. In general, by introducing nominal propositions, we can extend the expressivity of the underlying formalism. As shown in (Blackburn and Seligman, 1995; Blackburn and Seligman, 1997) hybrid logic is stronger than propositional modal logic. For instance, we can formulate the

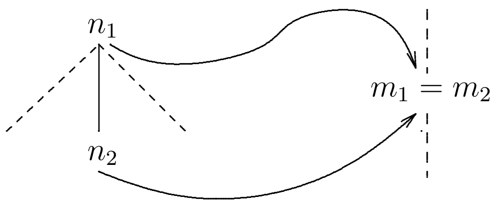


Figure 3: T/D-tree: n_1 and n_2 are internal nodes of the same elementary represented by m_1 and m_2.

until operator, or by $i \wedge \Box\neg i$ we can demand that the underlying modal reachability relation is irreflexive. Either of these properties fails to be expressible by means of propositional modal logic. On the other hand, we can specify the standard translation from hybrid logic to classical first-order logic. Therefore hybrid logic cannot be stronger than first-order logic. Moreover, as shown in (Schlingloff, 1992; Palm, 1997) the expressive power of the first-order logic for trees and the temporal logic for trees is identical. Since we can formulate the until-operator by means of hybrid logic, we obviously reach the expressivity of the temporal logic and the first logic on trees. However, the more crucial aspect of our formalism is the link operator $\heartsuit$ which allows to identify particular sets of tree nodes in the derived tree by referring to the same node in the derivation tree. Consequently, *HL*$_{TAG}$ describes first-order definable sets of derivation trees; the expressivity for the derived tree obviously depends on the properties of linking function τ. Next we discuss some restrictions on τ leading to tree adjoining grammars.

3. TAG Axioms for *HL*$_{TAG}$

Obviously, by the language *HL*$_{TAG}$ we can describe derived TAG trees and their corresponding derivation trees in an appropriate manner. However, so far it is unclear, what the necessary properties of a T/D tree are in order to describe valid TAG-trees and their derivations. Likewise, we want to know whether a given *HL*$_{TAG}$ formula φ is TAG-satisfiable, i.e. whether the set of T/D satisfying φ represents a certain TAG. The answer to either question is the set of TAG axioms for the language *HL*$_{TAG}$. Hence, a T/D-tree would be TAG generated if and only if it meets these axioms, and a *HL*$_{TAG}$-formula φ is TAG-satisfiable if and only if it is consistent with these axioms, i.e. φ and the axioms are satisfiable.

Before we turn to the axioms in detail, we examine the construction and the structural properties of a T/D-tree by a given TAG derivation. For simplification purposes we put some restriction on the kind of TAGs considered here. At first, we restrict our formalism to the adjunction operation and ignore substitution. Nevertheless it is possible to simulate a substitution by an adjunction. Further, we assume that nodes, where adjunction is possible, are marked by the special auxiliary proposition *adj* and, correspondingly, all non-adjunction nodes must fail *adj*. Moreover, an adjunction node must be an inner node of an elementary tree, i.e. it cannot be the root or some leaf. As a consequence, we obtain only TAG trees where an adjoined tree is completely surrounded by the elementary tree it was adjoined to. This leads to the following lemma:

Lemma 3.1
Let $T/D = \langle (t, v_t), (d, V_d), \tau\rangle$ *be a TAG-generated T/D-tree and* $n_1, n_2 \in t$, $m_1, m_2 \in d$ *with* $m_1 = \tau(n_1)$, $m_2 = \tau(n_2)$ *and* $n_2 = n_1.r$ *for some* $1 \leq r \leq k_t$. *Then exactly one of the following cases must be true:*

1. $m_1 = m_2$
2. $m_1.s = m_2,$ *for some* $1 \leq s \leq k_d$
3. $m_1 = m_2.s,$ *for some* $1 \leq s \leq k_d$

□

This lemma considers the properties of a pair of immediately dominating nodes n_1 and n_2 in the derived TAG tree. In the first case, both nodes belong to the same instance of an elementary tree. Therefore, they are linked to the same node in the derivation tree, as illustrated in Figure 3. The second case n_2 is the root of an adjoined tree. By the assumption we made above, the parent of a root node must be a node of the tree where the adjunction took place. Therefore n_1 must be linked with the parent of the derivation tree node that is linked with n_2, see Figure 4.

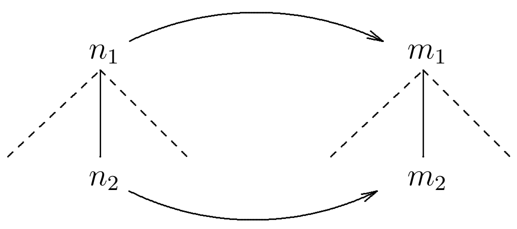


Figure 4: T/D-tree: n_2 is the root node of the elementary represented by m_2

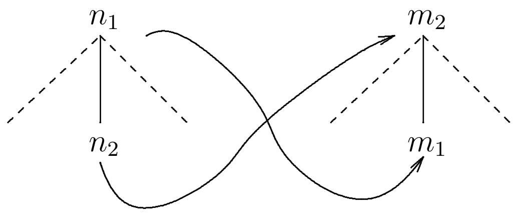


Figure 5: T/D-tree: n_1 is the foot node of the adjoined elementary tree represented by m_1

In the third case, n_1 is the foot node of an adjoined tree and, by assumption, each of its children must be a node of the tree where the adjunction took place. Consequently, m_1 must be a child of m_2, see Figure 5. Finally, due to above assumptions, no other case is possible.

Now we turn to the TAG axioms of *HL*$_{TAG}$ which ensure that a given formula describes a TAG. For the general tree axioms we refer to the similar modal tree logic as presented for example in (Blackburn, Meyer-Viol and de Rijke, 1996). However, the more interesting issue are the TAG axioms. They should ensure that *HL*$_{TAG}$ formulae only describe TAG-generated T/D-trees. For simplification, we introduce two auxiliary propositions *foot* and *root* that mark the corresponding nodes of an adjoined elementary tree. The TAG-axioms standing below assert the correct distribution of the auxiliary propositions *root* and *foot* and the correct linking between the derived and the derivation tree.

(D1)	$T_{root}\!: \mathit{root} \wedge \heartsuit D_{root}$	(associating the root nodes)
(D2)	$(i \wedge \mathit{root} \wedge \heartsuit k \wedge j\!: (\mathit{root} \wedge \heartsuit k)) \rightarrow j\!: i$	(unique root)
(D3)	$(i \wedge \mathit{foot} \wedge \heartsuit k \wedge j\!: (\mathit{foot} \wedge \heartsuit k)) \rightarrow j\!: i$	(unique foot)
(D4)	$(i \wedge \heartsuit k \wedge j\!: (\mathit{root} \wedge \heartsuit k)) \rightarrow j\!: (i \vee \Diamond i)$	(root domination)
(D5)	$\bigcirc_r \mathit{root} \leftrightarrow \bigcirc_r \heartsuit i \wedge \heartsuit \bigcirc i$	(link properties of the root)
(D6)	$\mathit{foot} \leftrightarrow \heartsuit i \wedge \circledast \heartsuit \bigcirc i$	(link properties of the root)
(D7)	$\neg \mathit{foot} \wedge \bigcirc_r \neg \mathit{root} \leftrightarrow \heartsuit i \wedge \bigcirc_r \heartsuit i$	(link properties of the inner nodes)

The first axiom asserts that in a t/d-tree T/D the underlying initial tree of the derive tree T is linked with the root of the derivation tree D. Actually, it is sufficient that (D1) only links the root node of the derived tree *root* with root of the derivation tree. The correct linking of the remaining nodes of the initial tree follows from (D7). In order to access the root nodes of either tree, we assume two special nominal propositions T_{root} and D_{root} referring to the root nodes of T and D, respectively. The next two axioms (D2) and (D3) ensure that every instance of an elementary tree occurring in T/D has a unique root and a foot. We consider a root (or foot) node with the nominal i that is linked with derivation tree with the nominal k. Then every root (or foot) node that is linked with k must be identical to i. Moreover (D4) asserts that all nodes of the same instance of an elementary tree are dominated by the root node of this instance. Finally, the axiom (D5), (D6) and (D7) ensure the local structural properties mentioned in Lemma 3.1. By (D5), the r-th child of a node meets the proposition *root* if and only if the successor relationship also applies to the derivation tree nodes corresponding to them. By (D6), a node is a foot node if and only if it is linked to the node whose parent is associated with all children of the node considered. Finally, (D7) asserts that all pairs of immediately dominating nodes share the same instance of an elementary tree, if neither the upper one is its foot nor the lower one is its root.

Obviously, due to Lemma 3.1 and the properties of a TAG derivation, every T/D-tree that is generated by a given TAG must meet these axioms. Thus, these axioms are sound with respect to tree adjoining grammars. However, the opposite direction is less obvious. It states that every T/D-tree satisfying these axioms must be generated by a tree adjoining grammar. Next we describe a tree-extraction formalism that establishes this:

1. We arbitrarily select a leaf of the derivation tree with some nominal k and, further, we consider all nodes in the derived tree that are linked with k.
2. By the axioms (D2) and (D3) there must be a unique root and foot and by (D4) all nodes the are linked with k are weakly dominated by the the root. In addition, since k has no child, no other tree was adjoined. Therefore, due to (D5), (D6) and (D7) all nodes that are linked with k define a coherent tree section in the tree t.
3. We extract the tree section as identified previously, and we replace it by a single adjunction node that is linked with the parent of k.
4. We remove k in the derivation tree
5. We repeat the steps above until a single node in the derivation tree remains.
6. Due to (D1) the remaining structure defines the underlying initial tree of that TAG-tree. The trees we extracted above are the corresponding elementary trees.

This formalism illustrates how to construct a TAG for any given t/d-tree satisfying the axioms (D1) to (D7) such that the resulting TAG generates the given T/D-tree. In general, we obtain that a T/D-tree is TAG generated, if and only if it meets these axioms at every node of the derived tree.

Moreover this formalism can be extended t in the following way. So far we know that every HL_{TAG} formula that is consistent with the TAG axioms (D1) to (D7) specifies a set of trees where each member is generated by a certain TAG. We briefly sketch a method that constructs a corresponding TAG for a given TAG-consistent HL_{TAG} formula. Therefore, we combine the above extraction formalism with an ordinary method of constructing tree models, especially tree automata, from a given modal tree description. The desired result are two linked tree-automaton for the derived tree and the derivation tree. Instead of linking to certain tree nodes of the derivation tree, we employ links to the states of the corresponding tree automaton. Then we can apply a slightly modified version of the extraction method to these tree automata. Instead of extracting trees, this modified version considers subtree automata. The final result is a (finite) set of tree automata where each of them represents a set of initial trees. In addition, the resulting automaton for the derivation tree expresses possible adjunction operations for these automata.

In order to construct these automata, we can employ, for instance, well-known labeled tableau methods as described in (Goré, 1999), which were adopted for our purposes. The overall goal is to construct a set of simple tree automata representing the initial trees of the TAG described and another special tree automaton representing the corresponding derivation trees. Using labeled formulae in the tableau, we can indicate the node and the tree the formulae considered must apply to. A crucial part of the tableau system concerns the construction of the r^{th} successor of some node by the formula $\bigcirc_r \varphi$:

$$(\bigcirc_r)\ \frac{\alpha, \sigma :: \bigcirc_r \varphi}{\begin{array}{l|l|l} \alpha, \sigma :: \heartsuit i & \alpha, \sigma :: \heartsuit \bigcirc_s i & \alpha, \sigma :: \heartsuit i \\ \alpha, \sigma.r :: \varphi & \alpha.s, \sigma.r :: \varphi & \alpha', \sigma.r :: \varphi \\ \alpha, \sigma.r :: \heartsuit i & \alpha.s, \sigma.r :: \heartsuit i & \alpha', \sigma.r :: \heartsuit \bigcirc_s i \\ \alpha, \sigma :: \neg \mathit{foot} & \alpha, \sigma.r :: \mathit{root} & \alpha, \sigma :: \mathit{foot} \\ \alpha, \sigma.r :: \neg \mathit{root} & & \end{array}}$$

where $\alpha = \alpha'.s$. The premise of this rule considers the node σ of the derived tree that is associated with the node α of the derivation tree and $\bigcirc_r \varphi$ must hold at σ. Obviously this rule states again the situation described in Lemma 3.1. In the first case (see Figure 6), the new successor node $\sigma.r$ belongs to the same elementary tree as its parent, so both are associated with the same node α in the derivation tree. For the second case (see Figure 7), this rule generates a root node of an adjoined tree, so the new successor $\sigma.r$ is associated with $\alpha.s$, i.e. the corresponding successor. Note that we also construct the s-successor of α. Finally, in the third case (see Figure 8) σ denotes a foot node, so the successor $\sigma.r$ must be associated with the parent of α.

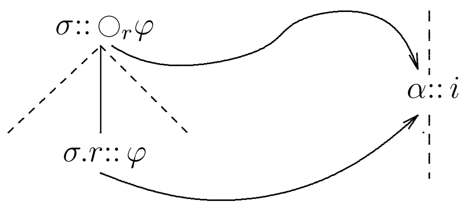


Figure 6: tableau rule for $\bigcirc_r\varphi$: inner nodes

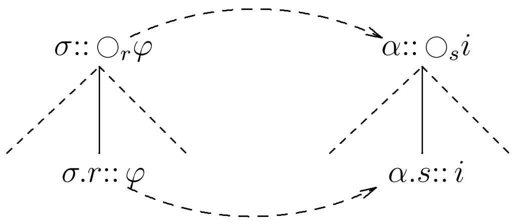


Figure 7: tableau rule for $\bigcirc_r\varphi$: root node

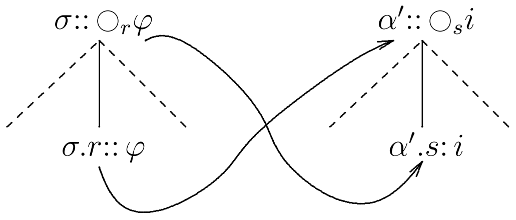


Figure 8: tableau rule for $\bigcirc_r\varphi$: foot node

Most other rules of this labeled tableau calculus are more or less straightforward or result from the requirements of the TAG axioms. To obtain tree automata from the tableau, we define classes of equivalent labels including the same set of formulae:

$$\alpha, \sigma \approx \alpha', \sigma' \text{ iff } \{\varphi \mid \alpha, \sigma :: \varphi\} = \{\varphi \mid \alpha', \sigma' :: \varphi\}$$

Obviously, the number of such equivalence classes is finite, since the number of occurring subformulae is finite as well. Every class defines some state of a tree automaton, and the immediate dominance relation leads to the state transition relation. Accordingly, we can extract the tree automaton for each elementary tree α by selecting the states including the label α. Then a gap in the immediate dominance relation indicates an adjunction node which must be handled correspondingly. Eventually, we also obtain a tree automaton for the derivation tree.

4. Conclusions

Specifying sets of trees beyond context-free grammars requires additional structural information. For the TAG-approach presented here, we combined the derived tree and the derivation tree leading to so-called T/D-trees. While the derived tree actually is the object of consideration, the derivation tree serves as a kind of storage for the required additional information. The linking function from the derived tree to the derivation tree combines the TAG tree nodes sharing the same instance of an elementary tree. Therefore we can access the underlying elementary trees and, consequently, we can decide whether a tree is TAG-generated.

For the formal description we employed hybrid logic which provides sufficient expressivity to specify TAG axioms and further constraints on TAGs. On the other hand, the modal foundation of hybrid logic, offers simple formulations that are easier to handle than classical logic. The result is a simple proof system for TAGs, which can be used to verify certain formal properties for a given TAG or, as we have sketched briefly, to construct a TAG from a given formal specification

An open question, so far, is the expressive power of our formalism. Obviously, it is possible to specify sets of T/D-trees that fall out of the scope of TAGs. For instance, it is possible specify complete binary trees. Nevertheless, such a specification would violate the TAG axioms.

References

Blackburn, P. 2000a. Internalizing Labelled Deduction. *Journal of Logic and Computation*, 10(1):137–168.

Blackburn, P. 2000b. Representation, Reasoning, and Relational Structures: a Hybrid Logic Manifesto. *Logic Journal of the IGPL*, 8(3):339–365.

Blackburn, P., A. Burchard and S. Walter. 2001. Hydra: a tableaux-based prover for basic hybrid logic. In C. Areces and M. de Rijke, editors, *Proceedings of Methods for Modalities 2*, Amsterdam, The Netherlands, November.

Blackburn, P., W. Meyer-Viol and M. de Rijke. 1996. A Proof System for Finite Trees. In H. Kleine Büning, editor, *Computer Science Logic*, LNCS, vol. 1092. Springer Verlag, pages 86–105.

Blackburn, P. and J. Seligman. 1995. Hybrid Languages. *Logic Journal of Logic, Language and Information*, 4(1):251–272.

Blackburn, P. and J. Seligman. 1997. What are Hybrid Languages. In M. Kracht, H. Wansing and M. Zakharyshev, editors, *Advances in Modal Logic '96*. CSLI Publications, Stanford University.

Blackburn, P. and M. Tzakova. 1998. Hybrid Completeness. *Logic Journal of the IGPL*, 6(4):625–650.

Blackburn, P. and M. Tzakova. 1999. Hybrid Languages and Temporal Logic. *Logic Journal of the IGPL*, 7(1):27–54.

Goré, R. 1999. Tableau Methods for Modal and Temporal Logic. In M. D'Augustino, D. Gabbay, R. Hähnle and J. Posegga, editors, *Handbook of Tableau Methods*. Kluwer, Dordrecht, pages 297–396.

Michaelis, J., U. Mönnich and F. Morawietz. 2000. Derivational Minimalism in Two Regular and Logical Steps. In *Proceedings of TAG+5, Paris*, pages 163–170.

Morawietz, F. and U. Mönnich. 2001. A Model-Theoretic Description of Tree Adjoining Grammars. In *Formal Grammar Conference/MOL Conference, Helsinki*, Electronical Notes in Theoretical Computer Science, vol. 53. Elsevier Science.

Palm, A. 1997. *Transforming Tree Constraints into Rules of Grammars*. DISKI, volume 173. St. Augustin: infix-Verlag.

Palm, A. 2000. Structure Sharing in Tree-Adjoining Grammars. In *Proceedings of TAG+5*.

Rogers, J. 1999. Generalized Tree-Adjoining Grammars. In *Proceedings of 6th Meeting on Mathemathics of Language (MOL6)*.

Schlingloff, B.-H. 1992. Expressive Completeness of Temporal Logic for Trees. *Journal of Applied Non-Classical Logics*, 2:157–180.

Thatcher, J.W. and J.B. Wright. 1968. Generalized Finite Automata Theory with an Application to Decision Porblems of Second-Order Logic. *Mathematical System Theory*, 2:57–81.

Tzakova, M. 1999. Tableaux calculi for hybrid logics. In N. Murray, editor, *Conference on Tableaux Calculi and Related Methods (TABLEAUX), Saratoga Springs, USA*, LNAI, vol. 1617. Springer Verlag, pages 278–292.

Tree-Adjoining Grammars as Abstract Categorial Grammars

Philippe de Groote
LORIA UMR n° 7503 – INRIA
Campus Scientifique, B.P. 239
54506 Vandœuvre lès Nancy Cedex – France
e-mail: Philippe.de.Groote@loria.fr

1. Introduction

We recently introduced abstract categorial grammars (ACGs) (de Groote, 2001) as a new categorial formalism based on Girard linear logic (Girard, 1987). This formalism, which derives from current type-logical grammars (Carpenter, 1996; Moortgat, 1997; Morrill, 1994; Oehrle, 1994), offers some novel features:

- Any ACG generates two languages, an abstract language and an object language. The abstract language may be thought as a set of abstract grammatical structures, and of the object language as the set of concrete forms generated from these abstract structures. Consequently, one has a direct control on the parse structures of the grammar.
- The langages generated by the ACGs are sets of linear λ-terms. This may be seen as a generalization of both string-langages and tree-langages.
- ACGs are based on a small set of mathematical primitives that combine via simple composition rules. Consequently, the ACG framework is rather flexible.

Abstract categorial grammars are not intended as yet another grammatical formalism that would compete with other established formalisms. It should rather be seen as the kernel of a grammatical framework — in the spirit of (Ranta, 2002) — in which other existing grammatical models may be encoded. This paper illustrates this fact by showing how tree-adjoining grammars (Joshi and Schabes, 1997) may be embedded in abstract categorial grammars.

This embedding exemplifies several features of the ACG framework:

- The fact that the basic objects manipulated by an ACG are λ-terms allows higher-order operations to be defined. Typically, tree-adjunction is such a higher-order operation (Abrusci, Fouqueré and Vauzeilles, 1999; Joshi and Kulick, 1997; Mönnich, 1997).
- The flexibility of the framework allows the embedding to be defined in two stages. A first ACG allows the tree langage of a given TAG to be generated. The abstract language of this first ACG corresponds to the derivation trees of the TAG. Then, a second ACG allows the corresponding string language to be extracted. The abstract language of this second ACG corresponds to the object language of the first one.

2. Abstract Categorial Grammars

This section defines our notion of an abstract categorial grammar. We first introduce the notions of *linear implicative types*, *higher-order linear signature*, *linear λ-terms* built upon a higher-order linear signature, and *lexicon*.

Let A be a set of atomic types. The set $\mathscr{T}(A)$ of *linear implicative types* built upon A is inductively defined as follows:

1. if $a \in A$, then $a \in \mathscr{T}(A)$;
2. if $\alpha, \beta \in \mathscr{T}(A)$, then $(\alpha \multimap \beta) \in \mathscr{T}(A)$.

A *higher-order linear signature* consists of a triple $\Sigma = \langle A, C, \tau \rangle$, where:

1. A is a finite set of atomic types;

2. C is a finite set of constants;

3. $\tau : C \to \mathscr{T}(A)$ is a function that assigns to each constant in C a linear implicative type in $\mathscr{T}(A)$.

Let X be a infinite countable set of λ-variables. The set $\Lambda(\Sigma)$ of *linear λ-terms* built upon a higher-order linear signature $\Sigma = \langle A, C, \tau \rangle$ is inductively defined as follows:

1. if $c \in C$, then $c \in \Lambda(\Sigma)$;

2. if $x \in X$, then $x \in \Lambda(\Sigma)$;

3. if $x \in X$, $t \in \Lambda(\Sigma)$, and x occurs free in t exactly once, then $(\lambda x.\, t) \in \Lambda(\Sigma)$;

4. if $t, u \in \Lambda(\Sigma)$, and the sets of free variables of t and u are disjoint, then $(t\, u) \in \Lambda(\Sigma)$.

$\Lambda(\Sigma)$ is provided with the usual notion of capture avoiding substitution, α-conversion, and β-reduction (Barendregt, 1984).

Given a higher-order linear signature $\Sigma = \langle A, C, \tau \rangle$, each linear λ-term in $\Lambda(\Sigma)$ may be assigned a linear implicative type in $\mathscr{T}(A)$. This type assignment obeys an inference system whose judgements are sequents of the following form:

$$\Gamma \vdash_{\Sigma} t : \alpha$$

where:

1. Γ is a finite set of λ-variable typing declarations of the form '$x : \beta$' (with $x \in X$ and $\beta \in \mathscr{T}(A)$), such that any λ-variable is declared at most once;

2. $t \in \Lambda(\Sigma)$;

3. $\alpha \in \mathscr{T}(A)$.

The axioms and inference rules are the following:

$$\vdash_{\Sigma} c : \tau(c) \quad \text{(cons)}$$

$$x : \alpha \vdash_{\Sigma} x : \alpha \quad \text{(var)}$$

$$\frac{\Gamma, x : \alpha \vdash_{\Sigma} t : \beta}{\Gamma \vdash_{\Sigma} (\lambda x.\, t) : (\alpha \multimap \beta)} \quad \text{(abs)}$$

$$\frac{\Gamma \vdash_{\Sigma} t : (\alpha \multimap \beta) \quad \Delta \vdash_{\Sigma} u : \alpha}{\Gamma, \Delta \vdash_{\Sigma} (t\, u) : \beta} \quad \text{(app)}$$

Given two higher-order linear signatures $\Sigma_1 = \langle A_1, C_1, \tau_1 \rangle$ and $\Sigma_2 = \langle A_2, C_2, \tau_2 \rangle$, a lexicon $\mathscr{L} : \Sigma_1 \to \Sigma_2$ is a realization of Σ_1 into Σ_2, i.e., an interpretation of the atomic types of Σ_1 as types built upon A_2 together with an interpretation of the constants of Σ_1 as linear λ-terms built upon Σ_2. These two interpretations must be such that their homomorphic extensions commute with the typing relations. More formally, a *lexicon* $\mathscr{L}$ from Σ_1 to Σ_2 is defined to be a pair $\mathscr{L} = \langle F, G \rangle$ such that:

1. $F : A_1 \to \mathscr{T}(A_2)$ is a function that interprets the atomic types of Σ_1 as linear implicative types built upon A_2;

2. $G : C_1 \to \Lambda(\Sigma_2)$ is a function that interprets the constants of Σ_1 as linear λ-terms built upon Σ_2;

3. the interpretation functions are compatible with the typing relation, *i.e.*, for any $c \in C_1$, the following typing judgement is derivable:
$$\vdash_{\Sigma_2} G(c) : \hat{F}(\tau_1(c)),$$
where $\hat{F}$ is the unique homomorphic extension of F.

We are now in a position of defining the notion of abstract categorial grammar. An *abstract categorial grammar* is a quadruple $\mathscr{G} = \langle \Sigma_1, \Sigma_2, \mathscr{L}, s \rangle$ where:

1. Σ_1 and Σ_2 are two higher-order linear signatures; they are called the *abstract vovabulary* and the *object vovabulary*, respectively ;

2. $\mathscr{L} : \Sigma_1 \to \Sigma_2$ is a lexicon from the abstract vovabulary to the object vovabulary;

3. s is an atomic type of the abstract vocabulary; it is called the *distinguished type* of the grammar.

The *abstract language* generated by $\mathscr{G}$ ($\mathcal{A}(\mathscr{G})$) is defined as follows:

$$\mathcal{A}(\mathscr{G}) = \{t \in \Lambda(\Sigma_1) \mid\ \vdash_{\Sigma_1} t\colon s \text{ is derivable}\}$$

In words, the abstract language generated by $\mathscr{G}$ is the set of closed linear λ-terms, built upon the abstract vocabulary Σ_1, whose type is the distinguished type s. On the other hand, the *object language* generated by $\mathscr{G}$ ($\mathcal{O}(\mathscr{G})$) is defined to be the image of the abstract language by the term homomorphism induced by the lexicon $\mathscr{L}$:

$$\mathcal{O}(\mathscr{G}) = \{t \in \Lambda(\Sigma_2) \mid \exists u \in \mathcal{A}(\mathscr{G}).\ t = \mathscr{L}(u)\}$$

3. Representing Tree-Adjoining Grammars

In this section, we explain how to construct an abstract categorial grammar that generates the same tree langage as a given tree-adjoining grammar.

Let $G = \langle \Sigma, N, I, A, S\rangle$ be a tree-adjoining grammar, where Σ, N, I, A, and S are the set of terminal symbols, the set of non-terminal symbols, the set of initial trees, the set of auxiliary tree, and the distinguished non-terminal symbol, respectively. We associate to G an ACG $\mathscr{G}^G = \langle \Sigma_1^G, \Sigma_2^G, \mathscr{L}^G, s^G\rangle$ as follows.

The set of atomic types of Σ_1^G is made of two copies of the set of non-terminal symbols. Given $\alpha \in N$, we write α_S and α_A for the two corresponding atomic types. Then, we associate a constant

$$c_T : \gamma_{1A} \multimap \cdots \gamma_{mA} \multimap \beta_{1S} \multimap \cdots \beta_{nS} \multimap \alpha_S$$

to each initial tree T whose root node is labelled by α, whose substitution nodes are labeled by $\beta_1, \ldots, \beta_n$, and whose interior nodes are labeled by $\gamma_1, \ldots, \gamma_m$. Similarly, we associate a constant

$$c_{T'} : \gamma_{1A} \multimap \cdots \gamma_{mA} \multimap \beta_{1S} \multimap \cdots \beta_{nS} \multimap \alpha_A \multimap \alpha_A$$

to each auxiliary tree T' whose root node is labelled by α, whose substitution nodes are labeled by $\beta_1, \ldots, \beta_n$, and whose interior nodes are labeled by $\gamma_1, \ldots, \gamma_m$. Finally, we also associate to each non-terminal symnbol $\alpha \in N$, a constant I_α of type α_A. This concludes the specification of the abstract vocabulary.

The object vocabulary Σ_2^G allows labelled trees to be represented. Its set of atomic types contains only one element : τ (for *tree*). Then, its set of constants consists in:

1. constants of type τ corresponding to the terminal symbols of G;

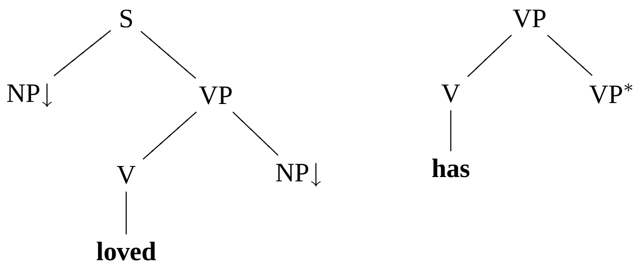
2. for each non-terminal symbol α, constants

$$\alpha_i : \underbrace{\tau \multimap \cdots \tau}_{i \text{ times}} \multimap \tau$$

for $1 \le i \le k$, where k is the maximal branching of the interior nodes labelled with α that occur in the initial and auxiliary trees of G.

Clearly, the terms of type τ that can be built by means of the above set of constants correspond to trees whose frontier nodes are terminal symbols and whose interior nodes are labelled with non-terminal symbols.

It remains to define the lexicon $\mathscr{L}^G$. The rough idea is to represent the initial trees as trees (i.e., terms of type τ) and the auxiliary trees as functions over trees (i.e., terms of type $\tau \multimap \tau$). Consequently, for each $\alpha \in N$, we let $\mathscr{L}^G(\alpha_S) = \tau$ and $\mathscr{L}^G(\alpha_A) = \tau \multimap \tau$. Accordingly, the susbstitution nodes will be represented as first-order λ-variables of type τ , and the adjunction nodes as second-order λ-variables of type $\tau \multimap \tau$. The object representation of the elementary trees is then straightforward. Consider, for instance, the following initial tree and auxiliary tree:



According to our construction, the two abstract constants corresponding to these trees have the following types:

$$C_{\text{loved}} : \mathrm{S}_A \multimap \mathrm{VP}_A \multimap \mathrm{V}_A \multimap \mathrm{NP}_S \multimap \mathrm{NP}_S \multimap \mathrm{S}_S \quad \text{and} \quad C_{\text{has}} : \mathrm{VP}_A \multimap \mathrm{V}_A \multimap \mathrm{VP}_A \multimap \mathrm{VP}_A$$

Then, the realization of these two constants is as follows:

$$\begin{array}{rcl} \mathscr{L}^G(C_{\text{loved}}) & = & \lambda F.\, \lambda G.\, \lambda H.\, \lambda x.\, \lambda y.\, F\,(\mathrm{S}_2\, x\, (G\,(\mathrm{VP}_2\,(H\,(\mathrm{V}_1\,\textbf{loved}))\, y))) \\ \mathscr{L}^G(C_{\text{has}}) & = & \lambda F.\, \lambda G.\, \lambda H.\, \lambda x.\, F\,(\mathrm{VP}_2\,(G\,(\mathrm{V}_1\,\textbf{has}))\,(H\,x)) \end{array}$$

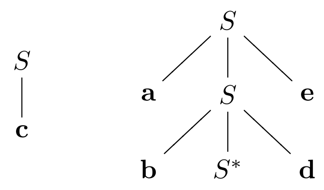
In order to derive actual trees, the second-order variables should eventually disappear. The abstract constants I_α have been introduced to this end. Consequently they are realized by the identity function, i.e., $\mathscr{L}^G(I_\alpha) = \lambda x.\, x$.

Finally, the distinguished type of $\mathscr{G}^G$ is defined to be S_S. This completes the definition of the ACG $\mathscr{G}^G$ associated to a TAG G. Then, the following proposition may be easily established.

PROPOSITION *Let G be a TAG. The tree-language generated by G is isomorphic to the object language of the ACG $\mathscr{G}^G$ associated to G.* □

4. Example

Consider the TAG with the following initial tree and auxiliary tree:



It generates a non context-free language whose intersection with the regular language $a^*b^*c\,d^*e^*$ is $a^nb^nc\,d^ne^n$. According to the construction of Section 3, this TAG may be represented by the ACG, $\mathscr{G} = \langle \Sigma_1, \Sigma_2, \mathscr{L}, S\rangle$, where:

$$\begin{array}{l} \Sigma_1 = \langle\ \{S_S, S_A\}, \{c_i, c_a, I\}, \\ \qquad \{c_i \mapsto (S_A \multimap S_S), \\ \qquad c_a \mapsto (S_A \multimap (S_A \multimap (S_A \multimap S_A))), \\ \qquad I \mapsto S_A\}\ \rangle \end{array}$$

$$\begin{array}{l} \Sigma_2 = \langle\ \{\tau\}, \{\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{e}, S_1, S_3\}, \\ \qquad \{\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{e} \mapsto \tau, \\ \qquad S_1 \mapsto (\tau \multimap \tau), \\ \qquad S_3 \mapsto (\tau \multimap (\tau \multimap (\tau \multimap \tau)))\}\ \rangle \end{array}$$

$$\begin{array}{l} \mathscr{L} = \langle\ \{S_S \mapsto \tau, \\ \qquad S_A \mapsto (\tau \multimap \tau)\}, \\ \qquad \{c_i \mapsto \lambda f.\, f\,(S_1\,\mathbf{c}), \\ \qquad c_a \mapsto \lambda f.\, \lambda g.\, \lambda h.\, \lambda x.\, f\,(S_3\,\mathbf{a}\,(g\,(S_3\,\mathbf{b}\,(h\,x)\,\mathbf{d}))\,\mathbf{e}), \\ \qquad I \mapsto \lambda x.\, x\}\ \rangle \end{array}$$

5. Extracting the string languages

There is a canonical way of representing strings as linear λ-terms. It consists of encoding a string of symbols as a composition of functions. Consider an arbitrary atomic type σ, and define the type '*string*' to be $(\sigma \multimap \sigma)$. Then, a string such as '$abbac$' may be represented by the linear λ-term:

$$\lambda x.\, a\,(b\,(b\,(a\,(c\,x)))),$$

where the atomic strings 'a', 'b', and 'c' are declared to be constants of type $(\sigma \multimap \sigma)$. In this setting, the empty word is represented by the identity function:

$$\epsilon \triangleq \lambda x.\, x$$

and concatenation is defined to be functional composition:

$$\alpha + \beta \triangleq \lambda\alpha.\, \lambda\beta.\, \lambda x.\, \alpha\,(\beta\,x),$$

which is indeed an associative operator that admits the identity function as a unit.

This allows a second ACG, $\mathscr{G}'^G$, to be defined. Its abstract vocabulary is the object vocabulary Σ_2^G of $\mathscr{G}^G$. Its object vocabulary allows string of terminal symbols to be represented. Its lexicon interprets each constant of type τ as an atomic string, and each constant α_i as a concatenation operator. This second ACG, $\mathscr{G}'^G$, extracts the yields of the trees. Then, by composing $\mathscr{G}^G$ with $\mathscr{G}'^G$, one obtains an ACG which generates the same string-language as G.

Let us continue the example of Section 4. The second ACG, $\mathscr{G}' = \langle \Sigma_1', \Sigma_2', \mathscr{L}', S' \rangle$, is defined as follows:

$$\Sigma_1' = \Sigma_2$$

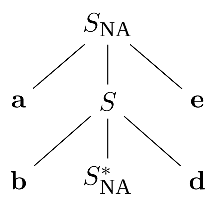
$$\begin{aligned}\Sigma_2' = \langle\ & \{\sigma\}, \{a, b, c, d, e\},\\ & \{a, b, c, d, e \mapsto (\sigma \multimap \sigma)\}\ \rangle\end{aligned}$$

$$\begin{aligned}\mathscr{L}' = \langle\ & \{\tau \mapsto (\sigma \multimap \sigma)\},\\ & \{\mathbf{a} \mapsto \lambda x.\, a\,x,\\ & \ \mathbf{b} \mapsto \lambda x.\, b\,x,\\ & \ \mathbf{c} \mapsto \lambda x.\, c\,x,\\ & \ \mathbf{d} \mapsto \lambda x.\, d\,x,\\ & \ \mathbf{e} \mapsto \lambda x.\, e\,x,\\ & \ S_1 \mapsto \lambda f.\, \lambda x.\, f\,x,\\ & \ S_3 \mapsto \lambda f.\, \lambda g.\, \lambda h.\, f\,(g\,(h\,x))\}\ \rangle\end{aligned}$$

6. Expressing Adjoining constraints

Adjunction, which is enabled by second-order variables at the object level, is explicitly controlled at the abstract level by means of types. This typing discipline may be easily refined in order to express adjoining constraints such as selective, null, or obligatory adjunction.

Consider again the TAG given in Section 4. By adding the following null adjunction constraints on its auxiliary tree:



one obtains a grammar that generates exactly the non context-free language $a^n b^n c\, d^n e^n$. These constraints may be expressed in a simple and natural way. It suffices to exclude the constrained nodes from the arguments of the λ-term corresponding to the auxiliary tree. This gives the following modified ACG:

$$\begin{aligned}\Sigma_1 = \langle\ & \{S_S, S_A\}, \{c_i, c_a, I\},\\ & \{c_i \mapsto (S_A \multimap S_S),\\ & \ c_a \mapsto (S_A \multimap S_A),\\ & \ I \mapsto S_A\}\ \rangle\end{aligned}$$

$$\begin{aligned}\Sigma_2 = \langle\ & \{\tau\}, \{\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{e}, S_1, S_3\},\\ & \{\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{e} \mapsto \tau,\\ & \ S_1 \mapsto (\tau \multimap \tau),\\ & \ S_3 \mapsto (\tau \multimap (\tau \multimap (\tau \multimap \tau)))\}\ \rangle\end{aligned}$$

$$\begin{aligned}\mathscr{L} = \langle\ & \{S_S \mapsto \tau,\\ & \ S_A \mapsto (\tau \multimap \tau)\},\\ & \{c_i \mapsto \lambda f.\, f\,(S_1\,\mathbf{c}),\\ & \ c_a \mapsto \lambda f.\, \lambda x.\, S_3\,\mathbf{a}\,(f\,(S_3\,\mathbf{b}\,x\,\mathbf{d}))\,\mathbf{e},\\ & \ I \mapsto \lambda x.\, x\}\ \rangle\end{aligned}$$

The other kinds of adjunction constraints may be expressed in a similar way.

References

Abrusci, M., C. Fouqueré and J. Vauzeilles. 1999. Tree-adjoining grammars in a fragment of the Lambek calculus. *Computational Linguistics*, 25(2):209–236.

Barendregt, H.P. 1984. *The lambda calculus, its syntax and semantics*. revised edition. North-Holland.

Carpenter, B. 1996. *Type-Logical Semantics*. Cambridge, Massachussetts and London England: MIT Press.

de Groote, Ph. 2001. Towards Abstract Categorial Grammars. In *Association for Computational Linguistics, 39th Annual Meeting and 10th Conference of the European Chapter, Proceedings of the Conference*, pages 148–155.

Girard, J.-Y. 1987. Linear Logic. *Theoretical Computer Science*, 50:1–102.

Joshi, A. K. and S. Kulick. 1997. Partial Proof Trees as Building Blocks for a Categorial Grammar. *Linguistic & Philosophy*, 20:637–667.

Joshi, A. K. and Y. Schabes. 1997. Tree-adjoining grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of formal languages*, volume 3. Springer, chapter 2.

Mönnich, U. 1997. Adjunction as substitution. In G.-J. Kruijff, G. Morrill and D. Oehrle, editors, *Formal Grammar*, pages 169–178. FoLLI.

Moortgat, M. 1997. Categorial Type Logic. In J. van Benthem and A. ter Meulen, editors, *Handbook of Logic and Language*. Elsevier, chapter 2.

Morrill, G. 1994. *Type Logical Grammar: Categorial Logic of Signs*. Dordrecht: Kluwer Academic Publishers.

Oehrle, R. T. 1994. Term-labeled categorial type systems. *Linguistic & Philosophy*, 17:633–678.

Ranta, A. 2002. Grammatical Framework, a type-theoretical grammar formalism. Working paper, submitted for publication.

Residuation, Structural Rules and Context Freeness

Gerhard Jäger
University of Potsdam/ZAS Berlin

1. Introduction

This paper deals with the issue of the generative capacity of a certain version of type logical categorial grammar. Originally categorial grammar in the modern sense was invented in three varieties in the late fifties and early sixties. Bar-Hillel (1953) developed applicative categorial grammar, a bidirectional version of older type theoretic systems tracing back to the work of Polish logicians in the early twentieth century. Few years later, Lambek (1958) proposed his calculus of syntactic types that is known as the (associative) "Lambek calculus" nowadays (abbreviated as "Ł"). A short time later he published a non-associative version of this calculus in (Lambek, 1961), which is known as the "Non-associative Lambek calculus" **NL**. These two systems are the first instances of "Type Logical Grammars", i.e. the deductive machinery of the grammar formalism is a substructural type logical calculus.

The issue of the position of these grammar formalisms within the Chomsky hierarchy has intrigued mathematical linguists from the beginning. It was settled first for applicative categorial grammar by Bar-Hillel, Gaifman and Shamir (1960). They establish the weak equivalence of this version of categorial grammars with the context free languages.

For the type logical categorial grammars, this problem was settled fairly late. Buszkowski (1986) established the the product free fragment of the non-associative Lambek calculus defines exactly the context free languages, and Kandulski (1988) showed that this result carries over to full **NL**. Finally, (Pentus, 1993) gives a proof that the associative Lambek calculus Ł is weakly equivalent to the context free grammars as well.

It was already conjectured in (Chomsky, 1957) that context free grammars are not expressive enough to give an adequate description of the grammar of natural languages. This was formally proved in (Shieber, 1985). So it seems that none of the tree basic varieties of categorial grammar provides an adequate grammar formalism for linguistics. This shortcoming motivated to move to multimodal systems, i.e. type logical grammars that employ several families of connectives and certain interactions between them. This idea was first explored in (Moortgat, 1988) and (Morrill, 1990), and systematized in (Moortgat, 1996). There it is assumed that it is sufficient to use a finite ensemble of residuation connectives plus some interaction postulates between them to come to terms with the empirical facts of natural language. This idea is confirmed but trivialized by (Carpenter, 1999), where it is proved that multimodal type logical grammars are equivalent in generative power to Turing machines.

Considering Carpenter's proof, it seems intuitively obvious that the unrestricted use of interaction postulates is responsible for this increase in generative capacity, while the notion of multimodality as such has no such effect. This is partially confirmed by Jäger (2001). This article gives a proof that enriching the associative Lambek calculus with pairs of unary residuation connectives without interaction postulates does not increase generative power; the resulting system still describes exactly the context free languages. This is established by a straightforward extension of Pentus' construction.

For a non-associative base logic, several important results in this connection have been obtained by Maciej Kandulski (see (Kandulski, 1995; Kandulski, 2002)). He generalizes his result from (Kandulski, 1988) to the commutative version of the non-associative Lambek calculus, and to multimodal logics comprising an arbitrary number of different families of residuation connectives of any arity, but without structural rules.

Kandulski's results are all based on an axiomatization of the type logics underlying the grammars in question and a process of proof normalization within this axiomatic calculus. The present paper presents alternative proofs of these theorems that are based on the Gentzen style sequent presentation of the logics involved. This new proof strategy leads to generalizations of Kandulski's results in two respects: Any combination of residuated connectives with any of the structural rules Permutation, Contraction and Expansion lead to type logical grammars that recognized only context free languages, and this also holds if we admit non-atomic designated types.

The structure of the paper is as follows. We first focus on the simplest multimodal extension of the non-associative Lambek calculus, namely the calculus **NL**$\diamond$ from (Moortgat, 1996). In section 2 we introduce the necessary technical notions, and section 3 presents the proof that grammars based on **NL**$\diamond$ recognized exactly the context free languages. In section 4 we generalize these results to residuation modalities of arbitrary arity, and to calculi using the structural rules Permutation, Contraction or Expansion. Section 5 summarizes the findings and points to desiderata for further research.

2. Technical Preliminaries

2.1. NL

The non-associative Lambek calculus **NL** is the weakest substructural logic. Its logical vocabulary consists of one binary product $\bullet$ and its left and right residuation, the two directed implications $\backslash$ and $/$. More formally, the types of **NL** are defined recursively over some finite alphabet of atomic types $\mathcal{A}$ as

$$\mathcal{F} =:: \mathcal{A}|\mathcal{F}\backslash\mathcal{F}|\mathcal{F} \bullet \mathcal{F}|\mathcal{F}/\mathcal{F}$$

The calculus itself can be defined as a set of arrows, i.e. objects of the form $A \to B$, where A and B are types of **NL**. In its axiomatic presentation, the calculus comprises the identity axiom and the Cut rule. We use the upper case Latin letter $A, B, C, ...$ as meta-variables over types.

$$\frac{}{A \to A}\,id$$

$$\frac{A \to B \qquad B \to C}{A \to C}\,Cut$$

The behavior of the logical connectives is governed by the residuation laws

$$B \to A\backslash C \text{ iff } A \bullet B \to C \text{ iff } A \to C/B$$

Lambek also gives a Gentzen style sequent presentation of **NL**. A sequent consists of an antecedent and a succedent, where the antecedent is a binary tree over types and the succedent a single type. We write trees as terms formed from types and the binary operation $\circ$. Formally, the set of **NL**-trees is thus given by

$$\mathcal{T} =:: \mathcal{F}|(\mathcal{T} \circ \mathcal{T})$$

The upper case Latin letters $X, Y, Z, ...$ are meta-variables over trees of types. $X[Y]$ is a tree X containing a sub-tree Y, and $X[Z]$ is the result of replacing the sub-tree Y in X by Z.

$$\frac{}{A \Rightarrow A}\,id \qquad \frac{X \Rightarrow A \qquad Y[A] \Rightarrow B}{Y[X] \Rightarrow B}\,Cut$$

$$\frac{X[A \circ B] \Rightarrow C}{X[A \bullet B] \Rightarrow C}\,\bullet L \qquad \frac{X \Rightarrow A \qquad Y \Rightarrow B}{X \circ Y \Rightarrow A \bullet B}\,\bullet R$$

$$\frac{X \Rightarrow A \qquad Y[B] \Rightarrow C}{Y[B/A \circ X] \Rightarrow C}\,/L \qquad \frac{X \circ A \Rightarrow B}{X \Rightarrow A/B}\,/R$$

$$\frac{X \Rightarrow A \qquad Y[B] \Rightarrow C}{Y[X \circ A\backslash B] \Rightarrow C}\,\backslash L \qquad \frac{A \circ X \Rightarrow B}{X \Rightarrow A\backslash B}\,\backslash R$$

Figure 1: Sequent presentation of the non-associative Lambek calculus **NL**

We write "**NL** $\vdash X \Rightarrow A$" iff the sequent $X \Rightarrow A$ is derivable in the sequent calculus. The axiomatic and the sequent presentation of **NL** are equivalent in the sense that every derivable arrow is also a derivable sequent, and replacing all occurrences of $\circ$ in a derivable sequent by the product $\bullet$ yields a derivable arrow.

It is easy to see that all rules in the sequent calculus except Cut have the subformula property. Lambek proved Cut elimination for the sequent calculus, which establishes decidability.

2.2. NL$\diamond$

(Moortgat, 1996) extends the format of type logics in two ways. He considers calculi that comprise more than one family of residuated operators, and he generalizes Lambek's binary operators to the n-ary case. In the present paper, we will be mainly concerned with one of the simplest version of such a multimodal system, namely the combination of one binary product and its accompanying implications with one unary product and its residuated counterpart. The resulting logic is dubbed **NL**$\diamond$.

The logical vocabulary of **NL**$\diamond$ extends the vocabulary of **NL** with two unary connectives, $\diamond$ and $\Box^{\downarrow}$. So the set of **NL**$\diamond$-types is over the atoms $\mathcal{A}$ is given by

$$\mathcal{F} =:: \mathcal{A}|\mathcal{F}\backslash\mathcal{F}|\mathcal{F}\bullet\mathcal{F}|\mathcal{F}/\mathcal{F}|\diamond\mathcal{F}|\Box^{\downarrow}\mathcal{F}$$

They form a pair of residuated operators, i.e. their logical behavior is governed by the residuation law

$$A \to \diamond B \text{ iff } \Box^{\downarrow}A \to B$$

The axiomatic presentation of **NL**$\diamond$consists just of the axioms and rules of **NL** plus the above residuation law.

(Moortgat, 1996) also gives a sequent presentation of **NL**$\diamond$. Now the trees that occur in the antecedent of a sequent is composed from types by two operators, a binary one ($\circ$), and a unary one ($\langle\cdot\rangle$), corresponding to the two products $\bullet$ and $\diamond$. So we have

$$\mathcal{T} =:: \mathcal{F}|(\mathcal{T}\circ\mathcal{T})|\langle\mathcal{T}\rangle$$

Moortgat's sequent calculus for **NL**$\diamond$ is obtained by extending the sequent calculus for **NL** with the following four rules, i.e. a rule of use and a rule of proof for both $\diamond$and $\Box^{\downarrow}$.

$$\frac{X \Rightarrow A}{\langle X\rangle \Rightarrow \diamond A}\diamond L \qquad \frac{X[\langle A\rangle] \Rightarrow B}{X[\diamond A] \Rightarrow B}\diamond R$$

$$\frac{X[A] \Rightarrow B}{X[\langle\Box^{\downarrow}A\rangle] \Rightarrow B}\Box^{\downarrow}R \qquad \frac{\langle X\rangle \Rightarrow A}{X \Rightarrow \Box^{\downarrow}A}\Box^{\downarrow}R$$

Figure 2: Sequent rules for the unary modalities in **NL**$\diamond$

As in **NL**, all sequent rules of **NL**$\diamond$ have the subformula property. By proving Cut elimination for **NL**$\diamond$, (Moortgat, 1996) thus establishes decidability, and he also proves the equivalence of the axiomatic with the sequent presentation.

2.3. Logic and grammars

A type logic like **NL**$\diamond$ is the deductive backbone of a type logical grammar. The grammar itself consists just of the lexicon, i.e. an assignment of types to lexical items, and a collection of designated types (which is sometimes tacitly assumed to be the singleton set $\{s\}$, but I assume a more general notion of grammar here).

Definition 1 (NL$\diamond$-grammar) *An **NL**$\diamond$-grammar over an alphabet Σ is a pair $\langle\mathcal{L},\mathcal{D}\rangle$, where $\mathcal{L}$, the lexicon, is a finite relation between Σ^+ and the set of **NL**$\diamond$-types $\mathcal{F}$, and the set of designated types $\mathcal{D}$ is a finite subset of $\mathcal{F}$.*

(The definitions for "**NL**-grammar", "Ł-grammar" etc. are identical.) A string from Σ^+ is recognized by an **NL**$\diamond$-grammar G if it is a concatenation of lexical items, and replacing each of these items by one of their lexical types leads to the yield of some binary tree of types from which a designated category is derivable. Formally this reads as follows:

Definition 2 (Recognition) *Let $G = \langle\mathcal{L},\mathcal{D}\rangle$ be an **NL**$\diamond$-Grammar over Σ. A string $w \in \Sigma^+$ is recognized by G iff*

- $w = v_1 \cdots v_n$,
- *there are $A_1,\ldots,A_n \in \mathcal{F}$ such that for all $1 \le i \le n : \langle v_i, A_i\rangle \in \mathcal{L}$,*

- *there is a tree X and a type $S \in \mathcal{D}$ such that* **NL**$\Diamond \vdash X \Rightarrow S$*, and $A_1 \ldots A_n$ is the yield of X.*

The notion of recognition for other categorial calculi is similar—the only difference being the underlying calculus and thus the derivability relation in the last clause.

3. Generative Capacity

In this section I will present and discuss the main result of this paper, the weak generative equivalence between context free grammars and **NL**$\Diamond$-grammars. The inclusion of the context free languages in the class of **NL**$\Diamond$-recognizable languages is easy to show; the proof immediately follows from Kandulski's (1988) analogous proof for **NL** (which is itself a straightforward adaption of the corresponding proof for applicative categorial grammars from (Bar-Hillel, Gaifman and Shamir, 1960)).

Lemma 1 *Every context free language L is recognized by some* ***NL***$\Diamond$*-grammar.*

Proof: Kandulski (1988) proves that the class of **NL**-grammars recognizes exactly the context free languages. Thus there is an **NL**-Grammar $G = \langle \mathcal{L}, \mathcal{D} \rangle$ that recognizes L. From the facts that all sequent rules of cut-free **NL**$\Diamond$ have the subformula property and that all sequent rules of **NL** are also rules of **NL**$\Diamond$ it follows that **NL**$\Diamond$ is a conservative extension of **NL**. In other words, if an **NL**$\Diamond$-sequent $X \Rightarrow A$ is derivable in **NL**$\Diamond$ and does not contain occurrences of $\Diamond, \Box^{\downarrow}$, and $\langle \cdot \rangle$, it is also **NL**-derivable. The structural connective $\langle \cdot \rangle$ only cooccurs with the modalities $\Diamond$ or $\Box^{\downarrow}$ in derivable sequents. (This can easily be shown by induction over sequent derivations.) So if a derivable **NL**$\Diamond$-sequent does neither contain $\Diamond$ nor $\Box^{\downarrow}$, it is also **NL**-derivable. To decide whether a string is recognized by G or not it is sufficient to restrict attention to sequents that only involve types from $\mathcal{L}$ or $\mathcal{D}$. For this fragment, the derivability relations defined by **NL** and **NL**$\Diamond$ coincide. Therefore G still recognizes L if it is conceived as an **NL**$\Diamond$-grammar. $\dashv$

To prove that a given variety of type logical grammar recognizes only context free languages, it is sufficient to show that the relevant fragment of the underlying logic can be axiomatized by using only finitely many axioms and the Cut rule. (This strategy has in fact been pursued in all such proofs from the literature that were mentioned in the introduction.) Pentus (1993) proof for the context freeness of Ł is an especially simple and elegant implementation of this idea (even though the proof for the correctness of his construction is quite complex). Consider an Ł-grammar $G = \langle \mathcal{L}, \mathcal{D} \rangle$. It comprises finitely many types (either as lexical or as designated types), and hence there is some upper limit n for the complexity of types in G, where the complexity of a type is identified with the number of connectives occurring in it. The first important insight of his proof is that to one does not need the entire calculus Ł but just those fragment of it that only uses types with a complexity $\leq n$ to determine which language G recognizes. Let us call this fragment of Ł Ł(n). The central lemma of the proof establishes that Ł(n) can be axiomatized by the set of its sequents that have at most two antecedent types, and the Cut rule. Since there are only finitely many types occurring in Ł(n), this set of axiom is finite.

The same construction can be applied to **NL**$\Diamond$ as well. The adequacy of the construction is in fact much easier to prove here then for Ł. We first show that every **NL**$\Diamond$-sequent can be represented as the result of Cut in such a way that the premises of this Cut rule do not involve types that are more complex than the most complex type in the original sequent. Furthermore, any position in the original sequent can be chosen as the target position for the Cut application.

Lemma 2 *Let $X[Y] \Rightarrow A$ be a theorem of* ***NL***$\Diamond$*. Then there is a type B such that*

1. ***NL***$\Diamond \vdash Y \Rightarrow B$
2. ***NL***$\Diamond \vdash X[B] \Rightarrow A$
3. *There is a type occurring in $X[Y] \Rightarrow A$ which contains at least as many connectives as B.*

Proof: We prove the lemma by induction over sequent derivations. For the base case id the lemma is obviously true. So let us suppose the lemma holds for the premises of a sequent rule. We have to demonstrate that it holds for the conclusion as well. If **NL**$\Diamond \vdash X[Y] \Rightarrow A$ and B has the required properties, we call B a *witness for Y* (with respect to the sequent $X[Y] \Rightarrow A$).

Suppose that $X[Y] \Rightarrow A$ is a premise of a sequent rule and $X'[Y'] \Rightarrow B$ is the conclusion, where Y' is the substructure corresponding to Y. Then for any type C, if $\vdash Y \Rightarrow C$, then $\vdash Y' \Rightarrow C$. (Either Y and Y' are identical, or Y' is the result of applying a rule of use to Y, which is derivability preserving.) Furthermore, if $\vdash X[D] \Rightarrow A$ for some type D, then $\vdash X'[D] \Rightarrow B$ as well. (Either $X[D]$ and $X'[D]$ are simply identical, or X' is the result of applying a rule of use to X with Y as an inactive part. In the latter case, it does not matter for derivability if we replace Y with D.) Finally (Moortgat, 1996) proves that the sequent calculus of **NL**$\diamond$ enjoys the subformula property. Thus if B is a witness for Y in the premise, it is also a witness for Y' in the conclusion (because the most complex type from the premise occurs in the conclusion as subtype). So to complete the induction step, we only have to consider cases where a substructure in the conclusion does not correspond to any substructure in any of the premises of a sequent rule. This applies to all types that are created via a left introduction rule. It is obvious though that each type is a witness for itself, so the induction step holds for these cases as well. So the only cases that remain to be considered are the left hand sides of the conclusions in $\bullet R$ and $\diamond R$. In either case, the type on the right hand side is a witness for the left hand side as a whole. This completes the proof. $\dashv$

From this it follows immediately that a Pentus style axiomatization is possible for **NL**$\diamond$ as well.

Lemma 3 *Let* ***NL***$\diamond(n) = \{A \Rightarrow B|$***NL***$\diamond \vdash A \Rightarrow B \& max(\#A, \#B) \leq n\} \cup \{\langle A\rangle \Rightarrow B|$***NL***$\diamond \vdash \langle A\rangle \Rightarrow B \& max(\#A, \#B) \leq n\} \cup \{A \circ B \Rightarrow C|$***NL***$\diamond \vdash A \circ B \Rightarrow C \& max(\#A, \#B, \#C) \leq n\}$*, where $\#A$ is the number of connectives occurring in A. Furthermore, let $X \Rightarrow A$ be an* ***NL***$\diamond$*-derivable sequent such that no type in it contains more than n connectives. Then $X \Rightarrow A$ is derivable from* ***NL***$\diamond(n)$ *and Cut.*

Proof: We prove the lemma by induction over the number of structural operators (i.e. $\circ$ and $\langle\cdot\rangle$) in X. If X is a single type, the lemma is obviously true. So let us assume that $X = Y[Z]$, where $Z = B_1 \circ B_2$ or $Z = \langle B\rangle$. According to lemma 2, there is a witness C for Z such that $\vdash Z \Rightarrow C, Y[C] \Rightarrow A$, and $\#C \leq n$. Then $Z \Rightarrow C \in$ **NL**$\diamond(n)$ by definition and $Z \Rightarrow C$ is derivable from **NL**$\diamond(n)$ and Cut by induction hypothesis. Thus $X \Rightarrow A$ is derivable from **NL**$\diamond(n)$ and Cut as well. $\dashv$

This leads directly to the inclusion of the class of **NL**$\diamond$-recognizable languages in the class of context free languages.

Lemma 4 *Every* ***NL***$\diamond$*-recognizable language is context free.*

Proof: Let an **NL**$\diamond$-grammar $G_1 = \langle \mathcal{L}, \mathcal{D}\rangle$ (with $\mathcal{L}$ being the lexicon and $\mathcal{D}$ the set of designated types) be given. We construct an equivalent CFG G_2 in the following way: The terminal elements of G_2 are the lexical items of G_1. The non-terminals are all **NL**$\diamond$-types A with $\#A \leq n$, where n is the maximal number of connectives in a single type occurring in G_1. Besides we have a fresh non-terminal S which is the start symbol. Productions are

- $\{A \rightarrow B|B \Rightarrow A \in$ **NL**$\diamond(n)\}\cup$
- $\{A \rightarrow B|\langle B\rangle \Rightarrow A \in$ **NL**$\diamond(n)\}\cup$
- $\{A \rightarrow B, C|B \circ C \Rightarrow A \in$ **NL**$\diamond(n)\}\cup$
- $\{A \rightarrow v|\langle v, A\rangle \in \mathcal{L}\}\cup$
- $\{S \rightarrow A|A \in \mathcal{D}\}$

If $v_1 \ldots v_m$ is recognized by G_1, then there is a **NL**$\diamond$-derivable sequent $X \Rightarrow B$ such that B is a designated category, the yield of X is $A_1 \ldots A_m$, and $\langle v_i, A_i\rangle \in \mathcal{L}$ for $1 \leq i \leq m$. By the construction of G_2, $S \rightarrow^*_{G_2} B$. Due to lemma 3 and the construction of G_2, thus $S \rightarrow^*_{G_2} A_1 \ldots A_n$, and by the construction of G_2, this leads to $S \rightarrow^*_{G_2} v_1 \ldots v_n$. So $v_1 \ldots v_n$ is recognized by G_2.

No suppose $v_1 \ldots v_m$ is recognized by G_2. This means that $S \rightarrow^*_{G_2} v_1 \ldots v_n$. By the construction of G_2, there must be a $B \in \mathcal{D}$ and $A_1 \ldots A_n$ with $\langle v_i, A_i\rangle \in \mathcal{L}$ such that $B \rightarrow^*_{G_2} A_1 \ldots A_n$. Hence there must be a derivation from B to some structure X such that $A_1 \ldots A_n$ is the yield of X. All rules involved in this derivation originate from **NL**$\diamond(n)$, and since all rules in **NL**$\diamond(n)$ are **NL**$\diamond$-derivable, $\vdash X \Rightarrow B$. Hence $v_1 \ldots v_n$ is recognized by G_1. $\dashv$

The Lemmas 1 and 4 jointly give the main result of this section:

Theorem 1 ***NL***$\diamond$*-grammars recognize exactly the context free languages.*

Proof: Immediate. $\dashv$

4. Generalizations

The concept of residuated logical connectives can readily be generalized to n-ary operations for arbitrary n. Such this systems have been considered at various places in the context of categorial grammar, including (Moortgat, 1996) and (Kandulski, 2002). A multimodal logic of pure residuation ("**LPR**" henceforth) is characterized by a family of modes $\mathcal{M}$ and a function δ that assigns each mode an arity, i.e. a natural number. If $f \in \mathcal{M}$ is a mode of arity $\delta(f) = m$, it defines $m + 1$ m-ary connectives: an m-ary product operator $f_\bullet$, and m implications $\{f^i_\rightarrow | 1 \leq i \leq m\}$. As for **NL** and **NL**$\Diamond$, there is an axiomatic formulation for any **LPR** having the identity axiom as only axiom, the Cut rule and a collection of residuation laws. The laws for the binary and unary operators given above are generalized to the general case in the following way:

$$\forall f \forall i \leq \delta(f) : f_\bullet(A_1, \cdots, A_{\delta(f)}) \rightarrow B \text{ iff } A_i \rightarrow f^i_\rightarrow(A_1, \ldots, A_{i-1}, B, A_{i+1}, \ldots, A_{\delta(f)})_i$$

The sequent calculus for a given **LPR** over the set of modes $\mathcal{M}$ is also a straightforward extrapolation from **NL**$\Diamond$. Antecedents of sequents are now terms built from types by means of structural operators. There is one structural operator $f_\circ$ for every $f \in M$ with arity $\delta(f)$.

$$\frac{}{A \Rightarrow A}\, id \qquad\qquad \frac{X \Rightarrow A \qquad Y[A] \Rightarrow B}{Y[X] \Rightarrow B}\, Cut$$

$$\frac{\forall i \leq \delta(f) : X_i \Rightarrow A_i}{f_\circ(X_1, \cdots, X_{\delta(f)}) \Rightarrow f_\bullet(A_1, \cdots, A_{\delta(f)})}\, f_\bullet L \qquad \frac{X[f_\circ(A_1, \cdots, A_{\delta(f)})] \Rightarrow B}{X[f_\bullet(A_1, \cdots, A_{\delta(f)})] \Rightarrow B}\, f_\bullet R$$

$$\frac{\forall i \leq \delta(f), i \neq j : X_i \Rightarrow A_i \qquad Y[B] \Rightarrow C}{Y[f_\circ(X_1, \cdots, X_{i-1}, f_\rightarrow(A_1, \cdots, A_{i-1}, B, A_{i+1} \cdots, A_{\delta(f)}), X_{i+1}, \cdots, X_{\delta(f)})] \Rightarrow C}\, f_\rightarrow L$$

$$\frac{f_\circ(A_1, \cdots, A_{i-1}, X, A_{i+1}, \cdots, A_{\delta(f)}) \Rightarrow B}{X \Rightarrow f_\rightarrow(A_1, \cdots, A_{i-1}, B, A_{i+1}, \cdots, A_{\delta(f)})}\, f_\rightarrow R$$

Figure 3: Sequent rules for **LPR**

The equivalence of the axiomatic with the sequent presentation, as well as Cut elimination for all instances of **LPR** can be proven by arguments that are entirely parallel to the corresponding proofs for **NL**.

The results from the previous section on **NL**$\Diamond$ carry over to all instances of **LPR** without further ado.

Lemma 5 *Let $X[Y] \Rightarrow A$ be a theorem of **LPR**. Then there is a type B such that*

1. ***LPR*** $\vdash Y \Rightarrow B$
2. ***LPR*** $\vdash X[B] \Rightarrow A$
3. *There is a type occurring in $X[Y] \Rightarrow A$ which contains at least as many connectives as B.*

Proof: Parallel to the proof of lemma 2. $\dashv$

So there is a finite axiomatization of any fragment of an **LPR** that has an upper limit for the complexity of the types involved. The notions of an **LPR**-grammar and of the language recognized by such a grammar can be adapted from the corresponding notions related to **NL**$\Diamond$ in an obvious way. Finite axiomatizability thus amounts to the fact that every **LPR**-grammar is equivalent to some context free grammar. On the other hand, any **NL**-grammar is an **LPR**-grammar, and since any context free language is recognized by some **NL**-grammar, we obtain the following generalization of theorem 1:

Theorem 2 ***LPR**-grammars recognize exactly the class of context free languages.*

Proof: Analogous to the corresponding proof for **NL**$\Diamond$. $\dashv$

Up to now we only considered calculi of pure residuation, i.e. calculi that do without any structural rules. One might wonder what impact the presence of structural rules has on the generative capacity. We will consider the structural rules Associativity (A), Permutation (P), Contraction (C), Expansion (E) and Weakening (W) in the sequel, as applying to some distinguished binary mode f. For simplicity, we write $\circ$ instead of $f_\circ$ below. The sequent versions of these rules are given in figure 4. (The double line in A indicates that these are actually two rules, left associativity and right associativity.)

$$\frac{X[Y \circ (Z \circ W)] \Rightarrow A}{\overline{X[(Y \circ Z) \circ W] \Rightarrow A}}\,A \qquad \frac{X[Y \circ Z] \Rightarrow A}{X[Z \circ Y] \Rightarrow A}\,P$$

$$\frac{X[Y \circ Y] \Rightarrow A}{X[Y] \Rightarrow A}\,C \qquad \frac{X[Y] \Rightarrow A}{X[Y \circ Y] \Rightarrow A}\,E$$

$$\frac{X[Y] \Rightarrow A}{X[Y \circ Z] \Rightarrow A}\,W$$

Figure 4: Structural Rules

Would the proof for lemma 5 still go through if we add some of these rules to the calculus? Certainly not for A. If the induction hypothesis would hold for the sequent on top, this would not guarantee that there is a witness for $Y \circ Z$, and likewise for the opposite direction. Likewise, the induction step would not work for W, because the induction hypothesis—the lemma holds for the premise sequent—does not guarantee that there is a witness for Z. It doesn't work for C either because we cannot be sure whether the witness for the two occurrences of Y in the premise are identical. If they aren't, contraction cannot be applied anymore after replacing the two Ys by their witnesses. However, P and E are well-behaved.

Lemma 6 *Let* $\mathbf{C}$ *be some multimodal calculus that comprises the sequent rules for* ***LPR*** *and a subset of the structural rules* $\{P, E\}$ *for each binary mode of* $\mathbf{C}$*. Let* $X[Y] \Rightarrow A$ *be a theorem of* $\mathbf{C}$*. Then there is a type* B *such that*

1. $\mathbf{C} \vdash Y \Rightarrow B$
2. $\mathbf{C} \vdash X[B] \Rightarrow A$
3. *There is a type occurring in* $X[Y] \Rightarrow A$ *which contains at least as many connectives as* B.

Proof: By induction over sequent derivations. For the logical rules, the induction step was established above. As for P, the witness of any substructure of X in the conclusion is identical to the corresponding witness in the premise, and likewise for substructures of Y and Z. As for E, the witness for a substructure of X in the conclusion is inherited from the premise. Suppose Y has a substructure Z, and we want to know whether there is a witness for the occurrence of Z in the left occurrence of Y in the conclusion. By hypothesis, we know that for some type B with a complexity that is not more complex than the most complex type in the premise sequent, it holds that $\mathbf{C} \vdash X[Y[B]] \Rightarrow A$, and $\mathbf{C} \vdash B \Rightarrow Z$. By applying E, we obtain $\mathbf{C} \vdash X[Y[B] \circ Y[B]] \Rightarrow A$, and by Cut we get $\mathbf{C} \vdash X[Y[B]] \circ Y[Z] \Rightarrow A$. $\dashv$

Thus we have

Theorem 3 *A type logical grammar that is based on a calculus which extends a version of* ***LPR*** *with* P *or* E *for some of its binary modes recognizes only context free languages.*

Proof: Immediate. $\dashv$

Let me conclude this section with some remarks on the relation of my results to (Kandulski, 1995) and (Kandulski, 2002). The central theorem of the latter work is almost identical to my theorem 2 (and my theorem 1 is just a corollary of this). Kandulski gives an axiomatization for **LPR**using Cut as the only rule, and he shows that derivations in this axiomatic calculus can be normalized in such a way that only finitely instances of the axioms are relevant for a given **LPR**-grammar. There is a minor difference between his results and mine: Kandulski requires that the set of designated types of a type logical grammar is a singleton containing only one atomic type. The restriction to atomic types is in fact essential for his proof to go through. The same holds *ceteris paribus* for the context freeness proof for **NL** in (Kandulski, 1988). In this respect the results from the present paper are somewhat more general. Furthermore, (Kandulski, 1995) presents a proof that grammars based on **NL**+P only recognize context free languages. Again, the proof makes essential use of the restriction to atomic designated types.

Even though the results obtained in the present paper have an considerable overlap with Kandulski's prior work, the proof strategy used here is novel, and arguably simpler. It is also easier to generalize, as the application to Expansion illustrates.

5. Conclusion

In this paper a new strategy for proving the context freeness of a class of type logical grammars was proposed. The basic idea for the construction of context free grammars from type logical grammars is adapted from (Pentus, 1993). The proof of the correctness of the construction is different (and much simpler) though; it is based on a property of sequent derivations that can be seen as a variant of Roorda's (1991) interpolation lemma. It was shown that this property is shared by all multimodal logics of pure residuation, i.e. any pure or mixed calculi using only families of residuated operators of arbitrary arity. Prominent instances of this family of logics are **NL** and **NL**$\diamond$. It was furthermore proved that this property of sequent calculi is preserved by adding the structural rules of Permutation or Expansion. Any categorial grammar based on one of these logics recognizes a context free language. Conversely, by a slight variation of Cohen's (1967) argument for Ł it can be shown that any context free language is recognized by some **LPR** comprising at least one binary product, but no structural rules.

Further work is required to gain a deeper understanding on the impact of structural rules on generative capacity. The proof strategy that was proposed in this paper can be used as a recipe to establish context freeness for extensions of **LPR** with certain structural postulates, including interaction postulates involving several modes. However, it is not always applicable, as the example of the associative Lambek calculus demonstrates. So it would be desirable to identify sufficient conditions when structural rules preserve context freeness.

References

Bar-Hillel, Yehoshua. 1953. A quasi-arithmetical notation for syntactic description. *Language*, 29:47–58.

Bar-Hillel, Yehoshua, C. Gaifman and E. Shamir. 1960. On Categorial and Phrase Structure Grammars. *Bulletin of the Research Council of Israel*, F(9):1–16.

Buszkowski, Wojciech. 1986. Generative Capacity of Nonassociative Lambek Calculus. *Bulletin of the Polish Academy of Sciences: Mathematics*, 34:507–518.

Carpenter, Bob. 1999. The Turing-completeness of multimodal categorial grammars. Papers presented to Johan van Benthem in honor of his 50th birthday. European Summer School in Logic, Language and Information, Utrecht.

Chomsky, Noam. 1957. *Syntactic Structures*. The Hague: Mouton.

Cohen, Joel M. 1967. The equivalence of two concepts of Categorial Grammar. *Information and Control*, 10:475–484.

Jäger, Gerhard. 2001. On the Generative Capacity of Multimodal Categorial Grammars. to appear in *Journal of Language and Computation*.

Kandulski, Maciej. 1988. The equivalence of nonassociative Lambek categorial grammars and context-free grammars. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 34:41–52.

Kandulski, Maciej. 1995. On commutative and nonassociative syntactic calculi and categorial grammars. *Mathematical Logic Quarterly*, 41:217–135.

Kandulski, Maciej. 2002. On Generalized Ajdukiewicz and Lambek Calculi and Grammars. manuscript, Poznan University.

Lambek, Joachim. 1958. The Mathematics of Sentence Structure. *American Mathematical Monthly*, 65:154–170.

Lambek, Joachim. 1961. On the Calculus of Syntactic Types. In Roman Jakobson, editor, *Structure of Language and Its Mathematical Aspects*. Providence, RI.

Moortgat, Michael. 1988. *Categorial Investigations. Logical and Linguistic Aspects of the Lambek Calculus*. Dordrecht: Foris.

Moortgat, Michael. 1996. Multimodal linguistic inference. *Journal of Logic, Language and Information*, 5(3/4):349–385.

Morrill, Glyn. 1990. Intensionality and Boundedness. *Linguistics and Philosophy*, 13:699–726.

Pentus, Martin. 1993. Lambek grammars are context-free. In *Proceedings of the 8th Annual IEEE Symposium on Logic in Computer Science*. Montreal.

Roorda, Dirk. 1991. *Resource logics: Proof-theoretical investigations*. Ph.D. thesis, University of Amsterdam.

Shieber, Stuart. 1985. Evidence against the non-context-freeness of natural language. *Linguistics and Philosophy*, 8:333–343.

A Note on the Complexity of Associative-Commutative Lambek Calculus

Christophe Costa Florêncio
Utrecht University

1. Introduction

In this paper the NP-completeness of the system **LP** (associative-commutative Lambek calculus) will be shown. The complexity of **LP** has been known for some time, it is a corollary of a result for multiplicative intuitionistic linear logic (**MILL**)[1] from (Kanovich, 1991) and (Kanovich, 1992).

We show that this result can be strengthened: **LP** remains NP-complete under certain restrictions. The proof does not depend on results from the area of linear logic, it is based on a simple linear-time reduction from the minimum node-cover problem to recognizing sentences in **LP**.

2. Definitions

First some definitions are in order:

Definition 1 *The* degree *of a type is defined as*

$$\begin{array}{lcl} \mathrm{degree}(A) & = & 0 \textit{ if } A \in \mathrm{Pr} \\ \mathrm{degree}(B\backslash A) & = & 1 + \mathrm{degree}(A) + \mathrm{degree}(B) \\ \mathrm{degree}(A/B) & = & 1 + \mathrm{degree}(A) + \mathrm{degree}(B) \end{array}$$

In other words, the degree of a type can be determined by counting the number of operators it contains.

Definition 2 *The* Order *of a type is defined as*

$$\begin{array}{lcl} \mathrm{order}(A) & = & 0 \textit{ if } A \in \mathrm{Pr} \\ \mathrm{order}(B\backslash A) & = & \max(1 + \mathrm{order}(A) + \mathrm{order}(B)) \\ \mathrm{order}(A/B) & = & \max(1 + \mathrm{order}(A) + \mathrm{order}(B)) \end{array}$$

Definition 3 *A* domain subtype *is a subtype that is in domain position, i.e. for the type $((A/B)/C)$ the domain subtypes are B and C.*
For the type $(C\backslash(B\backslash A))$ the domain subtypes are C and B.

A range subtype *is a subtype that is in range position, i.e. for the type $((A/B)/C)$ the range subtypes are (A/B) and A.*
For the type $(C\backslash(B\backslash A))$ the range subtypes are $(B\backslash A)$ and A.

In an applicaton $A/B, B \vdash A$ or $B, B\backslash A \vdash A$ the type B is an argument *and A/B and $B\backslash A$ are known as* functors*.*

Definition 4 *Let $G = (V, E)$ be an undirected graph, where V is a set of nodes and E is a set of edges, represented as tuples of nodes. A* node-cover *of G is a subset $V' \subseteq V$ such that if $(u, v) \in E$, then $u \in V'$ or $v \in V'$. That is, each node 'covers' its incident edges, and a node cover for G is a set of nodes that covers all the edges in E. The* size *of a node-cover is the number of nodes in it.*

The node-cover problem *is the problem of finding a node-cover of minimum size (called an* optimal node-cover*) in a given graph.*

The node-cover problem can be restated as a decision problem*: does a node-cover of given size k exist for some given graph?*

Proposition 5 *The decision problem related to the node-cover problem is* NP*-complete, The node-cover problem is* NP*-hard.*

This problem has been called one of the 'six basic NP-complete problems' in (Garey and Johnson, 1979).

1. The systems **LP** and **MILL** are identical up to derivation from the empty sequent, i.e. the only difference is that $\vdash n/n$ is not derivable in **LP**.
The system **MILL** is closely related to **MILL1**, another system that has interesting linguistic applications, see (Moot and Piazza, 2001).

3. Complexity of LP

Theorem 6 *Deciding membership for the unidirectional product-free fragment of* **LP**, with all types restricted to a maximum degree of 2 and a maximum order of 1, is NP-complete in $|\Sigma|$.

`Proof`: It is well known that **LP** is in NP.

What remains to be shown is existence of a p-time reduction from an NP-complete problem. Let $G = (E, V)$ be an undirected graph, $ne = |E|$. Let $C = \mathrm{C}(G)$ be a minimum node cover of G, and $\min(G) = |\mathrm{C}(G)|$. The graph G can be reduced to a grammar $Gr = \mathrm{gram}(G)$ as follows:

1. Assign s to s.
2. Let f be the function that maps node V_n to type v_n. For every edge $E_x \in E$, where $E_x = \langle V_y, V_z\rangle$, let $v_y = f(V_y), v_z = f(V_z)$. Assign types $v_y\backslash v_y$, $v_y\backslash(s\backslash s)$ and $v_z\backslash v_z$, $v_z\backslash(s\backslash s)$ to symbol v_x.
3. For every node $V_n \in V$, assign $f(V_n) = v_n$ to `node`.

The intuition behind this reduction is that `node` stands for any node in G, and e_x for the *connection* of edge E_x to any of the two nodes it is incident on.

Note that this reduction always yields a unidirectional product-free grammar, with all types restricted to a maximum degree of 2 and a maximum order of 1. Also note that this reduction sets $|\Sigma|$ to the number of edges plus two.

We will now show that accepting a sentence s of the form $\mathsf{s}\,\underbrace{\mathsf{node}\ldots\mathsf{node}}_{i\ \text{times}}\ \mathsf{v}_1\ldots\ \mathsf{v}_{ne}$ as being in $\mathrm{L}(\mathrm{gram}(G))$ while rejecting $\mathsf{s}\,\underbrace{\mathsf{node}\ldots\mathsf{node}}_{i-1\ \text{times}}\ \mathsf{v}_1\ldots\ \mathsf{v}_{ne}$ will indicate that there is a node cover of size i for G. Simply iterating from $i = 1$ to $i = ne$ will lead to acceptance when $i = \min(G)$.

Parsing such a sentence will yield a *solution*: one can collect the assignments to the symbol `node` used in the derivation to obtain a minimum node cover.

Let T be some set of types (taken from the assignments to `node` in $\mathrm{gram}(G)$) assigned to the substring $\underbrace{\mathsf{node}\ldots\mathsf{node}}_{i\ \text{times}}$ of s. Let U be some set of types assigned to the substring $\mathsf{v}_1\ldots\mathsf{v}_{ne}$ under the same restrictions.

1. Assume that $i < \min(G)$. Since by the form of s $|T| \leq i$, $|T| < \min(G)$, so for every minimum node cover C, there is a $V_n \in C$ such that $f(V_n) \notin T$. Since for every edge $\langle V_y, V_z\rangle \in E$, there is some v_n in s that has been assigned either the type $v_x\backslash v_x$ or $v_x\backslash(s\backslash s)$, $v_x = f(V_y)$ or $v_x = f(V_z)$.

   Since for every edge $\langle V_y, V_z\rangle \in E$, $f(V_y) \in C$ or $f(V_z) \in C$, there is some v_m in s that has been assigned $v_n\backslash v_n$ or $v_n\backslash(s\backslash s)$, $v_n \notin T$.

   Since $\Gamma, pT, \Gamma' \not\vdash_{\mathbf{LP}} \Gamma, \Gamma'$ (where pT is a primitive type), in order to derive (just) s, all the types in T have to occur as argument to an application in the derivation. Given the form of $\mathrm{gram}(G)$ this is possible just if the functor is a type assigned to $\mathsf{v}_{1\leq n\leq ne}$. Thus $s_{1\leq i<\min(G)} \notin \mathrm{L}(\mathrm{gram}(G))$.

2. Assume $i = \min(G)$. Then there is a T such that $|T| = i$. Let Tc be $\{f(V_n)|V_n \in C\}$, for some C. Given s and assignments of types such that for each $1 \leq p < ne$, $v_p\backslash(s\backslash s)$ occurs at most once …

   Since **LP** is associative and commutative any rearrangment is allowed during a derivation. This property can be used to 'sort' the assignments to the symbols `node` and v_n in the following way: each occurrence of `node` (assigned type $v_x \in Tc$) is followed by all v_n's that are assigned type $v_x\backslash v_x$, followed by a single v_n assigned $v_n\backslash(s\backslash s)$. The substring thus obtained is associated with a sequent that derives $(s\backslash s)$. The whole of s minus s, can be arranged into a number of these substrings, and since $A\backslash A, A\backslash A \vdash_{\mathbf{LP}} A\backslash A$, the associated sequent will derive $s\backslash s$. Since s is only assigned s in $\mathrm{gram}(G)$, we finally get the derivation $s, s\backslash s \vdash s$.

This shows that the reduction given is indeed a reduction from an NP-complete problem. □

Example: Reducing $G = (\{(1,2), (1,3), (3,4), (2,4)\}, \{1,2,3,4\})$ will yield

$$
\text{gram}(G):\quad
\begin{array}{lcl}
\mathsf{s} & \mapsto & s\\
\mathsf{v}_1 & \mapsto & v_1\backslash v_1, v_1\backslash(s\backslash s), v_2\backslash v_2, v_2\backslash(s\backslash s)\\
\mathsf{v}_2 & \mapsto & v_1\backslash v_1, v_1\backslash(s\backslash s), v_3\backslash v_3, v_3\backslash(s\backslash s)\\
\mathsf{v}_3 & \mapsto & v_3\backslash v_3, v_3\backslash(s\backslash s), v_4\backslash v_4, v_4\backslash(s\backslash s)\\
\mathsf{v}_4 & \mapsto & v_2\backslash v_2, v_2\backslash(s\backslash s), v_4\backslash v_4, v_4\backslash(s\backslash s)\\
\mathsf{node} & \mapsto & v_1, v_2, v_3, v_4
\end{array}
$$

The corresponding minimal node cover is $\{1,4\}$ or $\{2,3\}$.
As a final remark, note that there exists an alternative reduction $\text{gram}'(G)$:

1. Assign s to s.

2. For every edge $E_x \in E$, where $E_x = \langle V_y, V_z\rangle$, let $v_y = f(V_y), v_z = f(V_z)$. Assign types $v_y\backslash v_y$ and $v_z\backslash v_z$ to symbol e_x.

3. For every node $V_n \in V$, assign $v_x\backslash(s\backslash s)$ to c and $f(V_n) = v_n$ to node.

Example: Applying this procedure to the same graph yields:

$$
\text{gram}'(G):\quad
\begin{array}{lcl}
\mathsf{s} & \mapsto & s\\
\mathsf{v}_1 & \mapsto & v_1\backslash v_1, v_2\backslash v_2\\
\mathsf{v}_2 & \mapsto & v_1\backslash v_1, v_3\backslash v_3\\
\mathsf{v}_3 & \mapsto & v_3\backslash v_3, v_4\backslash v_4\\
\mathsf{v}_4 & \mapsto & v_2\backslash v_2, v_4\backslash v_4\\
\mathsf{node} & \mapsto & v_1, v_2, v_3, v_4\\
\mathsf{c} & \mapsto & v_1\backslash(s\backslash s), v_2\backslash(s\backslash s), v_3\backslash(s\backslash s), v_4\backslash(s\backslash s)
\end{array}
$$

Accepting a sentence of the form $\mathsf{s}\,\underbrace{\mathsf{node}\ldots\mathsf{node}}_{i\ \text{times}}\ \mathsf{v}_1\ldots\ \mathsf{v}_{ne}\,\underbrace{\mathsf{c}\ldots\mathsf{c}}_{i\ \text{times}}$ as being in $\text{L}(\text{gram}(G))$ will indicate that there is a node cover of size i for G. Again, iterating from $i = 1$ to $i = ne$ will lead to acceptance when $i = \min(G)$.

4. Example Derivations

Given graph $G = (\{(1,2),(1,3),(3,4),(2,4)\},\{1,2,3,4\})$, the grammar $\text{gram}(G)(G)$ and sentence '`s node node  v1 v2 v3 v4`' ($i = 4$) we get the solutions shown in Figures 1 and 2.

$$
\dfrac{
\dfrac{\text{s} \vdash s \qquad
\dfrac{\dfrac{\text{node} \vdash v_1 \quad \text{v1} \vdash v_1\backslash v_1}{\text{node} \circ \text{v1} \vdash v_1}[\backslash E] \qquad \text{v2} \vdash v_1\backslash(s\backslash s)}{(\text{node} \circ \text{v1}) \circ \text{v2} \vdash s\backslash s}[\backslash E]}
{\text{s} \circ ((\text{node} \circ \text{v1}) \circ \text{v2}) \vdash s}[\backslash E]
\qquad
\dfrac{\dfrac{\text{node} \vdash v_4 \quad \text{v3} \vdash v_4\backslash v_4}{\text{node} \circ \text{v3} \vdash v_4}[\backslash E] \qquad \text{v4} \vdash v_4\backslash(s\backslash s)}{(\text{node} \circ \text{v3}) \circ \text{v4} \vdash s\backslash s}[\backslash E]
}{
(\text{s} \circ ((\text{node} \circ \text{v1}) \circ \text{v2})) \circ ((\text{node} \circ \text{v3}) \circ \text{v4}) \vdash s
}[\backslash E]
$$

$$
\begin{array}{cl}
(\text{s} \circ ((\text{node} \circ \text{v1}) \circ \text{v2})) \circ ((\text{node} \circ \text{v3}) \circ \text{v4}) \vdash s & \\ \hline
(\text{s} \circ ((\text{node} \circ \text{v1}) \circ \text{v2})) \circ (\text{node} \circ (\text{v3} \circ \text{v4})) \vdash s & [ass] \\ \hline
(\text{s} \circ (\text{node} \circ (\text{v1} \circ \text{v2}))) \circ (\text{node} \circ (\text{v3} \circ \text{v4})) \vdash s & [ass] \\ \hline
((\text{s} \circ \text{node}) \circ (\text{v1} \circ \text{v2})) \circ (\text{node} \circ (\text{v3} \circ \text{v4})) \vdash s & [ass] \\ \hline
(\text{s} \circ \text{node}) \circ ((\text{v1} \circ \text{v2}) \circ (\text{node} \circ (\text{v3} \circ \text{v4}))) \vdash s & [ass] \\ \hline
(\text{s} \circ \text{node}) \circ (((\text{v1} \circ \text{v2}) \circ \text{node}) \circ (\text{v3} \circ \text{v4})) \vdash s & [ass] \\ \hline
(\text{s} \circ \text{node}) \circ ((\text{node} \circ (\text{v1} \circ \text{v2})) \circ (\text{v3} \circ \text{v4})) \vdash s & [comm]
\end{array}
$$

Figure 1: A derivation for '`s node node v1 v2 v3 v4`' corresponding to the minimum node cover $\{v_1, v_4\}$.

$$\dfrac{\mathrm{s} \vdash s \qquad \dfrac{\dfrac{\mathrm{node} \vdash v_2 \quad \mathrm{v1} \vdash v_2\backslash v_2}{\mathrm{node} \circ \mathrm{v1} \vdash v_2}\,[\backslash E] \qquad \mathrm{v4} \vdash v_2\backslash(s\backslash s)}{(\mathrm{node} \circ \mathrm{v1}) \circ \mathrm{v4} \vdash s\backslash s}\,[\backslash E]}{\mathrm{s} \circ ((\mathrm{node} \circ \mathrm{v1}) \circ \mathrm{v4}) \vdash s}\,[\backslash E] \qquad \dfrac{\dfrac{\mathrm{node} \vdash v_3 \quad \mathrm{v2} \vdash v_3\backslash v_3}{\mathrm{node} \circ \mathrm{v2} \vdash v_3}\,[\backslash E] \qquad \mathrm{v3} \vdash v_3\backslash(s\backslash s)}{(\mathrm{node} \circ \mathrm{v2}) \circ \mathrm{v3} \vdash s\backslash s}\,[\backslash E]$$

$$\dfrac{}{(\mathrm{s} \circ ((\mathrm{node} \circ \mathrm{v1}) \circ \mathrm{v4})) \circ ((\mathrm{node} \circ \mathrm{v2}) \circ \mathrm{v3}) \vdash s}\,[\backslash E]$$

$$\dfrac{(\mathrm{s} \circ ((\mathrm{node} \circ \mathrm{v1}) \circ \mathrm{v4})) \circ ((\mathrm{node} \circ \mathrm{v2}) \circ \mathrm{v3}) \vdash s}{(\mathrm{s} \circ ((\mathrm{node} \circ \mathrm{v1}) \circ \mathrm{v4})) \circ (\mathrm{node} \circ (\mathrm{v2} \circ \mathrm{v3})) \vdash s}\,[ass]$$

$$\dfrac{}{(\mathrm{s} \circ (\mathrm{node} \circ (\mathrm{v1} \circ \mathrm{v4}))) \circ (\mathrm{node} \circ (\mathrm{v2} \circ \mathrm{v3})) \vdash s}\,[ass]$$

$$\dfrac{}{((\mathrm{s} \circ \mathrm{node}) \circ (\mathrm{v1} \circ \mathrm{v4})) \circ (\mathrm{node} \circ (\mathrm{v2} \circ \mathrm{v3})) \vdash s}\,[ass]$$

$$\dfrac{}{(\mathrm{s} \circ \mathrm{node}) \circ ((\mathrm{v1} \circ \mathrm{v4}) \circ (\mathrm{node} \circ (\mathrm{v2} \circ \mathrm{v3}))) \vdash s}\,[ass]$$

$$\dfrac{}{(\mathrm{s} \circ \mathrm{node}) \circ (((\mathrm{v1} \circ \mathrm{v4}) \circ \mathrm{node}) \circ (\mathrm{v2} \circ \mathrm{v3})) \vdash s}\,[ass]$$

$$\dfrac{}{(\mathrm{s} \circ \mathrm{node}) \circ ((\mathrm{node} \circ (\mathrm{v1} \circ \mathrm{v4})) \circ (\mathrm{v2} \circ \mathrm{v3})) \vdash s}\,[comm]$$

$$\dfrac{}{(\mathrm{s} \circ \mathrm{node}) \circ (\mathrm{node} \circ ((\mathrm{v1} \circ \mathrm{v4}) \circ (\mathrm{v2} \circ \mathrm{v3}))) \vdash s}\,[ass]$$

$$\dfrac{}{(\mathrm{s} \circ \mathrm{node}) \circ (\mathrm{node} \circ (\mathrm{v1} \circ (\mathrm{v4} \circ (\mathrm{v2} \circ \mathrm{v3})))) \vdash s}\,[ass]$$

$$\dfrac{}{(\mathrm{s} \circ \mathrm{node}) \circ (\mathrm{node} \circ (\mathrm{v1} \circ ((\mathrm{v2} \circ \mathrm{v3}) \circ \mathrm{v4}))) \vdash s}\,[comm]$$

Figure 2: A derivation for '`s node node v1 v2 v3 v4`' corresponding to the minimum node cover $\{v_2, v_3\}$.

References

Garey, Michael R. and David S. Johnson, editors. 1979. *Computers and Intractability. A Guide to the Theory of NP-completeness.* Freeman, New York.

Kanovich, Max I. 1991. The multiplicative fragment of linear logic is NP-complete. ITLI Prepublication Series X-91-13, University of Amsterdam.

Kanovich, Max I. 1992. Horn programming in linear logic is NP-complete. In *Proceedings, Seventh Annual IEEE Symposium on Logic in Computer Science*, pages 200–210. IEEE Computer Society Press, 22-25 June.

Moot, Richard and Mario Piazza. 2001. Linguistic applications of first order multiplicative linear logic. *Journal of Logic, Language and Information*, 10(2):211–232.

Turning Elementary Trees into Feature Structures

Alexandra Kinyon
University of Pennsylvania
Institute for Research in Cognitive Science
Suite 400A, 3401 Walnut Street
Philadelphia, PA 19104-6228
kinyon@linc.cis.upenn.edu
http://www.cis.upenn.edu/∼kinyon

1. Introduction

The richness of the TAG formalism allows one to encode a wide variety of linguistic information in the topology of an elementary tree (ex: argument vs adjunct distinction, information about "belonging" to a given subcategorization class, constraints on word order, realization of one or more syntactic phenomena such as passive, wh extraction etc...). Given a number of "linguistic" information and constraints applying to lexical items, a grammar developer will encode this information into one or several elementary trees. However, proceeding in the opposite direction is far less obvious : given an elementary tree, which crucial pieces of linguistic information are provided by this tree, and how can these pieces be clearly isolated and represented ?

In this paper we try to answer that question. More precisely, we investigate the different possibilities for extracting from each elementary tree of a TAG a feature structure which captures and clearly separates each piece of essential linguistic information conveyed by that elementary tree. [1]

We examine **existing implementations** of wide-coverage TAG grammars. Current existing wide-coverage TAG grammars are either semi-automatically generated from an abstract level of syntactic representation such as MetaRules or a MetaGrammar, or are "plain" grammars that do not resort to such an intermediate step of abstract syntactic generalization [2]. Therefore, after explaining in the first part of this paper why featurizing a TAG is useful, we then discuss the featurization of these three types of TAGs: We examine the "featurization" of a TAG semi-automatically generated with a MetaGrammar, the featurization of a TAG generated with MetaRules, and finally, several options for featurizing a plain TAG (hand-crafted, or automatically acquired from a treebank). We argue that automatic featurization is desirable, and show that such an automation is achievable thanks to existing implementations, either via a MetaGrammar approach or via a MetaRules approach, even in order to featurize plain grammars.

2. Why featurize a TAG ?

By featurizing a elementary tree, we mean capturing its main linguistic characteristics and representing them in a feature structure. Of course, several kinds of information may be envisioned, depending on the application. Still, a general need for featurization appears for tasks as diverse as improving Supertagging accuracy, Text Generation, Corpus annotation and search, and extraction of framework independent linguistic information.

2.1. Improving Supertagging accuracy

Supertagging consists in assigning one or several elementary tree(s) to each lexical item in a text (Srinivas, 1997). Good SuperTagging accuracy is obtained with medium-size grammars[3]. However, as the size of a TAG increases from a few hundred trees for hand-crafted grammars to a few thousand trees for automatically generated grammars, the accuracy of Supertagging deteriorates (just as for traditional POS tagging, the POS accuracy

1. Since we are concerned with *linguistic* information, we leave aside approaches which focus on capturing common *mathematical* or *topological* properties of elementary trees such as those discussed in (Kallmeyer, 1999) (logical formula) or (Halber, 2001) (linear types).
2. A "plain" TAG can be either hand-crafted (e.g. (XTAG Research Group, 2001)) or acquired automatically from a TreeBank (e.g. (Chiang, 2000)).
3. For example (Chen, Bangalore and Vijay-Shanker, 1999) report up 98.4% accuracy with an average of 4.8 supertags per word, using the hand-crafted Xtag grammar for English, comprising approx. 600 trees.

deteriorates with a large POS tagset). To remedy this problem, F. Toussenel and A. Nasr, following a proposal from (Chen, 2001), are associating a "feature vector" to each supertag i.e. a set of of features which capture the salient characteristics of a SuperTag. The idea is then to train a supertagger on each element of the vector in order to reduce sparse data problems. (Chen, 2001) proposes features such as *Part of speech, Subcategorization....By-passive, Wh movement, Topicalized*, but does not detail how to obtain these features from elementary trees [4]. As we discuss below, this kind of features can be organized and extracted from Supertags in a systematic way.

2.2. Text generation

Another application for featurization is text generation. For example, (Bangalore and Rambow, 2000) discuss featurization for text generation as a way to *improve error analysis and allow for better stochastic Supertag assignments models*. For lack of space we refer to their paper page 39 for a list of features they use. [5] Also for text generation, (Danlos, 2000) uses the G-TAG formalism, and the notion of *T-features* which can take the value + or - to identify non-canonical syntactic constructions (i.e. to refer to elementary trees). For instance, if one considers the transitive family, the binary T-features *T-passive, T-without-arg-2* are used to distinguish the salient features of four distinct elementary trees in that family. A given feature structure will then yield a node in a derivation tree. From that derivation tree, a sentence exhibiting the correct syntactic phenomena will be generated

We do not discuss into details the other numerous potential applications of featurizing a TAG, but still mention the use of featurization for sentence planning and query-answer applications (Stone and Doran, 1997), for easy search of annotated corpora (the idea being that it is easier, esp. for non TAG experts, to search a feature such as *relativized object*=+ in a corpus where each word is annotated with a feature structure representing its main syntactic characteristic, rather than making queries on the shape of Supertags identifying relativized-objects) (Kinyon, 2000). Finally, featurizing a TAG is a way to achieve framework independence. This can be used to share resources between lexicons, to compile TAGs into different frameworks, and more generally to define a notion of *syntactic interlingua*.

However, to achieve such a goal, one must agree among all the featurizations proposed, on what kind of linguistic featurization is desirable and/or achievable.

4. For a complete list of features proposed see his PhD dissertation pp246-247. Another discussion related to training probabilistic models taking into account syntactic transformations, independently of TAGs, may be found in (Eisner, 2001)
5. Since they deal with both probabilistic supertagging, and text generation, we have made the arbitrary choice of classifying their work in this section instead of the previous one.

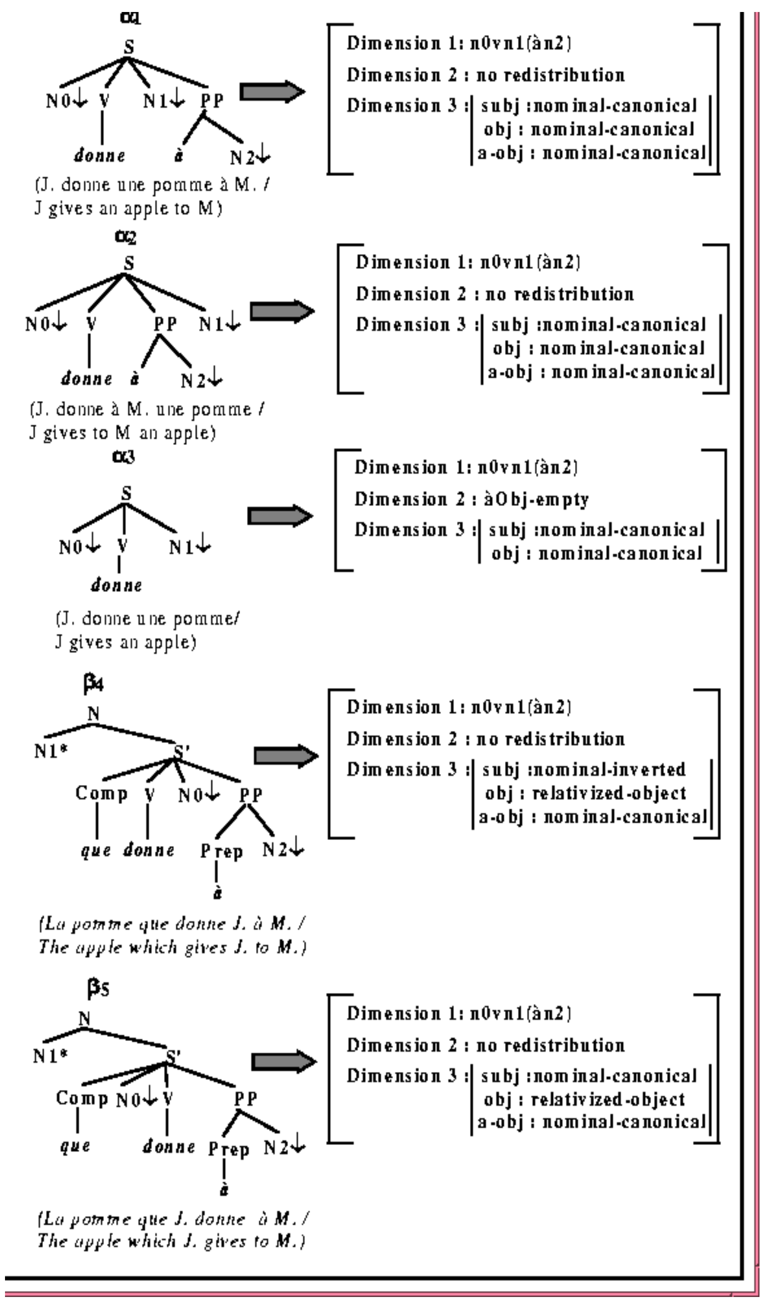

α1
S
N0↓ V N1↓ PP
donne à N2↓
(J. donne une pomme à M. / J gives an apple to M)
Dimension 1: n0vn1(àn2)
Dimension 2 : no redistribution
Dimension 3 : subj :nominal-canonical
obj : nominal-canonical
a-obj : nominal-canonical
α2
S
N0↓ V PP N1↓
donne à N2↓
(J. donne à M. une pomme / J gives to M an apple)
Dimension 1: n0vn1(àn2)
Dimension 2 : no redistribution
Dimension 3 : subj :nominal-canonical
obj : nominal-canonical
a-obj : nominal-canonical
α3
S
N0↓ V N1↓
donne
(J. donne une pomme/ J gives an apple)
Dimension 1: n0vn1(àn2)
Dimension 2 : àObj-empty
Dimension 3 : subj :nominal-canonical
obj : nominal-canonical
β4
N
N1* S'
Comp V N0↓ PP
que donne Prep N2↓
à
(La pomme que donne J. à M. / The apple which gives J. to M.)
Dimension 1: n0vn1(àn2)
Dimension 2 : no redistribution
Dimension 3 : subj :nominal-inverted
obj : relativized-object
a-obj : nominal-canonical
β5
N
N1* S'
Comp N0↓ V PP
que donne Prep N2↓
à
(La pomme que J. donne à M. / The apple which J. gives to M.)
Dimension 1: n0vn1(àn2)
Dimension 2 : no redistribution
Dimension 3 : subj :nominal-canonical
obj : relativized-object
a-obj : nominal-canonical

Figure 1: Featurization of elementary trees with a MetaGrammar for French: each elementary tree is associated to a feature structure which contains the main syntactic characteristics of the tree (i.e. the final classes of the hierarchy which allowed the generation of that tree).

3. Featurizing with a MetaGrammar

(Candito, 1996) has originally developed in LISP a tool to semi-automatically generate a 5000-tree wide coverage TAG for French (Abeille, Candito and Kinyon, 1999) as well as a large grammar for Italian (Candito, 1999). The idea is to use an additional layer of linguistic description called the MetaGrammar (MG), which imposes a general organization for syntactic information in a three-dimensional hierarchy :

- Dimension 1: initial subcategorization

- Dimension 2: valency alternations and redistribution of functions
- Dimension 3: surface realization of arguments.

Each terminal class in dimension 1 describes a possible initial subcategorization (i.e. a tree family). Each terminal class in dimension 2 describes a list of ordered redistributions of functions (e.g. it allows to add an argument for causatives). Each terminal class in dimension 3 represents the surface realization of a surface function (ex : pronominalized, wh-extracted etc.). Each class in the hierarchy corresponds to the partial description of a tree (Rogers and Vijay-Shanker, 1994). An elementary tree is generated by inheriting from exactly one terminal class from dimension 1, one terminal class from dimension 2, and n terminal classes from dimension 3 (where n is the number of arguments of the elementary tree being generated). Since the hierarchy summarizes the main linguistic pieces of information that will be used to generate a given elementary tree, when using such a MetaGrammar to generate a TAG, it is possible to keep track of which terminal classes each elementary tree inherits from. Therefore, such a TAG can easily be "featurized" as can be seen on figure 1. This implementation was not included in Candito's tool, but (Crabbe and Gaiffe, 2002) recently reimplemented a MetaGrammar generator in Java, which incorporates such a featurization facility. In addition, (Crabbe and Gaiffe, 2002)'s tool offers flexibility in the sense that one or more features may be associated to each class in the hierarchy, and that the feature(s) associated to a given class may be the name of that class, but may also be any other feature deemed appropriate. This flexibility has obvious advantages, but it may also prove to be an inconvenient in the sense that there is no constraints based on linguistic principles. For example, their implementation does not enforce a three-dimensional hierarchy. It is up to the MetaGrammar writer to make sure that the hierarchy is linguistically adequate. Nonetheless, it is a very useful tool to automatically featurize a TAG.

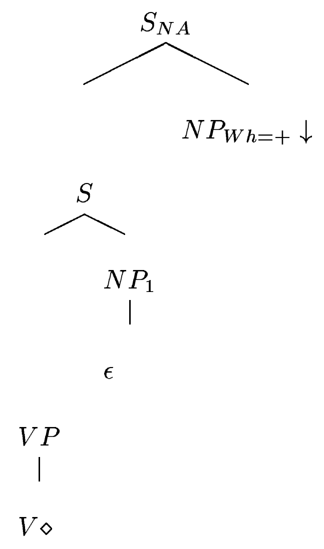
4. Featurizing with MetaRules

The idea of MetaRules is presented in (Becker, 2000). A MetaRule takes as input an elementary tree and outputs a new, and generally more complex, elementary tree. Therefore, in order to create a TAG, one can start from one *canonical* elementary tree for each subcategorization frame and a finite number of MetaRules which model syntactic transformations (e.g. passive, wh-questions etc) and automatically generate a full-size grammar. [6] Within the Xtag project (Prolo, 2002) was the first one to implement a large TAG grammar for English based on MetaRules and to tackle with the numerous practical implementation issues left unsolved. [7] This grammar consists of 1008 elementary trees and was generated from 57 canonical trees and 21 MetaRules within the Xtag project. It is quite straightforward, when the grammar is generated with MetaRules, to keep track of which MetaRules were applied to generate a given elementary tree. The main characteristic of an elementary tree can therefore be represented in a feature structure with one feature taking as value a subcategorization frame, and 21 binary features (one for each metarule) having the value + if the metarule was applied to generate that tree, and the feature - if it was not. Figure 2 illustrates such a featurization. So this is a second possibility to achieve the automatic featurization of a TAG.

5. Featurizing a plain grammar

To featurize a *plain* grammar, hand-crafted or automatically acquired, one can start from a finite set of features that are judged linguistically interesting (e.g. such as those suggested by (Chen, 2001). One can then perform the tedious task of manually examining each elementary tree in the TAG to assign a value to each of those features for each of these trees (possibly with the help of some degree of automation e.g. macros, pearl scripts etc...). However, ensuring constitency with such a hand-featurization does not seem realistic : ideally we would want a given feature structure to correspond to a unique elementary tree. At the very least, we would want a consistent featurization of all the trees. For example, having a *subcategory* feature for the trees in some family but not for all families would be problematic. Similarly, having a *relativized* feature for only some of the trees exhibiting a relativized argument would be equally unsatisfactory. This type of inconsistency could easily arise though esp.

6. Since we examine **the** one practical existing implementation of MetaRules, we do not take position on the theoretical status of *syntactic transformations* nor on the best way to ensure the finiteness of the generation process (i.e. via finite closure or ordering of the rules etc.) We refer to (Becker, 2000), or (Evans, Gazdar and Weir, 2000) for a slightly passionate discussion on this issue.
7. Unfortunately, apart from (Prolo, 2002), no detailed description of this implementation has been published to this day.



Family	= transitive
Passive	= +
Passive-from-PP	= -
DropBy	=+
Gerund	= -
WhSubject	= +
WhSentSubj	= -
WhNPObj	= -
WhSmallNPObj	= -
WhApObj	= -
WhAdvObj	= -
WhPPObj	= -
RelAdjNoW	= -
RelAdjW	= -
RelSubjNoW	= -
RelSubjNoWForPassive	= -
RelSubjW	= -
RelObjNoW	= -
RelObjW	= -
RelPPObj	= -
PROSubject	= -
Imperative	= -

Figure 2: Featurization with MetaRules: elementary tree for *What was eaten* and its associated feature structure (i.e. MetaRules applied to generate the tree are marked +)

when the plain grammar is augmented : when new trees are added to the grammar, manual featurization obliges one to go over the feature structure associated to each new tree in the grammar. Therefore, automatic featurization is higly desirable even in the case of a hand-crafted TAG : given a TAG, a program needs to systematically extract the salient linguistic information from each elementary tree. Note that this amounts to extracting a higher level of syntactic information from the grammar (i.e. a set of MetaRules or the hierarchy nodes of a MetaGrammar). The question remains as to whether it is preferable to extract a MetaGrammar (yielding non binary features) or MetaRules (yielding, at least with the existing implementation, binary features).

It is generally believed that the MetaGrammar approach is preferable because it allows *real* featurization, instead of just a feature vector. This is highly debatable and may be simply due to the fact that the linguistic basis of the MetaGrammar approach have been more cleanly defined i.e. the MG incorporate the notion of syntactic function, and the distinction between subcategorization, valency alternations and realization of arguments.

For Metarules, the notion of initial subcategorization is de-facto captured since a meta-rule grammar is initially generated from a set of elementary trees. In addition to that, it seems highly desirable and feasible to incorporate the notions of syntactic function, valency distribution and realization of arguments into the notion of metarules.

This amounts to formulating metarules in terms of syntactic functions and separating two types of MetaRules :

- Alternation metarules (ordered)
- Realization metarules (unordered)

Moreover, in practice, Metarules are more compact than MetaGrammars for developing *real* grammars (but this point is beyond the scope of this work). The bottom line is that MetaRules and MetaGrammars are not necessarily as different as they are thought to be, and that automatically extracting an abstract level of syntactic representation from a plain TAG yields in fact a hybrid representation that is between the notion of MG and that of MetaRules.

6. Conclusion

We have (briefly) discussed diverse applications for which extracting the salient linguistic characteristics of a TAG via featurization is interesting. We have shown that existing TAG implementations based on MetaGrammar and MetaRules allow automatic featurization. We have argued that automatic featurization is preferable even in the case of *plain* grammars, whether hand-crafted or automatically acquired; and that a MetaGrammar approach is not necessarily preferable to a MetaRule approach. We also would have liked to present the partial implementation which we have done on the PennTreebank to acquire a hybrid level of syntactic representation, mixing MetaGrammar and MetaRules characteristics but we will have to wait for a more suitable presentation format.[8] From this abstract level of syntactic representation, one can generate a TAG, but also potentially grammars in another format (such as LFG, HPSG, dependency grammars etc...). Hence, an abstract level of syntactic representation (MG, MetaRules or Hybrid) can be an interesting syntactic interlingua candidate for compiling one grammar framework into another.

6.1. Acknowlegements

We thank B. Crabbe, C. Prolo, F. Toussenel for their helpful comments on earlier drafts of this paper.

References

Abeille, A., M.H. Candito and A. Kinyon. 1999. FTAG: current status and parsing scheme. In *VEXTAL-99*, Venice.
Bangalore, S and O. Rambow. 2000. Using TAGs, a Tree Model, and a Language Model for Generation. In *TAG+5*, Paris.
Becker, T. 2000. Patterns in metarules for TAG. In Abeille Rambow, editor, *Tree Adjoining Grammars*, CSLI.
Candito, M.H. 1996. A principle-based hierarchical representation of LTAGs. In *COLING-96*, Copenhagen.
Candito, M.H. 1999. *Representation modulaire et parametrable de grammaires electroniques lexicalisees*. Ph.D. thesis, Univ. Paris 7.
Chen, J. 2001. *Towards Efficient Statistical Parsing using Lexicalized Grammatical Information.* Ph.D. thesis, Univ. of Delaware.
Chen, J., S. Bangalore and K. Vijay-Shanker. 1999. New Models for Improving Supertag Disambiguation. In *EACL99*, Bergen.
Chiang, D. 2000. Statistical parsing with an automatically-extracted TAG. In *ACL-00*, Hong-Kong.
Crabbe, B. and B. Gaiffe. 2002. A new Metagrammar Compiler. In *Proc. TAG+6*, Venice.
Danlos, L. 2000. G-TAG: A lexicalized formalism for text generation. In Abeille Rambow, editor, *Tree Adjoining Grammars*, CSLI.
Eisner, J. 2001. *Smoothing a Probabilistic Lexicon via Syntactic Transformations.* Ph.D. thesis, Univ. of Pennsylvania.
Evans, R., G. Gazdar and D. Weir. 2000. Lexical rules are just lexical rules. In Abeille Rambow, editor, *Tree Adjoining Grammars*, CSLI.
Halber, A. 2001. *Strategie d'analyse pour la comprehension de la parole: une approche TAG.* Ph.D. thesis, ENST-Paris.
Kallmeyer, L. 1999. *Tree Description Grammars and Underspecified Representations.* Ph.D. thesis, Univ. of Tubingen.
Kinyon, A. 2000. Hypertags. In *COLING-00*, Sarrebrucken.
Kinyon, A. and C. Prolo. 2002. Identifying verb arguments and their syntactic function in the Penn Treebank. In *LREC 02*, Las Palmas.
Prolo, C. 2002. Generating the Xtag English grammar using metarules. In *Proc. TAG+6*, Venice.
Rogers, J. and K. Vijay-Shanker. 1994. Obtaining trees from their description: an application to TAGS. In *Computational Intelligence 10:4*.
Srinivas, B. 1997. *Complexity of lexical descriptions and its relevance for partial parsing.* Ph.D. thesis, Univ. of Pennsylvania.
Stone, M. and C. Doran. 1997. Sentence Planning as description using TAG. In *ACL97*, Madrid.
XTAG Research Group. 2001. A Lexicalized Tree Adjoining Grammar for English. Technical Report IRCS-01-03, IRCS, University of Pennsylvania.

8. Meanwhile, we refer the reader to (Kinyon and Prolo, 2002) for an incomplete overview of how tree families are extracted from the PennTreebank with an interlingua appraoch.

On the Affinity of TAG with Projective, Bilexical Dependency Grammar

Tom B.Y. Lai[1,2] Changning Huang[3] and Robert W.P. Luk[4]
City University of Hong Kong[1], Tsinghua University[2], Microsoft Research Beijing[3], Polytechnic University of Hong Kong[4]

1. Introduction

This paper describes a projective, bilexical dependency grammar, and discusses its affinity with TAG. Common features of the two formalisms include a tree-like surface syntactic structure and readiness for a lexicalised treatment. TAG surface structures built from elementary and auxiliary trees by means of substitution and adjunction can correspond to trees consisting entirely of lexical nodes and dependency arcs. Lexical anchors in TAG, a well-motivated notion, can also be accommodated in the dependency grammar formalism, provided it is recognized that the dependent, as well as the governor, can have a vote about the formation of a dependency relation. It is noted, however, that mirroring obligatory adjuncts in TAG in dependency grammar can be problematic.

2. Dependency Analysis

2.1. Projective Dependency Structures

Though not supported by all schools of dependency grammar (Tesnière, 1959), some followers of dependency grammar assume that there is a projective surface or back-bone dependency structure. The theoretical foundation of this tradition can be traced to Gaifman (1965) and Hays (1964), and is summed up in the following well-formedness conditions for dependency structures in Robinson (1970):

- one and only one element is independent;
- all others depend directly on some element;
- no elements depend directly on more than one other;
- if A depends directly on B and some element C intervenes between them (in linear order of string), then C depends directly on A or on B or some other intervening element.

These conditions say, in effect, that conforming dependency structures are representable by trees without crossing branches. Of courses, as in other grammar formalisms that pre-suppose a context-free syntactic structure back-bone, additional linguistic constraints can be incorporated in the formalism by means of various mechanisms, e.g. feature unification.

2.2. Dependency Structures without Phrasal Nodes

In the dependency grammar formalism (Lai and Huang, 1998; Lai and Huang, 2000) discussed in this paper, dependency structures are trees consisting entirely of *lexical* nodes. For example, the dependency *tree* for (1a), taken from Abeillé (1993), is (1b):

(1) a.

Jean	dort	beaucoup
Jean	sleeps	much

‘Jean sleeps a lot.’

b.

```
             dort
               |
 ---------------------------
 | subj             adjunct |
 |                          |
Jean                    beaucoup
```

When a coarser degree of granularity is warranted by the situation, the actual lexical items in the tree nodes can be replaced by their syntactic categories.

2.3. Statistical Dependency Analysis

In the computational linguistics community, dependency structures are often parsed, exploiting their affinity with phrase-structure structures, with the help techniques used with context-free grammars (Hellwig, 1986). On the other hand, Collins (1996) uses ‘bilexical’ co-ocurrence probabilities (of governors and their head daughters) to estimate the likelihood of phrase-structures in the syntactic analysis of sentences. In a recent effort on Chinese (Lai *et al.*, 2001), bilexical probabilities have been used directly to derive, without direct reference to context-free grammar and phrasal structures, dependency structures using a CYK-like algorithm (Eisner, 1996; Eisner, 2000).

The probabilistic model in Lai and Huang (2000) uses conditional probabilities defined in terms of dependencies. Factors considered include both dominance and ‘function’, as well as other contextual factors like relative proximity to the governor. No phrase structures are generated and the dependency structures consist of only binary bilexical relations. A CYK-like algorithm is used to construct optimal non-constituent structures that ‘span’ chunks of contiguous words until the ‘span’ covers the whole input.

In the experiment reported, a training set of about 40 M of text was taken from a two-gigabyte Chinese newspaper corpus. A lexicon of about 60,000 entries was generated. The performance of the statistical dependency parser was gauged against the annotations in training corpus. For the more stringent criterion of getting both the dominance relation and the functional label correct at the same time, closed and open test averages were 94.7% (95.6% correct in the training corpus) and 74.2% (94.9% correct in the training corpus).

3. Dependency and Lexicalized TAG

The formal properties of this dependency grammar formalism can be compared with those of lexicalised TAG as described in (Abeillé, 1993; Abeillé and Rambow, 2000).

3.1. Initial Trees and Substitution

TAG trees are derived by applying the operations of substitution and adjunction to *initial* trees and *auxiliary* trees respectively. Initial trees like (2) account for the complements in the projection of a subcategorizing word (e.g. *dort* ‘sleeps’).

(2)
```
                V
                |
     ---------------------
     |                   |
     N                   V
    --->                 |
                        dort
```

In this example, the arrow attached to the node N indicates that a similar initial tree for a noun (an N-tree) can replace the node by the substitution operation. To avoid confusion, we avoid the use of the word ‘head’, but *dort* can be safely called the ‘anchor’ of the tree, which forms a part of its lexical property.

In the dependency grammar formalism, a verb like *dort* will subcategorize for each of its complement dependents.

(3)
```
dort
  |
  | subj
  |
Jean
```

A word subcategorizes for each of its complement dependents separately. Positional constraints can be added to handle multiple complements.

3.2. Auxiliary Trees and Adjunction

In TAG, adjuncts are accounted for by auxiliary trees and the operation of adjunction. For example, the word *beaucoup* ‘a lot’ is the anchor of the auxiliary tree (4):

(4)
```
            V
            |
 -----------------
 |                |
 V*              Adv
                  |
               beaucoup
```

This auxiliary tree is a lexical property of the anchor *beaucoup*. It can be used to replace a V-tree (a sub-tree) in a syntactic structure. The replaced V-tree is then, used to replace, in turn, the V* node in the auxiliary tree. The *beaucoup* auxiliary tree is ‘adjoined’ to a tree representing the sentence *Jean dort* to obtain *Jean dort beaucoup*.

In TAG, an adjunct is on the same level as or higher than the head word of the phrase in the *derived* syntactic tree as in (5a) or (5b).

(5) a.
```
         V
         |
   -------------
   V           Adv
   |            |
  dort      beaucoup
```

b.
```
         VP
         |
    -------------
   VP           Adv
   |             |
   V         beaucoup
   |
  dort
```

c.
```
  dort
   |
  ---------
          |
       beaucoup
```

TAG is probably correct in letting adverbs (e.g. *beaucoup*) decide that they are to be adjoined to verbs.

In the dependency grammar formalism, a verb (e.g. *dort*) governs an adverb (e.g. *beaucoup*). (See (5c) and the dependency structure for *Jean dort beaucoup* in (1b).) However, there is nothing to prevent a dependent adjunct from determing (as in TAG) what it should be adjoined to.

Where only initial trees and substitutions are involved, it is obvious that TAG derived trees can be pruned into dependency structures. (Dependency relation labels like ‘subject’ can obtained from lexical information or from the configuration of the tree.) Conversely, dependency structures can also be fleshed out to form TAG derived trees (with minimal structure). Adjunction makes the situation somewhat more complicated as the adjunct has to be placed higher up than the ‘head’ word in the TAG derived tree. This is possible because the substitution-adjunction distinction is obtainable from dependency relation labels (e.g. ‘adjunct’). in the dependency structure.

Similar grammatical information can be stored in the lexicon in either formalism. Additional mechanisms, e.g. feature unificaiton, can also be added on top of the tree backbone in both formalisms.

4. Some Complications

The adjuncts (daughters in dependency grammars) that we have looked at are *optional*.

However, with the presence of *adjunction constraints* or *top* and *bottom features* in TAG, adjuncts can either be optional or obligatory.

4.1. Auxiliary Adjunction

One kind of obligatory adjuncts discussed in Abeillé and Rambow (2000) are auxiliaries like *has* and *is* in *has seen* and *is seen*.

(6) a.
```
        S
        |
------------------
|                |
NP               VP: OA((6b), (6c))
-->              |
                 V
                 |
                seen
```
b.
```
        VP
        |
 ------------------
 |                |
Aux               VP*
 |
has
```
c.
```
        VP
        |
 ------------------
 |                |
Aux               VP*
 |
 is
```

The initial tree (6a) associated with *seen* (a past participle form) has a VP node with the *obligatory adjunction constraint* $OA(6b, 6c)$ indicating that an adjunction operation must be applied to it using the auxiliary trees (6b) or (6c).

In a dependency grammar, the first decision to be made is whether the governor should be the participle *seen*, or the auxiliary *has* (or *is*). One can very well follow TAG and say that the participle is the governor and the auxiliary is an adjunct daughter. A mechanism can be added to stipulate that, given that the governing verb is in participle form, the adjunct auxiliary is obligatory.

An ‘obligatory’ adjunct may sound weird to those who are accustomed to associating the term ‘adjunct’ with optional dependents, but it should be noted that formally, or mathematically, there is nothing to censor this usage. Anyway, there is obviously a sense in which an auxiliary is not subcategorized for like the ‘complement’ arguments of a predicate.

In Chinese, adverbial particles like *le* (perfective marker) are often said to be words rather than morphological affixes. Given this practice, considering these words to be adjuncts as in TAG is not an unreasonable way to account for their occurrence with the governor predicate. It should perhaps be noted, though, that this will also mean that an adjunct can come between the governor predicate and subcategorized complements as in:

(7)

na	le	dian	qian
take	PERF	some	money

‘taken/took some money’

4.2. Modals and Raising Predicates

In Abeillé and Rambow (2000), modals, like *can*, and other raising predicates, like *difficult* and *possible*, are also considered to be obligatory adjuncts.

In TAG, even though modals are adjuncts, they occupy a higher position than the verb predicates in the derived trees.

(8)
```
                S
                |
      ---------------------
      |                   |
      NP                  VP
      |                   |
    John            ----------------
                    |              |
                    V              VP
                    |              |
                   can             V
                                   |
                                  swim
```

In dependency grammar, however, the modal will have to be a daughter of the predicate verb.

With other raising predicates, there may be a difficulty. For example, placing *difficult* under (*to*) *read* in the dependency structure for *difficult to read* will be a remarkable commitment.

It must be said that this is not a problem with TAG. Raising predicate are higher up in derived syntactic tree anyway. This is a problem with dependency grammar only.

5. Concluding Remarks

The formal properties of TAG are well understood (Joshi, Vijay-Shanker and Weir, 1991), and the close relation between TAG and dependency grammar have been known (Rambow and Joshi, 1997)[1]. In this paper, in particular, we have noted that the basic TAG mechanisms of substitution and adjunction go well with a projective bilexical dependency grammar approach in general. We have however also noted that coping with the TAG notion of obligatory adjuncts, e.g. as applied to modals and other raising predicates, can be problematic.

To a certain extent, the above observations support the idea of trying to abstract away from particular grammar formalisms when marking the surface syntactic structures of a corpus, for example, as suggested by the annotators of the Chinese PennTreeBank (Xia *et al.*, 2000). It must, however, also be noted that trying to mirror TAG derived trees may be complicated sometimes.

References

Abeillé, Anne. 1993. *Les nouvelles syntaxes*. Paris: Armand Colin.

Abeillé, Anne and Owen Rambow. 2000. Tree Adjoining Grammar: An Overview. In Anne Abeillé and Owen Rambow, editors, *Tree Adjoining Grammars: Formalisms, Linguistic Analysis and Processing*. CSLI Publications, Stanford, pages 1–68.

Collins, Michael. 1996. A New Statistical Parser Based on Bigram Lexical Dependencies. In *Proceedings of 34th Annual Meeting of ACL (ACL'96)*, Santa Cruz.

Eisner, Jason. 1996. Three New Probabilistic Models for Dependency Parsing: An Exploration. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*, pages 340–345, Copenhagen, August.

Eisner, Jason. 2000. Bilexical Grammars and Their Cubic-time Parsing Algorithms. In Bunt and Nijholt, editors, *Advances in Probabilistic and Other Parsing Technologies*. Kluwer Academic Publishers, Dordrecht, pages 29–62.

Gaifman, Haim. 1965. Dependency Systems and Phrase-Structure Systems. *Information and Control*, 8:304–337.

Hays, David. 1964. Dependency Theory: A Formalism and Some Observations. *Language*, 40:511–525.

Hellwig, Peter. 1986. Dependency Unification Grammar. In *Proceedings of the 11th International Conference on Computational Linguistics (COLING'86)*, pages 195–199.

Joshi, Aravind, J. Vijay-Shanker and David J. Weir. 1991. The Convergence of Mildly Context-Sensitive Grammar Formalisms. In P. Sells, Shieber S. and T. Warsaw, editors, *Foundational Issues in Natural Language Processing*. MIT Press, Cambridge, Mass., pages 31–81.

Lai, Tom B.Y. and Changning Huang. 1998. An Approach to Dependency Grammar for Chinese. In Yang Gu, editor, *Studies in Chinese Linguistics*. Linguistic Society of Hong Kong, Hong Kong, pages 143–163.

Lai, Tom B.Y. and Changning Huang. 2000. Dependency-based Syntactic Analysis of Chinese and Anotation of Parsed Corpus. In *Proceedings of ACL'2000*, pages 255–262, Hong Kong, October.

1. Thanks to an anonymous reviewer for pointing out and providing this reference.

Lai, Tom B.Y., C.N. Huang, Ming Zhou, J.B. Miao and Tony K.C. Siu. 2001. Span-based Statistical Dependency Parsing of Chinese. In *Proceedings of NLPRS'2001*, pages 667–684, Tokyo, November.

Rambow, Owen and Aravind Joshi. 1997. A Formal Look at Dependency Grammars and Phrase-Structure Grammars, with Special Consideration of Word-Order Phenomena. In Leo Wanner, editor, *Recent Trends in Meaning-Text Theory*. John Benjamins, Amsterdam and Philadelphia, pages 167–190.

Robinson, Jane. 1970. Dependency Structures and Transformation Rules. *Language*, 46:259–285.

Tesnière, Lucien. 1959. *Elements de syntaxe structurale*. Paris: Klincksieck.

Xia, Fei, Martha Palmer, Nianwen Xue, Mary Ellen Okurowski, John Kovarik, Fu-Dong Chiou, Shizhe Huang, Tony Kroch and Mitch Marcus. 2000. Developing Guidelines and Ensuring Consistency for Chinese Text Annotation. In *LREC-2000*, Athens.

The Theory of Control Applied to the Prague Dependency Treebank (PDT)

Jarmila Panevová, Veronika Řezníčková, and Zdeňka Urešová
Charles University, Prague, Czech Republic {panevova,rez,uresova}@ufal.mff.cuni.cz

1. Introduction

One of the most difficult issues within corpora annotation on an underlying syntactic level is the restoration of nodes omitted in the surface shape of the sentence, but present on the “underlying” or “deep” syntactic level. In the present paper we concentrate on such type of nodes which are omitted due to the phenomenon usually called grammatical “control” with regard to their respective anaphoric relations. In particular, we extend the notion of control to nominalization and demonstrate how this relation is captured in the Prague Dependency Treebank.

The theory of control is present within Chomsky’s framework of Government and Binding (using the terms *verb of control*, *controller* and *controllee*, cf. Chomsky, 1980), but also within many other formal frameworks, e.g. GPSG (Sag and Pollard, 1991) or categorial grammar (Bach, 1979). We analyse this phenomenon within the framework of the dependency grammar, theoretically based on the Functional Generative Description (FGD, cf. Sgall, Hajičová and Panevová, 1986).

In FGD, on the “underlying” or “tectogrammatical” level, control is a relation of an obligatory or an optional referential dependency between a *controller* (antecedent) and a *controllee* (empty subject of the nonfinite complement (= *controlled clause*)). The controller is one of the participants in the valency frame of the governing verb (Actor (ACT), Addressee (ADDR), or Patient (PAT)). The controlled clause functions also as a filler of a dependency slot in the valency frame of the governing verb, being labeled as Patient or Actor. The empty subject of the controlled clause may have the function of different dependency relations to its head word (the infinitive): Actor, or, with passivization of the controlled clause, Addressee or Patient (cf. Koktová, 1992).

2. Capturing of “control” phenomena in the PDT

In the present section we focus on the capturing of the phenomenon of control in the Prague Dependency Treebank (PDT), a three-layer annotated corpus of Czech, basically conceived of in accordance with the theoretical assumptions of the FGD (for more information about PDT cf. Hajič: Tectogrammatical Representation: Towards a Machine Transfer in Machine Translation, this volume).

2.1. Restoration of deletions and capturing of coreferential relations in the PDT

One of the basic principles of annotation of the PDT at the tectogrammatical level concerns also restoration of deletions: in the cases of deletion in the surface sentence, nodes are introduced into the tectogrammatical tree to 'recover' a deleted word. It includes also a restoration of deleted participants of valency frames of verbs. When the nodes deleted in constructions of control are restored, annotators should indicate coreferential relations between the arguments in positions of the controller and the controllee. For labeling these coreferential relations the following attributes (grammatemes) of the general scheme are relevant:

COREF(erence) - the value of this attribute is the lexical value of the antecedent of the given anaphoric node (this node itself may be present on the surface, or deleted)

ANTEC(edent) – the value of this attribute corresponds to the functor of the antecedent with grammatical coreference[1]

CORNUM – refers to the antecedent of the given node[2].

The *Controllee* gets the special lemma *Cor.*

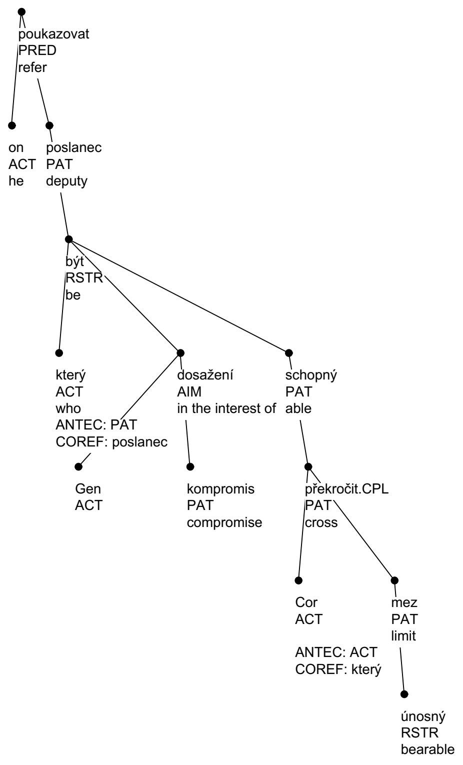
Let us present here some illustrative examples of rather complicated sentences from our annotated corpus that exhibit relations between the arguments in positions of the controller and the controllee.

* Supported by the Ministry of Education of the ČR Project LN00A0063.

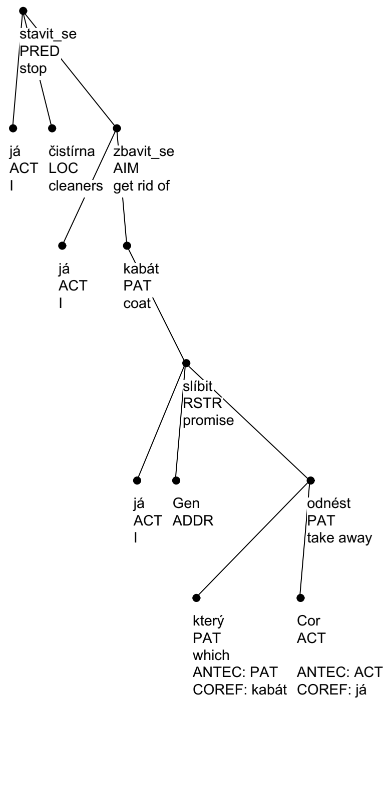
1. For the difference between the textual and the grammatical coreference see Hajičová, Panevová and Sgall, 2000.

2 Technically, the CORNUM is the only attribute that has to be marked, since the attributes COREF and ANTEC can be then easily extracted from the referred-to node. For the reason of perspicuity we refer to all the three attributes separately.

(1) Poukazuje na poslance, kteří jsou v zájmu dosažení kompromisu schopni překročit únosnou mez.
'He refers to deputies who are able in the interest of the compromise to cross the bearable limit.'



(2) Musím se stavit v čistírně, abych se zbavil toho kabátu, který jsem slíbil odnést.
'I must stop at the cleaners to get rid of the coat (which) I promised to take away.'



2.2. Survey of views on "control" phenomena with verbs in the FGD

2.2.1. Classification of verbs of control with controlled infinitive clauses

Koktová and Panevová classify the verbs of control according to the type of its valency frame and to the functions of the controlled infinitive clause and the controller in the valency frame of the verb of control (see Koktová, 1992, and Panevová 1986, 1996). According to this classification the following basic groups of verbs of control should be recognized (we leave out here some groups with really rare types of verbs of control, e.g. verbs with the so-called Slavonic Accusative with Infinitive, e.g. *Viděl Karla přicházet* (lit. *He saw Charles to-come*)):

1. The controlled infinitive clause functions as Patient: three groups of verbs of control in Czech can be distinguished, namely verbs in the valency frame of which the Controller is:

i) ACT (e.g. *Jan se bojí zůstat doma sám* (*John is afraid to stay at home alone*))

ii) ADDR (e.g. *Redaktor doporučil autorovi provést několik změn v textu* (*An editor recommended the author to make several changes in the text*))

iii) ACT or ADDR (the verb *slíbit* (*promise*) with the Controller functioning as ACT: e.g. *Jan slíbil matce vrátit se domů před půlnocí* (*John promised his mother to return at home before midnight*); the same verb with the Controller functioning as ADDR e.g. *Rodiče slíbili dětem užít si prázdniny ve stanu u rybníka* (lit.: *The parents promised (their) children to enjoy the holidays in a tent by a lake*))

2. The controlled infinitive clause functions as Actor: especially the "predicate" of control (expressed by a copula with an evaluative or modal adjective) is taken into account (e.g. *Je snadné číst tu knihu* (*It is easy to read the book*))

3. The controlled infinitive clause can have also another function, as cases based on the operation of raising (e.g. *Viktor se zdá být chytrý* (*Viktor seems to be clever*)) and the function of attribute (e.g. *Viktor nesmí propást šanci vyhrát* (*Viktor may not miss the occasion to win*)).

2.2.2. Extension of verbs of control also to the so-called "analytical predicates"

The most typical verbs of control (belonging to the group (1)(i)) are modal verbs (e.g. *moci (can), smět (may), chtít (want), muset (must), mít (have to)*) and so-called "phase verbs" (e.g. *začít (begin), zůstat (stay), přestat (stop)*). While describing the phenomenon of control, it seems to be necessary to extend the understanding of the notion of modal verb also to another synonymous expressions of these verbs. Thus the function of modal verbs is undertaken not only by "modal verbs in the wider sense" (*umět (be able), dovést (know how to do sth), dokázat (manage), zdráhat se (hesitate), odmítat (refuse)* etc.) but also by "analytical predicates" with modal meaning (the verb *mít* (*have*) plus a noun, e.g. *mít schopnost* (lit. *have an ability*), *dar (*lit. *have a gift / talent), potřebu (have an urge to do sth), příležitost (have an opportunity), šanci (have a chance)*; the verb *být* (*be*) plus a modal adjective, e.g. *být schopen (be able), ochoten (be willing), povinen (be obliged)).*

Also some verbs from other semantic groups of verbs of control can be expressed by some type of "analytical predicate". For example verbs expressing intent, e.g. *hodlat* (*intend), snažit se (try)*, can be paraphrased by predicates *mít v úmyslu (úmysl), záměr* (lit. *have an intention*), *mít v plánu (plán)* (lit. *have a plan*), *mít tendenci* (lit. *have a tendency*) etc.*; být připraven (be ready), odhodlán (be determined)* etc. (they belong also to the group (1)(i)). Verbs expressing the meaning *"umožnit někomu udělat něco" (make it possible for somebody to do something)* can be paraphrased by analytical predicates *dát někomu šanci (příležitost) udělat něco* (lit. *give somebody a chance (an opportunity) to do sth*) (these verbs belong to the group (1)(ii)).

2.2.3. Verbs of control with controlled nominalizations

Panevová (1996) deals not only with controlled infinitive verb structures but also with some types of nominalizations where an omission of an argument is also based on the "control" properties of the head (governing) word and must be interpreted as coreferentiality. The group of verbs that offer the possibility for controlled nominalization includes for example verbs from the semantic group of causing a change of a physical and/or mental state, e.g. přisoudit (adjudge), osočit (accuse), podezírat (suspect): Paní podezírá komornou z krádeže stříbrných příborů (The lady suspects the chamber-maid of the theft of silver covers)).

2.3. Nominalizations in constructions of control

The restoration of deletions in PDT includes not only the restoration of all obligatory participants and obligatory free modifications of verbs deleted at the surface shape of the sentence, but also the restoration of obligatory members of valency frames of postverbal nouns and adjectives formed by the process of nominalization.

2.3.1. From verbs to nouns

By nominalizations we understand:

a) Nouns derived from verbs by productive means (e.g. *rozhodnutí (decision making)*, *obžalování* (*accusing*) or nouns derived from verbs by non-productive means or by the zero suffix (e.g. *rada* (*advise*), *slib* (*promise*))

b) Nouns derived from the predicative adjective (e.g. *on je schopen udělat* (*he is able to do sth*) *→ jeho schopnost napsat knihu (his ability to write a book)*, *on je povinen udělat* (*he is obliged / required to do sth*) *→ jeho povinnost vydat majetek (his duty / obligation to release possession)*

c) Deverbative adjectives, it seems that only predicative deverbative adjectives can occur with control (e.g. *dívka je schopna studovat* (*the girl is able to study*)*→ dívka schopná studovat (a girl able to study) , osoba je povinna platit daně (the person is obliged to pay taxes) → osoba povinná platit daně (a person obliged to pay taxes)*

d) Nouns which were a part of an analytical predicate (e.g. *Petr má šanci vyhrát (Peter has a chance to win*) *→ Petrova šance vyhrát* (*Peter's chance to win*)*, Petr má právo odvolat se (Peter has a right to appeal) → Petrovo právo odvolat se (Peter's right to appeal).*

Some of the nouns derived from this type of analytical predicates, especially from those with the meaning of intent, do not express grammatical coreference, e.g. *nápad vydat knihu (an idea to publish a book)* (cf. also Panevová, 1996).

2.3.2. Types of nominalized constructions of control

Considering the possibility of a nominalization of both the governing as well as the dependent verb, we deal with four types of constructions of control:

1. The infinitive clause depends on a finite verb (e.g. *radil nechodit (he advised not to go)*, *slíbil napsat (he promised to write);*
2. The infinitive clause depends on a nominalization of a finite verb (e.g. *rada nechodit* (*an advice not to come*), *slib napsat* (*a promise to write*));
3. The nominalization of the embedded verb depends on a finite verb (e.g. *obvinil někoho z vyvolání problému (he charged a person with a raising of a problem), vyžadoval odpuštění daní (he claimed exemption of the taxes))*;
4. The nominalization of the embedded verb depends on a nominalization of a finite verb (e.g. *obvinění z vyvolání problému (an accusation of a raising of a problem), snaha o podplacení (an attempt for corruption)).*

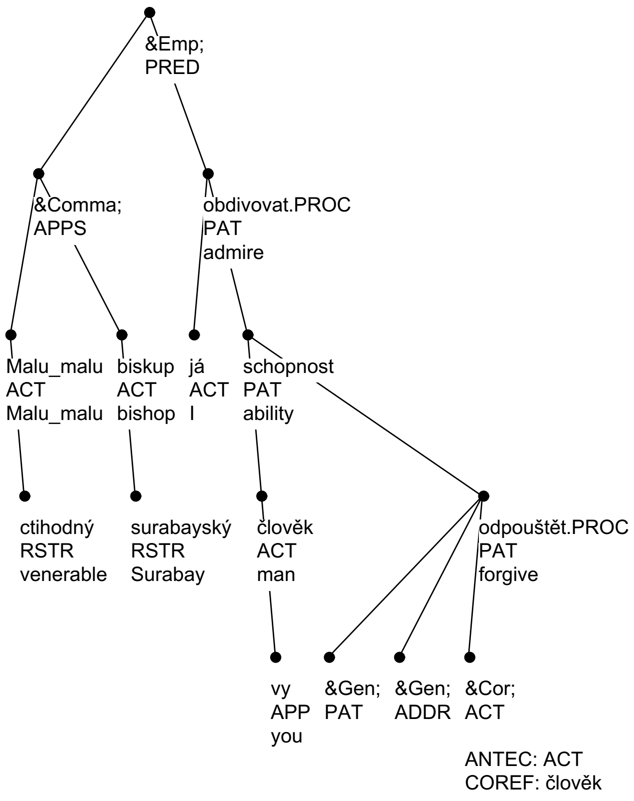
However, it is necessary to say that not all groups of verbs of control mentioned in section 2.2.1. allow for its nominalization or for a nominalization of its controlled infinitive clause:

- Verbs of control from the groups (1)(i), (ii) and (iii) may occur in all four types of constructions of control (e.g. verbs *slíbit* (*promise*), *vyžadovat* (*require, claim*), *snažit se* (*try*): *slíbit napsat* (*to promise to write*), *slib napsat* (*a promise to write*), *slíbit napsání* (*to promise writing*), *slib napsání* (*a promise of writing*)
- Verbs of control from the group (2) allow only for the nominalization of the dependent verb (*Je snadné číst tu knihu* (*It is easy to read the book*) - *Četba této knihy je snadná* (*The reading of this book is easy*)
- Verbs from the group (3) do not allow nominalization in constructions of control.

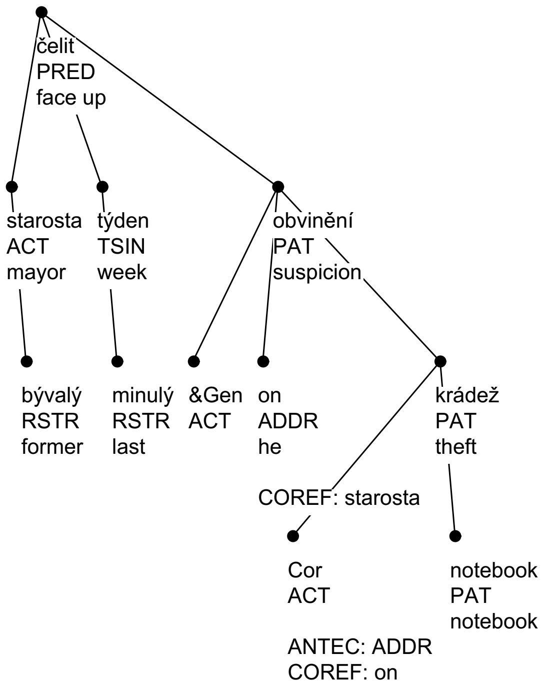
Verbs mentioned in section 2.2.3. may occur only in construction types (3) and (4) (e.g. verbs *podezírat* (*suspect*), *obvinit* (*accuse*): *podezírat z krádeže* (*to suspect of theft*), *podezření z krádeže* (*a suspicion of theft*), but **podezírat krást* (*to suspect to steal*), **podezření krást* (*a suspicion to steal*)).

Let us present here some illustrative examples of nominalized constructions of control from our annotated corpus:

(3) Ctihodný Malu-malu, biskup Surabayský: Obdivuju schopnost Vašich lidí odpouštět.
'The venerable Malu-malu, the bishop of Surabaya: I admire the ability of your people to forgive.'



(4) Bývalý starosta od minulého týdne čelí obvinění z krádeže notebooku.
'The former mayor has been facing up to suspicion of theft of the notebook since the last week.'



2.3.3. Coreferential relations in nominalized constructions

Nominalized constructions retain the same coreferential relations between the Controller and the Controllee which were realized in constructions with the corresponding verbs of control. Thus, e.g. the nominalized constructions of verbs from the group (1)(iii) mentioned in section 2.2.1. offer the possibility for the Controller to be an Actor or an Addressee. These features are illustrated in the following examples:

1. Constructions in which the Actor of the governing postverbal noun and the Actor of the dependent noun (derived from the predicate expressed by a copula with an adjective) are identical:

(5) jeho slib poslušnosti
derived from the construction *slíbil, že bude poslušný* (*he promised to be obedient*)
'his promise of obedience'

The controllee in the valency frame of the dependent noun (i.e. *poslušnost* (*obedience*)) gets the lemma *Cor* and the functor ACT. Its attributes for coreferential relations are filled in by the following values: COREF: *on* (*he*), ANTEC: ACT.

2. Constructions in which the Actor of the dependent noun (derived from the predicate expressed by a copula with an adjective) is identical to the ADDR of the governing postverbal noun:

(6) slib beztrestnosti
derived from the construction *slíbili mu, že bude beztrestný* (*they promised him to be exempt from punishment*)
'a promise of impunity'

The Controllee in the valency frame of the dependent noun (i.e. *beztrestnost* (*impunity*)) gets the lemma *Cor* and the functor ACT. Its attributes for coreferential relations are filled in by the following values: COREF: *on* (*he*), ANTEC: ADDR.

3. Conclusion

In the present paper we sum up how the “control” phenomenon is treated in the framework of the FGD and demonstrate how annotators capture the control properties in the PDT. We also presented the extension of the notion of verbs of control to the so-called “analytical predicates”, but especially to the nominalized constructions of control. We showed that the nominalized constructions retain the same coreferential relations between the Controller and the Controllee as those realized in constructions with the respective verbs of control.

References

Bach, Emmon. 1979. Control in Montague grammar. Linguistic Inquiry 10: 515-531.
Chomsky, Noam. 1980. On Binding. Linguistic Inquiry 11: 1-46.
Hajič, Jan. 2002. Tectogrammatical Representation: Towards a Machine Transfer in Machine Translation. In: Proceedings of TAG+6, 2002, Venice, Italy, this volume.
Hajičová, Eva, Jarmila Panevová, and Petr Sgall. 2000. Coreference in Annotating a Large Corpus. In: Proceedings of LREC 2000, 1: 497-500, Athens, Greece.
Koktová, Eva. 1992. On New Constraints on Anaphora and Control. Theoretical Linguistics 18: 102-178.
Panevová, Jarmila. 1986. The Czech Infinitive in the Function of Objective and the Rules of Coreference. In: *Language and Discourse: Test and Protest* (Festschrift for P. Sgall, ed. by J. Mey), Amsterdam – Philadelphia: John Benjamins, 123-142.
Panevová, Jarmila. 1996. More Remarks on Control. In: *Prague Linguistic Circle Papers*, 2 (eds. E. Hajičová, O. Leška, P. Sgall, Z. Skoumalová). J. Benjamins Publ. House: Amsterdam - Philadelphia, 101-120.
Sag, Ivan, and Carl Pollard. 1991. An Integrated Theory of Complement Control. Language 67: 63-113.
Sgall, Petr, Eva Hajičová and Jarmila Panevová. 1986. *The Meaning of the Sentence in Its Semantic and Pragmatic Aspects*. Dordrecht: Reidel and Prague: Academia.

Systematic Grammar Development in the XTAG Project

Carlos A. Prolo
University of Pennsylvania

1. Introduction

The XTAG Project (Joshi, 2001) is an ongoing project at the University of Pennsylvania since about 1988, aiming at the development of natural language resources based on Tree Adjoining Grammars (TAGs) (Joshi and Schabes, 1997). Perhaps the most successful experience in it has been the construction of a wide-coverage Lexicalized TAG for English (LTAG) (Doran *et al.*, 2000; XTAG Research Group, 2001), based on ideas initially developed in (Krock and Joshi, 1985).

As the grammar grew larger, the process of consistent grammar development and maintenance became harder (Vijay-Shanker and Schabes, 1992). Driven by locality principles, each elementary tree for a given lexical head is expected to contain its projection, and slots for its arguments (e.g., (Frank, 2001)). As consequence, the number of required elementary trees grows huge for a grammar with reasonable coverage of syntactic phenomena. Under the XTAG project, for engineering reasons, the grammar has been split up in (roughly) two main components[1]: a set tree templates lexicalized by a syntactic category, and a lexicon with each word selecting its appropriate tree templates. Although various syntactic categories have multiple syntactic frames available (e.g., prepositions may have different kinds of arguments, nouns and adjectives may have arguments or not, etc.), it is the verbs that exhibit the most wild variety of domains of locality: from the 1226 template trees in the XTAG grammar, 1008 are for verbs, more than 82%. That happens because the grammar tries to capture in elementary trees the locality for each of the diverse syntactic structures related transformationally to each other (the effect of long distance movement is captured by adjunction of the intervening material). Examples of required tree templates are: declarative transitive; ditransitive passive with wh-subject moved; and intransitive with PP object with the PP-object relativized.

As early noticed by (Vijay-Shanker and Schabes, 1992) the information regarding syntactic structure and feature equations in (feature-based) LTAGs is repeated across templates trees in a quite regular way, that perhaps could be more concisely captured than by just having a plain set of elementary trees. Besides the obvious linguistic relevance, as a pure engineering issue, the success of such enterprise would result in enormous benefits for grammar development and maintenance.

Several approaches have been proposed in the literature describing compact representations methods for LTAGs, of which, perhaps the best known are (Vijay-Shanker and Schabes, 1992), (Candito, 1996; Candito, 1998), (Evans, Gazdar and Weir, 1995; Evans, Gazdar and Weir, 2000), (Xia *et al.*, 1998; Xia, 2001), and (Becker, 1993; Becker, 1994; Becker, 2000). We describe in this paper how we combined Becker's metarules with a hierarchy of rule application to generate the verb tree templates of the XTAG English grammar, from a small initial set of trees.

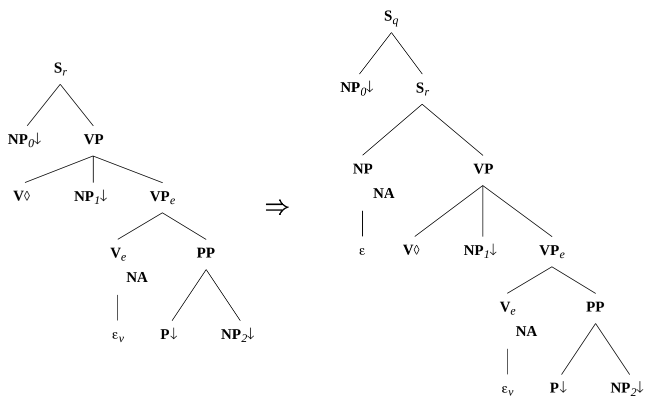
2. Metarules

A subsystem for interpreting metarules was initially introduced into the XTAG development system by Tilman Becker, from 1993 to 1995, based on his ideas in (Becker, 1993; Becker, 1994) with subsequent additions over the years, reaching a stable form as documented (by this author) in (XTAG Research Group, 1998). Although it has been topically used since then, as an auxiliary tool to reduce the effort spent in grammar development (e.g., to generate the trees for an updated analysis of relative clauses, using the former trees as starting point), this paper describes the first effort to effectively use them to generate the full XTAG grammar verb trees.[2]

* We are indebted to all members of the XTAG Group that participated of the valuable discussions during the realization of this work, and in particular to Alexandra Kinyon for her comments on this paper.

1. For a more accurate description of the architecture, see (XTAG Research Group, 2001) or (Doran *et al.*, 2000).
2. This effort is already mentioned in (Doran *et al.*, 2000, p. 388). There has been some confusion on the issue, perhaps driven by a somewhat ambiguous statement in (Becker, 2000, p. 331): "In this paper, we present the various patterns which are used in the implementation of metarules which we added to the XTAG system (Doran et al. 2000)". The work of Becker conceived and developed the idea of metarules for TAGs and also created the original implementation of the metarule interpreter as part of the XTAG software. However, he only created the necessary example patterns to support the concepts of metarules while the work described here is the first to actually evaluate metarules in-the-large as part of the XTAG project.

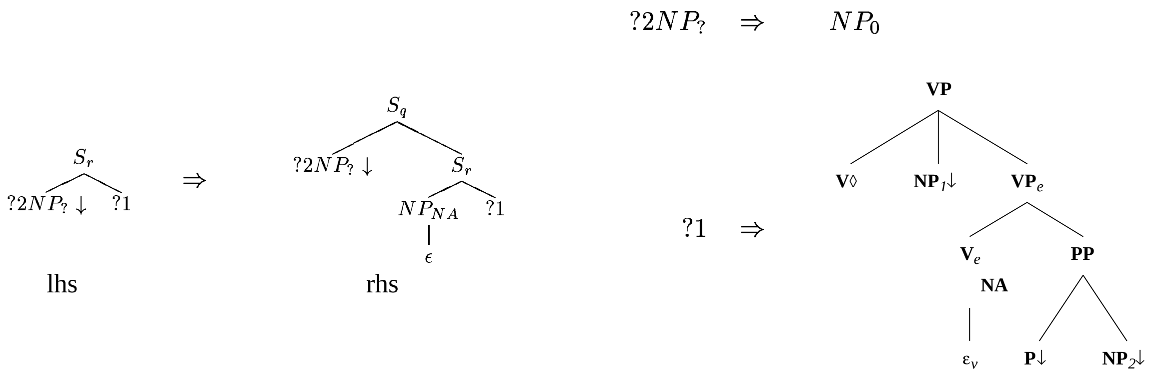
We present in this section a brief introduction to Becker's metarules.[3] Consider the trees in Figure 1 anchored by verbs that take as arguments an NP and a PP (e.g., *put*). The one to the left corresponds to its declarative structure; the other to the wh-subject extracted form. Despite their complexity, they share most of their structure: the only differences being the wh-site (higher NP in the right tree) and the trace at subject position. That observation would not be very useful if the differential description we have made was idiosyncratic to this pair, which is not the case. Clearly, many other pairs all over the grammar will share the same differential description.



(a) declarative (b) subject extracted

Figure 1: Some related trees for the verb put

Figure 2.a shows a metarule for wh-subject extraction, that captures the similarities mentioned above. It describes how to automatically generate the tree in Figure 1.b, given as input the tree in Figure 1.a. Here is how it works. First the input tree has to match the *left-hand side* of the metarule, *lhs* in Figure 2.a, starting from their roots. In the example the lhs tree, requires the candidate tree to have its root labeled S_r. Then, its leftmost child has to be an NP, as indicated by the node $?2NP_?$ in lhs: $?2$ indicates it is the variable $\#2$; $NP_?$ indicates we need an NP, regardless of the subscript. Next, the lhs tree requires the rest of the tree to match variable $?1$. That is trivial, because such variables with just an identification number are "wild cards" that match any range of subtrees. The matches of each variable in lhs, for the application to the input tree in Figure 1.a, are shown in Figure 2.b.



a) Metarule for wh-movement of subject b) Variables Matching for the tree in Figure 1.a

Figure 2: A metarule for wh-movement of subject applied to the tree of Figure 1.a

Had the matching process failed the metarule would not apply and no new tree would have been generated. Since in the example above the matching succeeded, the processor move to the final step, which is to generate the new tree. We look at the *right-hand side* of the metarule *rhs* and just replace the instances of the variables there

3. For a more comprehensive view, including linguistic motivations and the sort of patterns it allows, we refer the reader to (Becker, 2000). The actual set of metarules we used can be obtained upon request to this author.

SUBCATEGORIZATION GROUP	No. OF FAMS.	TOTAL No. OF TREES
Intransitive	1	12
Transitive	1	39
Adjectival complement	1	11
Ditransitive	1	46
Prepositional complement	4	182
Verb particle constructions	3	100
Light verb constructions	2	53
Sentential Complement (full verb)	3	75
Sentential Subject (full verb)	4	14
Idioms (full verb)	8	156
Small Clauses/Predicative	20	187
Equational ”be”	1	2
Ergative	1	12
Resultatives	4	101
It Clefts	3	18
Total	57	1008

Table 1: Current XTAG Grammar Coverage

with their matched values, obtaining the tree in Figure 1.b. The same process can be applied for the many other pairs related by the same metarule.

Variables like ?1 above, with no category specification, are indeed more powerful than the above example allow us to see. For instance, they allow us to express dominance relations. Additionally a single metarule application may result multiple matchings and therefore multiple output trees.

In a feature-based grammar as the one we are focusing on, to create tree structures without the proper feature equations is of little use. On the other hand, experience has shown that feature equations are much harder to maintain correct and consistent in the grammar than the tree structures. The XTAG metarules use features in two ways: as matching requirements, and for transformation purposes. Both positive and negative matching can be specified (that is, one can state that match will happen only if the input tree does not have a certain feature equation). Regarding the transformations, feature equations can be inserted, deleted, maintained, or modified, when generating the new tree from the matched one. A few imperative commands have proved very useful, e.g. “replace all NPs with index 1 by NPs with index w in all equations”.

3. A hierarchy for the application of the metarules

The set of verbal trees can be seen as a subset of the Cartesian product of three dimensions: subcategorization (e.g., transitive, intransitive), redistribution (e.g., passive), and realization (e.g., wh-subject movement) – discounted, of course, combinations blocked by linguistic constraints (e.g., there can not be object movement in intransitives). The verb trees in the XTAG English grammar are organized in families that roughly reflect a subcategorization frame. Hence, each family contains trees for each combination of redistribution and realization alternatives compatible with the subcategorization frame. The *base* tree of a family is the one corresponding to its declarative usage (no redistribution, arguments in canonical position). Table 1 summarizes the current coverage of the XTAG English grammar. The grouping of the families is just for presentational convenience.

Becker(1993; 1994; 2000) proposes that a grammar is the closure of the set of base trees under metarule application, raising a heated discussion on the unboundedness of the process of recursive application. We understand the issue is artificial and we show in this section that a simple ordering mechanism among the metarules suffices.[4]

Our strategy for generation of the verbal trees is as follows. There is a unique ordered set of 21 metarules (Table 2). For each family, we start with the base, declarative tree, apply the sequence of metarules, and the result is the whole family of trees. The sequence of metarules are applied in a way we call cumulative mode of application represented in Figure 3. The generated set start with the declarative tree. The first metarule is applied to the set, generating new trees, which are themselves included in the generated set. Then the second rule is applied, and so on, until the sequence is finished.

Redistribution rules are applied before realization rules. It is usual for a metarule to fail to apply to many of the already generated trees. Partly, this is due to the obvious fact that not all rules are compatible with any given

4. Notice that in the context of TAGs, metarules are used “off-line” to generate a finite grammar, a bounded process, which is radically different from their use in the Transformational Grammar tradition or in any other “on-the-fly” environment.

Metarule	Description
passive passive-fromPP	Generate the passive form Passive form for PP complements: ”The results were accounted for by the theory”
dropby	Passive without by-clause
gerund	Trees for NPs like in ”John eating cake (is unbelievable)”
wh-subj wh-sentsubj	Wh-subject movement Wh-subject movement for sentential subjects
wh-npobj wh-smallnpobj wh-apobj wh-advobj wh-ppobj	NP extraction from inside objects NP extraction from inside objects for small clauses AP complement extraction ADVP complement extraction PP complement extraction
rel-adj-W rel-adj-noW rel-subj-W rel-subj-noW rel-subj-noW-forpassive rel-obj-W rel-obj-noW rel-ppobj	Adjunct relative clause with wh-NP Adjunct relative clause with (possibly empty) complementizer Subject relative clause with wh-NP Subject relative clause with complementizer Subject relative clause with complementizer for passives NP Object relative clause with wh-NP NP Object relative clause with complementizer PP Object relative clause
imperative	Imperative
PRO	PRO Subject

Table 2: Metarules used to generate the verb families of the XTAG English Grammar

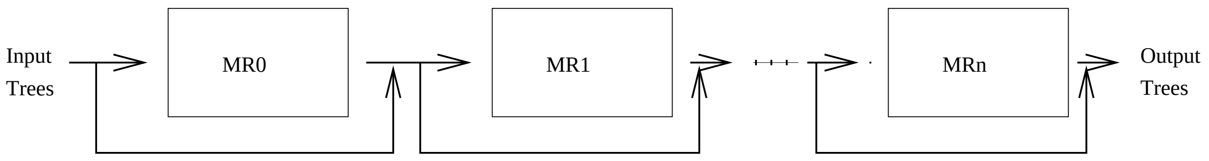


Figure 3: Cumulative application of metarules

subcategorization frame or after a redistribution metarule has been applied to it. But also, because the linear order is a simplification of what in fact should be a partial order, e.g. subject relativization should not apply to a wh-subject extracted tree. The constraints expressed in the metarules are responsible for blocking such applications.

We chose one of the largest families, with 52 trees, for verbs like *put* that take both an NP and a PP as complements, to detail the process of generation. For the sake of simplicity we omit the 26 relative clause trees. The remaining 25 trees [5] are described in Table 3, and the generation graph is shown in Figure 4. Numbers assigned to the trees in the Table are used to refer to them in the Figure.

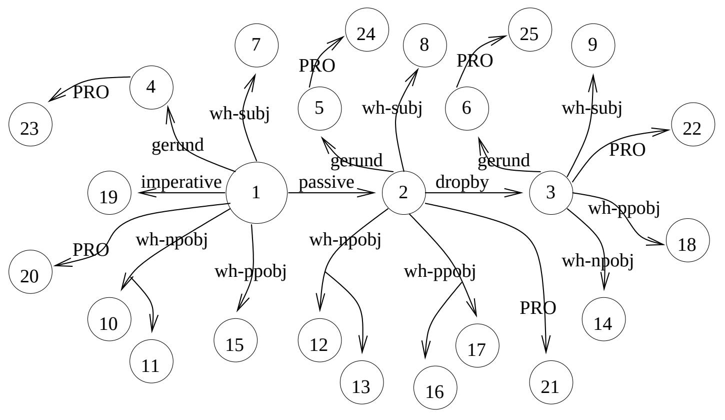


Figure 4: Partial generation of the *put* family using Metarules

5. There is one tree, for nominalization with determiner, we have found not worth generating. We comment on that ahead.

Number	Description	Example
1	Declarative	He put the book on the table
2	Passive w. by	The book was put on the table by him
3	Passive w.o. by	The book was put on the table
4	Gerundive nominals	He putting the book on the table was unexpected
5	Gerundive for passive w. by	The book being put on the table by him ...
6	Gerundive for passive w.o. by	The book being put on the table ...
7	Subject extraction	Who put the book on the table ?
8	Subj. extr. from passive w. by	What was put on the table by him ?
9	Subj. extr. from passive w.o. by	What was put on the table ?
10	1st obj extraction	What did he put on the table ?
11	2nd obj NP extraction	Where did he put the book on ?
12	2nd obj NP extr. from pass. w. by	Where was the book put on by him ?
13	Agent NP extr. from pass. w. by	Who (the hell) was this stupid book put on the table by ?
14	2nd obj NP extr. from pass. w.o. by	Where was the book put on ?
15	PP obj extr	On which table did he put the book ?
16	PP obj extr. from pass. w. by	On which table was the book put by him ?
17	By-clause extr. from pass. w. by	By whom was the book put on the table ?
18	PP obj extr. from pass. w.o. by	On which table was the book put ?
19	Imperative	Put the book on the table !
20	Declarative with PRO subject	I want to [PRO put the book on the table]
21	Passive w. by w. PRO subject	The cat wanted [PRO to be put on the tree by J.]
22	Passive w.o. by w. PRO subject	The cat wanted [PRO to be put on the tree]
23	Gerundive nominals with PRO subject	John approved of [PRO putting the cat on the tree]
24	Gerundive nominals for passive w. by w. PRO subject	The cat approved of [PRO being put on the tree by J.]
25	Gerundive nominals for passive w.o. by w. PRO subject	The cat approved of [PRO being put on the tree]

Table 3: Partial view of the trees from the *put* family

4. Evaluation and final remarks

An important methodological issue is that the grammar was generated towards a pre-existent English Grammar. So we can claim that the evaluation was quite accurate. Differences between the generated and pre-existent trees had to be explained and discussed with the group of grammar developers. Often this led to the discovery of errors and better ways of modeling the grammar. The purpose of this work was to generate the 57 verb families (1008 trees) from only the corresponding 57 declarative trees (or so) plus 21 metarules, a quite compact initial set. More importantly a compact set that can be effectively used for grammar development.[6] We turn now to a short inventory of problems found as well as some interesting observations:

1. We undergenerate:
   (a) There are about 20 idiosyncratic trees not generated, involving trees for "-ed" adjectives, restricted to transitive and ergative families, and Determiner Gerund trees, which lack a clear pattern across the families.[7] These trees should be separately added to the families. Similarly, there are 10 trees involving punctuation in the sentential complement families which are not worth generating automatically.
   (b) We overlooked the it-cleft families with peculiar tree structures, and the equational *be* family with two trees.
   (c) We do not handle yet: the passivization of the second object (from inside a PP) in families for idiomatic expressions ("The warning was taken heed of"); the occurrence of the "*by phrase*" before sentential complements ("I was told by Mary that ..."); and wh-extraction of sentential complements and of exhaustive PPs. Except for the first case all can be easily accounted for.
2. We overgenerate: we generate 1200 trees (instead of 1008).[8] However things are not so bad as they look at first: 206 of them are for passives related to idioms and they are fruit of some pragmatism in the group. It is acknowledged the existence of a certain amount of overgeneration in the tree families due to the separation between the lexicon and the tree templates. For instance, it is widely known that not all transitive verbs can undergo passivization. But the transitive family contains passive trees. The reconciliation can be made through

6. Of course we would not be very happy with a compact representation resembling a "zipped" file.
7. For instance, the nominalization of the transitive verb *find* selects a prepositional complement introduced by the preposition *of*: "The finding of the treasure (by the pirates) was news for weeks." But the "of" insertion is not uniform across families: cf. "the accounting for the book."
8. Which means more than an excess of 192 trees since there is also some undergeneration, already mentioned.

features assigned to verbs that allow blocking the selection of the particular tree. However in the family for verb particle with two objects (e.g., for “John opened up Mary a bank account”), the four lexical entries were judged not to undergo passivization and the corresponding trees (64) were omitted from the family. It is not surprising then that the metarules overgenerate them. Still, 100 out of the 206 are for passives in idiom families which are currently not in the grammar and are definitely lexically dependent. The other 42 overgenerated passives are in the light verb families. There a few other cases of overgeneration due to lexically dependent judgments, not worth detailing. Finally, a curious case involved empty elements that could be generated at slightly different positions that are not distinguished at surface (e.g., before or after a particle). The choice for having only one alternative in the grammar is of practical nature (related to parsing efficiency) as opposed to linguistic.

3. Limitations to further compaction: All the metarules for wh-object extraction do essentially the same, but currently they cannot be unified. Further improvements in the metarule system implementation could solve the problem at least partially, by allowing to treat symbols and indices as separate variables. A more difficult problem is some subtle differences in the feature equations across the grammar (e.g., causing the need of a separate tree for relativization of the subject in passive trees). By far, feature equations constitute the hardest issue to handle with the metarules.

4. A metarule shortcoming: currently they do not allow for the specification of negative structural constraints to matching. There is one feature equation related to punctuation that needed 5 separate metarules (not described above) to handle (by exhaustion) the following constraint: the equation should be added if and only if the tree has some non-empty material after the verb which is not a “by-phrase”.

5. Other cases: a separate metarule was needed to convert foot nodes into substitution nodes in sentential complement trees. This families departs from the rest of the grammar in that their base tree is an auxiliary tree to allow extraction from the sentential complement. But the corresponding relative clauses have to have the S complement as a substitution node. This is more an engineering than a conceptual problem.

References

Abeille, Anne and Owen Rambow, editors. 2000. *Tree Adjoining Grammars: formalisms, linguistic analysis and processing.* Stanford, CA, USA: CSLI.

Becker, Tilman. 1993. *HyTAG: A new Type of Tree Adjoining Grammars for Hybrid Syntactic Representation of Free Word Order Lang uages*. Ph.D. thesis, Universität des Saarlandes.

Becker, Tilman. 1994. Patterns in metarules. In *Proceedings of the 3rd TAG+ Conference*, Paris, France.

Becker, Tilman. 2000. Paterns in Metarules for TAG. In Abeille and Rambow (Abeille and Rambow, 2000), pages 331–342.

Candito, Marie-Helene. 1996. A Principle-Based Hierarchical Representation of LTAGs. In *Proceedings of the 16th CoLing (COLING'96)*, pages 194–199, Copenhagen, Denmark.

Candito, Marie-Helene. 1998. Building Parallel LTAG for French and Italian. In *Proceedings of the 36th Annual Meeting of the ACL and 16th CoLing*, pages 211–217, Montreal, Canada.

Doran, Christine, Beth Ann Hockey, Anoop Sarkar, B. Srinivas and Fei Xia. 2000. Evolution of the XTAG System. In Abeille and Rambow (Abeille and Rambow, 2000), pages 371–404.

Evans, Roger, Gerald Gazdar and David Weir. 1995. Encoding Lexicalized Tree Adjoining Grammars with a Nonmonotonic Inheritance Hierarchy. In *Proceedings of the 33rd Annual Meeting of the ACL*, pages 77–84, Cambridge, MA, USA.

Evans, Roger, Gerald Gazdar and David Weir. 2000. 'Lexical Rules' are just lexical rules. In Abeille and Rambow (Abeille and Rambow, 2000), pages 71–100.

Frank, Robert. 2001. *Phrase Structure Composition and Syntactic Dependencies.* to be published.

Joshi, Aravind K. 2001. The XTAG Project at Penn. In *Proceedings of the 7th International Workshop on Parsing Technologies (IWPT-2001)*, Beijing, China. Invited speaker.

Joshi, Aravind K. and Yves Schabes. 1997. Tree-Adjoining Grammars. In *Handbook of Formal Languages, vol. 3*. Springer-Verlag, Berlin, pages 69–123.

Krock, Anthony S. and Aravind K. Joshi. 1985. The linguistic relevance Tree Adjoining Grammar. Technical Report MS-CIS-85-16, University of Pennsylvania.

Vijay-Shanker, K. and Yves Schabes. 1992. Structure Sharing in Lexicalized Tree-Adjoining Grammars. In *Proceedings of the 14th CoLing (COLING'92)*, pages 205–211, Nantes, France.

Xia, Fei. 2001. *Investigating the Relationship between Grammars and Treebanks for Natural Languages.* Ph.D. thesis, Department of Computer and Information Science, University of Pennsylvania.

Xia, Fei, Martha Palmer, K. Vijay-Shanker and Joseph Rosenzweig. 1998. Consistent Grammar Development Using Partial-Tree Descriptions for Lexicalized Tree-Adjoining Grammars. In *Proceedings of the 4th International Workshop on Tree Adjoining Grammars (TAG+4)*, Philadelphia, USA.

XTAG Research Group, The. 1998. A Lexicalized Tree Adjoining Grammar for English. Technical Report IRCS 98-18, University of Pennsylvania.

XTAG Research Group, The. 2001. A Lexicalized Tree Adjoining Grammar for English. Technical Report IRCS 01-03, University of Pennsylvania.

A Formal Proof of Strong Equivalence for a Grammar Conversion from LTAG to HPSG-style

Naoki Yoshinaga,[†] Yusuke Miyao,[†] and Jun'ichi Tsujii[†‡]
† University of Tokyo ‡ CREST, JST (Japan Science and Technology Corporation)

1. Introduction

This paper presents a sketch of a formal proof of strong equivalence,[1] where both grammars generate equivalent parse results, between any LTAG (Lexicalized Tree Adjoining Grammar: Schabes, Abeille and Joshi (1988)) G and an HPSG (Head-Driven Phrase Structure Grammar: Pollard and Sag (1994))-style grammar converted from G by a grammar conversion (Yoshinaga and Miyao, 2001). Our proof theoretically justifies some applications of the grammar conversion that exploit the nature of strong equivalence (Yoshinaga *et al.*, 2001b; Yoshinaga *et al.*, 2001a), applications which contribute much to the developments of the two formalisms.

In the past decades, LTAG and HPSG have received considerable attention as approaches to the formalization of natural languages in the field of computational linguistics. Discussion of the correspondences between the two formalisms has accompanied their development; that is, their linguistic relationships and differences have been investigated (Abeillé, 1993; Kasper, 1998), as has conversion between two grammars in the two formalisms (Kasper *et al.*, 1995; Tateisi *et al.*, 1998; Becker and Lopez, 2000). These ongoing efforts have contributed greatly to the development of the two formalisms.

Following this direction, in our earlier work (Yoshinaga and Miyao, 2001), we provided a method for converting grammars from LTAG to *HPSG-style*, which is the notion that we defined according to the computational device that underlies HPSG. We used the grammar conversion to obtain an HPSG-style grammar from LTAG (The XTAG Research Group, 2001), and then empirically showed strong equivalence between the LTAG and the obtained HPSG-style grammar for the sentences in the ATIS corpus (Marcus, Santorini and Marcinkiewicz, 1994). We exploited the nature of strong equivalence between the LTAG and the HPSG-style grammars to provide some applications such as sharing of existing resources between the two grammar formalisms (Yoshinaga *et al.*, 2001b), a comparison of performance between parsers based on the two different formalisms (Yoshinaga *et al.*, 2001a), and linguistic correspondence between the HPSG-style grammar and HPSG. As the most important result for the LTAG community, through the experiments of parsing within the above sentences, we showed that the empirical time complexity of an LTAG parser (Sarkar, 2000) is higher than that of an HPSG parser (Torisawa *et al.*, 2000). This result is contrary to the general expectations from the viewpoint of the theoretical bound of worst time complexity, which is worth exploring further. However, the lack of the formal proof of strong equivalence restricts scope of the applications of our grammar conversion to grammars which are empirically attested the strong equivalence, and this prevents the applications from maximizing their true potential. In this paper we give a formal proof of strong equivalence between any LTAG G and an HPSG-style grammar converted from G by our grammar conversion in order to remove such restrictions on the applications.

2. Grammar conversion

We start by stating our definition of an HPSG-style grammar, and then briefly describe our algorithm for converting grammars from LTAG to HPSG-style. We hope that the reader will refer to the cited literature (Yoshinaga and Miyao, 2001) for a more detailed description.

We defined *an HPSG-style grammar*, the form of the output of our conversion, according to the computational architecture which underlies HPSG (Pollard and Sag, 1994). An HPSG-style grammar consists of *lexical entries* and *ID grammar rules*, each of which is described with typed feature structures (Carpenter, 1992). A lexical entry for a word must express the characteristics of the word, such as its subcategorization frame and grammatical category. An ID grammar rule must represent the constraints on the configuration of immediate constituency, and

* This research was funded by JSPS Research Fellowships for Young Scientists.

1. Chomsky (1963) first introduced the notion of strong equivalence between grammars, where both grammars generate the same set of structural descriptions (e.g., parse trees). Kornai and Pullum (1990) and Miller (1999) used the notion of isomorphism between sets of structural descriptions to provide the notion of strong equivalence across grammar formalisms, which we have adopted in our research.

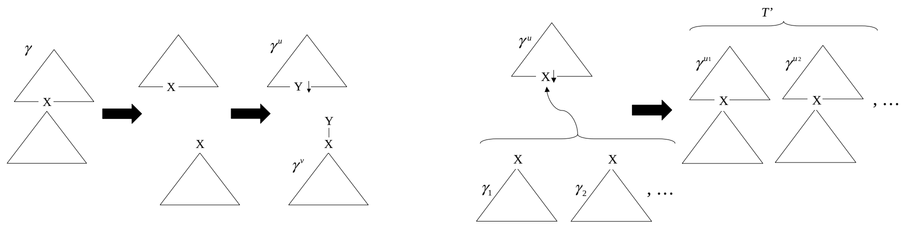


Figure 1: Sketch for the division transformation (left) and the substitution transformation (right)

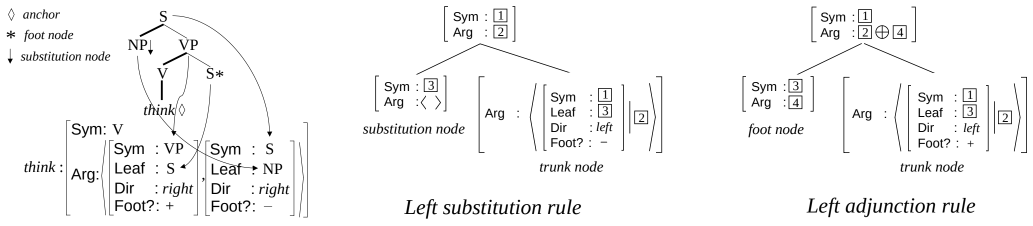


Figure 2: A conversion from a canonical elementary tree to an HPSG lexical entry (left) and grammar rules: the substitution rule (center) and adjunction rule (right)

not be a construction-specific rule specified by lexical characteristics. The formal definition of an HPSG-style grammar converted from LTAG G is given later in Section 3.3.

Our conversion algorithm consists of two kinds of conversion; i) a conversion from LTAG into *canonical LTAG*, LTAG which consists only of *canonical elementary trees*, and ii) a conversion from the canonical LTAG into an HPSG-style grammar. Canonical elementary trees are tree structures satisfy the following conditions; Condition 1: A tree must have only one anchor, and Condition 2: Every branching structure in a tree must contain trunk nodes. *Trunk nodes* are nodes on *a trunk* which is a path from an anchor to the root node. We call a subtree of depth $n(\geq 1)$ that includes no anchor *a non-anchored subtree*. Elementary trees which violate Condition 1 are converted into canonical ones by dividing them into single-anchored parts (*the division transformation*: the left-hand side of Figure 1). Elementary trees which violate Condition 2 are initially divided into multiple subtrees by the division transformation, each of which has at most one anchor, and then converted into canonical ones by substituting the deepest nodes in the non-anchored subtrees with every initial tree (*the substitution transformation*: the right-hand side of Figure 1). We give the formal definition of these transformations later in Section 3.2. Conversion of a canonical elementary tree is straightforward; that is, we traverse the trunk of a canonical elementary tree from its anchor to root, regard the leaf nodes as the anchor's arguments, and store the symbols of the leaf nodes and the trunk nodes as Leaf and Sym features respectively in a stack (Arg feature in the left-hand side of Figure 2), where Dir and Foot? features are the direction of the leaf node relative to the trunk and the type of the leaf node, respectively. A set of pre-determined rules manipulates the stack to emulate substitution and adjunction; namely, substitution rules (the center of Figure 2) and adjunction rules (the right-hand side of Figure 2).

3. A formal proof of strong equivalence

The whole proof consists of two pieces, each of which respectively proves that strong equivalence is guaranteed before and after the two conversions mentioned in the previous section.

3.1. Definitions

We first define LTAG, according to the definition of TAG given by (Vijay-Shanker, 1987). We then define *a derivation tree*, which is a structural description of LTAG, and introduce the notion of strong equivalence.

We hereafter denote a tree as a set of pairs $\langle p, X\rangle$ where $p \in \mathcal{N}^*$, which is a free monoid of the set of natural numbers, and $X \in V$, which is a finite set of alphabets (Gorn, 1962). For example, a tree in the left-hand side of Figure 2 is denoted as $\{(\epsilon, S), (\epsilon\cdot 1, NP)(\epsilon\cdot 2, VP), (\epsilon\cdot 2\cdot 1, V), (\epsilon\cdot 2\cdot 2, S), (\epsilon\cdot 2\cdot 1\cdot 1, think)\}$. An inequality $p \leq q$ is satisfied if and only if there is a $r \in \mathcal{N}^*$ such that $q = p\cdot r$. Another inequality $p < q$ is satisfied if and only if $p \leq q$ and $p \neq q$.

Definition 3.1 (Lexicalized Tree Adjoining Grammar (LTAG)) *Lexicalized Tree Adjoining Grammar G*[2] *is a quintuplet (Σ, NT, S, I, A) where Σ and NT are a finite set of terminal symbols and a finite set of nonterminal symbols respectively, S is a distinguished nonterminal symbol called the start symbol, and I and A are a finite set of initial trees and a finite set of auxiliary trees respectively.*

Here, an elementary tree $\gamma \in A \cup I$ is a tree whose leaf nodes are labeled by $X \in NT \cup S$ or $x \in \Sigma$, and whose internal nodes are labeled by $X \in NT \cup S$. The symbol of one leaf node in an auxiliary tree $\beta \in A$ is identical to that of its root node, and is specially marked for a foot node. Note that more than one leaf nodes called anchors in an elementary tree γ are labeled with $x \in \Sigma$, and leaf nodes other than anchors and foot nodes are marked for substitution nodes.

We denote adjunction and substitution of several trees $\gamma_1, \ldots, \gamma_k$ into a tree γ at k distinct addresses $a_1, \ldots, a_k$ by $\gamma' \to \gamma[a_1, \gamma'_1] \ldots [a_k, \gamma'_k]$ where $k \geq 1$, and $[a_i, \gamma'_i]$ indicates substitution at a_i of γ'_i in the case where a_i is a substitution node, or indicates adjunction at a_i of γ'_i in the case where a_i is an internal node. We call this production as *a derivation* for γ if all of the addresses of the substitution nodes in γ are included in $a_1, \ldots, a_k$. A derivation for γ without substitution and adjunction is denoted as $\gamma' \to \epsilon$.

We use the above notations to define *a derivation tree*, which represents the history of combinations of trees and is a structural description of LTAG.

Definition 3.2 (Derivation trees) *A derivation tree Υ_G for LTAG $G = (\Sigma, NT, S, I, A)$ is defined by a set of derivations as follows:*

$$\Upsilon_G = \{\gamma'_i \to \epsilon \mid 1 \leq i \leq m, \gamma_i \in A \cup I\} \bigcup D_G$$

where $D_G \subset \{\gamma'_i \to \gamma_i[a_1, \gamma'_{i_1}] \ldots [a_k, \gamma'_{i_k}] \mid k \geq 1, i > m, \gamma_i, \gamma_{i_j} \in A \cup I\}$. The derivation tree Υ_G must satisfy the following condition: γ'_i can appear once respectively in the left-hand side and the right-hand side of derivations except that one distinguished elementary tree γ_S, which is the root of the derivation tree Υ_G and γ'_S can appear once in the left-hand side of the derivation, because γ_i can adjoin or substitute once.[3] *Note that the inequality $i > i_j \geq 1$ is necessary to avoid cyclic applications of substitution and adjunction among elementary trees.*

Finally, we give the definition of strong equivalence between two given grammars G_1 and G_2.

Definition 3.3 (Strong equivalence) *Two given grammars G_1 and G_2 are strongly equivalent if and only if there is a bijective (i.e., one-to-one and onto) function which maps a set of structural descriptions of G_1, $T_D(G_1)$, to a set of structural descriptions of G_2, $T_D(G_2)$.*

In what follows, we assume that structural descriptions of LTAG are derivation trees in which the root node of γ_S is labeled by the start symbol S in the definition 3.2.

3.2. Proof of strong equivalence for the two tree transformations

In this section we give a proof that strong equivalence is guaranteed for grammars before and after the two tree transformations. In this abstract, We omit the proof of the substitution procedure, because the substitution transformation is exactly the same as the one that Schabes and Waters (1995, pp. 494–495) defined and proved in their strong lexicalization procedure of CFG into Lexicalized Tree Insertion Grammar.

The division transformation is formalized in the following lemma.

Lemma 3.1 (The division transformation) *Let $G = (\Sigma, NT, S, I, A)$ be LTAG. Let $\gamma \in A \cup I$ be an elementary tree and let μ be an internal node with address p of γ that is labeled by X and be not on the spine. We divide γ at μ and obtain two trees γ^u, γ^v as follows. Let γ^u be a subtree except that a node labeled by $Y \notin NT \cup S$ is added to its root node, and let γ^v be a supertree, except that the symbol of μ is relabeled by the symbol $Y \notin NT$ and by marking it for substitution as shown in Figure 1. Define $G' = (\Sigma, NT \cup \{Y\}, S, I', A')$ where I' and A' are created as follows:*

If $\gamma \in I$ then $I' = (I - \{\gamma\}) \cup \{\gamma^u, \gamma^v\}$ and $A' = A$
If $\gamma \in A$ then $I' = I \cup \{\gamma^v\}$ and $A' = (A - \{\gamma\}) \cup \{\gamma^u\}$

Then, G' is strongly equivalent to G; that is, there is a one-to-one onto mapping from the set of derivation trees $T_D(G')$ generated by G' to the set of derivation trees $T_D(G)$ generated by G for the same sentence.

2. Due to limitations of space, we omit the notion of adjoining constraints and the proof including the notion in this abstract, and then assume all internal nodes take selective adjoining constraints.
3. The condition implies that no trees can substitute or adjoin to two different nodes.

Proof We show that there is a one-to-one mapping from a derivation tree $\Upsilon_{G'} \in T_D(G')$ to a derivation tree $\Upsilon_G \in T_D(G)$.

Assume each derivation tree $\Upsilon_{G'}$ consists of elementary trees $\{\gamma_1, \ldots, \gamma_n\}$, $\gamma_j \in A \cup I$ for $1 \leq j \leq n$. Then, we can represent the derivation tree $\Upsilon_{G'}$ by the set of derivations as shown in the definition 3.2.

Since we assume that a derivation tree is rooted by an elementary tree whose symbol of the root node is S, every occurrence of γ^v in $\Upsilon_{G'}$ must always accompany with γ^u and vice versa. In the following procedure, we construct a one-to-one mapping from $\Upsilon_{G'}$ to Υ_G by replacing every occurrence of γ^u which takes a substitution of γ^v with γ in derivations in $\Upsilon_{G'}$.

1. When $\gamma^u \notin \{\gamma_1, \ldots, \gamma_n\}$ or $\gamma^v \notin \{\gamma_1, \ldots, \gamma_n\}$, $\Upsilon_{G'}$ includes neither γ^u nor γ^v. $\Upsilon_{G'}$ therefore consists of $\gamma_i \in (A \cup I - \{\gamma\}) \subset A \cup I$, there is exactly the same derivation tree Υ_G in $T_D(G)$.
2. When $\gamma^u \in \{\gamma_1, \ldots, \gamma_n\}$, we can construct one derivation tree Υ_G from Υ'_G as follows.
   (a) We first replace every occurrence of γ'^u in the right-hand side of derivations with γ'.
   (b) We next replace every derivation whose left-hand side is either γ'^u or γ'^v.
      i. When a root node with address ϵ of γ^v takes substitution or adjunction, a pair of two derivations whose left-hand side is γ'^u and γ'^v is denoted as $\gamma'^u \to \gamma^u[a_1, \gamma'_1] \ldots [a_{h-1}, \gamma'_{h-1}][p, \gamma'^v]$ and $\gamma'^v \to \gamma^v[\epsilon, \gamma'_h][b_{h+1}, \gamma'_{h+1}] \ldots [b_k, \gamma'_k]$, where $k > h \geq 1$. Here we assume $a_i \not\geq p$ for $1 \leq i < h$ without loss of generality. We replace these two derivations with the following derivation:

$$\gamma' \to \gamma[a_1, \gamma'_1] \ldots [a_{h-1}, \gamma'_{h-1}][p, \gamma'_h][p \cdot 1 \cdot b_{h+1}, \gamma'_{h+1}] \ldots [p \cdot 1 \cdot b_k, \gamma'_k]$$

      ii. If a root node with address ϵ in γ^v takes neither adjunction nor substitution, we can also replace a pair of two derivations whose left-hand side are respectively γ'^u and γ'^v with one derivation whose left-hand side is γ' in a similar way as above.
   (c) By repeating the above replacements at most the number of pairs of two derivations for γ^u and γ^v, we can obtain a set of derivations Υ_G without γ'^u and γ'^v. The replacement in (a) is valid since γ^u includes both root node and foot node of γ, and thus γ can substitute or adjoin every node at which γ^u does. In the procedure (b), we replace exactly the same number of γ'^u as the procedure (a). The resulting derivations including γ' is valid in G because γ' appear only once in the right-hand side and the left-hand side of the derivations, respectively.

The resulting derivation tree Υ_G is the same as $\Upsilon_{G'}$ except that every occurrences of γ^u which takes a substitution of γ^v with γ. Since γ^u which takes a substitution of γ^v is the same as γ except that one internal node is added, this does not cause effect on the frontier string. Also, when $\Upsilon^1_{G'}, \Upsilon^2_{G'}$ are mapped to $\Upsilon^1_G, \Upsilon^2_G$ and Υ^1_G and Υ^2_G are equivalent, $\Upsilon^1_{G'}$ and $\Upsilon^2_{G'}$ are also equivalent owing to the formulation of the above mapping.

On the other side, we can also construct a one-to-one onto mapping from Υ_G to $\Upsilon_{G'}$ by replacing every occurrence of γ in Υ_G by γ^u which takes a substitution of γ^v. Due to limitations of space, we omit the proof here.

In this way, we can construct a one-to-one onto mapping from a derivation tree $\Upsilon_{G'} \in T_D(G')$ to a derivation tree $\Upsilon_G \in T_D(G)$ for the same sentence. This indicates that G is strongly equivalent to G'. □

3.3. Proof of strong equivalence for the conversion from canonical LTAG to HPSG-style

In this section, we prove that strong equivalence is guaranteed for the latter part of our grammar conversion, that is, a conversion from canonical LTAG G to an HPSG-style grammar G'. In the following proof, we first introduce the notion of *origination* for every Sym and Leaf feature in HPSG lexical entries. We next define *an HPSG parse,* which is a structural description of an HPSG-style grammar. We then prove the strong equivalence by giving a one-to-one onto mapping from a derivation tree by G to an HPSG parse by G'.

Definition 3.4 (An HPSG-style grammar converted from LTAG) *Given canonical LTAG $G = (\Sigma, NT, S, I, A)$, an HPSG-style grammar G' converted from G is denoted by quituplet $(\Sigma, NT, S, \Delta, R)$ where $\delta_i \in \Delta$ is a lexical entry converted from $\gamma_i \in A \cup I$ and R is substitution and adjunction rules. δ_i is denoted as follows: $\delta_i = (s_0, (s_1, l_1, d_1, t_1), \ldots, (s_k, l_k, d_k, t_k))$ where $k \geq 1$, s_0 is the symbol of the mother node of the anchor in γ_i, and $s_j \in \Sigma \cup NT, l_j \in \Sigma \cup NT, d_j \in \{right, left\}, t_j \in \{+, -\}$ are values of Sym, Leaf, Dir, Foot? features in the j-th element of the Arg feature in δ_i. When the length of the Arg feature of δ_i is 0, δ_i is denoted as $\delta_i = (s_0, \phi)$.*

First, we introduce the notion of *origination* for the Sym and Leaf features in HPSG lexical entries in order to define *an HPSG parse*, which represents the histories of rule applications to lexical entries and is a structural description of an HPSG-style grammar. We hereafter assume that each HPSG lexical entry δ_i is converted from a canonical elementary tree γ_i. We define the origination of the feature in δ_i as $\langle p, \gamma_i \rangle$, which indicates that the value of the feature originates from the symbol of a node with address p in γ_i.

Next, we define *a rule history* for δ_i, which is a history of rule applications to a lexical entry δ_i in the parse tree. We then follow the parse tree from an anchor of δ_i to root, and then assign each rule application as an element of the rule history for δ_i if and only if the applied rule pops an element which originates from an element of the Arg feature in δ_i. Assume that δ_i is denoted as the one given in the definition 3.4. A rule history for δ_i is denoted as follows, where the origination of l_j and the feature unified with l_j are $\langle a_j, \gamma_i \rangle$ and $\langle b, \gamma_{i_j} \rangle$, respectively.

1. When $\gamma_i \in I$, no application of the adjunction rule is assigned to δ_i as an element of the rule history for δ_i. The rule history is then denoted as $\delta'_i \rightarrow \delta_i[a_1, \delta'_{i_1}] \ldots [a_k, \delta'_{i_k}]$.

2. When $\gamma_i \in A$, one application of the adjunction rule is assigned to δ_i as an element of the rule history for δ_i. The rule history for δ_i is then denoted as $\delta'_i \rightarrow \delta_i[a_1, \delta'_{i_1}] \ldots [a_{h-1}, \delta'_{i_{h-1}}][b, \delta_{i_h}][a_{h+1}, \delta'_{i_{h+1}}] \ldots [a_k, \delta'_{i_k}]$ where $t_h = +$.

When the length of the Arg feature of δ_i is 0, a rule history for δ_i is denoted by $\delta'_i \rightarrow \epsilon$.

Definition 3.5 (HPSG parses) *Given canonical LTAG $G = (\Sigma, NT, S, I, A)$ and an HPSG-style grammar $G' = (\Sigma, NT, S, \Delta, R)$ converted from G, an HPSG parse $\Psi_{G'}$ is denoted by a set of rule histories for $\delta_i \in \Delta$ as follows:*

$$\Psi_{G'} = \{\delta'_i \rightarrow \epsilon \mid 1 \leq i \leq m, \gamma_i \in I\} \bigcup A_{G'} \bigcup B_{G'}$$

where $A_{G'}$ is a set of rule histories for δ_i converted from $\gamma_i \in I$, and $B_{G'}$ is a set of rule histories for δ_i converted from $\gamma_i \in A$, and elements in $A_{G'}$ and $B_{G'}$ are denoted as the ones in the above paragraph where $i > m$.

Since the above HPSG parse Ψ_G must uniquely correspond to the parse tree, we require some conditions on Ψ_G. First, δ'_i where $\gamma_i \in I$ can appear once respectively in the left-hand side and the right-hand side of rule histories except that one distinguished lexical entry δ_S where δ'_S appears once in the left-hand side of the rule history for δ_S. Second, δ'_i where $\gamma_i \in A$ must appear only once in the left-hand side of the rule history for δ_i. Third, $1 \leq i_j < i$ for the rule history for δ_i where $\gamma_i \in I$. Fourth, $1 \leq i_j < i$ where $j \neq h$, and $i_h > i$, for the rule history for δ_i where $\gamma_i \in A$. The third and fourth conditions are necessary to avoid cyclic applications of grammar rules to lexical entries.

Lemma 3.2 *Let $G = (\Sigma, NT, S, I, A)$ and G' be LTAG and an HPSG-style grammar converted from G, respectively. Then, we can map a derivation tree Υ_G by G one-to-one onto to an HPSG parse $\Psi_{G'}$ by G'.*

Proof In the following proof, we first show a mapping from $\Psi_{G'}$ to a set of derivations $\Upsilon_{G'}$, and then show that $\Upsilon_{G'}$ is a valid derivation by G.

Suppose an HPSG parse denoted as the one given in the definition 3.5. We can map it to a set of derivations $\Upsilon_{G'}$ in the following procedure. For each δ_i where $\gamma_i \in A$, we eliminate $[b, \delta_{i_h}]$, which corresponds to an application of the adjunction rule, and add the element $[b, \delta'_i]$ to the right-hand side of the rule history for δ_{i_h}. Then, we obtain a set of derivations $\Upsilon_{G'}$ by replacing δ_{i_j} and δ'_{i_j} with γ_{i_j} and γ'_{i_j} in the rule history for δ_i and by assigning it as the derivation for γ_i. This mapping is one-to-one because a pair operation of an elimination of $[b, \delta_{i_h}]$ and an addition of $[b, \delta'_i]$ is one-to-one mapping.

Following the definition 3.2, we show that $\Upsilon_{G'}$ is a valid derivation tree by G. First, every substitution and adjunction in the derivations in $\Upsilon_{G'}$ must be valid in G. Since the substitution and adjunction rules preserve the order of the elements in the Arg feature of δ_i, substitution rules always unify the symbol of the substitution node with the symbol of the root node of γ_{i_j}, which represents the same constraint as the one on which substitution imposes. We can give the similar argument for an adjunction rule. The substitution and adjunction in the derivations in $\Upsilon_{G'}$ are then valid in G. Second, all addresses in the substitution nodes of γ_i must be included in its derivation. This is apparently guaranteed by the definition of the rule history for δ_i. Third, γ'_i can appear only once respectively in the right-hand side and the left-hand side of the derivations. This is apparently guaranteed for γ'_i where $\gamma_i \in I$ by the definition 3.5, and is guaranteed for γ'_i where $\gamma_i \in A$ because δ'_i does not appear in the right-hand side of rule histories, $[b, \delta_{i_h}]$ appears only once in the rule history for δ_i, and the elimination of $[b, \delta_{i_h}]$ accompanies the addition of $[b, \gamma'_i]$ once to the right-hand side of the derivation for γ_{i_h}. Fourth, the elements in the right-hand side of the derivation for γ_i must be $[a_j, \gamma'_{i_j}]$ where $i_j < i$. This is apparently guaranteed for γ'_i where $\gamma_i \in I$ by the definition 3.5, and is guaranteed for γ'_i where $\gamma_i \in A$ because the addition of $[b, \gamma'_i]$ for the derivation for γ'_{i_h} satisfy $i_h > i$ due to the definition 3.5.

The frontier string is preserved before and after this mapping from $\Psi_{G'}$ to $\Upsilon_{G'}$, because δ_i stores the same LP constraints between δ_i and δ_j for $i \neq j$ as the constraints between γ_i and γ_j. Then, an HPSG parse $\Psi_{G'}$ by G' mapped one-to-one to a derivation tree $\Upsilon_{G'}$ which is valid in G.

On the other side, we can construct a mapping from Υ_G to an HPSG parse Ψ_G as the inverse procedure for the above mapping from $\Psi_{G'}$ to $\Upsilon_{G'}$. The obtained Ψ_G is a valid HPSG parse by G' because we can give a similar argument for the validity of the rule histories in Ψ_G. $\square$

Hence, strong equivalence is guaranteed for a conversion from canonical LTAG to an HPSG-style grammar. The two proofs given here and in the previous section prove the strong equivalence between any LTAG G and an HPSG-style grammar converted from G by our grammar conversion.

4. Conclusion

In this research, we proved that strong equivalence is guaranteed between any LTAG grammar G and an HPSG-style grammars converted from G by our grammar conversion. Our proof theoretically justifies some applications of the grammar conversion that exploit the nature of strong equivalence (Yoshinaga *et al.*, 2001b; Yoshinaga *et al.*, 2001a), applications which contribute much to the developments of the two formalisms.

References

Abeillé, Anne. 1993. *Les nouvelles syntaxes: grammaires d'unification et analyse du français*. Armanda Colin. in French.

Becker, Tilman and Patrice Lopez. 2000. Adapting HPSG-to-TAG compilation to wide-coverage grammars. In *Proc. of TAG+5*, pages 47–54.

Carpenter, Bob. 1992. *The Logic of Typed Feature Structures*. Cambridge University Press.

Chomsky, Noam. 1963. Formal properties of grammar. In R. D. Luce, R. R. Bush and E. Galanter, editors, *Handbook of Mathematical Psychology*, volume II. John Wiley and Sons, Inc., pages 323–418.

Gorn, Saul. 1962. Processors for Infinite Codes of Shannon-Fano type. In *Proc. of the Symposium on Mathematical Theory of Automata*, pages 223–240.

Kasper, Robert. 1998. TAG and HPSG. Talk given in the tutorial session at TAG+4.

Kasper, Robert, Bernd Kiefer, Klaus Netter and K. Vijay-Shanker. 1995. Compilation of HPSG to TAG. In *Proc. of ACL 1995*, pages 92–99.

Kornai, A. and G. K. Pullum. 1990. The X-bar Theory of Phrase Structure. *Language*, 66:24–50.

Marcus, Mitchell, Beatrice Santorini and Mary Ann Marcinkiewicz. 1994. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Miller, Philip H. 1999. *Strong Generative Capacity*. CSLI publications.

Pollard, Carl and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press and CSLI Publications.

Sarkar, Anoop. 2000. Practical Experiments in Parsing using Tree Adjoining Grammars. In *Proc. of TAG+5*, pages 193–198.

Schabes, Yves, Anne Abeille and Aravind K. Joshi. 1988. Parsing strategies with 'Lexicalized' grammars: Application to Tree Adjoining Grammars. In *Proc. of COLING 1992*, pages 578–583.

Schabes, Yves and Richard C. Waters. 1995. Tree Insertion Grammar: A Cubic-Time Parsable Formalism that Lexicalizes Context-Free Grammar without Changing the Tree Produced. *Computational Linguistics*, 21(4):479–513.

Tateisi, Yuka, Kentaro Torisawa, Yusuke Miyao and Jun'ichi Tsujii. 1998. Translating the XTAG English Grammar to HPSG. In *Proc. of TAG+4*, pages 172–175.

The XTAG Research Group. 2001. A Lexicalized Tree Adjoining Grammar for English. http://www.cis.upenn.edu/~xtag/.

Torisawa, Kentaro, Kenji Nishida, Yusuke Miyao and Jun'ichi Tsujii. 2000. An HPSG Parser with CFG Filtering. *Natural Language Engineering*, 6(1):63–80.

Vijay-Shanker, K. 1987. *A Study of Tree Adjoining Grammars*. Ph.D. thesis, Department of Computer & Information Science, University of Pennsylvania.

Yoshinaga, Naoki and Yusuke Miyao. 2001. Grammar conversion from LTAG to HPSG. In *Proc. of the Sixth ESSLLI Student Session*, pages 309–324.

Yoshinaga, Naoki, Yusuke Miyao, Kentaro Torisawa and Jun'ichi Tsujii. 2001a. Efficient LTAG parsing using HPSG parsers. In *Proc. of Pacific Association for Computational Linguistics (PACLING 2001)*, pages 342–351.

Yoshinaga, Naoki, Yusuke Miyao, Kentaro Torisawa and Jun'ichi Tsujii. 2001b. Resource sharing among HPSG and LTAG communities by a method of grammar conversion from FB-LTAG to HPSG. In *Proc. of ACL/EACL 2001 Workshop on Sharing Tools and Resources for Research and Education*, pages 39–46.

Parsing MCS Languages with Thread Automata

Éric Villemonte de la Clergerie
INRIA

1. Introduction

Generalizing ideas presented for 2-stack automata in (Éric Villemonte de la Clergerie, 2001), we introduce *Thread Automata* [TA], a new automata formalism that may be used to describe a wide range of parsing strategies (in particular top-down prefix-valid [pv] strategies) for many Mildly-Context Sensitive [MCS] grammatical formalisms (Weir, 1988), including CFG, TAG (Joshi, 1987), Multi-Component TAG (Weir, 1988), and Linear Context-Free Rewriting Systems [LCFRS] (Weir, 1992).

As suggested by their name, the underlying idea of TA is that several lines of computation (*threads*) are followed during parsing, only one being active at any time. Threads may start sub-threads, may terminate, and may be suspended to give control to their parent or one of their direct descendants.

Intuitively, a thread may be used to recognize a constituent while new sub-threads are started to recognize its sub-constituents. Because a thread may be suspended and resumed several times, we can recognize *discontinuous* constituents with *holes* such as auxiliary trees in TAG. More generally, TA may handle complex interleaving of discontinuous constituents as shown by Fig. 4(a) for the constituents B and C. TA may also be used to parse a sub-class of Range Concatenation Grammars [RCG] (Boullier, 2000b), which covers LCFRS.

Though TA exhibit strong expressive power, they still ensure good operational and complexity properties. Indeed, we propose a simple *Dynamic Programming* [DP] interpretation for TA that ensures tabular parsing in polynomial worst-case complexity for space and time w.r.t. the length of the input string.

If we focus in this paper on top-down pv parsing strategies, it is not because we believe them to be the most important ones, but rather because we think it is important to cover the full spectrum of parsing strategies. Moreover, a tabular parser which handles pv parsing strategies may usually be easily adapted to handle other kinds of strategies, the converse being not true. For instance, there already exists a systematic non pv parsing algorithm for RCG, but we are unaware of any systematic pv parsing algorithm for them.

2. Thread Automata

Formally, a Thread Automaton is a tuple $(\mathcal{N}, \Sigma, S, F, \kappa, \mathcal{K}, \delta, \mathcal{U}, \Theta)$ where

- Σ (resp. $\mathcal{N}$) denotes a finite set of terminal (resp. non-terminal) symbols, with two distinguished initial and final non-terminals S and F;
- Θ is a finite set of transitions;
- κ denotes a partial function from $\mathcal{N}$ to some other finite set $\mathcal{K}$ and is used to capture the amount of information consulted in a thread to trigger the application of some kinds of transitions; [1]
- $\mathcal{U}$ is a finite set of labels used to identify threads. It is also used by the partial function δ to *drive* computations by specifying which threads (subthreads or parent) may potentially be created or resumed at some point, δ being defined from $\mathcal{N}$ to 2^{Δ} where $\Delta = \{\perp\} \cup \mathcal{U} \cup \{u^s | u \in \mathcal{U}\}$ and $\perp \notin \mathcal{U}$. The two functions κ and δ being often consulted in conjunction, we note $\kappa\delta(A) = (\kappa(A), \delta(A))$ and $(a, d) \in \kappa\delta(A)$ if $a = \kappa(A)$ and $d \in \delta(A)$.

A *thread* is a pair $p{:}A$ where $p = u_1 \dots u_k \in \mathcal{U}^\star$ is a (possibly empty) path, and A some non-terminal symbol from $\mathcal{N}$. The empty path is denoted by ϵ. A *thread store* $\mathcal{S}$ is a finite set of threads (denoting a partial function from $\mathcal{U}$ to $\mathcal{N}$) such that its associated path set $\ker(\mathcal{S}) = \{p | p{:}A \in \mathcal{S}\}$ is closed by prefix (ie., $pu \in \ker(\mathcal{S}) \Rightarrow p \in \ker(\mathcal{S})$). We will often confuse a thread with its path.

A TA configuration is a tuple $\langle l, p, \mathcal{S} \rangle$ where l denotes the current position in the input string, p the path of the active thread, and $\mathcal{S}$ a thread store with $p{:}A \in \mathcal{S}$ for some non-terminal A. The initial configuration is

1. This function, while not essential, is useful to reduce complexity w.r.t. grammar sizes, as illustrated for TAGs (Section 3).

$c_{\text{init}} = \langle 0, \epsilon, \{\epsilon{:}S\}\rangle$ and the final one $c_{\text{final}} = \langle n, u, \{\epsilon{:}S, u{:}F\}\rangle$ where $u \in \delta(S) \cap \mathcal{U}$ and n denotes the length of the input string. A derivation step $c \vdash_{\tau} c'$ is performed using a transition $\tau \in \Theta$ of the following kind, where bracketed (resp. non-bracketed) parts denote contents of non active (resp. active) threads :

SWAP $B \overset{\alpha}{\longmapsto} C$: Changes the content of the active thread, possibly scanning a terminal.

$$\langle l, p, \mathcal{S} \cup p{:}B\rangle \vdash_{\tau} \langle l + |\alpha|, p, \mathcal{S} \cup p{:}C\rangle \qquad a_l = \alpha \text{ if } \alpha \neq \epsilon$$

PUSH $b \longmapsto [b]\, C$: Creates a new subthread.

$$\langle l, p, \mathcal{S} \cup p{:}B\rangle \vdash_{\tau} \langle l, pu, \mathcal{S} \cup p{:}B \cup pu{:}C\rangle \qquad (b, u) \in \kappa\delta(B) \wedge pu \notin \ker(\mathcal{S})$$

POP $[B]\, C \longmapsto D$: Terminates thread pu (if there is no existing subthread).

$$\langle l, pu, \mathcal{S} \cup p{:}B \cup pu{:}C\rangle \vdash_{\tau} \langle l, p, \mathcal{S} \cup p{:}C\rangle \qquad pu \notin \ker(\mathcal{S})$$

SPUSH $b\, [C] \longmapsto [b]\, D$: Resumes the subthread pu (if already created)

$$\langle l, p, \mathcal{S} \cup p{:}B \cup pu{:}C\rangle \vdash_{\tau} \langle l, pu, \mathcal{S} \cup p{:}B \cup pu{:}D\rangle \qquad (b, u^s) \in \kappa\delta(B)$$

SPOP $[B]\, c \longmapsto D\, [c]$: Resumes the parent thread p of pu

$$\langle l, pu, \mathcal{S} \cup p{:}B \cup pu{:}C\rangle \vdash_{\tau} \langle l, p, \mathcal{S} \cup p{:}D \cup pu{:}C\rangle \qquad (c, \bot) \in \kappa\delta(C)$$

Without restrictions, a thread may be suspended infinitely often. However, we will only consider h-TA, subclasses of TA where a thread may be suspended at most h times to return to its parent, which is a sufficient condition to ensure the termination of our tabular parsing algorithm (Section 6). Another key parameter is $d \leq |\mathcal{U}|$, the maximal number of subthreads of a thread that may be simultaneously alive. These two parameters h and d are sufficient to characterize the worst-case complexities of our tabular algorithm, namely $O(n^{2(1+h+dh)})$ for space and $O(n^{2(1+h+dh)+1})$ for time. For instance, TA with $h = 0$ and $d = 1$ are equivalent to Push-Down Automata and may be used to handle CFG, giving us optimal worst-case complexity $O(n^2)$ for space and $O(n^3)$ for time.

A finer complexity analysis (still to be confirmed) suggests that, modulo a generally satisfied condition on SWAP transitions[2], the worst-case complexities are actually $O(n^{2+h+dh+x})$ for space and $O(n^{3+h+dh+x})$ for time where $x = \min(h + dh, (l - d)(h + 1))$ and l is the maximal number of subthreads created by a thread. When assuming $l = d$, we get $x = 0$ and complexities $O(n^{2+h+dh})$ for space and $O(n^{3+h+dh})$ for time.

Fig. 1 lists the transitions of a thread automaton $\mathcal{A}$ that may be used to recognize the **COUNT-3** language $a^n b^n c^n$ with a derivation of $a^3 b^3 c^3$ illustrated by Fig. 2. The characteristics of $\mathcal{A}$ are $h = 2$ and $d = 1$.

(1)	$K \overset{a}{\longmapsto} s_1$	$\kappa\delta(s_1) = (K, \{1\})$	(9)	$K \longmapsto [K]\, K$	
(2)	$[s_1]$ void $\longmapsto s_2$ [void]	$\kappa\delta(s_2) = (\text{void}, \{\bot\})$	(10)	$S \longmapsto r_0$	$\kappa\delta(r_0) = (K, \{1\})$
(3)	void $[s_2] \longmapsto$ [void] s_3		(11)	$[r_0]$ void $\longmapsto r_1$ [void]	$\kappa\delta(r_1) = (\text{void}, \{1^s\})$
(4)	$s_3 \overset{b}{\longmapsto} s_4$	$\kappa\delta(s_1) = (\text{void}, \{1^s\})$	(12)	$[r_1]$ void $\longmapsto r_2$ [void]	$\kappa\delta(r_2) = (\text{void}, \{1^s\})$
(5)	$[s_4]$ void $\longmapsto s_5$ [void]	$\kappa\delta(s_5) = (\text{void}, \{\bot\})$	(13)	$[r_2]$ ret $\longmapsto$ ret	
(6)	void $[s_5] \longmapsto$ [void] s_6		(14)	$K \longmapsto [K]\, t_0$	$\kappa\delta(t_0) = (\text{void}, \{\bot\})$
(7)	$s_6 \overset{c}{\longmapsto} s_7$	$\kappa\delta(s_7) = (\text{void}, \{1^s\})$	(15)	void $[t_0] \longmapsto$ [void] t_1	$\kappa\delta(t_1) = (\text{void}, \{\bot\})$
(8)	$[s_7]$ ret $\longmapsto$ ret		(16)	void $[t_1] \longmapsto$ [void] ret	

Figure 1: TA transitions for **COUNT-3** language $a^n b^n c^n$

3. Parsing TAG

TAG parsing strategies may be encoded with TA in a straightforward way. The idea is to associate a thread to each elementary tree traversal. For instance, in Fig. 3, a thread p is started at R to traverse some elementary tree

2. The condition is that no sequence of SWAP transitions can loop from a configuration $\langle l, p, \mathcal{S} \cup p{:}A\rangle$ to a configuration $\langle r, p, \mathcal{S} \cup p{:}A\rangle$. That means for instance that we are not scanning regular expressions with Kleene stars using SWAP transitions.

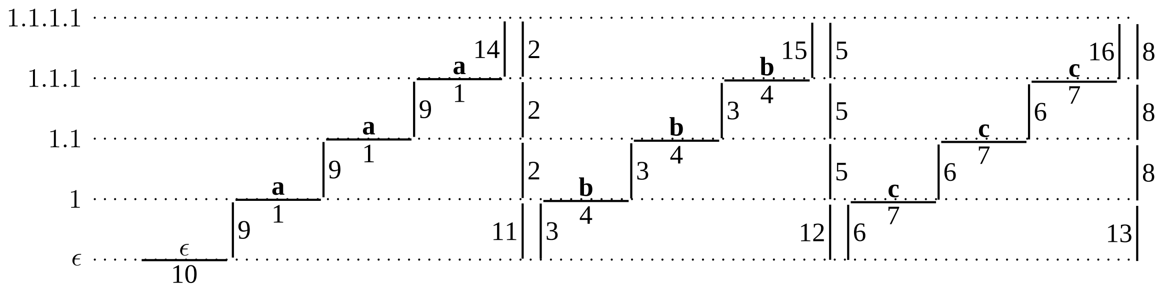
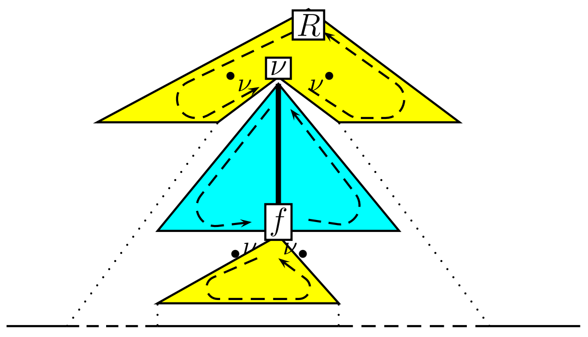


Figure 2: A derivation for $a^3b^3c^3$ starting from S

α and a subthread pu with $u = \mathrm{depth}(\nu)$ is started at node ν to handle an adjunction for non-terminal N (**Adj Call**). This subthread T selects and traverses some auxiliary tree β (**Adj Select**); is $\perp$-suspended when reaching the foot node of β to traverse the subtree $\alpha_{|\nu}$ rooted at ν (**Foot Suspend**); is resumed after the traversal of $\alpha_{|\nu}$ (**Foot Resume**); and is ended to return to its parent thread at the end of the traversal of β (**Adj Return**). [3]



(a) Traversals

Adj	$\kappa\delta({}^{\bullet}\nu) = (N^a, \{u\})$ $\kappa\delta(\nu_{\bullet}) = (\mathrm{adj}, \{u^s\})$ $u = \mathrm{depth}(\nu)$
Foot	$\kappa\delta({}_{\bullet}\mathrm{f}) = (\mathrm{foot}, \{\perp\})$

(Adj Call)	$N^a \longmapsto [N^a]\, N^a$
(Adj Select)	$N^a \longmapsto {}^{\bullet}\mathrm{r}_\beta$
(Foot Suspend)	$[{}^{\bullet}\nu]\ \mathrm{foot} \longmapsto {}_{\bullet}\nu\ [\mathrm{foot}]$
(Foot Resume)	$\mathrm{adj}\ [{}_{\bullet}\mathrm{f}] \longmapsto [\mathrm{adj}]\ \mathrm{f}_{\bullet}$
(Adj Publish)	$\mathrm{r}_\beta{}^{\bullet} \longmapsto \mathrm{ret}$
(Adj Return)	$[\nu_{\bullet}]\ \mathrm{ret} \longmapsto \nu^{\bullet}$

(b) Transitions

Figure 3: Traversing trees using TA transitions

When traversing a tree τ with a thread p, the maximal number d of simultaneously alive sub-threads of p is bounded by its depth, at most one subthread being started and still alive for each node along a path of τ. A thread may only be suspended once ($h = 1$). Hence we get complexities $O(n^{4+2d})$ for space and $O(n^{5+2d})$ for time. These complexities are not optimal for TAG but correspond to those mentioned in (Éric Villemonte de la Clergerie, 2001) for a very similar tabular parsing algorithm, which has proved to be efficient in practice for linguistic grammars.

The best known worst-case complexities for TAG are $O(n^4)$ or $O(n^5)$ for space and $O(n^6)$ for time, and correspond to tabular algorithms where the traversal of the subtree $\alpha_{|\nu}$ may be shared between the different traversals of α (relying on the fact that the traversal of $\alpha_{|\nu}$ may be done independently of the adjunctions started on the path from the root of α down to ν). It is worth noting that we have very good reasons to believe that we can also achieve the $O(n^6)$ time complexity using TA and our tabular parsing algorithm (see discussion in Section 5).

4. Parsing Multi-Component TAG

Multi-Component [MC] TAG allows the adjoining or substitution of a finite set of elementary trees on nodes of a single elementary tree (*tree-local* MC-TAG) or on nodes of elementary trees of another set (*set-local* MC-

3. Note that without a triggering function κ, the **(Foot Suspend)** transition would be of the form $[{}^{\bullet}\nu]\ {}_{\bullet}\mathrm{f} \longmapsto \nu_{\bullet}\ [{}_{\bullet}\mathrm{f}]$, explicitly referring to nodes of two distinct elementary trees, and leading to complexities in $O(|G|^2)$ instead of $O(|G|)$ where $|G|$ denotes the size of the grammar.

TAG). We provide some intuitions how TA may be used to encode prefix-valid parsing strategies for both kinds of MC-TAG, restricting ourselves to sets of auxiliary trees.[4]

Tree-local MC-TAG. The idea is to assign a thread p to the traversal (in any order) of a tree set Σ and subthreads of p to the traversal of each tree in Σ.

More formally, for the traversal in any order of a tree set $\Sigma = \{\beta_1, \dots, \beta_m\}$, we consider extended dotted points defined by

$$\Sigma{:}\rho\sigma \quad \rho \in \{\, {}^\bullet\mathrm{i},\ {}_\bullet\mathrm{i},\ \mathrm{i}_\bullet\, | i = 1 \dots m\}^\star \wedge \sigma \in \{\, \mathrm{i}^\bullet\, | i = 1 \dots m\}^\star$$

where ρ (resp. σ) is the list of trees in Σ which have been started but not yet completed (resp. completed). We note $\mathrm{ind}(\rho\sigma)$ the set of indices in $\{1, \dots, m\}$ occurring in $\rho\sigma$. The rightmost index of ρ states which tree of Σ we are currently working on.

The non-terminal set $\mathcal{N}$ of the automaton includes these extended dotted points $\Sigma{:}\rho\sigma$, as well as the dotted nodes for each node ν and the symbols N^a and N^s for each non-terminal N of the grammar. We consider the thread label set $\mathcal{U} = \{1, \dots, m\} \cup \{\mathrm{addr}(\tau, \nu) | \tau, \nu \text{ node of } \tau\}$ where $\mathrm{addr}(\tau, \nu)$ denotes the address of ν in τ.

We now associate to Σ the following (non exhaustive) set of transitions, where r_i denotes the root node of β_i and N_i the non-terminal label of r_i:

(Call set)	$N_i^a \longmapsto [N_i^a]\, N_i^a$	
(Start set with tree β_i)	$N_i^a \longmapsto \Sigma{:}\, {}^\bullet\mathrm{i}$	
(Resume set with tree β_i)	$N_i^a\, [\Sigma{:}\rho\sigma] \longmapsto [N_i^a]\, \Sigma{:}\rho\, {}^\bullet\mathrm{i}\, \sigma$	$i \notin \mathrm{ind}(\rho\sigma)$
(Start tree β_i)	$r_i \longmapsto [r_i]\ {}^\bullet\mathrm{r_i}$	$\kappa\delta(\Sigma{:}\rho\, {}^\bullet\mathrm{i}\, \sigma) = (r_i, \{i\})$
(Suspend β_i at foot)	$[\Sigma{:}\rho\, {}^\bullet\mathrm{i}\, \sigma]\, \mathrm{foot} \longmapsto \Sigma{:}\rho\, {}_\bullet\mathrm{i}\, \sigma\, [\mathrm{foot}]$	
(Suspend set at foot of β_i)	$[{}^\bullet\nu]\, \mathrm{foot} \longmapsto {}_\bullet\nu\, [\mathrm{foot}]$	$\kappa\delta(\Sigma{:}\rho\, {}_\bullet\mathrm{i}\, \sigma) = (\mathrm{foot}, \{\perp\})$
(Resume set after foot of β_i)	$\mathrm{adj}\, [\Sigma{:}\rho\, {}_\bullet\mathrm{i}\, \sigma] \longmapsto [\mathrm{adj}]\, \Sigma{:}\rho\, \mathrm{i}_\bullet\, \sigma$	
(Resume β_i after foot)	$\mathrm{adj}\, [{}_\bullet\mathrm{f}] \longmapsto [\mathrm{adj}]\ \mathrm{f}_\bullet$	$\kappa\delta(\Sigma{:}\rho\, \mathrm{i}_\bullet\, \sigma) = (\mathrm{adj}, \{i\})$
(End tree β_i)	$[\Sigma{:}\rho\, \mathrm{i}_\bullet\, \sigma]\, \mathrm{ret} \longmapsto \Sigma{:}\rho\, \mathrm{i}^\bullet\, \sigma$	
(Suspend set after tree β_i)	$[\nu_\bullet]\, \mathrm{ret} \longmapsto \nu^\bullet\, [\mathrm{ret}]$	$\kappa\delta(\Sigma{:}\rho\, \mathrm{i}^\bullet\, \sigma) = (\mathrm{ret}, \{\perp\}) \wedge \rho \neq \epsilon$
(End set)	$\Sigma{:}\sigma \longmapsto \mathrm{ret}$	$\mathrm{ind}(\sigma) = \{1, \dots, m\}$
(Return from set)	$[\nu_\bullet]\, \mathrm{ret} \longmapsto \nu^\bullet$	

For a node ν of some elementary tree τ with non-terminal label N, we set

Adj	$\kappa\delta({}^\bullet\nu) = (N^a, \{\mathrm{addr}(\tau, \nu)\} \cup \{\mathrm{addr}(\tau, \mu) \| \mu \neq \nu\}^s)$ $\kappa\delta(\nu_\bullet) = (\mathrm{adj}, \{\mathrm{addr}(\tau, \mu)\}^s)$
Foot	$\kappa\delta({}_\bullet\mathrm{f}) = (\mathrm{foot}, \{\perp\})$

In terms of complexity, the number h of $\perp$-suspensions is bounded by $2m$ where m denotes the maximal number of trees per set while d is bounded by $\frac{1}{2} \max_\tau |\tau|$ where $|\tau|$ is the size of τ (number of nodes). However, in our complexity analysis, we use dh to bound the number of suspensions of a thread due to its subthreads. For tree-local MC-TAG, one can check than we can replace dh by $2 \max_\tau |\tau|$ and get much better results.

Set-local MC-TAG. We consider a single thread to traverse all trees of a set Σ (without subthreads for trees of Σ), using extended dotted nodes of the form:

$$\Sigma{:}\rho\sigma \quad \rho \in (\{\, {}^\bullet\nu,\ {}_\bullet\nu,\ \nu_\bullet,\ \nu^\bullet\, | \nu \text{ a node of } \beta_i\} / \{\, \mathrm{r_1}^\bullet, \dots, \mathrm{r_m}^\bullet\})^\star \wedge \sigma \in \{\, \mathrm{r_1}^\bullet, \dots, \mathrm{r_m}^\bullet\}^\star$$

where ρ (resp. σ) gives indication about each uncompleted (resp. completed) traversal of trees in Σ. We note $\mathrm{ind}(\rho\sigma)$ the set of indices $i \in \{1, \dots, m\}$ for whose a dotted node of β_i occurs in $\rho\sigma$. The rightmost dotted node of ρ states which node of Σ we are currently working on.

The non-terminal set $\mathcal{N}$ of the automaton includes the extended dotted points $\Sigma{:}\rho\sigma$, as well as the symbols N^a and N^s for each non-terminal N of the grammar. We consider the thread label set $\mathcal{U} = \{\mathrm{addr}(\Sigma, \nu) | \Sigma, \nu \text{ node of } \Sigma\}$ where $\mathrm{addr}(\Sigma, \nu)$ denotes the unambiguous address of ν in the tree set Σ.

4. There is no special difficulty involved with substitution trees.

We associate to Σ the following set of transitions, most of them being straightforward extensions of the transitions for TAG listed in Fig. 3(b):

(Call set)	$N_i^a \longmapsto [N_i^a]\, N_i^a$	
(Start set with tree β_i)	$N_i^a \longmapsto \Sigma{:}\,{}^\bullet\mathrm{r_i}$	
(Add new tree β_i in set)	$N_i^a\, [\Sigma{:}\rho\sigma] \longmapsto [N_i^a]\, \Sigma{:}\rho\, {}^\bullet\mathrm{r_i}\, \sigma$	$i \notin \mathrm{ind}(\rho\sigma)$
(Foot suspend)	$[\Sigma{:}\rho\, {}^\bullet\nu\, \sigma]\, \mathrm{foot} \longmapsto \Sigma{:}\rho\, {}_\bullet\nu\, \sigma\, [\mathrm{foot}]$	
(Foot Resume)	$\mathrm{adj}\, [\Sigma{:}\rho\, {}_\bullet\mathrm{f_i}\, \sigma] \longmapsto [\mathrm{adj}]\, \Sigma{:}\rho\, \mathrm{f_i}{}_\bullet\, \sigma$	
(Suspend set after tree β_i)	$[\, \nu_\bullet]\, \mathrm{ret} \longmapsto\, \nu^\bullet\, [\mathrm{ret}]$	$\kappa\delta(\Sigma{:}\rho\, \mathrm{r_i}{}^\bullet\, \sigma) = (\mathrm{ret}, \{\bot\}) \wedge \rho \neq \epsilon$
(End set)	$\Sigma{:}\sigma \longmapsto \mathrm{ret}$	$\mathrm{ind}(\sigma) = \{1, \ldots, m\}$
(Return from set)	$[\, \nu_\bullet]\, \mathrm{ret} \longmapsto\, \nu^\bullet$	

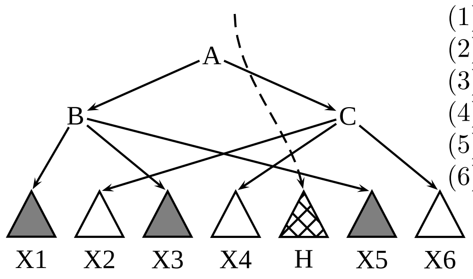
For a node ν of some elementary tree τ in set Σ and with non-terminal label N, we set

Adj	$\kappa\delta(\Sigma{:}\rho\, {}^\bullet\nu\, \sigma) = (N^a, \{\mathrm{addr}(\Sigma, \nu)\} \cup \{\mathrm{addr}(\Sigma, \mu) \mid \mu \neq \nu\}^s)$ $\kappa\delta(\Sigma{:}\rho\, \nu_\bullet\, \sigma) = (\mathrm{adj}, \{\mathrm{addr}(\Sigma, \mu)\}^s)$
Foot	$\kappa\delta(\Sigma{:}\rho\, {}_\bullet\mathrm{f_i}\, \sigma) = (\mathrm{foot}, \{\bot\})$

In terms of complexity, h is still bounded by $2m$ while d is now bounded by $\frac{1}{2} m \max_\tau |\tau|$ or, better, by $\frac{1}{2} \max_\Sigma |\Sigma|$ if we extend the notion of size to sets. Furthermore, as done for tree-local MC-TAG, one can easily check that we can replace dh by $2v$ in our complexity analysis where $v = \max_\Sigma |\Sigma|$. If the finer complexity analysis of Section 2 holds, we get a time complexity $O(n^{3+2(m+v)})$, to be compared with the time complexity $O(n^{2(m+v)})$ mentionned in (Boullier, 1999) for set-local MC-TAG.

5. Parsing Range Concatenation Grammars

Range Concatenation Grammars (Boullier, 2000b) are defined in terms of terminals, non-terminals, and range variables that may be instantiated by ranges of the input string. Many sub-classes of RCG may be identified and we characterize here a new one called *ordered simple RCG* [osRCG], equivalent to simple RCG which are themselves equivalent to Linear Context-Free Rewriting Systems (Weir, 1992). OsRCG are simple RCG where all ranges appearing in a literal are implicitly ordered: for instance $p(X_1X_2, X_3)$ means that range X_1 immediately precedes range X_2 which precedes X_3 (with some hole between X_2 and X_3). Fig. 4(a) shows an osRCG clause γ with two interleaved discontinuous sub-constituents B and C of clause head A, and a hole H inherited by A. Fig. 5 shows two simple osRCGs for the **COPY** language $\{ww|w \in \{a, b\}^*\}$ and for the **COUNT-**3 language $a^nb^nc^n$.



(a) Interleaved constituents

(1)	$A \longmapsto \gamma_{1.0}$	(7)	$[\gamma_{1.3}]\,\text{void} \longmapsto \gamma_{1.4}\,[\text{void}]$
(2)	$B \longmapsto [B]\, B$	(8)	$\text{void}\,[\gamma_{1.4}] \longmapsto [\text{void}]\,\gamma_{2.0}$
(3)	$[\gamma_{1.0}]\,\text{void} \longmapsto \gamma_{1.1}\,[\text{void}]$	(9)	$[\gamma_{2.0}]\,\text{void} \longmapsto \gamma_{2.1}$
(4)	$C \longmapsto [C]\, C$	(10)	$[\gamma_{2.1}]\,\text{void} \longmapsto \gamma_{2.2}$
(5)	$[\gamma_{1.1}]\,\text{void} \longmapsto \gamma_{1.2}\,[\text{void}]$	(11)	$\gamma_{2.2} \longmapsto \mathrm{ret}$
(6)	$[\gamma_{1.2}]\,\text{void} \longmapsto \gamma_{1.3}\,[\text{void}]$		

(b) TA transitions

Figure 4: osRCG clause γ : $A(X_1X_2X_3X_4, X_5X_6) \longrightarrow B(X_1, X_3, X_5)C(X_2, X_4, X_6)$

It is relatively straightforward to encode with TA a top-down pv parsing strategy for osRCG. Fig. 4(b) lists the transitions attached to γ while Fig. 6 illustrates how to apply these transitions to traverse γ where steps marked by (x) use transitions defined by other clauses. Fig. 1 also gives an idea of the transitions we get (with some simplifications) for the clauses r, s, t of the **COUNT-3** language of Fig. 5(b).

In terms of complexity, the TA associated with an osRCG G is characterized by $h = k-1$ the maximal number of holes in a constituent where k denotes the maximal arity of non-terminals of G and d the maximal number of

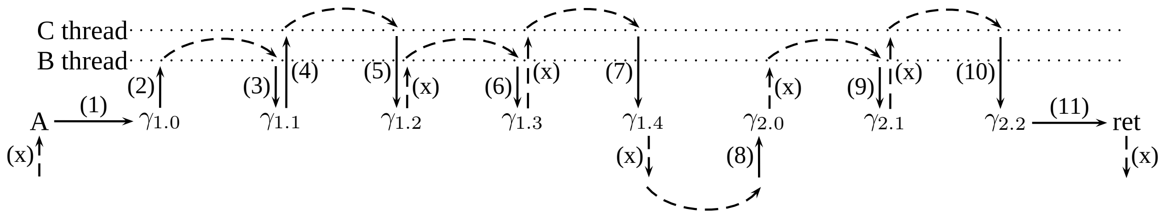
```
S(XY) -> K(X,Y).
K("","") -> .
K("a".X,"a".Y) -> K(X,Y).
K("b".X,"b".Y) -> K(X,Y).
```

(a) COPY language ww

```
S(ABC) -> K(A,B,C).                   % r
K("a".A,"b".B,"c".C) -> K(A,B,C).     % s
K("","","") -> .                      % t
```

(b) COUNT-3 language $a^n b^n c^n$

Figure 5: Two simple osRCG



	$\gamma_{1.0}$	$\gamma_{1.1}$	$\gamma_{1.2}$	$\gamma_{1.3}$	$\gamma_{1.4}$	$\gamma_{2.0}$	$\gamma_{2.1}$
$\kappa\delta$	$(B, \{1\})$	$(C, \{2\})$	(void, $\{1\}$)	(void, $\{2\}$)	(void, $\{\perp\}$)	(void, $\{1\}$)	(void, $\{2\}$)

Figure 6: Traversing γ

sub-constituents that are interleaved in a clause of G. [5] For instance, we get $h = d = 1$ for the **COPY** language and $h = k - 1$ and $d = 1$ for the **COUNT-k** language $\{a_1^n \dots a_k^n\}$.

The worst-case time complexity for RCG provided by (Boullier, 2000b) is $O(n^{k+v})$ where v is the maximal number of distinct range variables occurring in a clause. It is relatively difficult to compare with our results. However, if we suppose true our refined complexity analysis (Section 2) and assume $l = d$ for the maximal number l of literals in a clause, we can take $v = d(h + 1)$, which would give time complexities $O(n^{1+h+hd+d})$ for Boullier's algorithm applied on osRCG and $O(n^{3+h+hd})$ for our algorithm. Note it is possible to encode TAG as osRCG (Boullier, 2000a) in such a way that $h = 1$, $d = l = 2$ and $v = 4$, which would give the same time complexity $O(n^6)$ for both algorithms applied on TAG.

6. Dynamic Programming Interpretation

We don't provide in this extended abstract the full details of our tabular parsing algorithm, but only a simple intuition. Following a methodology presented in previous papers (Villemonte de la Clergerie and Alonso Pardo, 1998; Éric Villemonte de la Clergerie, 2001), our algorithm relies on a Dynamic Programming interpretation for TA which is based on the identification of a class of sub-derivations that may be represented by compact items and combined together to retrieve all derivations.

As shown in Fig. 7, for TA, such an elementary sub-derivation D

$$c_{\text{init}} \overset{\star}{\underset{d_0}{\vdash}} c_0 \overset{\star}{\underset{d_1}{\vdash}} \cdots c_{2i+1} \overset{\star}{\underset{d_{2i+1}}{\vdash}} c_{2i+2} \cdots c_{2m+1}$$

retraces the history of some thread $\pi = pu$ starting at c_0, reaching c_{2m+1} and suspended between $c_{2i+1} \overset{\star}{\underset{d_{2i+1}}{\vdash}} c_{2i+2}$ to return to either its parent thread p ($\perp$-suspension) or some sub-thread puv **still alive** at c_{2m+1} (v-suspension).

Using projections of configuration defined by $\overline{c} = \langle l, A \rangle$ and $\overline{c}^\kappa = \langle l, \kappa(A) \rangle$ where $c = \langle l, p, \mathcal{S} \cup p{:}A \rangle$, the essential information of D may be captured by an item $I = \overline{c_0}^\kappa / \mathcal{C} / \overline{c_{2m+1}}$ where $\mathcal{C} = v_1 : S_1 \dots v_i : S_i \dots v_m : S_m$, $S_i = \overline{c_{2i-1}}^\kappa \overline{c_{2i}}^\kappa$, and $v_i = v \in \mathcal{U}^\perp$ if $c_{2i-1} \overset{\star}{\underset{d_{2i}}{\vdash}} c_{2i}$ is a v-suspension. For instance, the derivation of Fig 7 gives the item $s/v : ab, \perp : cd, w : ef, v : gh/I$.

5. d may be strictly smaller than the number of sub-constituents.

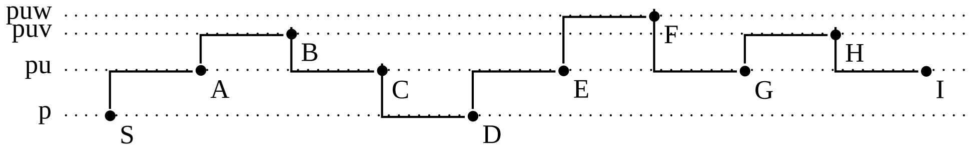


Figure 7: From a thread to item $s/v : ab, \perp : cd, w : ef, v : gh/I$

There are 5 application rules used to combine items and transitions, one for each kind of transitions. Except for the rules handling SWAP and PUSH transitions that only need an unique item, the other rules combine a transition with a *parent item* I related to some thread p and a *son item* J related to some subthread pu. These two items should *fit* together, in the sense that the holes of J should fill between the u-subparts of I and that one item should extend the other rightward. For instance, Fig. 8 shows a son item $J = a/\alpha', \perp : bc, \beta'/D$ which fits and extends a parent item $I = s/\alpha, u : ab, \beta/C$. These two items are combinable with a SPOP transition $[C]\, d \longmapsto E\, [d]$ to return an extended parent item $s/\alpha, u : ab, \beta, u : cd/E$.

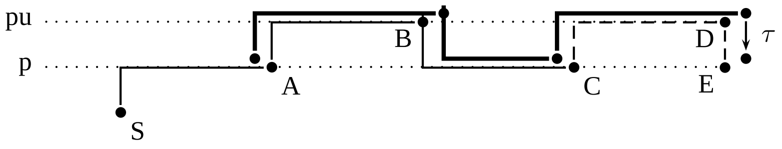


Figure 8: Combining SPOP transition $[C]\, d \longmapsto E\, [d]$ with $s/\alpha, u : ab, \beta/C$ and $a/\alpha', \perp : bc, \beta'/D$
to get item $s/\alpha, u : ab, \beta, u : cd/E$

The DP interpretation is complete and sound w.r.t. a straightforward evaluation of TA. As already mentioned, the upper-bounds for worst-case complexities are $O(n^{2(1+h+dh)})$ for space and $O(n^{2(1+h+dh)+1})$ for time, with better complexities in many cases. To provide some intuition, the space complexity is related to the maximal number h of $\perp$-suspensions and dh of v suspensions to keep trace in a item, with at most two positions in the input string per suspension and only one in many cases when the distance between the end position of a suspension and the start position of the next one belongs to some finite set independent of n. The time complexity is related to the number of distinct positions to consult when finding pairs of items that fit together to extend one of them. If we examine the case of Fig. 8, we see that we must consult all positions of parent item I to build the resulting item, check that some positions of I are also positions of the son item J, and consult the rightmost position of J to extend I. The other positions of J (in segments α' and β') may be ignored, which gives us the expected complexity.

7. Conclusion

We have introduced Thread Automata, a new automata formalism that may be used to describe a wide range of parsing strategies for a wide spectrum of MCS languages. An uniform tabular algorithm based on dynamic programming allows parsing with polynomial worst-case space and time complexities and will be soon implemented within system DyALog.

However, we still have to investigate the full spectrum of languages that may be covered by TA and explore how easy it is to describe parsing strategies for them. We would like also to explore the relationships of TA with other kinds of automata, such as (restricted) 2-Stack Automata, Embedded Push-Down Automata (Vijay-Shanker, 1988), or Tree-Walking transducers (Weir, 1992).

References

Boullier, Pierre. 1999. On Multicomponent TAG Parsing. In *TALN'99*, pages 321–326, Cargèse, Corse, France, July.
Boullier, Pierre. 2000a. On TAG Parsing. *Traitement Automatique des Langues (T.A.L.)*, 41(3):111–131. issued June 2001.
Boullier, Pierre. 2000b. Range Concatenation Grammars. In *Proceedings of the Sixth International Workshop on Parsing Technologies (IWPT2000)*, pages 53–64, Trento, Italy, February.
Joshi, Aravind K. 1987. An Introduction to Tree Adjoining Grammars. In Alexis Manaster-Ramer, editor, *Mathematics of Language*. John Benjamins Publishing Co., Amsterdam/Philadelphia, pages 87–115.

Éric Villemonte de la Clergerie. 2001. Refining tabular parsers for TAGs. In *Proceedings of NAACL'01*, June.

Vijay-Shanker, K. 1988. *A Study of Tree Adjoining Grammars*. Ph.D. thesis, University of Pennsylvania, January. Available as Technical Report MS-CIS-88-03 LINC LAB 95 of the Department of Computer and Information Science, University of Pennsylvania.

Villemonte de la Clergerie, Eric and Miguel A. Alonso Pardo. 1998. A tabular interpretation of a class of 2-Stack Automata. In *Proc. of ACL/COLING'98*, August.

Weir, David. 1988. *Characterizing Mildly Context-Sensitive Grammar Formalisms*. Ph.D. thesis, University of Pennsylvania.

Weir, David. 1992. Linear context-free rewriting systems and deterministic tree-walking transducers. In *Proc. of ACL'92*.

Evaluation of LTAG Parsing with Supertag Compaction

Olga Shaumyan, John Carroll and David Weir
School of Cognitive and Computing Sciences
University of Sussex
Brighton, BN1 9HQ
UK
{*olgas,johnca,davidw*}*@cogs.susx.ac.uk*

1. Introduction

One of the biggest concerns that has been raised over the feasibility of using large-scale LTAGs in NLP is the amount of redundancy within a grammar's elementary tree set. This has led to various proposals on how best to represent grammars in a way that makes them compact and easily maintained (Vijay-Shanker and Schabes, 1992; Becker, 1993; Becker, 1994; Evans, Gazdar and Weir, 1995; Candito, 1996). Unfortunately, while this work can help to make the *storage* of grammars more efficient, it does nothing to prevent the problem reappearing when the grammar is processed by a parser and the complete set of trees is reproduced. In this paper we are concerned with an approach that addresses this problem of *computational* redundancy in the trees, and evaluate its effectiveness.

2. LTAG parsing

LTAG parsing involves (at least) the following two steps. Each word in the input sentence is associated with that set of (elementary) trees (also called supertags) from the grammar that it can anchor. In large-scale grammars such as the XTAG grammar (XTAG-Group, 1999) and the LEXSYS grammar (Carroll *et al.*, 1998b), due to lexical ambiguity, there are usually a great many trees that can anchor each word. Once all of these elementary trees or supertags have been found, the parser must explore ways in which they can be composed—using substitution and adjunction—to produce complete parses of the input string.

In various experiments using a large, automatically produced LTAG, Sarkar, Xia and Joshi (2000) measured the time to derive a set of shared derivation forests representing all derivations for each sentence. They used a grammar with 6,789 tree templates and 2,250 sentences of length 21 words or less, and concluded that the amount of syntactic lexical ambiguity and the number of clauses in a sentence are more significant factors in determining the time taken to compute a parse forest than sentence length.

To date, the most popular way of addressing the computational problem of lexical ambiguity in LTAG parsing involves supertag filtering, where another step is included in the parsing processes (between the two phases described above) which involves filtering out some of the possible supertags for words in the sentence (Joshi and Bangalore, 1994; Bangalore, 1997a; Bangalore, 1997b; Chen, Bangalore and Vijay-Shanker, 1999). This can dramatically reduce the time it takes to find all ways in which supertags can be combined together into complete parses. Sarkar et al. demonstrate the potential benefit: parsing their 2,250 sentences with all supertags took 548,000 seconds, but this reduced to 21,000 seconds when the maximum number of supertags per word was limited to 60, and to a mere 31.2 seconds when lexical ambiguity was completely eliminated.

However, a drawback of this approach is that, since supertag filtering cannot be 100% accurate, a proportion of desirable supertags are filtered out, resulting in some parses being lost. For example in Sarkar et al.'s experiment, a limit of 60 supertags per word resulted in over 40% of the sentences receiving no parse at all.

3. Elementary computation sharing

There is an alternative approach to the problem of lexical ambiguity in parsing that removes some of the computational redundancy that results from lexical ambiguity[1]. Given some parsing algorithm, each elementary

* We are extremely grateful to Fei Xia for providing us with the grammar used in these experiments, and to Fei Xia and Anoop Sarkar for the help and advice they have given.

1. Note that the approach described in this section could be combined with supertag filtering.

tree can be viewed as invoking some fragment of computation (an elementary computation). Evans and Weir (1998) showed that elementary computations corresponding to bottom-up parsing can be expressed as finite state automata (FSA). All elementary computations for the supertags associated with a word can be combined into a single FSA. By minimizing this automaton (using standard minimization algorithms) sharing of elementary computation is achieved. The hope is that this will lead to significant reductions in parsing time.

To date, this proposal has only received limited evaluation. Carroll *et al.* (1998a) demonstrated that for a large hand-crafted grammar the number of states was significantly reduced by merging and minimizing the FSA associated with a word. For example, the numbers of states in the automaton for the word *come* (associated with 133 supertags) was reduced from 898 to 50, for *break* from 1240 to 68, and *give* from 2494 to 83.

This paper improves on this evaluation in two ways: firstly, the grammar used is automatically acquired, so we are not open to the charge that it was designed to make this technique work particularly well; secondly, we measure parse time, not just numbers of states for individual words. Even when the number of states is significantly reduced it is not clear that parse time (as opposed to recognition time) will drop. This is because in order that parse trees be recoverable from the parse table, a considerable amount of book-keeping is required when the table is being completed. This increases both space and time requirements.

4. Experimental evaluation

We used a grammar that was automatically induced by Fei Xia (1999) from sections 00–24 of the Wall Street Journal Penn Treebank II corpus (Marcus, Santorini and Marcinkiewicz, 1993). This is very similar to the grammar used by Sarkar, Xia and Joshi (2000) and Sarkar (2000), though slightly larger, containing around around 7,500 elementary trees.

We implemented the algorithm described by Evans and Weir (1997) and Evans and Weir (1998), the details of which are not repeated here. Prior to parsing, the grammar is precompiled as follows. For each word, the set of trees that it can anchor is determined. This results in a total of 11,035 distinct tree sets. For each of these tree sets we first build what we refer to as an unmerged FSA. This automaton contains a separate progression of transitions for each of the trees in the set; using these automata for parsing gives a conventional LTAG parsing algorithm which we used to give a baseline for our evaluation. To evaluate the approach of Evans and Weir we implemented a parser that used minimized versions of the automaton with sharing of common elementary computation fragments.

There are 80,538 non-minimized automata (involving 488,421 states). Thus there is a total of 80,538 occurrences of one of the grammar's elementary trees in the 11,035 tree sets. When these nonminimized automata are minimized we have one automaton for each of the 11,035 tree sets; these automata contain a total of 153,022 states. Thus, minimization gives an overall compaction of a factor of 3.19. In order to determine the computational benefit of elementary computation sharing we ran both the merged and unmerged parsers on a set of 14,272 test sentences of lengths 1–45 words taken from sections 00–24 of the Penn Treebank corpus. The results are shown in Table 1[2]. It is clear that numbers of items and CPU time are smaller for the merged parser, and that the savings increase with longer sentences.

Given a tokenized sentence to be parsed, the time shown includes time to make a chart corresponding to the sentence length, look up definitions for each word and seed the chart with them, and then fill the chart. The results show that the parse time for the merged parser is around 0.6 that of the umerged parser, and that this ratio is fairly consistent as the length of sentence increases. This is shown in Figure 1.

5. Discussion

We have presented an empirical evaluation of the automaton-based LTAG parsing algorithm presented by Evans and Weir (1998). We used a grammar automatically generated from Penn Treebank trees with two parsers: one in which elementary trees were processed individually, and one in which overlapping elementary computations were shared. The results show that merging elementary computations results in a significant, though not spectacular, reduction in parse time, despite the increased amount of book-keeping required to make recovery of parse trees possible. In future work we plan to determine exactly how much this book-keeping adds to parse time by implementing a version of the merged parser in which book-keeping is omitted.

2. We ran both parsers on one 750MHz processor of a unloaded Sun Blade 1000 workstation with 1.5GB memory.

One rather significant drawback of this evaluation is that the amount of lexical ambiguity in this grammar is far less than is found in large-scale wide-coverage grammars such as the XTAG grammar (XTAG-Group, 1999). Although the tree-bank includes examples of a wide variety of syntactic constructions, for any individual word, the number of syntactic contexts (corresponding to alternative supertag possibilities for that word) that actually occur in the Penn Tree Bank is generally far less than those that would be included in the lexical entry for that word in a wide-coverage grammar. This is particularly true for words with low frequency. In future work we plan to look into ways of obtaining more complete mapping from a lexical item to the set of supertags it can anchor.

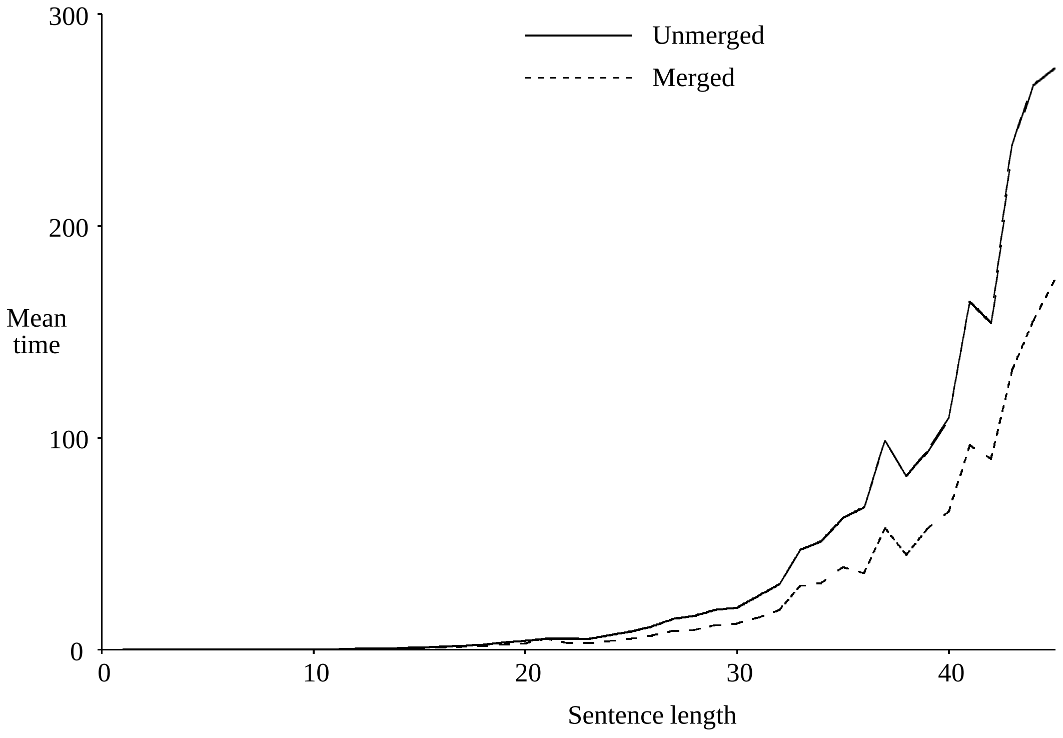


Figure 1: Comparison of Running Times

References

Bangalore, Srinivas. 1997a. *Complexity of Lexical Descriptions and its Relevance to Partial Parsing*. Ph.D. thesis, University of Pennsylvania, Philadelphia.

Bangalore, Srinivas. 1997b. Performance Evaluation of Supertagging for Partial Parsing. In *Proceedings of the Fifth International Workshop on Parsing Technologies*.

Becker, Tilman. 1993. *HyTAG: A new type of Tree Adjoining Grammar for hybrid syntactic representation of free word order languages*. Ph.D. thesis, Universitat des Saarlandes.

Becker, Tilman. 1994. Patterns in metarules. In *Proceedings of the Third International Workshop on Tree Adjoining Grammars*, pages 9–11.

Candito, Marie-Hélène. 1996. A principle-based hierarchical representation of LTAGs. In *Proceedings of the 16th International Conference on Computational Linguistics*, Copenhagen, Denmark, August.

Carroll, John, Nicolas Nicolov, Olga Shaumyan, Martine Smets and David Weir. 1998a. Grammar Compaction and Computation Sharing in Automaton-based Parsing. In *Proceedings of the First Workshop on Tabulation in Parsing and Deduction*, pages 16–25.

Carroll, John, Nicolas Nicolov, Olga Shaumyan, Martine Smets and David Weir. 1998b. The LEXSYS Project. In *Proceedings of the Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks*, pages 29–33.

Chen, John, Srinivas Bangalore and K. Vijay-Shanker. 1999. New Models for Improving Supertag Disambiguation. In *Proceedings of the Eighth Conference of the European Chapter of the Association for Computational Linguistics*.

Evans, Roger, Gerald Gazdar and David Weir. 1995. Encoding Lexicalized Tree Adjoining Grammars with a Nonmonotonic Inheritance Hierarchy. In *Proceedings of the 33rd Meeting of the Association for Computational Linguistics*, pages 77–84.

Evans, Roger and David Weir. 1997. Automaton-based Parsing For Lexicalized Grammars. In *Proceedings of the Fifth International Workshop on Parsing Technologies*, pages 66–76.

Evans, Roger and David Weir. 1998. A structure-sharing parser for lexicalized grammars. In *Proceedings of the 36th Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics*, pages 372–378.

Joshi, Aravind and Srinivas Bangalore. 1994. Disambiguation of super parts of speech (or supertags): almost parsing. In *Proceedings of the 15th International Conference on Computational Linguistics*, pages 154–160.

Marcus, Mitchell, Beatrice Santorini and Mary Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Sarkar, Anoop. 2000. Practical Experiments in Parsing using Tree Adjoining Grammars. In *Proceedings of the Fifth International Workshop on Tree Adjoining Grammars and Related Frameworks*.

Sarkar, Anoop, Fei Xia and Aravind Joshi. 2000. Some Experiments on Indicators of Parsing Complexity for Lexicalized Grammars. In *Efficiency in Large-Scale Parsing Systems*. Workshop held at COLING 2000.

Vijay-Shanker, K. and Yves Schabes. 1992. Structure Sharing in Lexicalized Tree-Adjoining Grammar. In *Proceedings of the 14th International Conference on Computational Linguistics*, pages 205–211.

Xia, Fei. 1999. Extracting Tree Adjoining Grammars from Bracketed Corpora. In *Proceedings of the 5th Natural Language Processing Pacific Rim Symposium(NLPRS-99)*.

XTAG-Group, The. 1999. A Lexicalized Tree Adjoining Grammar for English. Technical Report `http://www.cis.upenn.edu/˜xtag/ tech-report/tech-report.html`, The Institute for Research in Cognitive Science, University of Pennsylvania.

Sentence length	# items unmerged	# items merged	mean time unmerged	mean time merged
1	3	1	0.0	0.0
2	19	5	0.0	0.0
3	104	17	0.0	0.0
4	327	52	0.0	0.0
5	681	121	0.0	0.0
6	1256	235	0.0	0.0
7	2298	473	0.0	0.0
8	3923	864	0.1	0.0
9	5844	1347	0.1	0.1
10	9022	2180	0.2	0.1
11	12674	3203	0.2	0.2
12	18000	4649	0.4	0.2
13	26110	6928	0.5	0.4
14	34074	9165	0.7	0.5
15	47564	12969	1.0	0.8
16	62771	17481	1.3	1.0
17	80515	22809	1.8	1.4
18	99121	27909	2.3	1.7
19	128028	36790	3.3	2.5
20	163347	47322	4.1	2.9
21	193701	56268	5.1	5.2
22	277740	80430	5.1	3.1
23	274474	81562	5.0	3.1
24	354912	101143	6.8	4.1
25	427291	124919	8.5	5.3
26	532109	154792	10.9	6.7
27	683355	195608	14.5	8.9
28	731932	208338	15.9	9.3
29	855873	253130	18.8	11.6
30	873492	258383	19.7	12.3
31	1089989	314794	25.2	15.1
32	1291749	371601	30.8	18.6
33	1838306	556519	47.2	30.2
34	1917227	574944	51.1	31.6
35	2364987	710872	62.1	38.9
36	2487632	651374	67.1	36.1
37	3381691	982343	98.6	57.4
38	2864371	780416	82.0	44.7
39	3290281	979203	93.4	57.1
40	3755657	1106993	109.5	65.1
41	4993534	1467100	164.2	96.4
42	4843654	1380099	154.3	90.0
43	7071346	1983426	238.5	132.1
44	7655510	2282781	266.5	155.3
45	7772779	2317031	274.3	174.3

Table 1: Mean numbers of items and parse times (CPU seconds) per sentence, for sentences of length 1–45 words.

Korean–English MT and S-TAG

Mark Dras and Chung-hye Han
Macquarie University *Simon Fraser University*

1. Introduction

An early motivation for Synchronous TAG (S-TAG) (Shieber and Schabes, 1990) was machine translation (Abeillé, Schabes and Joshi, 1990). Abeillé *et al* note that traditionally difficult problems outlined by Dorr (1994)—for example, categorial, thematic, conflational, structural and lexical divergences—have been used to argue for the necessity of an explicit semantic representation. However, many of these divergences are not problems for an S-TAG-based approach. Synchronous TAG translation models thus allow us to explore the question of the extent to which a semantic representation is actually necessary.

S-TAG was redefined by Shieber (1994) for both theoretical and practical reasons, introducing the requirement that the derivation trees of target and source be isomorphic. Under this definition it has been noted (Shieber, 1994; Dras and Bleam, 2000) that there are mappings that cannot be described under S-TAG. This was the motivation for meta-level grammars (Dras, 1999), by which two TAG grammars can be paired while retaining their original properties, as under standard S-TAG, allowing for a description of mappings that include unbounded non-isomorphisms (Dras and Bleam, 2000).

This work on exploring how S-TAG (with and without meta-level grammars) can be used for MT has only been applied to languages that are closely related—English, French, Italian and Spanish. In this paper we aim to take a much more widely differing pair of languages, English and Korean, to investigate the extent to which syntactic mappings are satisfactory.

English and Korean have a wide range of differences: rigid SVO word order in English vs verb-final with free word order in Korean, the largely analytic structure of English vs the agglutinative structure of Korean with its complex morphology, optional subject and object and the absence of number and articles in Korean, and many others. These all suggest that a meta-level grammar will be necessary as there are various many-to-one or many-to-many mappings between derivation tree nodes (i.e., there will be few cases where a single elementary tree corresponds to another single elementary tree, which has been the case with closely related languages).

Although there is an implemented Korean/English MT system that includes a TAG Korean parser as a source language analysis component (Han *et al.*, 2000), this system as a whole is based on Meaning Text Theory (Mel'čuk, 1988), an enriched dependency formalism. Thus, it requires a conversion component that converts the TAG parser output to a dependency notation. As pointed out in Palmer *et al.* (2002), however, this conversion process resulted in a loss of crucial information such as predicate-argument structure encoded in TAG elementary trees, which had negative consequences in the translation results. This then provides further motivation to explore the feasibility of applying a single TAG-based formalism to modeling and implementing a Korean/English MT system.

As a first step towards exploring the extent to which an S-TAG style approach can successfully model these widely different languages, we have taken from a parallel English-Korean Treebank twenty examples of divergent constructions (see Appendix). Each half has roughly 50,000 word tokens and 5,000 sentences. While the annotation guidelines for the Korean half was developed in Han, Han and Ko (2001) for this corpus, the English half follows the guidelines already developed for Penn English Treebank (Bies *et al.*, 1995), as closely as possible. The example pairs represent structures including copula, predicative/attributive adjective, passive, causative, interrogative, relative clause, complex verb, and modal construction, among others. We find that using a TAG-based meta-level grammar to model Korean/English correspondences for machine translation is quite feasible.

2. Analyses

In this section we discuss two example pairs of sentences, taken from the parallel Treebank, that illustrates several divergences, and how an S-TAG with meta-level grammar can handle them. The trees we use for the subgrammars for the sentences are extracted automatically from the Treebank using Lextract (Xia, Palmer and

Joshi, 2000).[1]

2.1. Korean complex NP vs. English modal

The sentence pair in (1) represents a modal construction. The key divergence is that the Korean uses a noun complement structure, while the English uses a modal adjective structure:

(1)	전차들은	그	능력을	개활지에서	충분히	발휘할	수	있습니다.
	tank-Plu-Top	that	ability-Acc	open-terrain-Loc	fully	show-Adnominal	possibility	be-Past-Decl

Tanks are able to fully demonstrate their potential in open terrain.

A closer but less natural translation of the Korean is *The possibility that tanks fully demonstrate their potential in open terrain exists*; the noun representing *possibility* is modified by an adnominal clause. The corresponding English translation contains *be able to* followed by an infinitival clause. The derivation trees are as in Figure 1, and the Lextract elementary trees grouped according to the translation pairing in Figure 2.

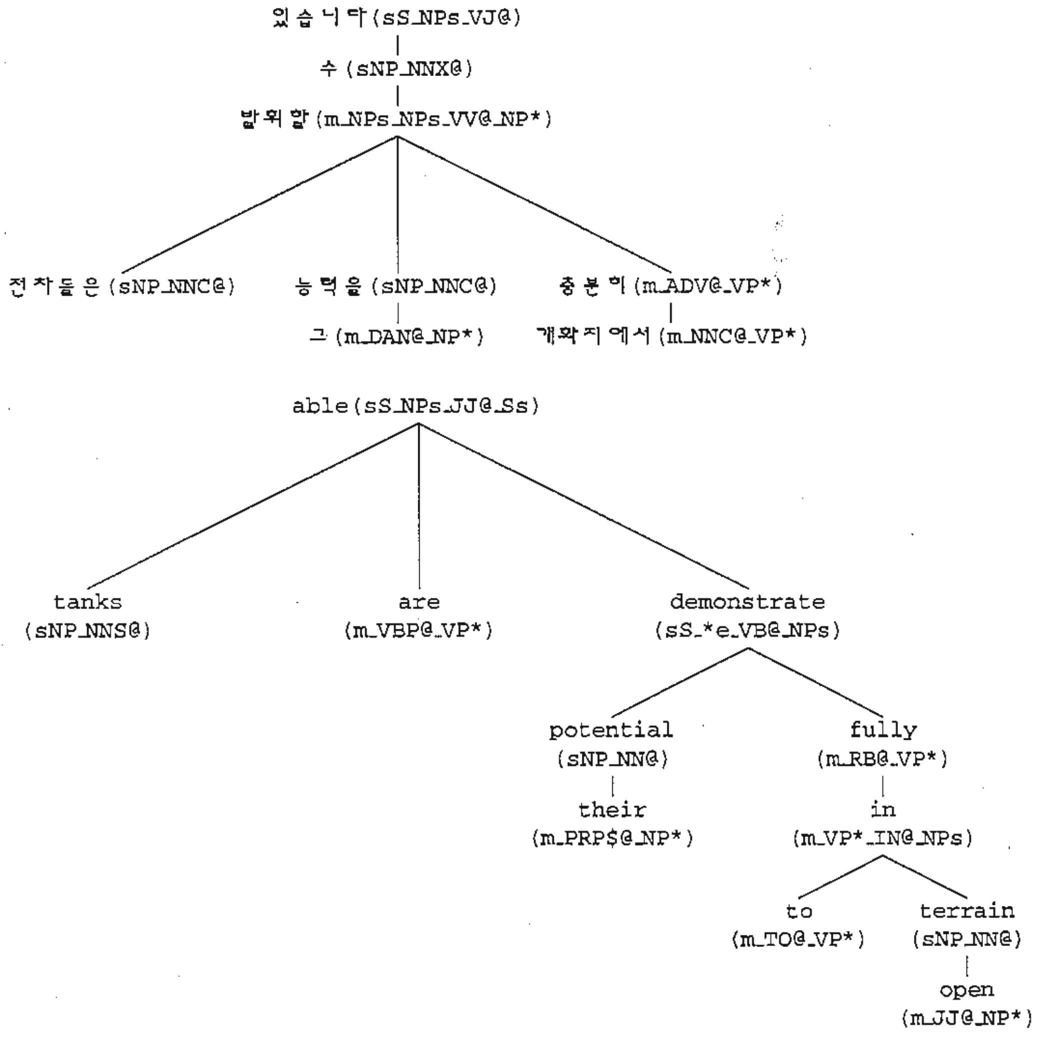


Figure 1: Derivation trees for (1)

1. Note that the Lextract trees do not contain features, although the corresponding Korean XTAG (Han *et al.*, 2000) trees do. We will make use of the features where necessary.

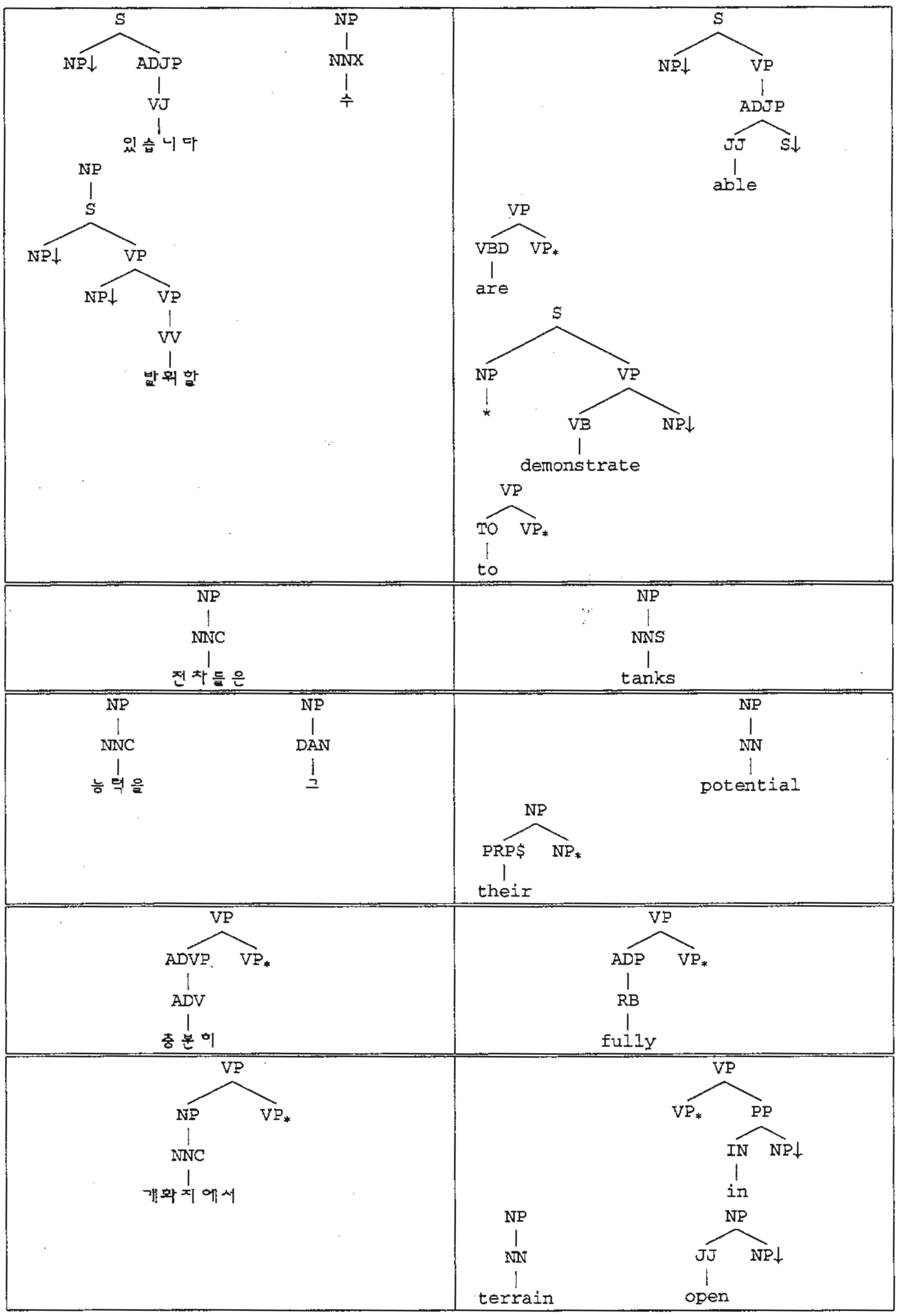


Figure 2: Lextract elementary trees for (1)

The trees are clearly far from isomorphic. The relationship between *able* and *are* is inverted between the corresponding Korean 있습니다 (*be*) and 수 (*possibility*), although *demonstrate* is the child of *able* (발휘할 and 수 respectively) in both. Most crucially, however, the infinitival *to* in English, attached to *demonstrate*, has no corresponding element in Korean; rather, *to* and *demonstrate* correspond to the single 발휘할 in Korean. But, given TAG's approach to modification, an unbounded number of modifiers (*fully*, the PP headed by *in*) can be inserted between *demonstrate* and *to*, giving an unbounded non-isomorphism. In other examples we have noted that this unbounded non-isomorphism is quite prevalent, occurring inter alia with nouns and determiners.

Other divergences attested in (1) are that *tanks* is an argument of *able*, but 전차들은 (*tanks*) is an argument of 발휘할 (*demonstrate*); and that the preposition *in* is represented by the suffix 서, a type of correspondence that occurs frequently because of the analytic–agglutinative language mismatch. Using the algorithm of Dras (1999), however, it is possible to construct a meta-level grammar to characterize appropriate paired substructures in the trees, as in Figure 3. The basic principle is that the divergent material is captured by the multi-level tree pairs (such as 19–A), in particular in cases with unbounded non-isomorphisms, where the recursive material (such as 19–D and 19–E) is factored out. The other structures that are not a cause of the isomorphism violation continue to to be paired by single-level tree pairs (either as in 19–B, or in cases not illustrated here where there is a single node corresponding to a lexicalized tree plus a substitution node).[2]

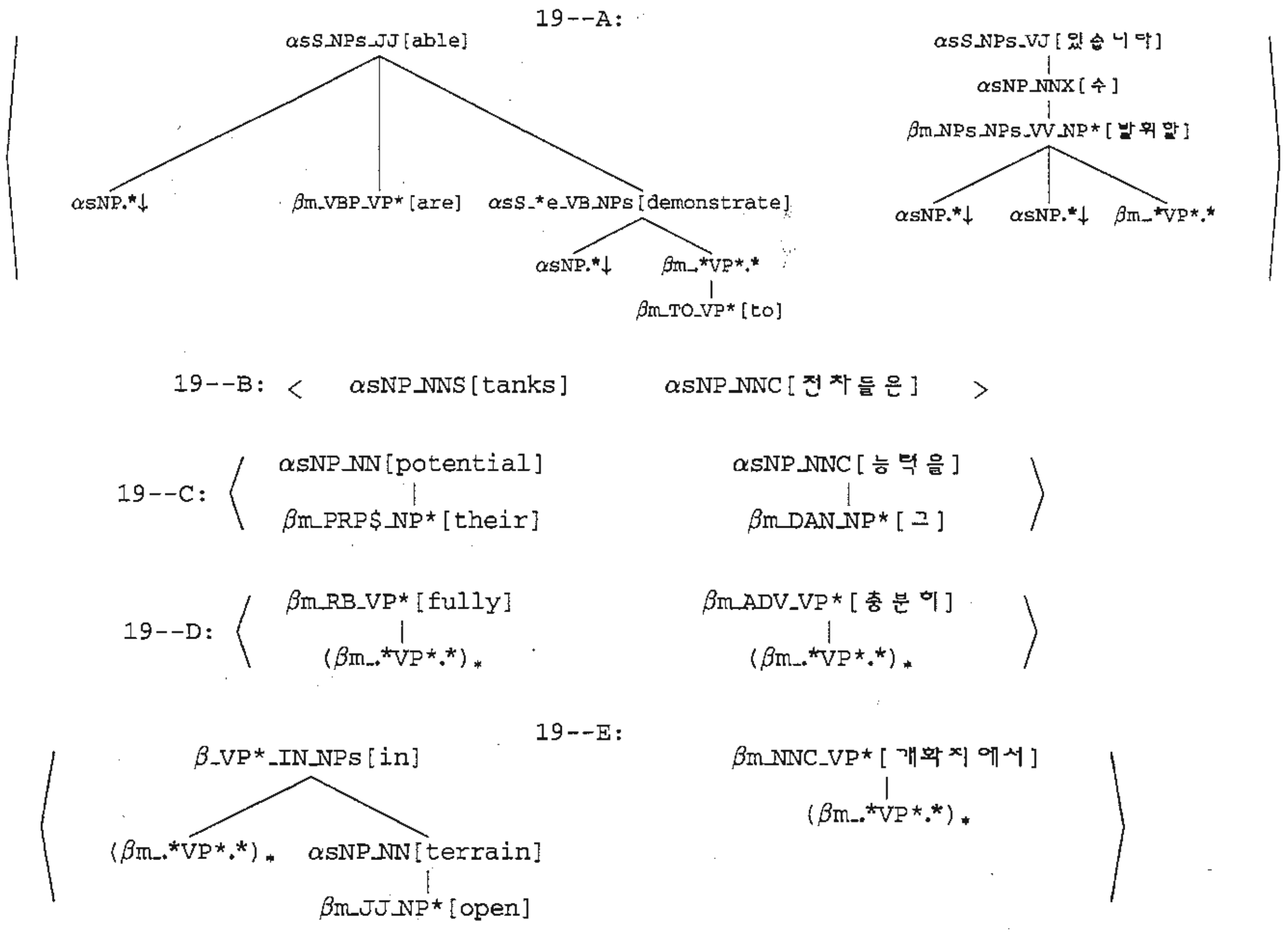


Figure 3: Meta-level grammar for (1)

The groupings that arise from the algorithm are fairly intuitive. 19–A represents the concept *the ability of X to demonstrate Y* (X here being *t*anks and Y *p*otential), with two consequent argument slots, and one slot where a modifier can be adjoined marked βm_*VP*.*. [3] 19–B and 19–D are straightforward; 19–C aggregates the nodes because in general Korean does not use determiners, so an English noun and determiner correspond to a

2. If a pairing of isomorphic trees was expressed by a meta-level TAG, all trees would be single-level.
3. This regular expression represents a node where any tree with a root whose label matches can adjoin; technically this is

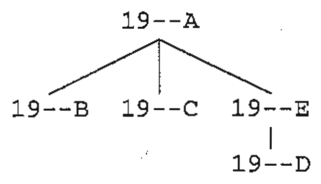


Figure 4: Meta-level derivation for (1)

single unit in Korean (although this is not the case here, we follow that general principle); and 19–E represents the correspondence between the English PP *in open terrain* and the single Korean 개활지에서. Under this meta-level grammar we have isomorphic meta-level derivation trees for English and Korean with structure as in Figure 4.

Note that, as a next step, the obvious generalisation is to have a single parametrized tree pair in cases like 19–A and 19–E. From 19–A we will have the same structure for *X are able to demonstrate Y*, *X are able to see Y*, and so on, with a Korean correspondent for each choice of verb. From 19–E we will have the same structure for *in open terrain*, *near open terrain*, and so on, with a corresponding Korean suffix for each choice of preposition. With the suffixes in Korean XTAG represented by features, the approach would be similar to that of Abeillé, Schabes and Joshi (1990) for cases where the French and English share a feature-related attribute like number.

For the example here it could be argued that perhaps *to* 'should' be in the same tree as *demonstrate*, and that in general there should not be separate elementary trees for function words. Frank (2001) argues for functional elements to be part of lexical elementary trees, and this is the principle used in building the large-scale French TAG grammar, although each has different ideas as to which trees functional elements should be included in. However, part of the aim of translating with S-TAG is to use already existing grammars; there are not special separate grammars for translation that have matching choices about function word treatment. And it is unlikely that all choices would match in any case, for example with determiners, which would be likely separate in English and French, but not in Korean.[4]

2.2. Copula constructions

Korean does not have an explicit copula; this gives rise divergences as in the sentence pair (2).

(2) 경기관총 분대장은 중사입니다.
light-machinegun squad-leader-Top sergeant-Cop-Decl
The light machinegun squad leader is a sergeant.

This is not problematic because of the way in which TAG conventionally represents copular constructions, where the predication is the root of the derivation and the copula is adjoined in. Derivation trees are as in Figure 5.

The feature of interest in this translation is the absence of Korean determiners, as mentioned in the previous example. The combined noun-determiner in English thus corresponds to only the noun in Korean; and there can be recursive intervening material (such as *light*, *machinegun* and *squad* between *the* and *leader*). Thus we again have an unbounded non-isomorphism, and we handle it with a meta-level grammar as in Figure 6.

3. Discussion

In our analysis of twenty sentence pair types (see Appendix) chosen to illustrate particular divergences not typically found between closely related languages, a TAG meta-level grammar is basically adequate for describing the mapping between them, using the algorithm of Dras (1999).

because the labels are really just features (Kasper *et al.*, 1995; Dras, Chiang and Schuler, 2002). Thus, slightly confusingly, there are three types of asterisk in a meta-level grammar. Firstly, there is the asterisk that is part of the name of an XTAG or Lextract tree; this is indicated by a normal aterisk *. Secondly, there is the asterisk to indicate a regular expression over these names; this is indicated by a bold asterisk *****. Thirdly, there is the asterisk to indicate a footnode in a meta-level auxiliary tree; this is indicated by a subscripted asterisk $_*$. All three occur in, for example, the right projection of 19–E.

4. In fact, the fact that F-TAG includes function words in lexical trees and the XTAG English grammar does not suggests that a meta-level grammar may be useful there also.

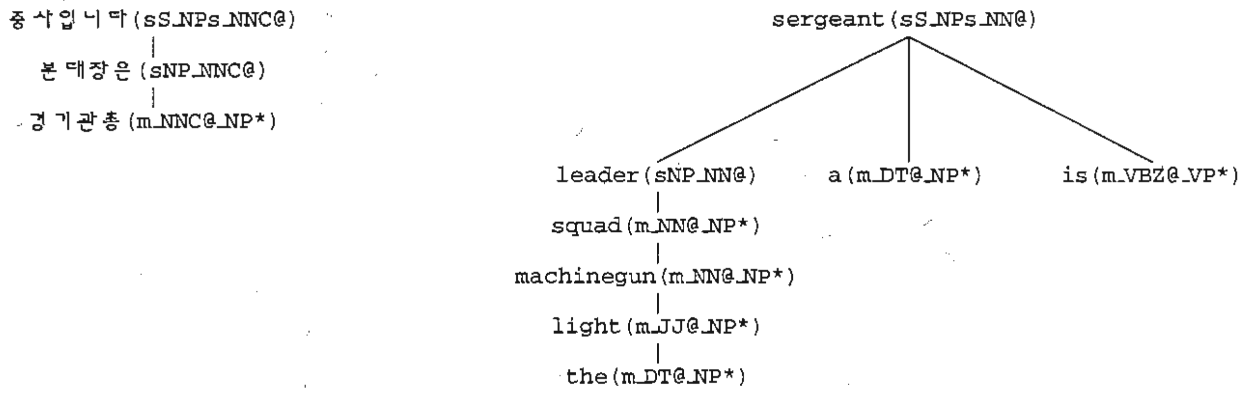


Figure 5: Derivation trees for (2)

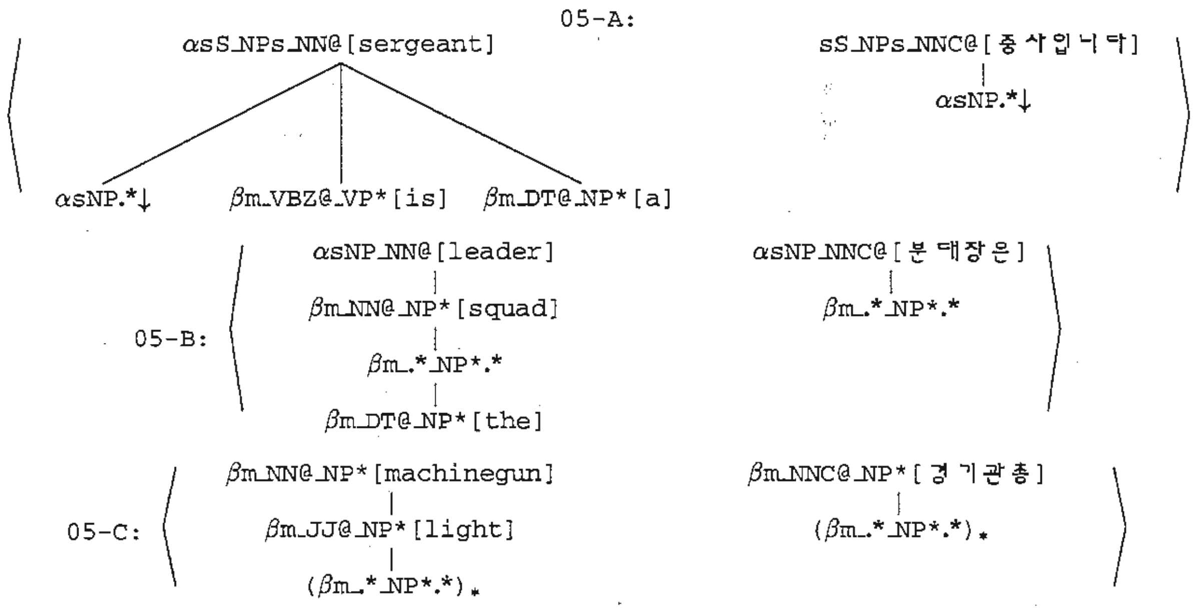


Figure 6: Meta-level grammar for (2)

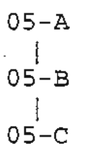


Figure 7: Meta-level derivation for (2)

The major exception is with some adverbial modifiers that can occur both sentence-initially and adjacent to VP without any semantic difference. Because TAG is fundamentally a constituent-based formalism, it is necessary to have two different trees for such modifiers (e.g., *soon*) depending on the location of the modifier (S-rooted and VP-rooted). Thus, in a sentence pair as in (3) in which *now* is VP-adjoined and 지금 ('now') is S-adjoined, it is not possible to build a reasonable TAG meta-level grammar. To see this examine the derivation trees given in Figure 8. Most nodes pair up straightforwardly (*on schedule* pairing with 계획대로, with the Korean containing a suffix to parallel *on*); the exceptions are the nodes for *now* and *proceeding*, which would have to be grouped together because of the different dominance relations (계획대로 being immediately dominated by 진행되고, but there being the possibility of unbounded intervening material between *proceeding* and *now*). This grouping of *proceeding* and *now* would be fairly unprincipled, as *now* is a case of recursive material that does not belong in an elementary tree pair at the meta-level. That is, a meta-level grammar is still formally adequate, but linguistically undesirable.

(3) 지금	그 공격 준비가	계획대로	진행되고	있습니다
now	that attack preparations-Nom	plan-as	proceed-Pass-Auxconn	be-Past-Decl

The attack preparations are now proceeding on schedule.

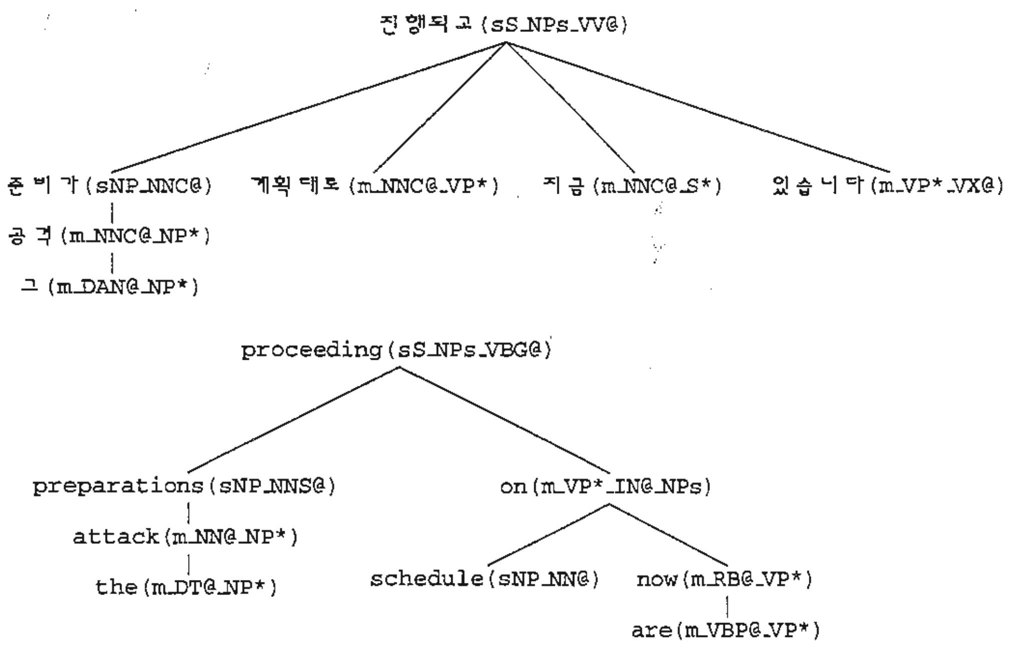


Figure 8: Derivation trees for (3)

However, no semantic difference will result if *now* were sentence-intial in the English, or if 지금 ('now') were adjacent to the verb 진행되고 ('proceed') in the Korean. This means that even if the Treebank translation does not allow a meta-level grammar, one is possible just by moving the modifier. From our initial exploration, then, a meta-level grammar appears to be a promising candidate for describing English-Korean translation.

The next stage of the work is to build a prototype system and use a Lextract-like approach to extract a meta-level grammar from the parallel Treebank. Lextract already provides us with elementary and derivation trees for Treebank pairs; the algorithm of Dras (1999) gives a systematic method for identifying paired substructures in derivation trees. Further, our prototype system will include a generation component (for Korean and/or English, depending on what the target language is) that generates derivation and derived trees from a given meta-level derivation structure.

A. Sample divergences

#simple declaratives

(4) 제가 대대장한테 관측 사항을 보고하였습니다.
I-Nom battalion-commander-to observation stuff-Acc report-Past-Decl
I reported my observations to the battalion commander.

#declarative with object scrambling

(5) 그들의 규모나 명칭은 저는 모릅니다.
their size-or designation-Top I-Top don't-know-Decl
I don't know their sizes or designations.

#attributive adjective

(6) 도로의 상태와 적 정황 중요한 요소가 됩니다.
road-Gen condition-Conj enemy situation be-important-Adnominal factor-Nom be-Decl
Road conditions and the enemy situation are key factors.

#predicative adjective

(7) 대대 정치 군관의 권한은 대단히 크지요.
battalion political office-Gen authority-Top very extensive-Decl
The authority of the battalion political officer is very extensive.

#copula sentence

(8) 경기관총 분대장은 중사입니다.
light-machinegun squad-leader-Top sergeant-Cop-Decl
The light machinegun squad leader is a sergeant.

#Korean passive morphology → English passive form

(9) 부대 명칭은 통상 암호로 하달됩니다.
unit designation-Top normally code-in transmit-Pass-Decl
Unit designations are normally transmitted in code.

#Korean passive morphology → English active form

(10) 지금 그 공격 준비가 계획대로 진행되고 있습니다.
now that attack preparations-Nom plan-as proceed-Pass-Auxconn be-Decl
The attack preparations are now proceeding on schedule.

#Korean active form → English passive form

(11) 그러니까 더 이상 그런 선전에는 속지 마!
so any more that propaganda-by-Top deceived don't
So don't be deceived by that propaganda anymore!

#lexical causative

(12) 눈보라가 교통을 마비시키었다.
snowstorm-Nom traffic-Acc paralysis-Cause-Past-Decl
The snowstorm paralyzed the traffic.

#structural causative

(13) 중대 특무장은 중대 성원들이 무기와 탄약을 갖도록
compay first-sergeant-Top company members-Nom weapons-Conj ammunition-Acc have
합니다.
make-Decl
The company first sergeant ensures that the members of the company have the weapons and ammunition.

#structural causative

(14) 그러면 대대부가 대대 공급반한테 탄약을 수송하도록 하였습니다
then battalion-HQ-Nom battalion supply-section-to ammunition-Acc transport-Caus do-Past-Decl
Our battalion HQ then had the ammunition brought in by the battalion's supply section.

#yes-no question

(15) 소대장의 호출대호가 바뀌었는가?
platoon-leader-Gen call-sign-Nom changed-Past-Int
Has the call sign of the platoon leader been changed?

#wh-question

(16) 전파 지향성 공중선은 어떤 무전기를 사용하는가?
radio(wave) directional antenna-Top what radio-Acc use-Int
What types of radios is the inclined beam antenna used with?

#relative clause

(17) 그 무선 전화수가 사용한 책은 컸다.
that radiotelephone operator-Nom use-Adnom book-Top big-Past-Decl
The book that the radiotelephone operator used was big.

#Korean morpheme → English word

(18) 포병 지원 부대가 대대에 배속되면 이들도 초단파망을
artillery support unit-Nom battalion-To attach-Pass-when these-persons-also microwaves-net-Acc
사용합니다.
use-Decl
When artillery support units are attached to the battalion, they would use the VHF network also.

#Korean noun and light verb → English verb

(19) 송신기나 수신기는 가끔 손질을 해야 합니다.
transmitter-and receiver-Top occasionally cleaning-Acc do-Auxconn must-Decl
One must clean the transmitters and receivers occasionally.

#Korean complex verb → English verb and adverb

(20) 그 편지 저한테 돌리어 주십시오.
that letter me-To handover-Auxconn give-Imp
Please give me back the letter.

#Korean complex verb → English verb and preposition

(21) 분대장은 그 부상병의 눈을 조심스럽게 들여다 보았습니다.
squad-leader-Top that wounded-soldier-Gen eye-Acc carefully take-in look-Past-Decl
The squad leader carefully looked into the eyes of the wounded soldier.

#Korean complex noun phrase → English modal auxiliary verb construction

(22) 전차들은 그 능력을 개활지에서 충분히 발휘할 수 있습니다.
tank-Plu-Top that ability-Acc open-terrain-Loc fully show-Adnominal possibility be-Past-Decl
Tanks are able to fully demonstrate their potential in open terrain.

#Korean intransitive verb → English transitive

(23) T-54 탱크는 발연했읍니다.
T-54 tank-Top smoke-emit-Past-Decl
The T-54 tank emitted smoke.

References

Abeillé, Anne, Yves Schabes and Aravind K. Joshi. 1990. Using lexicalized tree adjoining grammars for machine translation. In *Proceedings of the 13th International Conference on Computational Linguistics (COLING '90)*, Helsinki, Finland, August.

Bies, Ann, Mark Ferguson, Karen Katz and Robert MacIntyre. 1995. Bracketing Guidelines for Treebank II Style Penn Treebank Project.

Dorr, Bonnie. 1994. Machine Translation Divergences: A Formal Description and Proposed Solution. *Computational Linguistics*, 20(4):597–633.

Dras, Mark. 1999. A meta-level grammar: redefining synchronous TAG for translation and paraphrase. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL '99)*, pages 80–87.

Dras, Mark and Tonia Bleam. 2000. How Problematic are Clitics for S-TAG Translation? In *Proceedings of the Fifth International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+5)*, pages 241–244, Paris, France.

Dras, Mark, David Chiang and William Schuler. 2002. On Relations of Constituency and Dependency Grammars. *Journal of Language and Computation*.

Frank, Robert. 2001. Phrase Structure Composition and Syntactic Dependencies. MS. Johns Hopkins University, June.

Han, Chung-hye, Benoit Lavoie, Martha Palmer, Owen Rambow, Richard Kittredge, Tanya Korelsky, Nari Kim and Myunghee Kim. 2000. Handling Structural Divergences and Recovering Dropped Arguments in a Korean/English Machine Translation System. In John S. White, editor, *Envisioning Machine Translation in the Information Future*, Lecture Notes in Artificial Intelligence. Springer-Verlag, pages 40–53. Proceedings of the Association for Machine Translation in the Americas, AMTA 2000.

Han, Chunghye, Na-Rae Han and Eon-Suk Ko. 2001. Bracketing Guidelines for Penn Korean Treebank. Technical Report IRCS-01-10, Institute for Research in Cognitive Science, University of Pennsylvania.

Han, Chunghye, Juntae Yoon, Nari Kim and Martha Palmer. 2000. A Feature-Based Lexicalized Tree Adjoining Grammar for Korean. Technical Report IRCS-00-04, Institute for Research in Cognitive Science, University of Pennsylvania.

Kasper, Robert, Bernd Kiefer, Klaus Netter and K. Vijay-Shanker. 1995. Compilation of HPSG to TAG. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL '95)*.

Mel'čuk, Igor. 1988. *Dependency syntax: theory and practice* . Albany: State University of NY Press.

Palmer, Martha, Chung-hye Han, Anoop Sarkar and Ann Bies. 2002. Integrating Korean analysis components in a modular Korean/English machine translation system. MS. University of Pennsylvania and Simon Fraser University.

Shieber, Stuart M. 1994. Restricting the weak-generative capability of synchronous tree adjoining grammars. *Computational Intelligence*, 10(4).

Shieber, Stuart M. and Yves Schabes. 1990. Synchronous tree adjoining grammars. In *Proceedings of the 13th International Conference on Computational Linguistics (COLING '90)*, Helsinki, Finland, August.

Xia, Fei, Martha Palmer and Aravind Joshi. 2000. A Uniform Method of Grammar Extraction and Its Applications. In *Proceedings of EMNLP 2000.*

Tectogrammatical Representation: Towards a Minimal Transfer In Machine Translation

Jan Hajič*
Charles University, Prague, Czech Republic
hajic@ufal.mff.cuni.cz

1. Introduction

The Prague Dependency Treebank (PDT, as described, e.g., in (Hajič, 1998) or more recently in (Hajič, Pajas and Vidová Hladká, 2001)) is a project of linguistic annotation of approx. 1.5 million word corpus of naturally occurring written Czech on three levels ("layers") of complexity and depth: morphological, analytical, and tectogrammatical. The aim of the project is to have a reference corpus annotated by using the accumulated findings of the Prague School as much as possible, while simultaneously showing (by experiments, mainly of statistical nature) that such a framework is not only theoretically interesting but possibly also of practical use.

In this contribution we want to show that the deepest (tectogrammatical) layer of representation of sentence structure we use, which represents "linguistic meaning" as described in (Sgall, Hajičová and Panevová, 1986) and which also records certain aspects of discourse structure, has certain properties that can be effectively used in machine translation[1] for languages of quite different nature at the transfer stage. We believe that such representation not only minimizes the "distance" between languages at this layer, but also delegates individual language phenomena where they belong to - whether it is the analysis, transfer or generation processes, regardless of methods used for performing these steps.

2. The Prague Dependency Treebank

The Prague Dependency Treebank is a manually annotated corpus of Czech. The corpus size is approx. 1.5 million words (tokens). Three main groups ("layers") of annotation are used:

- the morphological layer, where lemmas and tags are being annotated based on their context;
- the analytical layer, which roughly corresponds to the surface syntax of the sentence,
- the tectogrammatical layer, or linguistic meaning of the sentence in its context.

In general, unique annotation for every sentence (and thus within the sentence as well, i.e. for every token) is used on all three layers. Human judgment is required to interpret the text in question; in case of difficult decisions, certain "tie-breaking" rules are in effect (of rather technical nature); no attempt has been made to define what type of disambiguation is "proper" or "improper" at what level.

Technically, the PDT is distributed in text form, with an SGML markup throughout. Tools are provided for viewing, searching and editing the corpus, together with some basic Czech analysis tools (tokenization, morphology, tagging) suitable for various experiments. The data in the PDT are organized in such a way that statistical experiments can be easily compared between various systems - the data have been pre-divided into training and two sets of test data.

In the present section, we describe briefly the Prague Dependency Treebank structure and its history.

2.1. Brief History of the PDT

The Prague Dependency Treebank project has started in 1996 formally as two projects, one for specification of the annotation scheme, and another one for its immediate "validation" (i.e., the actual treebanking) in the Institute of Formal and Applied Linguistics, Faculty of Mathematics and Physics at Charles University, Prague. The annotation part itself has been carried out in its Linguistic Data Lab. There has been broad cooperation at

* Supported by the Ministry of Education of the ČR Project LN00A0063 and by the NSF Grant 0121285.

1. We suppose the "classic" design of an MT system, namely, Analysis - Transfer - Synthesis (Generation). Although we believe that overall, our representation goes further than many other syntactico-semantic representations of sentence structure, we are far from calling it an interlingua, since it *can* in general have different realization in different languages for the same sentence.

(1)						
	Od	vlády	čekáme	autonomní	ekologickou	politiku
	od	vláda	čekat	autonomní	ekologická	politika
	RR--2--------	NNFS2-----A--	VB-P---1P-AA-	AAFS4----1A--	AAFS4----1A--	NNFS4-----A--

‘From the-government we-are-awaiting an-autonomous environment policy’

Figure 1: Example morphological annotation: form, lemma, tag

the beginning of the project, especially with the Institute of the Czech National Corpus which (in a similar vein to the British National Corpus) has been constituted at the time as the primary site for collection of and public access to large amounts of Czech contemporary texts[2]. A preliminary version of the PDT (called “PDT 0.5”) has been released in the summer of 1998, the first version containing the full volume of morphological and analytical annotation has been published by the LDC in the fall of 2001 (Hajič *et al.*, 2001). The funding for the project which currently concentrates on the tectogrammatical layer of annotation as described below is secured through 2004.

2.2. The Morphological Layer

The annotation at the morphological layer is an unstructured classification of the individual tokens (words and punctuation) of the utterance into morphological classes (*morphological tags*) and *lemmas*. The original word form is preserved, too, of course; in fact, every token has gotten its unique ID within the corpus for obvious reference reasons. Sentence boundaries are preserved and/or corrected if found wrong (as taken from the Czech National Corpus).

There is nothing unexpected at this level of annotation, since it follows closely the design of the Brown Corpus and of the tagged WSJ portion of the Penn Treebank. However, since it is a corpus of Czech, the tagset size used is 4257, with about 1100 different tags actually appearing in the PDT. The data has been double-annotated fully manually, our morphological dictionary of Czech (Hajič, 2001) has been used for generating a possible list of tags for each token from which the annotators selected the correct interpretation.

There are 13 categories used for morphological annotation of Czech: Part of speech, Detailed part of speech, Gender, Number, Case, Possessor’s Gender and Number, Person, Tense, Voice, Degree of Comparison, Negation and Variant. In accordance with most annotation projects using rich morphological annotation schemes, so-called positional tag system is used, where each position in the actual tag representation corresponds to one category (see Fig. 1).

2.3. The Analytical Layer

At the analytical layer, two additional attributes are being annotated:

- (surface) sentence structure,
- analytical function.

A single-rooted *dependency tree* is being built for every sentence[3] as a result of the annotation. Every item (token) from the morphological layer becomes (exactly) one node in the tree, and no nodes (except for the single “technical” root of the tree) are added. The *order* of nodes in the *original sentence* is being preserved in an additional attribute, but non-projective constructions are allowed (and handled properly thanks to the original token serial number). Analytical functions, despite being kept at nodes, are in fact names of the dependency relations between a dependent (child) node and its governor (parent) node. As stated above, only one (manually assigned) analytical annotation (dependency tree) is allowed per sentence.

According to the pure dependency tradition, there are no “constituent nodes”[4], as opposed e.g. to the mixed representations in the NEGRA corpus (Skut *et al.*, 1997) which contains the head annotation alongside the constituent structure; we are convinced the constituent nodes are in general not needed for deeper analysis, even though we found experimentally that for parsing, some of the annotation typically found at the constituent level might help

2. The ICNC has now over 0.5 billion words of Czech text available.
3. Sentence-break errors are manually corrected at the analytical layer as well.
4. And no equivalent markup either.

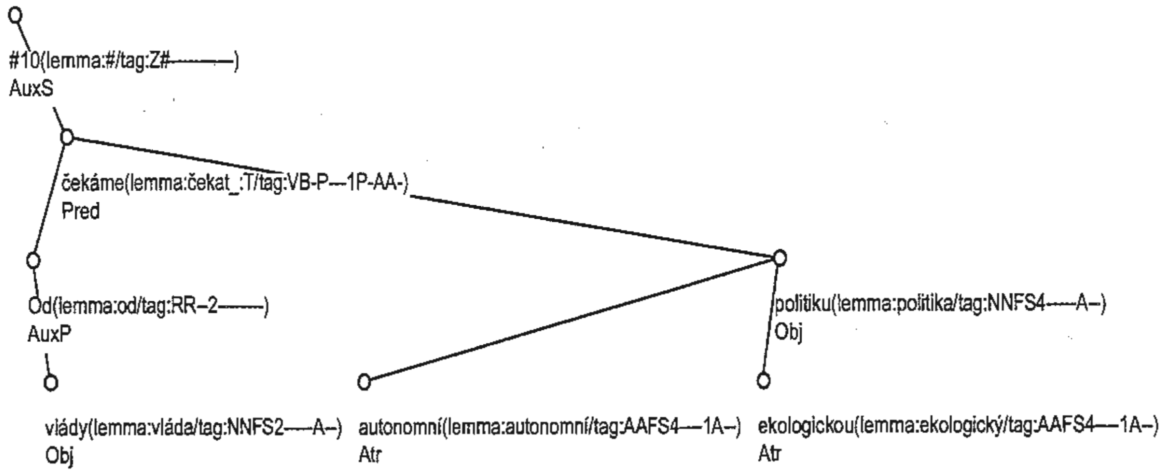

#10(lemma:#/tag:Z#———)
AuxS
čekáme(lemma:čekat_:T/tag:VB-P---1P-AA-)
Pred
Od(lemma:od/tag:RR--2———)
AuxP
politiku(lemma:politika/tag:NNFS4-----A--)
Obj
vlády(lemma:vláda/tag:NNFS2-----A--)
Obj
autonomní(lemma:autonomní/tag:AAFS4----1A--)
Atr
ekologickou(lemma:ekologický/tag:AAFS4----1A--)
Atr

Figure 2: Analytical annotation (sentence from Fig. 1): form, function (+ dependencies, preserved word order)

(such as subordinate clause root markup; for more details, see (Collins *et al.*, 1999)). However, there are still many "technical" dependencies left - we are here at the level of the surface syntax, and there is often no linguistic reason to create a dependency between e.g. an analytical verb form, or a punctuation and everything else, etc.

Coordination and apposition is handled using such "technical" dependencies, too: the conjunction is the head and the members are its "dependent" nodes. Common modifiers of the coordinated structure are also dependents of the coordinating conjunction, but they are not marked as coordinated structure members. This additional "coordinated structure member" markup (`_Co`, `_Ap`) gives an added flexibility for handling such constructions.

Ellipsis is not annotated at this level (no traces, no empty nodes etc.), but a special analytical function (`ExD`) is used at nodes that are lacking their governor, even though they (technically) do have a governor node in the annotation[5].

There are 24 analytical functions used[6], such as `Sb` (Subject), `Obj` (Object, regardless of whether the direct, indirect, etc.), `Adv` (Adverbial, regardless of type), `Pred`, `Pnom` (Predicate / Nominal part of a predicate for the (verbal) root of a sentence), `Atr` (Attribute in noun phrases), `Atv`, `AtvV` (Verbal attribute / Complement), `AuxV` (auxiliary verb - similarly for many other auxiliary-type words, such as prepositions (`AuxP`), subordinate conjunctions (`AuxC`), etc.), `Coord`, `Apos` (coordination/apposition "head"), `Par` (Parenthesis head), etc.

A simple example of the analytical level annotation of the sentence from Fig. 1 is in Fig. 2.

2.4. The Tectogrammatical Layer

The tectogrammatical layer is the most elaborated, complicated but also the most theoretically based layer of syntactico-semantic (or "deep syntactic") representation. The tectogrammatical layer annotation scheme is divided into four sublayers:

- dependencies and functional annotation,
- the topic/focus annotation including reordering according to the deep word order,
- coreference,
- the fully specified tectogrammatical annotation (including the necessary grammatical information).

5. It is the (recursively) closest parent that is physically present in the original sentence.
6. Not counting the additional coordination and special parenthetical markup which effectively triples that number.

As an additional data structure we use a syntactic lexicon, mainly capturing the notion of *valency*. The lexicon is not needed for the interpretation of the tectogrammatical representation itself[7], but it is helpful when working on the annotation since it defines when a particular node should be created that is missing on the surface. In other words, the notion of (valency-based) ellipsis is defined by the dictionary. But before describing the dictionary, let us describe the first ("core") sublayer of annotation.

Dependencies and Fnnctors

The tectogrammatical layer goes beyond the surface structure of the sentence, replacing notions such as "subject" and "object" by notions like "actor", "patient", "addressee" etc. The representation itself still relies upon the language structure itself rather than on world knowledge. The nodes in the tectogrammatical tree are *autosemantic words* only.[8]. Dependencies between nodes represent the relations between the (autosemantic) words in a sentence, for the predicate as well as any other node in the sentence. The dependencies are labeled by *functors*[9], which describe the dependency relations. Every sentence is thus represented as a dependency tree, the nodes of which are autosemantic words, and the (labeled) edges name the dependencies between a dependent and its governor.

Many nodes found at the morphological and analytical layers disappear[10] (such as function words, prepositions, subordinate conjunctions, etc.). The information carried by the deleted nodes is not lost, of course: the relevant attributes of the autosemantic nodes they belong to now contain enough information (at least theoretically) to reconstruct them.

Ellipsis is being resolved at this layer. Insertion of (surface-)deleted nodes is driven by the notion of *valency* (see below the section on Dictionary) and completeness (albeit not in its mathematical sense): if a word is deemed to be used in a context in which some of its valency frames applies, then all the frame's obligatory slots are "filled" (using regular dependency relations between nodes) by either existing nodes or by newly created nodes, and these nodes are annotated accordingly. Textual ellipsis (often found in coordination, direct speech etc.)[11] is resolved by creating a new node and copying all relevant information from its origin, keeping the reference as well.

Every node of the tree is furthermore annotated by such a set of grammatical features that enables to fully capture the meaning of the sentence (and therefore, to recover - at least in theory - the original sentence or a sentence with synonymous linguistic meaning).

The Dictionary (Syntactic, Valency Lexicon)

The tectogrammatical layer dictionary is viewed mainly as a valency dictionary of Czech. By *valency* (as theoretically defined in (Panevová, 1975); for recent account of the computational side and the actual dictionary creation, see (Skoumalová, Straňáková-Lopatková and Žabokrtský, 2001)) we mean the necessity and/or ability of (autosemantic) words to take other words as their dependents, as defined below.

Every dictionary entry is called a *lexia*, which may contain one or more alternative *(valency) frames*. A frame consists of a set of *(valency) slots*. Each slot contains a *function* section (the actual functor, and an indication whether the functor is obligatory[12]), and an associated *form* section. The form section has no direct relation to the tectogrammatical representation, but it is an important link to the analytical layer of annotation: it contains an (underspecified) analytical tree fragment that conforms to the analytical representation of a possible expression of the particular slot. Often, the form section is as simple as a small (analytical) subtree with one (analytical) dependency only, where the dependent node has a particular explicitly specified morphemic case[13]; equally often, it takes the form of a two-edge subtree with two analytical dependencies: one for a preposition (together with its case subcategorization) as the dependent for the surface realization of the root of the lexia itself, and one for the preposition's dependent (which is completely underspecified). However, the form section can be a subtree of any complexity, as it might be the case for phrasal verbs with idiomatic expressions etc.

7. Nor for further analysis (say, a logical one) based on it, nor (in the other direction) for generation (synthesis) of surface sentences.
8. By "autosemantic", as usual, we mean words that have lexical meaning, as opposed to just grammatical function.
9. At two levels of detail; here we ignore so-called *syntactic grammatemes*, which provide the more detailed subclassification.
10. Based on the principle of using only autosemantic words in the representation.
11. Nominal phrases, as used in headings, sports results, artifact names etc. are not considered incomplete sentences, even though they do not contain a predicate.
12. By "obligatory" we mean that this functor (slot) must be present at the tectogrammatical layer of annotation; this has immediate consequences for ellipsis annotation, cf. below.
13. Czech has seven morphemic cases: nominative, genitive, dative, accusative, vocative, locative, and instrumental, usually numbered 1 to 7. In the example in Fig. 1, the case takes the 5th position in the positional representation of the tag.

Moreover, the form section might be different for different expressions (surface realizations) of the lexia itself. For example, if the lexia is a verb and its surface realization is in the passive voice, the form of the (analytical) nodes corresponding to its (tectogrammatical) valency slots will be different than if realized in the active voice. However, relatively simple rules do exist to "convert" the active forms into the passive ones that work for most verbs; therefore, for such verbs, only the canonical (active) forms[14] are associated with the corresponding valency slots. For irregular passivization problems there is always the possibility to include the two (or more) different realizations explicitly into the dictionary.

A similar mechanism is in place for nominalizations. Verbal nouns typically share the function section of the valency frame with their source verbs, but the form section might be a regular or an irregular transform of the corresponding form section. Again, if the necessary transformation is regular, only the canonical form section need to be present (or even no frame at all, if the verb-to-noun derivation is regular in the function section as well).

Other issues are important in the design of the valency lexicon as well, such as reciprocity, information about verbs of control (Panevová, Řezníčková and Urešová, 2002), etc., but they are outside the scope of this rather brief discussion.

The issue of word sense(s) is not really addressed in the valency dictionary. Two lexias might have exactly the same set of valency frames (as defined above, i.e. including the form section(s) of the slot(s)); in such a case, it is assumed that the two words have different lexical meaning (polysemy)[15]. It is rather practical to leave this possibility in the dictionary (however "dirty" this solution is from the purist syntactic viewpoint), since it allows to link the lexias by a single reference to, e.g. the Wordnet senses (Pala and Ševeček, 1999). The lexical (word sense) disambiguation is, however, being solved outside the tectogrammatical level of annotation, even though eventually we plan to link the two, for obvious reasons. Then it will be possible to relate the lexias for one language to another in their respective (valency) dictionaries (at least for the majority of entires). From the point of view of machine translation, this will serve as an additional source of syntactically-based information of form correspondence between the two languages.

Topic, Focus and Deep Word Order

Topic and focus (Hajičová, Partee and Sgall, 1998) are marked, together with so-called deep word order reflected by the order of nodes in the annotation, is in general different from the surface word order, and all the resulting trees are projective by the definition of deep word order.

By *deep word order* we mean such (partial) ordering of nodes at the tectogrammatical layer that puts the "newest" information to the right, and the "oldest" information to the left, and all the rest inbetween, in the order corresponding to the notion of "communicative dynamism". Such an ordering is fully defined at each single-level subtree of the tectogrammatical tree; i.e., all sister nodes *together with their head* are fully ordered left-to-right. The order is relative to the immediate head only; therefore, there exists such a total ordering of the whole tectogrammatical tree that the tree is projective. We believe that tbe deep word order is language-universal for every utterance in the same context, unless, roughly speaking, the structural differences are "too big" (or, in other words, the corresponding translation is "too free").

In written Czech, the surface word order roughly corresponds to the deep word order (with the notable systematic exception of adjectival attributes to nouns, and some others), whereas the grammar of English syntax dictates in most cases a fixed order, and therefore the deep word order will be more often different (even though not always; even English has its means to shuffle words around to make the surface word order closer to the deep one, such as extraposition).

Coreference

Grammatical and some textual coreference is resolved and marked. This is subject to future work, despite some ongoing test annotation. Grammatical coreference (such as the antecedent of "which", "whom", etc., control etc.) is simpler and therefore we believe it will be done more easily and sooner that its textual counterpart. (For more on control in PDT, see (Panevová, Řezníčková and Urešová, 2002) in this volume.)

14. By "form" we mean the analytical tree fragment as defined above.
15. On the other hand, it is clear that two lexias that do *not* share the same set of frames must have different lexical meaning as well, unless truly synonymous at a higher level of analysis.

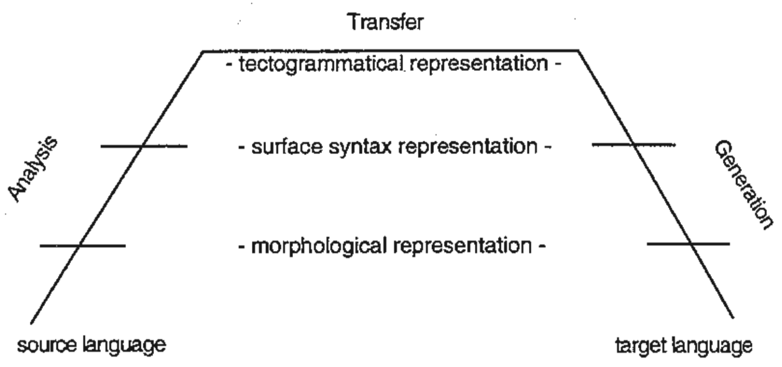

Transfer
- tectogrammatical representation -
- surface syntax representation -
- morphological representation -
Analysis
Generation
source language
target language

Figure 3: Transfer-based MT scheme with three levels of analysis and generation

3. Machine Translation and the Tectogrammatical Layer

The usual scenario of machine translation is Analysis - Transfer - Synthesis (Generation). It is commonly accepted wisdom that the deeper the analysis, the smaller the transfer and vice versa. It is equally clear that the deeper the analysis (and smaller and simpler the transfer), the longer the path from the source to the target languages, and therefore the more errors are likely to creep in. We in principle agree with this, since only careful experiments and variety of evaluations must be run to prove or disprove this. We would like to argue at this point, however, that (even though we have not done such convincing experiments yet), intuitively, there must be an advantage if the transfer end points are defined at a locally clean information saddle point with as least "dirt" from the other language as possible. There has been a number of attempts to use syntactic structure of a sentence to do MT; recently, the most succesful one is statistically based (Yamada and Knight, 2001). We propose here, however, to go to a "deeper" level of analysis.

3.1. The Overall Design

Fig. 3 shows the overall scheme of a transfer-based approach to machine translation. This triangle-based scheme[16] is currently considered the common scheme of all machine translation systems, whether they are of commercial nature (such as (Flanagan and McClure, 2002)) or of research nature ((Brown *et al.*, 1993), (Knight, 1999)) and regardless of their prevailing methodology (with the exception of very few interlingua-based systems (Cavalli-Sforza *et al.*, 2000)).

As Fig. 3 suggests, we propose three essential analysis steps and three generation steps:

- Morphological processing;
- Analytical (surface syntax) processing;
- Tectogrammatical processing (underlying syntax);

and, of course, transfer at the top of the processing "triangle"[17].

An output from one step is the input to the following step; thus we have here four representations of the data along the "up-leading" as well as the "down-leading" paths (from bottom to top):

- The *surface form* of the text (i.e., the actual input and output of the whole system).
- Unstructured *morphological representation* (cf. Sect. 2.2), i.e., an ordered list of lemmas and morphological tags. The order corresponds to the original word order of the sentence.

16. We should rather call it a "trapezoid" scheme, since the top is always cut off in it.
17. Word sense disambiguation (WSD) is not considered a separate step in this scenario, but of course it is taken care of at the tectogrammatical representation level, unless it is already solved while parsing to the tectogrammatical level (based on different valency frames of the words in question).

- Structured *analytical representation* (cf. Sect. 2.3), in the form of a dependency tree that contains all tokens from the morphological layer. Let's summarize that every token is annotated by the lemma and tag coming from the morphological layer, and by a pointer to its governing node and an analytical function naming the dependency relation. The left-to-right order of the nodes of the tree is still coming from the surface sentence word order, therefore causing non-projective trees at times.
- Structured *tectogrammatical representation* (for more details and the four sublayers of annotation, cf. Sect. 2.4) which does *not* contain the word form, lemma, morphological tag, analytical function, nor the surface dependency links. Instead, the tectogrammatical dependency, lexia[18] and the functor is used as the basic information here, supplemented by grammatemes that contain information about number, tense, degree of comparison only where it cannot be recovered from the lexia and function itself. In the full tectogrammatical representation, coreference and deep word order together with topic/focus is annotated as well.

Let us now illustrate how the correspondence among quite distant languages (English, Czech and Arabic) becomes more and more apparent (and straightforward) as we move up the translation "triangle". We will use the sentence *The only remaining baker bakes the most famous rolls north of Long River*, which translates to Czech as *Jediný zbývající pekař peče nejznámější rohlíky na sever od Dlouhé řeky* and to Arabic as (transcribed) *'al-xabbaaz 'al-'axiir 'al-baaqii yaśnacu 'ashhar 'al-kruasaanaat ilaa shimaal min Long River*.

3.2. Surface Form and Morphological Layer Correspondence

Even though the example sentence is quite straightforward to translate (certainly more easily than many sentences in the WSJ), it is clear that there are several unpleasant (non-)correspondences at the surface form, and similarly at the morphological level: articles have no correspondence in the Czech sentence, whereas in the Arabic counterpart, articles are in fact part of the Arabic words. Similarly, the superlative is expressed in Czech by circumfixing, whereas in English it is represented by several words and in Arabic there is a specific single word (*'ashhar*). The Arabic word order is different, too: the word for "baker" (*'al-xabbaaz*) precedes its attributes in the Arabic translation, but follows them in both Czech and English. Therefore methods based on very shallow analysis (i.e., morphological at most) will have trouble (at least) with different word counts, different word order, and, as usual, lexical choice (cf. further below).

3.3. Analytical Layer Correspondence

Fig. 4 shows the corresponding trees. The correspondence of the dependencies is more visible, but since the number of nodes is the same as on the morphological layer, the problems mentioned above did not disappear; on the contrary, the surface structure of the Arabic superlative construction (*<u>'ashhar</u> 'al-kruasaanaat*) even reverses the associated dependency relation (compared to both Czech and English, cf. *the most famous <u>rolls</u>*). Since the original word order is preserved in the analytical dependency tree, the shape of the tree does not correspond even for simple nominal phrases[19]. Overall, even though many dependencies do correspond to each other, there are still many dependencies that either do not correspond to anything in the other language, or are reversed.

3.4. Tectogrammatical Correspondence

Even though there is some similarity between languages at the surface dependency syntax level, the tectogrammatical structure displays often striking similarity, both in the structure and in the functor correspondence (Fig. 5), even though we say again that it is not meant to be an [artificial] interlingua[20].

Analytical Verb Forms

Verbs tend to use various auxiliaries to express person, tense, sometimes number and other morphological properties. We believe, however, that once the person, tense, number etc. is determined (disambiguated), then

18. Recall that "lexia" is the lexical unit reference at the tectogrammatical level, and thus it plays here a role similar to the "lemma" at the morphological and analytical layers.
19. Although the direction of dependencies does (*remaining* depends on *baker*, similarly in Arabic *'al-baaqii* depends on *'al-xabbaaz* and in Czech *zbývající* depends on *pekař*).
20. For example, compare the difference in the structure for "I like swimming" in English and "Ich schwimme gern" in German.

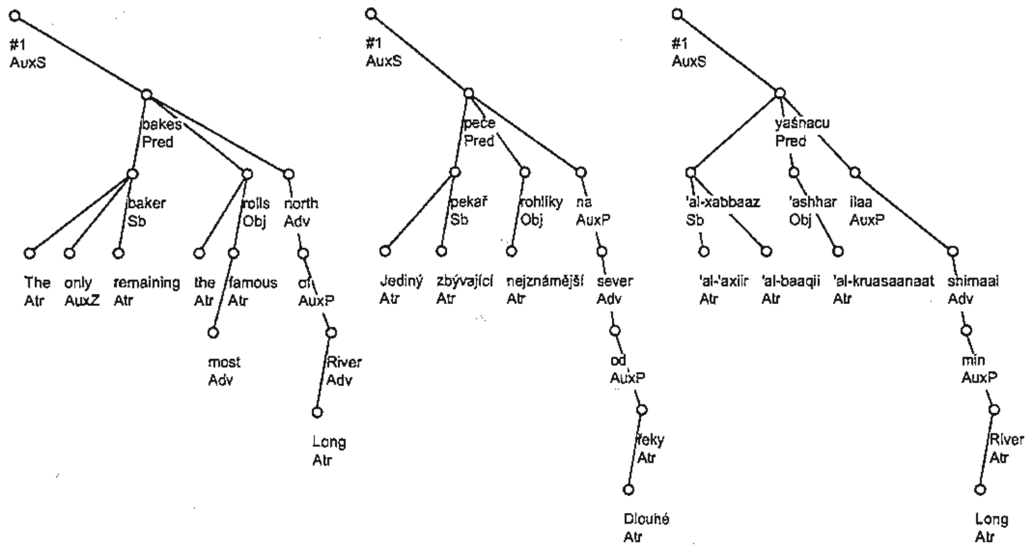


Figure 4: Analytical layer correspondence: *The only remaining baker bakes the most famous rolls north of Long River* in English, Czech and Arabic

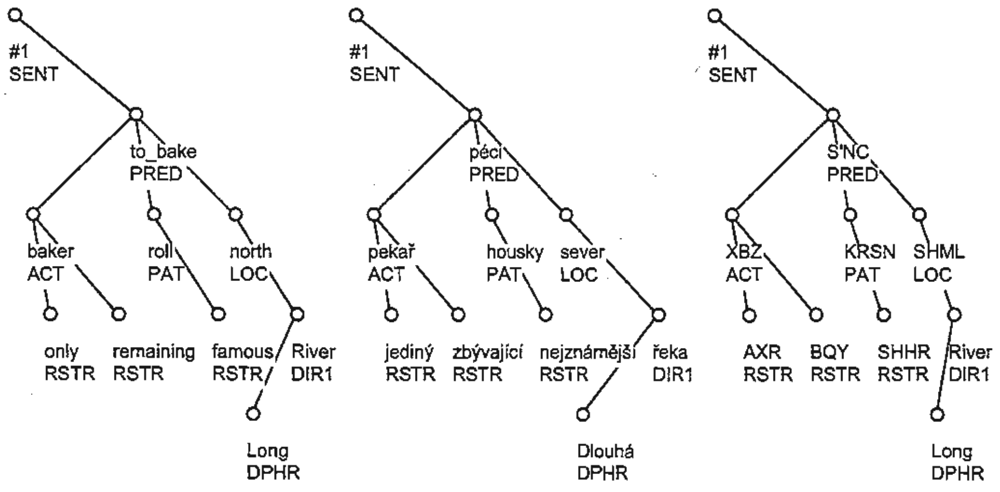


Figure 5: Tectogrammatical layer correspondence: *The only remaining baker bakes the most famous rolls north of Long River*, again in English, Czech and Arabic; for transparency reasons, only the lexia and functor are shown, and not grammatemes such as number.

there is no need to have separate nodes for each of the auxiliaries. The auxiliaries are completely determined by the language in question; therefore, we must be able to handle the insertion of appropriate auxiliaries at the generation stage, which is by its nature already monolingual. In the context of machine translation, this is especially useful: we have to take care of the main (autosemantic) verb only during the translation proper (the transfer phase), but not of the (presence or absence) of auxiliary source words. For example, the type of auxiliary in German perfect tense (*sein/haben*) is grammatically (or, lexically) based and has nothing to do with the other language, since that language might use quite different auxiliaries (or none at all, if it uses inflection to express the perfect tense).

Articles

Articles pose a difficult problem for translation into a language that has articles (such as English) from a language that does not (such as Czech). The choice of an article is hardly conditioned by words of the other language; rather, it is either determined grammatically, referentially, or by the topic/focus distinction described earlier. We thus believe that articles, as another class of non-autosemantic words, has no place at the tectogrammatical layer of representation, since the topic/focus and deep word order should be sufficient (together with the grammar rules of the target language) to insert the right articles at the right places. For example, the need to use "the" in front of every superlative is purely English-grammar-related and certainly does not stem from the language being translated from or to; the choice of "a" vs. "the" for a general word such as "keyboard" will be determined by the topic/focus annotation: if the word "keyboard" is in the topic, the definite article ("the") typically has to be used, otherwise "a" should be used instead.

Choice of Prepositions (and Morphemic Case)

Prepositions usually do exist across languages, even though they are not always used as separate words (cf. Hungarian and other agglutinative languages), and often a "default" translation can be found for every preposition. However, from the experience with inflective languages such as Czech, we consider prepositions and morphemic cases to be at the same "level" - if not just a form variant - expressing a particular tectogrammatical functor[21]. Therefore, when translating into English, we have to select prepositions, when translating e.g. into Czech we have to decide the case or preposition[22].

Even then, the relation between functors and prepositions/cases is not always straightforward, for at least two reasons:

- The choice of preposition is driven by usage in the target language (e.g., it depends on the noun used with the preposition or on some similar factor);

- The choice of preposition/case is driven by the governing word *and* by the functor of the dependent word (i.e. the one that has to get the preposition/case).

In both cases, the source language sentence representation does not help much. In the first case, we simply have to have a language model or similar knowledge of the target language[23] that simulates usage. In the second case, a valency dictionary of the target language (as defined in Sect. 2.4) comes in handy: once we are able to determine the correct target word (more precisely, the lexia as the translation of the source lexia), a valency dictionary entry gives matching functors and with each of them, its surface expression (by means of an underspecified analytical-level annotated subtree, mostly either just a case or a preposition with its own subcategorization for a morphemic case).

For words not having valency, their dependent nodes (as well as dependent nodes of all words with non-valency modifier functors) acquire their preposition or case as the default value for each functor.

21. Some regular "transformations" notwithstanding, such as in passivization, where the surface syntax expression also plays a role.
22. Prepositions have subcategorization for case, so for subcategorization-ambiguous prepositions the correct subcat frame must be selected together with the preposition.
23. A good language model (as used in automatic speech recognition systems) can actually help in many cases of target-language-related conditioning.

Word Order

Word order differs across languages, of course, sometimes wildly. English has its word order mostly grammatically given (meaning that the grammar dictates that sentences should in the SVO order, that the rules for systemic ordering must be followed, etc.); some exceptions in the grammar do allow for some word shuffling, such as extraposition. However, Czech word order is discourse-driven (and thus not so "free" as often mislabeled). The correct solution, in our opinion, to the word order problem is thus not to deal with it at the transfer level, but at the analysis level (determining the deep word order), and at the generation stage (using the determined deep word order to perhaps generate an extraposition, and using the grammar rules[24] of the target language to determine the correct word order).

Agreement (in the Generation Stage)

Grammatical agreement is again determined by the rules of the target language, and not by the translation itself. Its importance in English is low, obviously, but it is crucial for other languages. E.g., in Czech, every adjective has to agree in gender, number and case with its head noun. We propose to deal with this problem at the analytical level, once the analytical tree is built (which includes solving the word order issue, of course); it is not related to the tectogrammatical level in fact. Thus, for example, only the number is needed to be preserved (translated) at the tectogrammatical level[25], its dependent adjectives will be the populated by the correct morphemic values once also the case is determined by the rules described above, and once the gender of the noun is determined from the lexicon. The formation of the surface text is then easy through any morphological generator of the target language, since the word order has been defined in the preceding stages.

4. Conclusion

We have described the basic ideas and annotation scheme for the Prague Dependency Treebank, a reference corpus with three-level linguistic annotation for morphology, surface syntax, and so-called tectogrammatical layer representation. We have then argued that the tectogrammatical layer is suitable not only for various linguistic experiments, but also for practical use, specifically for machine translation systems, since it generalizes (and disambiguates) in such a way that it achieves - to a certain extent limited by "language meaning" - independence of both the source and target languages. We believe that our representation has the potential to improve the overall translation quality, and that the additional burden of deeper analysis will not outweight its benefits.

References

Brown, Peter, Stephen Della Pietra, Vincent Della Pietra and Robert Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(2).

Cavalli-Sforza, Violetta, Krzystof Czuba, Teruko Mitamura and Eric Nyberg. 2000. Challenges in Adapting an Interlingua for Bidirectional English-Italian Machine Translation. In *AMTA 2000*.

Collins, Michael, Jan Hajič, Eric Brill, Lance Ramshaw and Christopher Tillmann. 1999. A Statistical Parser for Czech. In *37th Meeting of the Association of Computational Linguistics*, pages 505–512, University of Maryland, College Park, MD, June 22nd–25th.

Flanagan, Mary and Steve McClure. 2002. SYSTRAN and the Reinvention of MT. In electronic form only, at SYSTRAN's web pages (`http://www.systransoft.com/IC/26459.html`).

Hajič, Jan. 1998. Building a Syntactically Annotated Corpus: The Prague Dependency Treebank. In *Festschrift for Jarmila Panevová*. Karolinum, Charles University Press, Prague, pages 106–132.

Hajič, Jan. 2001. *Disambiguation of Rich Inflection (Computational Morphology of Czech)*. Prague, Czech Republic: Faculty of Math. and Physics, Charles University. hab. thesis.

Hajič, Jan, Eva Hajičová, Petr Pajas, Jarmila Panevová, Petr Sgall and Barbora Vidová Hladká. 2001. The Prague Dependency Treebank. CD-ROM Catalog #LDC2001T10. ISBN 1-58563-212-0.

Hajič, Jan, Petr Pajas and Barbora Vidová Hladká. 2001. The Prague Dependency Treebank: Annotation Structure and Support. *IRCS Workshop on Linguistic Databases*, pages 105–114.

Hajičová, Eva, Barbara Partee and Petr Sgall. 1998. *Topic-Focus Articulation, Tripartite Structures and Semantic Content*. Dordrecht, Amsterdam, Netherlands: Kluwer Academic Publishers.

Knight, Kevin et al. 1999. EGYPT: a statistical machine translation toolkit. Summer Workshop'99, Johns Hopkins University, Baltimore, MD, `http://www.clsp.jhu.edu/ws99/projects/mt/toolkit`.

24. By grammar rules we mean here any kind of "rules"; it is expected that these rules will be learned within a statistical modeling framework.
25. One of the so-called grammatemes - invisible in our examples above - is devoted to this.

Pala, Karel and Pavel Ševeček. 1999. Final Report EuroWordNet-1,2, project LE4-8328. Technical report, EU Commission, Amsterdam, Sept. 1999. On the project CD-ROM.

Panevová, Jarmila. 1975. On Verbal Frames in Functional Generative Description. *Prague Bulletin of Mathematical Linguistics (PBML)*, 22 (Part I), 23 (Part II):3–40,17–52.

Panevová, Jarmila, Veronika Řezníčková and Zdeňka Urešová. 2002. The Theory of Control Applied in Tagging of the Prague Dependency Treebank. In Robert Frank, editor, *TAG+6 Workshop (this volume)*, Venice, May 20-23, 2002. Univ. of Pennsylvannia.

Sgall, Petr, Eva Hajičová and Jarmila Panevová. 1986. *The Meaning of a Sentence in its Semantic and Pragmatic Aspects.* Prague - Amsterdam: Academia - North–Holland.

Skoumalová, Hana, Markéta Straňáková-Lopatková and Zdeněk Žabokrtský. 2001. Enhancing the Valency Dictionary of Czech Verbs: Tectogrammatical Annotation. In *Text, Speech and Dialogue, Lecture Notes on Computer Science LNCS 2166*, pages 142–147, Železná Ruda, Czech Rep., Sept. 2001. Springer-Verlag.

Skut, Wojciech, Brigitte Krenn, Thorsten Brants and Hans Uszkoreit. 1997. An Annotataion Scheme for Free Word Order Languages. In *the Fifth Conference on Applied Natural Language Processing (ANLP-97)*, Washington, D.C., USA, March 1997. Association of Computational Linguistics.

Yamada, Kenji and Kevin Knight. 2001. A Syntax-Based Statistical Translation Model. In *39th Meeting of the Association of Computational Linguistics*, Toulouse, France, July 2001.

Clustering for Obtaining Syntactic Classes of Words from Automatically Extracted LTAG Grammars

Tadayoshi Hara[†], Yusuke Miyao[†], Jun'ichi Tsujii[†‡]
[†]*University of Tokyo* [‡]*CREST, JST (Japan Science and Technology Corporation)*

1. Introduction

We propose a method for obtaining syntactic classes of words from a lexicalized tree adjoining grammar (LTAG: Schabes, Abeillé and Joshi (1988)) automatically extracted from a corpus. Since *elementary trees* in LTAG grammars represent syntactic roles of a word, we can obtain syntactic classes by clustering words having the similar elementary trees. With our method, automatically extracted LTAG grammars will be arranged according to the syntactic classes of words, and the grammars can be improved from various points of view. For example, we can improve the coverage of the grammars by supplementing to a word the elementary trees of the syntactic class of the word.

An LTAG grammar consists of *elementary trees*, which determine the position where the word can be put in a sentence, that is, an elementary tree corresponds to a certain syntactic role. Hence, a syntactic class of a word is represented as a set of elementary trees assigned to the word. Since the words of the same syntactic class are expected to have similar elementary trees, we can obtain syntactic classes by clustering words having similar sets of elementary trees.

We applied our clustering algorithm to an LTAG grammar automatically extracted from sections 02–21 of the Penn Treebank (Marcus, Santorini and Marcinkiewicz (1994)), and investigated the obtained clusters with changing the number of clusters. We successfully obtained some of the clusters that correspond to certain syntactic classes. On the other hand, we could not obtain some clusters, such as the one for ditransitive verbs, and obtained the clusters that we could not associate clearly with syntactic classes. This is because our method was strongly affected by the difference in the number of words in each part-of-speech class. We concluded that, although our clustering method needs to be improved for practical use, it is effective to automatically obtain syntactic classes of words.

The XTAG English grammar (The XTAG Research Group (1995)) is a handmade LTAG grammar which is arranged according to syntactic classes of words, "*tree families*." Each tree family corresponds to a certain subcategorization frame, and determines elementary trees to be assigned to a word. Thanks to the tree families, the XTAG grammar is independent of a corpus. However, it needs considerable human effort to manually construct such a grammar.

Automatically extracted LTAG grammars are superior to manually developed grammars in the sense that it takes much less costs to construct the grammars. Chiang (2000) and Xia (1999) gave the methods of automatically extracting LTAG grammars from a bracketed corpus. They first decided a *trunk* path of the tree structure of a bracketed sentence, and the relationship (substitution or adjunction) between the trunk and branches. The methods then cut off the branches of them according to the relationship. Because the sentences used for extraction are in real-world texts, extracted grammars are practical for natural language processing. However, automatically extracted grammars are not systematically arranged according to syntactic classes their *anchors* belong to, like the XTAG grammar. Because of this, automatically extracted grammars tend to be strongly dependent on the corpus. This limitation can be a critical disadvantage of such extracted grammars when the grammars are used for various applications. Then, we want to arrange an extracted grammar according to the syntactic classes of words, without loosing the benefit for the cost.

Chen and Vijay-Shanker (2000) proposed the solution to the issue. To improve the coverage of an extracted LTAG grammar, they classified the extracted elementary trees according to the tree families in the XTAG English grammar. First, the method searches for a tree family that contains an *elementary tree template* of extracted elementary tree *et*. Next, the method collects other possible tree templates in the tree family and makes elementary trees with the anchor of *et* and the tree templates. By using tree families, the method can add only proper elementary trees that correspond to the syntactic class of anchors. Chen and Vijay-Shanker (2000) applied this method to an extracted LTAG grammar, and showed the improvement of the coverage of the grammar. Although their method showed the effectiveness of arranging a grammar according to syntactic classes, the method depends on

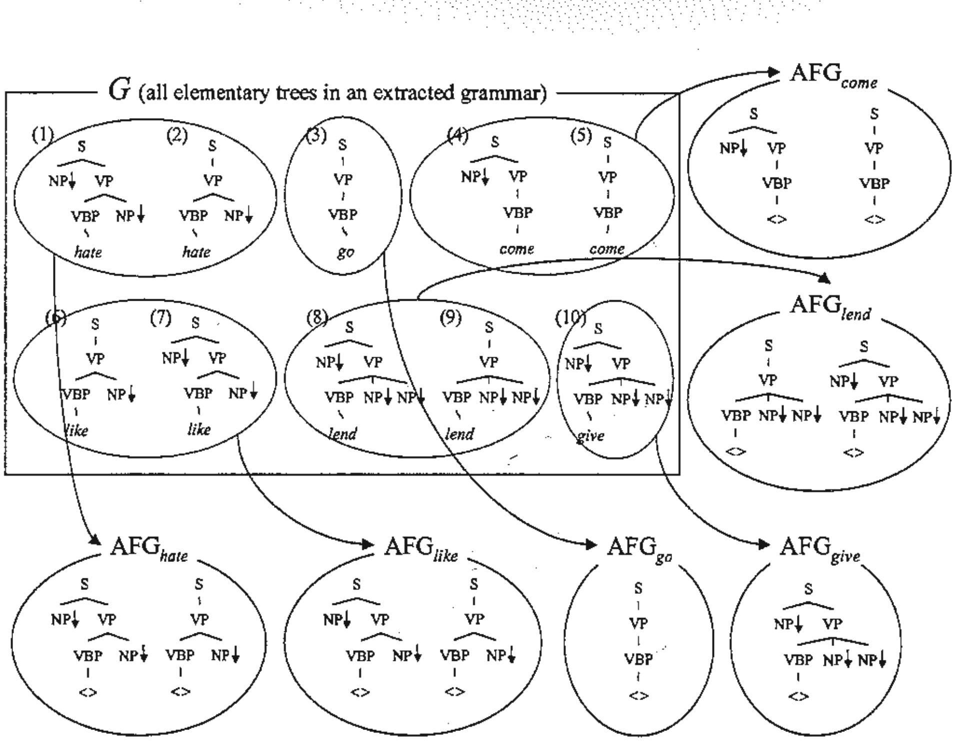


Figure 1: Obtaining AFGs from an extracted LTAG grammar

a manually developed grammar that takes considerable effort to construct. Since our method can automatically obtain syntactic classes, we can obtain this benefit without the considerable cost.

2. Methods

In this section, we give a method for automatically obtaining syntactic classes of words from an automatically extracted LTAG grammar. We first give the idea for our method, and then trace each step of our method with examples and formalization.

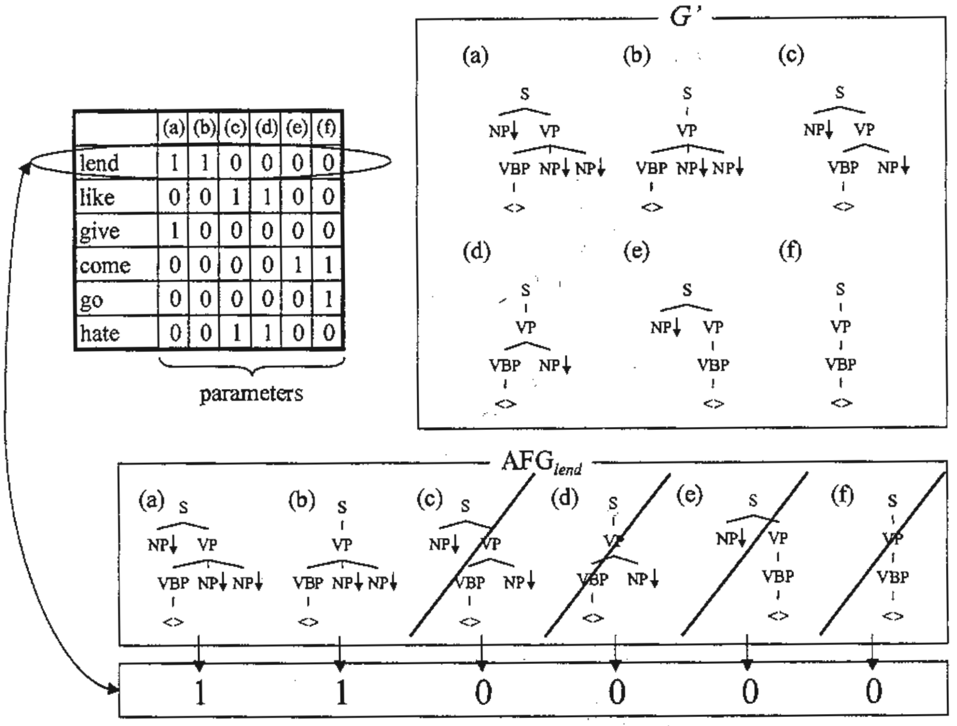
In LTAG grammars, syntactic roles are represented as elementary trees, and as in the XTAG grammar, a syntactic class of a word determines elementary trees to be assigned to the word. Given set W of words and set T of elementary tree templates, LTAG grammar G is defined as a subset of the product of them $G \subset T \times W$ that satisfies the following equation.

$$G = \{(t,w)|w \in W, t \in F(s(w))\} \tag{1}$$

where function s gives a syntactic class of word w, and function F gives a set of tree templates allowed by the syntactic class. Handmade LTAG grammars follow this formalization, for example, in the XTAG grammar, syntactic classes are represented as "tree families." However, automatically extracted LTAG grammars are not arranged according to syntactic classes, and lack elementary trees that do not appear in the training corpus. Therefore, we need to obtain s and F automatically.

The idea to achieve this task can be derived from the equation 1. From this formalization, we can see that words of the same syntactic class should have the same set of elementary trees. This indicates that, even when the grammar lacks some elementary trees, we can obtain syntactic classes by collecting words having the similar elementary trees. To achieve this, we apply a clustering method. First, we make *anchor's feature groups (AFGs)*, which represent possible syntactic roles of the word, and are objects for clustering. Next, we apply a clustering method to collect similar AFGs, and finally, we interpret obtained clusters.

The first step in our method is to make AFGs. An AFG for a word is a set of all tree templates assigned for the word. Figure 1 shows an example of AFGs. For example, an AFG for "*lend*" can be obtained by collecting all elementary trees for "*lend*" in the grammar, that is (8) and (9), and removing the anchor "*lend*" from them. The obtained tree templates correspond to the syntactic roles, for example, a declarative and an imperative for



	(a)	(b)	(c)	(d)	(e)	(f)
lend	1	1	0	0	0	0
like	0	0	1	1	0	0
give	1	0	0	0	0	0
come	0	0	0	0	1	1
go	0	0	0	0	0	1
hate	0	0	1	1	0	0

Figure 2: Parameters of AFGs for clustering

"ditransitive verbs." Formally, an AFG for word w is defined as follows.

$$A_G(w) = \{t | t \in T, (t, w) \in G\} \tag{2}$$

Suppose that we have "perfect" LTAG grammar G_P which consists of all elementary trees for composing syntactically correct sentences. From equation 1 and 2, we get

$$\forall w \in W \; F(s(w)) = A_{G_P}(w)$$

Therefore, we can obtain a syntactic class of word w from the AFG for w.

$$\forall w \in W \; s(w) = F^{-1}(A_{G_P}(w))$$

However, automatically extracted LTAG grammar G_E lacks some elementary trees, because a corpus for training does not contain all elementary trees of words[1]. Therefore, we cannot obtain syntactic classes of words just as above. However, we can assume that an extracted LTAG grammar is very similar to a perfect LTAG grammar, and an AFG for a word in G_E should be very similar to an AFG for the word in G_P, and not to other AFGs in G_P at all. Formally,

$$\underset{A_{G_P}(w')}{\operatorname{argmin}} \, d(A_{G_E}(w), A_{G_P}(w')) = A_{G_P}(w) = F(s(w)) \tag{3}$$

Function d gives a distance measure that indicates how given two sets are different. In addition, we can see that the words of the same syntactic class should have similar AFGs. Therefore, we can find the syntactic class by collecting words having similar AFGs.

In order to collect similar AFGs, we next apply a clustering method, *the K-means* (MacQueen (1967)), which groups "*objects*" close to each other. The distance between two objects is given by Euclidean distance between the two "*parameters*" of the objects. In our method, the "*object*" for clustering is a word characterized by a certain

1. Actually, an extracted LTAG grammar may contain some improper elementary trees which can not make a syntactically correct sentence. But in this discussion, we consider that there are not any such elementary trees in the grammar.

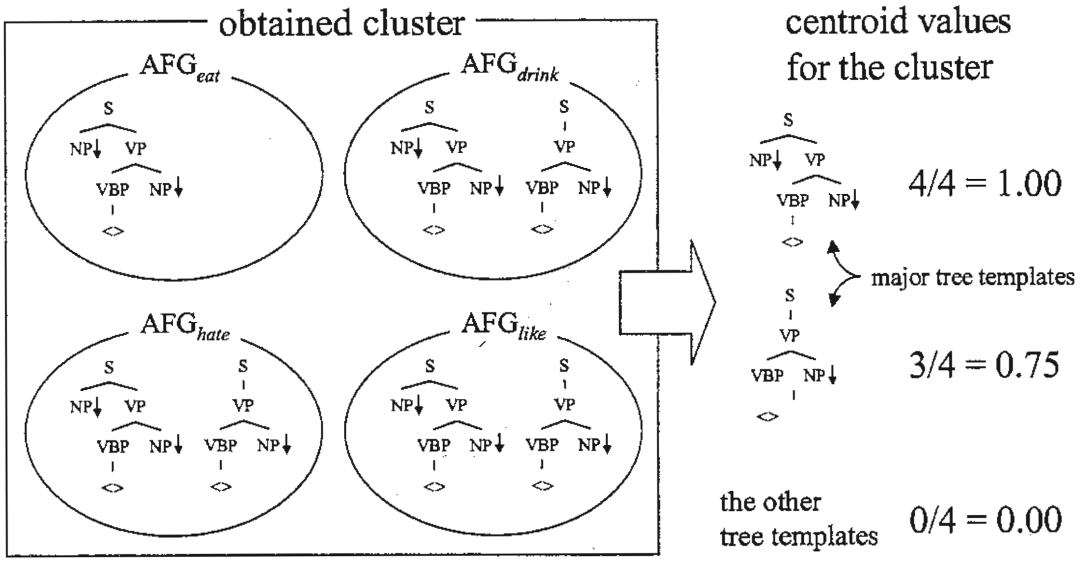


Figure 3: Centroid values for an obtained cluster

AFG, and the "*parameters*" for it are binary values for whether each tree template in the grammar is in the AFG or not. A table in Figure 2 shows example parameters. The grammar is the same as the example in Figure 1. A sequence of 0s and 1s in each row is a set of parameters for an AFG. For example, the AFG for "*lend*" contains tree templates (a) and (b), and therefore parameters for the two tree templates are both 1. The other parameters for AFG "*lend*" are all 0, because it does not contain the other tree templates in the extracted LTAG grammar.

The relation between obtained clusters and syntactic classes can be formalized as the following. The equation 3 shows our assumption that leads to the clustering method. We assume that the distance measure d of two sets, $T', T'' \subset T$ can be given as follows:

$$d(T', T'') = ||\vec{p}(T') - \vec{p}(T'')||$$

$$\vec{p}(T') = (x_i) \text{ where } x_i = \begin{cases} 1 & if\ t_i \in T' \\ 0 & otherwise \end{cases}$$

The vector $\vec{p}(T')$ is a parameter for AFG T'. According to the definition of d, equation 3 can be rewritten as follows.

$$\underset{A_{G_P}(w')}{\text{argmin}}\ ||\vec{p}(A_{G_E}(w)) - \vec{p}(A_{G_P}(w'))|| = A_{G_P}(w) \tag{4}$$

On the other hand, by clustering AFGs with parameters as defined above, the K-means algorithm makes clusters that satisfy the following equation:

$$\underset{centroid(c')\ \text{s.t.}\ c' \in C}{\text{argmin}}\ ||\vec{p}(A_{G_E}(w)) - \vec{p}(centroid(c'))|| = centroid(c) \text{ s.t. } A_{G_E}(w) \in c \tag{5}$$

where C is the set of all obtained clusters. *centroid*(c) is a "*centroid*" of the cluster c, which is an imaginary object with the average of the parameters for objects in c:

$$\vec{p}(centroid(c)) = \frac{\sum_{A_{G_E}(w) \in c} \vec{p}(A_{G_E}(w))}{|c|}$$

Comparing the equations 4 and 5, we can consider that *centroid*(c) corresponds to $A_{G_P}(w)$. This suggests that the obtained clusters of words correspond to sets of words which are in the same syntactic class.

At the last, we need to interpret obtained clusters. After the clustering, the method obtains words that have similar sets of tree templates, and the tree templates will indicate a syntactic class of the words. In particular, the set of tree templates common to most of those words are certain to correspond to the syntactic class of the words. Such

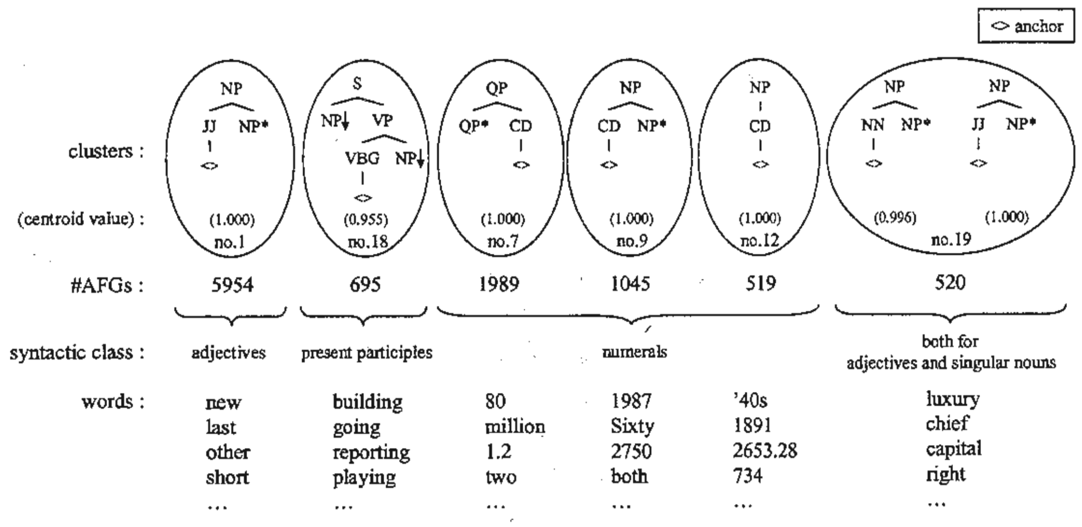
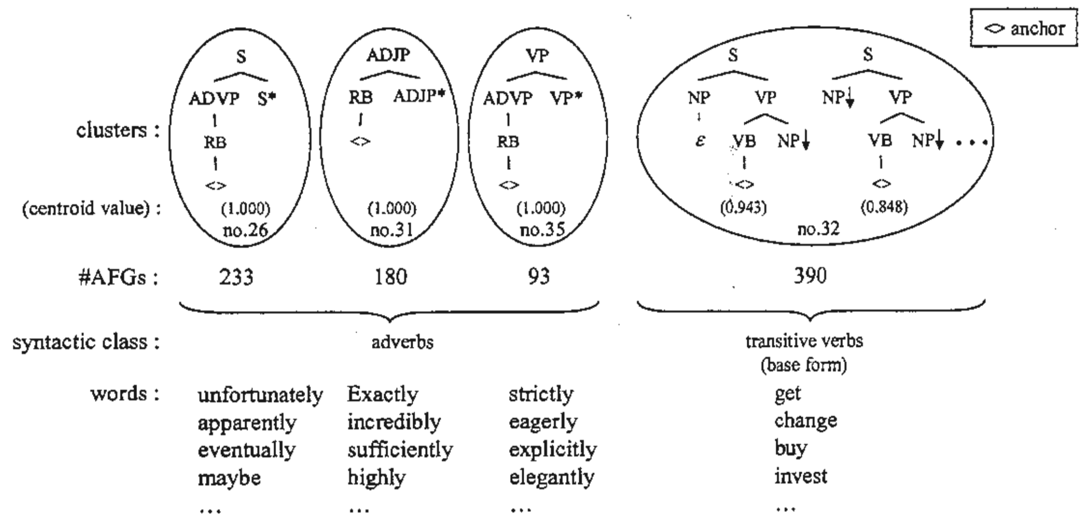


Figure 4: Clusters generated in the case of 20 clusters



<> anchor
clusters :
S ADVP S* RB <>
ADJP RB ADJP* <>
VP ADVP VP* RB <>
S NP VP ε VB NP↓ <> S NP↓ VP VB NP↓ <> ...
(centroid value) :
(1.000) no.26
(1.000) no.31
(1.000) no.35
(0.943) (0.848) no.32
#AFGs :
233
180
93
390
syntactic class :
adverbs
transitive verbs (base form)
words :
unfortunately apparently eventually maybe ...
Exactly incredibly sufficiently highly ...
strictly eagerly explicitly elegantly ...
get change buy invest ...

Figure 5: Newly classified syntactic classes in the case of 40 clusters

tree templates can be obtained by interpreting each element of the parameter for the centroid, a *centroid value*, as the probability that each tree template corresponds to a part of a syntactic class. We call such tree templates "*major tree templates*", and use them to associate clusters to syntactic classes of words. The left-hand side of Figure 3 shows an example of an obtained cluster. In this cluster, major tree templates represent the syntactic roles of a "transitive verb," and therefore, we can interpret the cluster as a "transitive verb" class.

3. Experiments

Our clustering algorithm was applied to an LTAG grammar automatically extracted from sections 02–21 of the Penn Treebank (Marcus, Santorini and Marcinkiewicz (1994)). The grammar is extracted by an algorithm similar to the one in Xia (1999). From 39,598 sentences, 2,571 elementary tree templates are extracted for 43,030 words. Accordingly 43,030 AFGs were obtained by our method. We then classified the obtained AFGs into clusters. Since we have no knowledge how many clusters are suitable for this task, we varied the number of clusters over 20, 40, 60, 80, and 100.

Increasing the number of clusters, we could observe how the classes of syntactic roles were being obtained in a more detailed way. In the case of 20 clusters shown in Figure 4, for example, the cluster No. 1 was for the

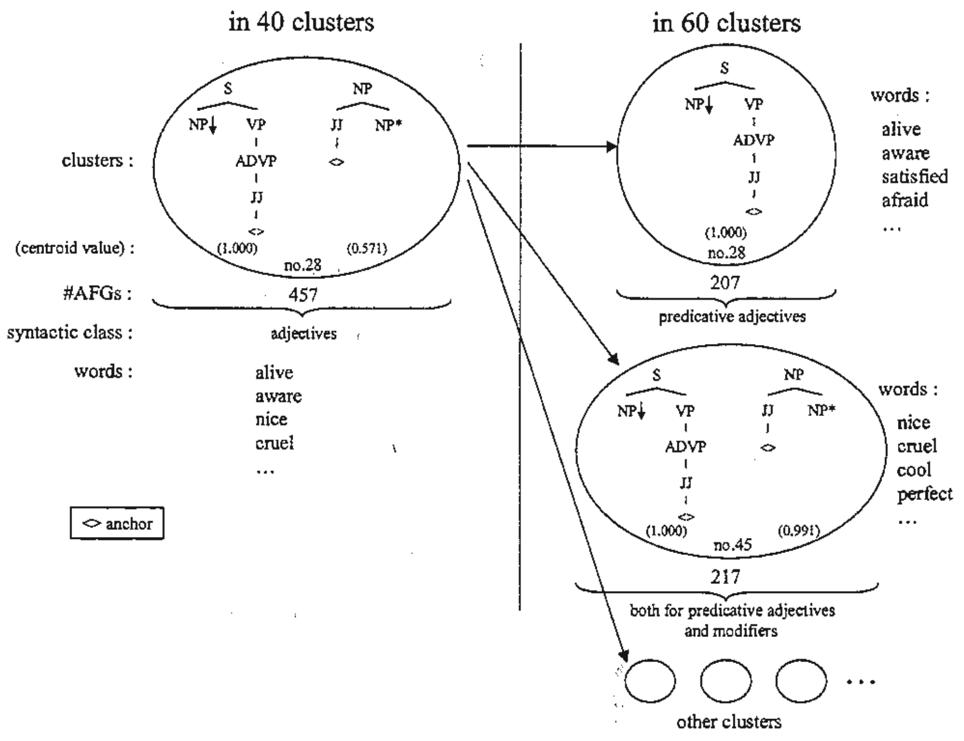


Figure 6: A divided class of syntactic roles

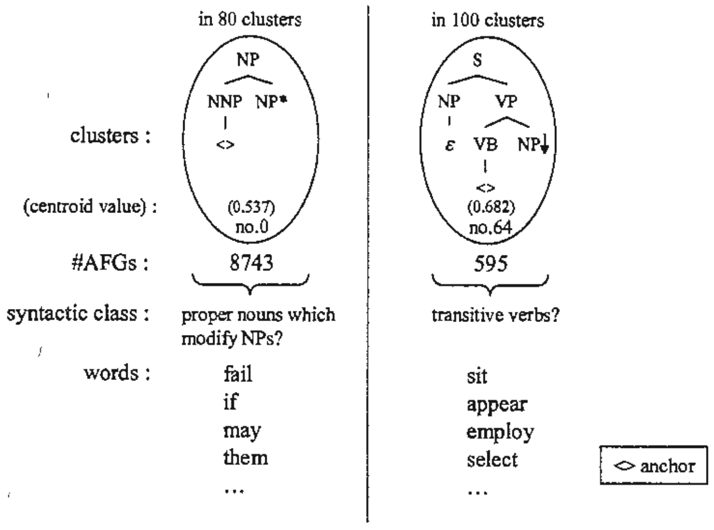


Figure 7: Ambiguous clusters

AFGs for adjectives; the cluster No. 18 was for present participles; the cluster No. 7, No. 9 and No. 12 were for numerals; the cluster No. 19 was for the words which can be both adjectives and singular nouns. In the case of 40 clusters, the clusters for more detailed syntactic classes were generated. For example, the clusters No. 26, 31, and 35 shown in Figure 5 were for adverbs that mainly modify a sentence, an adjective, a verb, respectively; the cluster No. 32 was for base form transitive verbs.

In addition, we could observe that there were some clusters made by dividing one cluster in the case of less clusters, and in both of which syntactic classes could be identified. For example (Figure 6), the cluster No. 28 in the case of 40 clusters was mainly for adjectives. About a half of AFGs in it were only for predicative adjectives and the rest of them were not. In the case of 60 clusters, the two types of AFGs were divided almost into two clusters, No. 28 and No. 45. We could see many AFGs only for predicative adjectives, such as "aware," "glad," "alive," in the cluster No. 28. In the other cluster (No. 45), we could see many AFGs for adjectives which could be both predicative adjectives and modifiers, such as "wrong," "hot," "certain." These clusters were generated by classifying the AFGs for one syntactic class in a more detailed way by increasing the number of clusters.

In the case of 80 clusters, there were some clusters whose centroid values of tree templates were no more than 0.5 (Figure 7). This meant that there was no tree template which was common to all AFGs in such clusters, and we could not identify syntactic classes for them. This would suggest that the number of clusters exceeded the one suitable for clustering the AFGs, and in consequence the AFGs were classified too finely.

However, this would not indicate that there would be less than 80 classes of syntactic roles in the grammar. When we focused on nouns, we could find too fine classification for them. In the case of 100 clusters, the number of them was no less than 37. On the other hand, clusters for verbs were few. In the case of 100 clusters, the number of them was 20, and expected clusters such as one for ditransitive verbs were not in those 20 clusters. The reason for this would be as follows.

A noun class contains many words, and AFGs for nouns in the grammar reflected this fact; the AFGs for nouns occupied about $18,427/43,030$ of all the AFGs in the grammar. On the other hand, a verb class contains not so many words and AFGs for verbs occupied $1,190/43,030$. Our clustering method treated all words equally regardless of parts-of-speech, and as a result, words in the noun class would be classified too finely, and words in the verb class, too roughly.

4. Conclusion

We proposed the method for automatically obtaining syntactic classes of words from automatically extracted LTAG grammars. We supposed that the class of the syntactic roles of a word corresponded to the set of elementary trees for the word, and attempted to obtain syntactic classes by clustering words that have similar elementary trees. The experiments showed that our method could obtain some clusters each of which represents a certain syntactic class. However, it also showed that our method would not obtain all syntactic classes properly, because it was affected by the differences in the number of words in each part-of-speech class. We should consider this result, and build an algorithm that would not mix various parts-of-speech, but would cluster groups of elementary trees for them separately. If we can obtain proper classes of syntactic roles, we will be able to apply various methods that improve and make good use of the extracted grammar. For example, we will be able to properly predict elementary trees which are not in the extracted grammar, by giving major tree templates to all words in the cluster. Such predicted elementary trees will improve the coverage of the grammar in a syntactically proper way.

References

Chen, John and K. Vijay-Shanker. 2000. Automated Extraction of TAGs from the Penn Treebank. In *Proc. of the 6th IWPT*.

Chiang, David. 2000. Statistical parsing with an automatically-extracted tree adjoining grammar. In *Proc. of the 38th ACL*, pages 456–463, October.

MacQueen, J. B. 1967. Some methods of classification and analysis of multivariate observations. In *Proc. of the Fifth Berkeley Symposium on Mathemtical Statistics and Probability*.

Marcus, Mitchell, Beatrice Santorini and Mary Ann Marcinkiewicz. 1994. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Schabes, Yves, Anne Abeillé and Aravind K. Joshi. 1988. Parsing strategies with 'Lexicalized' grammars: Application to Tree Adjoining Grammars. In *Proc. of the 12th COLING*, pages 578–583.

The XTAG Research Group. 1995. A Lexicalized Tree Adjoining Grammar for English. Technical Report IRCS 95-03, Institute for Research in Cognitive Science, University of Pennsylvania.

Xia, Fei. 1999. Extracting Tree Adjoining Grammars from Bracketed Corpora. In *Proc. of the 5th NLPRS*.

A New Metagrammar Compiler

B. Gaiffe, B. Crabbé, and A. Roussanaly
Loria

1. Why a new metagrammar compiler ?

Writing a TAG grammar manually is known to be a non trivial task (Abeillé 00; Doran et al. 94; Doran et al. 00; VS et al. 92). For that purpose, (Candito 96) has suggested a grammatical framework that allows linguists to describe the syntactic properties of a language at a higher level of abstraction . Given a set of classes, each of these containing a partial tree description, the compiler outputs a set of tree schemata.

In order to work on the organization of a syntactic lexicon, we needed an almost equivalent tool that would produce feature structures together with the tree schematas. Before developping a new tool, we criticized Candito's work (Candito 99) in considering the two following drawbacks:

1. the algorithm is closely linked to the specific linguistic description. As her analysis focusses on the study of verbal trees, the structuration is badly adapted to the description of non verbal units ;
2. the current implementation of the compiler is not flexible enough to be easily adapted to other input/output formats.

In the remainder of the paper we describe our tool and discuss two possible ways to implement a metagrammar for french verbs.

2. Design of our tool

In this section, we present the characteristics of the tool we implemented. Our compiler ressembles the one described in (Candito 96) and (Candito 99). We thus present the latter first in order to emphasize the differences between her proposition and ours.

2.1. Topological factorisation

In (VS et al. 92), the authors propose to use a logic (precisely defined in (Rogers et al. 94)) which describes elementary trees of a TAG grammar so that the topological informations shared by trees is factorized in an inheritance hierarchy.

In practice, information concerning trees may be factorized according to different points of view quite independant from each other. For instance, subcategorization information leads to a rather natural hierarchy while realizations of syntactic functions lead to another hierarchy altogether. Therefore, attempts to describe both hierarchies in a unique inheritance lattice either leads to having to make a copy of one hierarchy at each leaf of the other, or if multiple inheritance is allowed, multiplying links between leaves of the hierarchies.

2.2. Marie-Hélène Candito's Compiler

(Candito 96) and (Candito 99) precisely explains the preceding point and advocates three independant, linguistically motivated hierarchies which she calls dimensions:

1. subcategorization (which she represents in terms of initial syntactic functions)
2. syntactic function redistributions (which lead to final syntactic functions)
3. final functions realisations

The underlying idea is that a linguist only describes these three hierarchies, and an automatic tool completes the inheritance graph by crossing final classes of dimension 1 with the final classes of dimension 2 and further crossing the result with the final classes of dimension 3.

However, not all final classes of dimension 1 are to be crossed with all final classes of dimension 2. For instance, intransitive verbs do not admit passive constructions. In the same way, the resulting crossed classes of dimensions 1 plus 2 have to be crossed only with those final classes of dimension 3 that realize a final function actually occuring in the crossed class.

The linguist has thus to give crossing conditions together with his hierarchies in order to constrain the crossing process.

The algorithm implemented by (Candito 99) is thus the following:

```
dim12 = empty set
for each final class c1 of dimension 1
   for each final class c2 of dimension 2 (compatible with c1)
        create c12 that inherits c1 and c2 and add it to dim12
   end for
end for
res = empty set
for each c12 in dim12
   resOfC12 = {c12}
   for each final function ff appearing in c12
      for each class c3 of dimension 3 that realizes ff
         build new classes with each element of resOfC12 and c3
         resOfC12 = these new classes
      end for
   end for
   res = res U resOfC12
end for
compute minimal referents for each element of res.
```

As this pseudo-algorithm makes clear, some constants that denote tree nodes are labelled by a final syntactic function. They are actually also labelled by an initial syntactic function, probably in order to keep track of the process. In dimension 2, most of the job done by classes consists in modifying the functional assignment[1]. Moreover, final functions are maintained unique in any description by an additional mechanism that equates constants bearing the same final function.

2.3. Our proposition

Our initial motivation for developping a new meta-grammar compiler had to do with the lexicon: the grammar compiled with Candito's tool is organised in tree families (as is XTAG (Doran et al. 00)) and lemmas are associated to families. The anchoring of a tree thus consists in computing the lemma and the morpho syntactic features associated to a word form, getting the families associated with the lemma and finally attempting to substitute the lemma with the associated morpho syntactic features in the tree.

Such a process may of course fail, either because the morpho-syntactic features do not match (consider a tree dedicated to an imperative form, together with an infinitive word form) , or because the features associated to the lemma do not match (some transitive verbs, for instance, do not accept a passive form).

Our starting idea was then to generate trees together with a feature structure that globally describes each tree, and Candito's tool did not seem to permit that.

Since we were implementing a new tool anyway, we gave ourselves some additional constraints:

- avoiding non monotonous mechanisms such as the modification of the final functions
- not limiting *a priori* the number of dimensions:
  - the third dimension is *de facto* a collection of dimensions dedicated to realizing each of the possible functions

1. Final functions are initialized to the initial function

– three dimensions is perhaps not a good choice for other categories than verbs, or for other languages than French, English or Italian. (See for instance (Gerdes 02))

As we intend to produce trees together with a feature structure, classes of the meta-grammar contain a feature structure that describe its content. It then seems natural that these feature structures get combined through unification along the inheritance lattice. This feature structure is then a good mechanism to avoid unwanted class crossings. The example we mention of intransitive verbs that do not accepts passive forms may simply be taken into account by means of an attribute **transitive** with values **minus** for an intransitive verb and **plus** for all passive classes.

The remaining problem is to find a mechanism that allowes classes to cross. In (Candito 99) compiler, this mechanism relies on the three dimensions, but we do not want to rely on a fixed number of dimensions.

We thus decided to make explicit the reason why classes are to be crossed, typically a class of dimension 1 has to be crossed with a class of dimension 2 because it **needs** redistribution of syntactic functions. Classes of dimension 2 have to be crossed with classes of ex-dimension 3 because they **need** that their final functions be realized. Conversely, a class of ex-dimension 3 may **provide** the realization of a subject, or an object, or whatever other function.

A class in our system is then described by:

- a name
- a set of super-classes
- a description (which is a feature structure)
- a set of needs (atomic symbols)
- a set of providings (atomic symbols)
- a formula describing trees

When a class c12 inherits two classes c1 and c2, the descriptions are unified (in case of failure, c12 is not created), the set of needs is the union of the two set of needs minus the union of the providings, the set of providings is also the union of the providings minus the union of the needs and the formula is the conjunction of the two formulas.

The crossing mechanism then consists in computing all balanced final classes, that is classes whose set of needs and providings are empty[2].

Finally, the formulas corresponding to balanced final classes are used to compute minimal referents (Rogers et al. 94) together with the description associated with the corresponding class.

3. A survey on the linguistic applications

We made some experiments on french verbs, conforming as much as possible to the analysis of (Abeillé 91; Candito 99). Therefore, we give an overview of the way we describe verbs using three 'dimensions'[3].

Two ways to generate a grammar are given in the following sections. The first approach puts the focus on the topology of the trees. While it allows to identify the nodes representing the predicate and its arguments in the tree, it actually suffers from a major drawback due to the monotonicity of the system. That is, once a node is asserted in the process of generating a tree, it cannot be removed. This is problematic if, for instance, one wants to describe an agentless passive as the ellipsis of the predicate's agent[4].

The second approach comes closer to the functional analysis introduced by (Candito 99). We do a topologically free reasoning upon arguments and functions until we have specified a complete final functional subcategorization frame. The main interest of this approach is that the functional component of the grammar is not anymore

2. In order to keep with an associative and commutative mechanism, a cancelled need as well as a cancelled providing is not allowed to appear again.
3. Formally speaking, it should be clear that there are no dimensions anymore.
4. However, there is a trick : one can set a 'kill' attribute to a node. It means that the node will be removed from the tree after its generation. The principle is that the removed node's children (if any) become the children of the removed node's parent (the root of the tree cannot be removed).

mixed with the topological one. We are able to represent the agentless passive quite easily. But here, the relationship between the tree structure and the logical arguments is lost. As a comparison, we do not specify any topological information into the classes that belong to the equivalent of Candito's first and second dimension. The whole analysis is driven by the feature structures describing the classes.

Both approaches share some essentials ideas[5]. (1) Each tree which is generated represents the realization of a logical predicate and its arguments. (2) The focus is put on the functional organization of the grammar. Following these assumptions we specify through three successive layers the trees that represents the syntactic realization of that predicate. Each of these layers performs a mapping as follows :

$$\begin{tikzcd}
\text{Predicate Structure} \arrow[d, Rightarrow, "\text{Dimension 1}"] \\
\text{Initial- Function} \arrow[d, Rightarrow, "\text{Dimension 2}"] \\
\text{Final-Function} \arrow[d, Rightarrow, "\text{Dimension 3}"] \\
\text{Surface-Realization}
\end{tikzcd}$$

- Dimension 1 : maps the predicate arguments to an initial functional subcategorization frame.
- Dimension 2 : maps the initial subcategorization frame to a final functional subcategorization frame.
- Dimension 3 : maps the final subcategorization frame to a tree structure.

3.1. A node driven strategy

Following (Abeillé 91; Candito 99) each tree which belongs to a family represents a predicative structure. The first dimension is dedicated to mapping initial functions to the predicate's argument positions. Here we define a set of classes which represents the arguments and another set which defines the functions that are to be mapped on them. The final classes of this dimension are a list of all the valid mappings in French. For instance the final class *Subj0VObj1* is the class where the first argument is mapped with a subject and the second is mapped with a direct object[6].

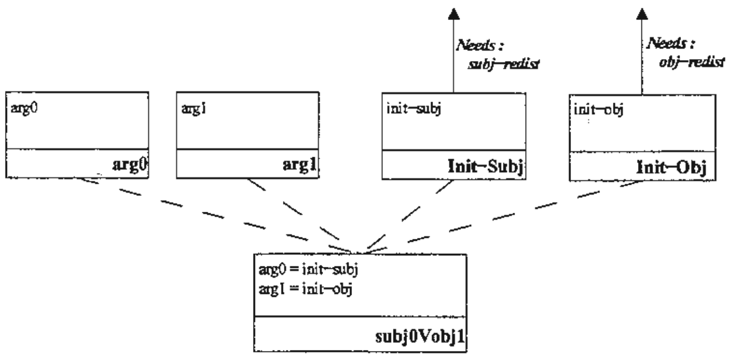


Figure 1: Overview of the first dimension

We express this mapping with the declaration of a constant[7]. Each of these quasi-nodes is equated with the one representing its associated function.

To be linguistically well formed, we impose the requirement that the crossed classes map each initial function to a final function. Then a class where an initial function node is defined contains a corresponding **need** for a final function (see fig. 3.1).

5. These ideas are already central in (Candito 99) work.
6. For the sake of clarity, we do not expand here to the whole (Abeillé 91) analysis in families. We do not consider here sentential arguments and verbal auxiliaries though they are important for the definitions of the families.
7. Following (Rogers et al. 94), a constant is denoting a node.

The second step in the generating process aims at defining a final subcategorization frame. Here we map the initial functions given above with final functions. As a side effect, the verb is given a morphological specification. For instance, through inheritance, the *full personal passive* maps the *initial-subject* to a *by-object*, the *initial object* to a *final-subject* and requests the predicate to be realized as a passive subtree. Furthermore, classes that introduce a final function emit the need that this function is realized.

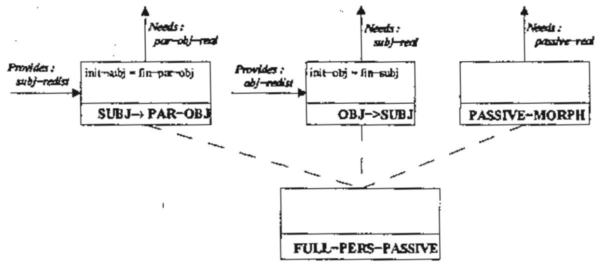


Figure 2: Overview of the second dimension

Our general strategy consists in manipulating functions through constants denoting nodes: instead of function features, we have such constants as *arg0, initial-subject, or final-object.* The redistributions are then performed by means of equalities between such constants. The final classes of dimension 2 express equalities between the nodes carrying the initial functions and the ones carrying the final functions. The interface with the first dimension is done through **providings** that satisfy the final-function **needs** of dimension 1 classes. For instance the *Full-personal-passive* class will inherit classes that provide the *subj-redist* and *obj-redist.* The content of the class reflects the mapping : the initial-subject node equals the final-by-object node and the initial-object node equals the final subject node (see fig. 3.1).

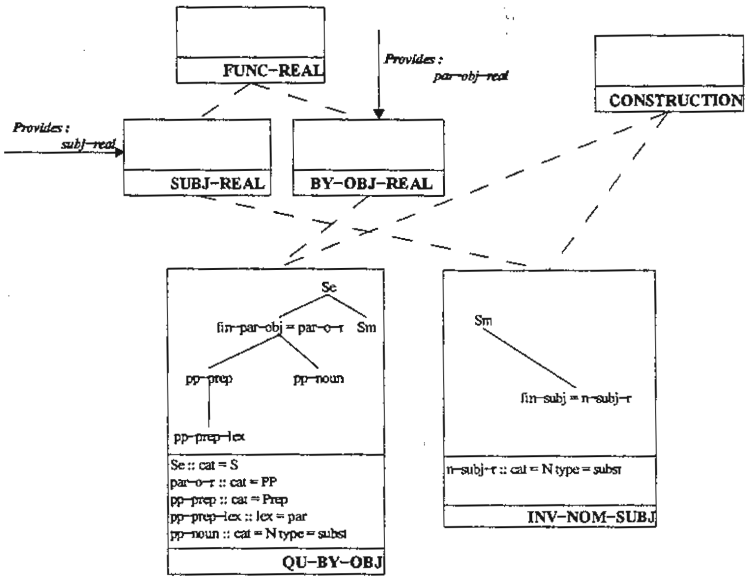


Figure 3: Overview of the third dimension

The functional realization 'dimensions', contains classes that actually yield trees. Basically, we view here a syntactic function as a label for a set of subtrees. Thus, this dimension groups all the subtrees and each of them is labelled as the representation of a function. The label assignment is performed through multiple inheritance as shown in figs. 3.1 and 3.1. Note that contrary to (Candito 99) approach, there are here two 'third dimensions'. One for the predicate's arguments and one for the predicate's head. The motivation is mainly methodological, we want to explicitly separate the functional and the topological part of the grammar.

Writing a metagrammar remains (at least for us) an experimental process. Other formalisms have been proposed that rely extensively either on feature structures (HPSG) or Linguistic functions (LFG). We experimented with a LFG inpired approach which allows us to deal with the topology of trees only in dimension 3. The general sketch is then to build feature structures in dimension 1 and 2 and to assemble the trees according to the specifications given by the feature structure in dimension 3.

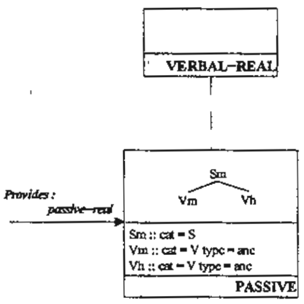


Figure 4: Overview of ‘another’ third dimension

3.2. A feature driven strategy

The feature structure descriptions, contained in classes and therefore associated to the produced trees at the end of the process, not only concern the anchoring, but may also actually describe the linguistic properties the tree is responsible for. Typically, it enables us to know that a tree is the representation of a two place predicate, that this predicate is a passive predicate, that the first argument is expressed as clitic and so forth. Thus the compiler allows to generate trees but also complex feature structures that are an explicit translation of what each tree ‘means’ linguistically.

In the previous approach we put the focus on the identification of particular nodes into the trees, and the feature structures associated to the classes only concerns the impossibilities in crossings. In the new approach we build complex feature structures and less complex formulas as lots of constants equated in rorder to represent redistributions simply disappear.

As an example, here are the features inherited by the following final classes:

- SUBJ0VOBJ1 :

$$\begin{bmatrix} \textsc{Pred} & \begin{bmatrix} \textsc{Head} & \begin{bmatrix} \textsc{Cat} & \text{Verb} \end{bmatrix} \\ \textsc{Subcat} & \left\langle \begin{bmatrix} \textsc{Init-Func} & \textsc{subject} \end{bmatrix}, \begin{bmatrix} \textsc{Init-Func} & \text{object} \end{bmatrix} \right\rangle \end{bmatrix} \end{bmatrix}$$

- FULL-PERS-PASSIVE :

$$\begin{bmatrix} \textsc{Pred} & \begin{bmatrix} \textsc{Head} & \begin{bmatrix} \textsc{Vmorph} & \text{passive} \end{bmatrix} \\ \textsc{Subcat} & \left\langle \begin{bmatrix} \textsc{Init-Func} & \textsc{subject} \\ \textsc{Fin-Func} & \textsc{by-obj} \\ \textsc{Cons} & \boxed{2} \end{bmatrix}, \begin{bmatrix} \textsc{Init-Func} & \text{object} \\ \textsc{Fin-Func} & \text{subject} \\ \textsc{Cons} & \boxed{1} \end{bmatrix} \right\rangle \\ \textsc{Cset} & \begin{bmatrix} \textsc{Subj} & \boxed{1} \\ \textsc{By-obj} & \boxed{2} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

- QUEST-BY-OBJ :

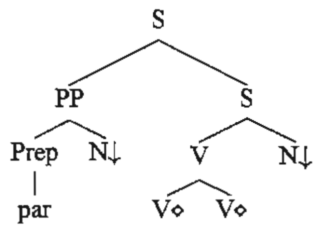
$$\begin{bmatrix} \textsc{Pred} & \begin{bmatrix} \textsc{Cset} & \begin{bmatrix} \textsc{Extraction} & \text{By-obj} \\ \textsc{By-obj} & \text{quest} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

- INV-SUBJ :

$$\begin{bmatrix} \textsc{Pred} & \begin{bmatrix} \textsc{Cset} & \begin{bmatrix} \textsc{Subj} & \text{inverted} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

In this approach, **needs** and **providings** are dispatched as they were in the previous approach. Classes of dimension 1 and 2 do not contain any formulas anymore. The third dimension realizes arguments as well as the predicate subtrees.

As a sample we generate the tree representing the schema that allows to analyze the sentence *Par qui sera accompagnee Marie ?* (*By whom will be accompanied Mary*) with the combination of the following final classes *subj0Vobj1, full-pers-passive, passive, inverted-subject, questioned-par-obj*(see figs. 3.1, 3.1, 3.1 and the feature structures given above) :



$$\Downarrow$$

$$\left[\begin{array}{ll}\textsc{Pred} & \left[\begin{array}{ll}\textsc{Head} & \left[\begin{array}{ll}\textsc{Cat} & \text{Verb}\\ \textsc{Vmorph} & \text{passive}\end{array}\right]\\ \textsc{Subcat} & \left\langle \left[\begin{array}{ll}\textsc{Init-Func} & \textsc{subject}\\ \textsc{Fin-Func} & \textsc{by-obj}\\ \textsc{Cons} & \boxed{2}\ \textsc{quest}\end{array}\right], \left[\begin{array}{ll}\textsc{Init-Func} & \text{object}\\ \textsc{Fin-Func} & \text{subject}\\ \textsc{Cons} & \boxed{1}\ \text{inverted}\end{array}\right]\right\rangle\\ \textsc{Cset} & \left[\begin{array}{ll}\textsc{Extraction} & \text{By-obj}\\ \textsc{Subj} & \boxed{1}\\ \textsc{Par-obj} & \boxed{2}\end{array}\right]\end{array}\right]\end{array}\right]$$

Notice that when we use this second approach (which anyway is an enhancement of the node driven approach), the feature structure keeps track of the successive mapping steps that are performed throughout the process of generation. This approach consists of not declaring any structural constraints in the two first dimensions[8]. This solution has the benefit of clearly splitting the functional from the topological part of the grammar. But at the time of this writing, we are not able to establish a link between the feature structures associated to the classes and the constants of the logical formulas used to generate the trees.

4. Conclusion

The tool developped so far, though rough and buggy[9] enables us to experiment with metagrammar writing. As we just mentioned it also raises interesting questions regarding the precise objects we are dealing with when describing a grammar. One of the main drawbacks of our implementation is the absence of relationship between the descriptive feature structure and the logical formulas. In our opinion, the root of this problem concerns what a TAG grammar really is: a set of trees gathered in families together with indices indexing nodes (cf. **n0Vn1**) are more than just elementary trees.

References

Abeillé, A., *Une grammaire lexicalisée d'arbres adjoints pour le français. Application à l'analyse automatique*, Doctoral dissertation, Université de Paris 7, 1991.

Abeillé, A, Candito, M.-H., "FTAG : A Lexicalized Tree Adjoining Grammar for French", *in* Abeillé, A. and Rambow, O. éd. *Tree Adjoining Grammars. Formalisms, Linguistic Analysis and Processing*, Stanford, CSLI, 2000.

Candito, M.-H., Candito, "A Principle Based Hierarchical Representation of LTAGs", *COLING*, 1996.

8. That also suppress the trouble related to node deletion as mentioned earlier for the analysis of the agentless passive.
9. . . . and available at *http://www.loria.fr/equipes/led/outils/mgc/mgc.html.*

Candito, M.-H.,*Organisation modulaire et paramétrable de grammaires électroniques lexicalisées. Application au français et à l'italien*, Doctoral dissertation, Université de Paris 7, 1999.
Doran, C, Egedi, D, Hockey, A., Srinivas, B., Zaidel, M., "XTAG System - A Wide Coverage Grammar for English", *COLING*, 1994.
Doran, C, Sarkar, A., Srinivas, B., Xia, F., "Evolution of the XTAG System", *in* Abeillé, A. and Rambow, O. éd. *Tree Adjoining Grammars. Formalisms, Linguistic Analysis and Processing*, Stanford, CSLI, 2000.
Gerdes, K., DTAG ? Attempts to generate a useful TAG for German using a metagrammar, TAG+6, 2002.
Joshi, A. K., Levy, L. S., Takahashi, M., "Tree Adjunct Grammars", *Journal of Computer Science*, 1975.
Kinyon, A., "Hypertags", *COLING*, 2000.
Lopez, P., Bonhomme, P., "Resources for Lexicalized Tree Adjoining Grammars and XML encoding : TagML", *LREC*, 2000.
Rogers, J., Vijay-Shanker, K., "Obtaining Trees from Their Descriptions : An Application to Tree-Adjoining Grammars", *Computational Intelligence*, 10, 4, 1994.
Vijay-Shanker, K., Schabes, Y., "Structure Sharing in Lexicalized Tree-Adjoining Grammars", *COLING*, 1992.

DTAG?

Kim Gerdes
Lattice, Université Paris 7

1. Introduction

This paper reports on work in progress on the creation of a metagrammar for German verbal constructions. Section 2 circumscribes the field we are working on: We describe known and less known problems and try to delineate the limits of a standard Tree Adjoining Grammar for German; we see which structures we get easily, and which structures we will never get. In Section 3 we put the German data in perspective to the other TAG languages, French and English, and we define and justify our choice to create a limited German TAG, designed for a specific generation task. We then propose in Section 4 some of the possible elementary trees that can live up to our expectations: Compromising on the semantic interpretability of the derivation structure as well as on the principles underlying TAG allows us to get most of the word orders necessary for the generation task. Yet, what we gain in usefulness in generation, we pay in linguistic descriptiveness. Last but not least, in Section 5 we discuss problems and limits of the metagrammar implementation and we give some indication on how the desired elementary tree sketches can be created and maintained with a metagrammar.

We presuppose the comprehension of the terms *metagrammar* and *topological field* of a German sentence, which we cannot define here. For details, see Gaiffe et al. 2002 and Kathol 1995 respectively.

2. Scrambled Minds

The two existing Tree Adjoining Grammars of interesting grammatical coverage have been created for English (XTAG, 1995) and French (FTAG, Abeillé 1991), two languages with quite rigid word order and little case marking. On the other hand it has long been shown that German is beyond the (derivative) generative capacity of TAG: There are no verbal elementary trees carrying the nominal arguments (and thus verifying the predicate-argument cooccurrence constraint) that can be combined to cover some of the (so called) scrambled word order of German (Becker et al. 1991, 92). Let us see where the problem is.

2.1. Argument Scrambling

Sentence (1) is the standard example for scrambling: The two constituents in the Mittelfeld (the positions between finite verb and non-finite verbs) have 'exchanged' their position. Note that this is a very natural order that even is the standard order if the direct and the indirect objects are pronominalized as in (1).

(1) a. Peter hat das Buch meinem Vater zu lesen versprochen.
Peter has the book to my father to read promised.
'Peter has promised to my father to read the book.'
b. Peter hat es ihm zu lesen versprochen
Peter has it him to read promised.
'Peter has promised him to read it.'

By definition, a control verb like *versprochen* 'promised' assigns a theta role to the subject just like the embedded verb. So the (semantic) predicate argument structure is shown in Figure 1a (leaving aside for the moment the tense auxiliary because of its uncertain semantic role). As the derivation structure of TAG is a tree, we have to restrict our analysis to one of the predicate-argument links, the subject role of the infinitive or the subject role of the control verb.

Suppose we wanted to follow the usual XTAG/FTAG analysis and have the control verb *versprochen* 'promised' govern its subject. Suppose further that we handled the auxiliary as an adjunction to the matrix verb. Our goal is then to obtain the derivation structure in Figure 1b. The first elementary tree of *versprochen* 'promised' (Figure 2) can adjoin to the root node of the infinitive and subsequently, its subject substitution node appears at the right place. However, in this case, its dative substitution node remains outside of the infinitive's elementary tree. We do not obtain the word order of sentence (1); the dative would be supposed to appear at the

* I would like to thank the XTAG group at the University of Pennsylvania and an anonymous reviewer for their helpful comments on my work. I am also grateful to Tilman Becker for his help finding the least ugly trees and keeping up my faith in TAG. I assume the customary responsibility for content and shortcomings.

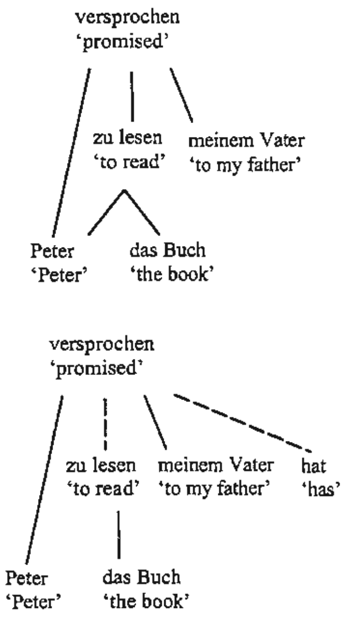


Figure 1: predicate-argument graph and derived tree

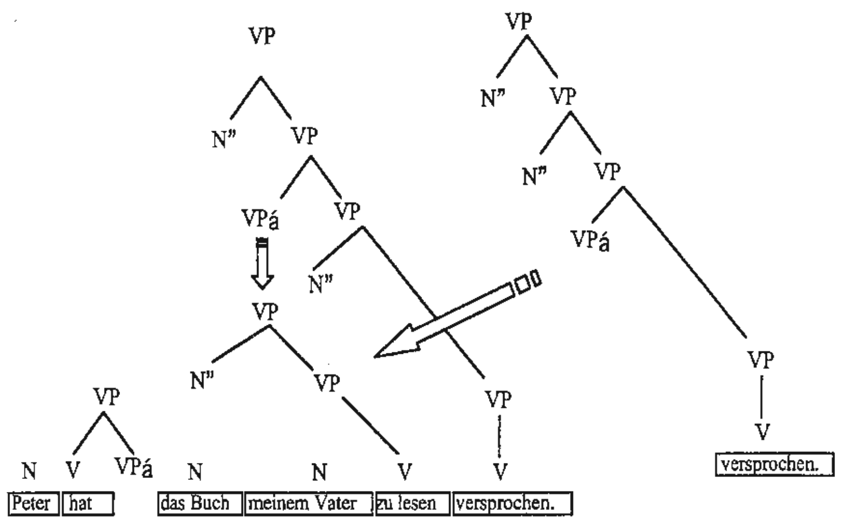


Figure 2: Scrambling is beyond TAG

right of the infinitive giving the so-called intraposition structure of German: *Peter hat das Buch zu lesen meinem Vater versprochen.* (Peter has the book to read to my father promised). The second elementary tree for *versprochen* 'promised' shown in Figure 2, allows on the contrary the dative argument to appear in the right place. Then, however, the subject node appears between the infinitive and its argument and the subject cannot be substituted at its Vorfeld (sentence initial) position.

The reader easily verifies that it actually does not make a difference if we decided to treat *versprochen* 'promised' as a raising verb, i.e. leaving the subject to the embedded verb *zu lesen* 'to read'. In this case we could not find a way to adjoin the auxiliary *hat* 'has' to its head *versprochen* 'promised'.[1] The only thing that would help for this particular case is to have the auxiliary carry the subject. This possibility could be a good choice if we wanted to develop a useful parsing grammar, as the case of a subject not neighboring the V2 position is very rare (mainly for negative pronouns like *niemand* 'nobody' or in spoken language with a strong accentuation). It would however weaken further the semantic interpretability of the derivation tree and does not solve the theoretical problem of the 'free'[2] placement of the arguments of the left and right bracket's verbs.

2.2. Adjunct Scrambling

It is less known that even with all the arguments in standard order, the position of the modifiers can block a correct TAG analysis. The preferred interpretation of sentence (2) has a prepositional phrase modifying the control verb *versprochen* 'promised', although this modifier finds itself between the infinitive *zu lesen* 'to read' and the infinitive's argument *das Buch* 'the book'

(2) Peter hat meinem Vater das Buch ohne Zögern zu lesen versprochen.
Peter has to my father the book without hesitation to read promised.
'Peter did not hesitate to promise to my father to read the book.'

In Fig. 3 we depict the corresponding elementary trees. The elementary tree of the adverbial modifier *ohne Zögern* 'without hesitation', cannot reach any node of its governor's elementary tree. The only possible TAG analysis will give us the semantically 'wrong' derivation tree where the reading is done with hesitation.

1. If the auxiliary does not adjoin to the past participle's tree, the information that a past participle adjoined would have to be passed through the infinitive's tree. This puts lexical information into the features and breaks all usual TAG principles.
2. Of course this order is only free in the sense that it does not affect the predicate argument structure of the sentence. The order depends on the communicative structure of the sentence.

2.3. Relatively Difficult

German relative clauses put up two hurdles for TAGs: The inner structure of the relative clause and its placement in the main clause.

The standard TAG formalism excludes trees with more than one foot node (as this would complicate considerably the derivation structure). Thus, we cannot express the idea that an adjunct turns its governor into an adjunct itself. However, this is the case for relativized adverbial modifiers. Take (3) as an example.

(3) der Balkon auf dem er singt.
the balcony on which he sings
'the balcony he sings on'

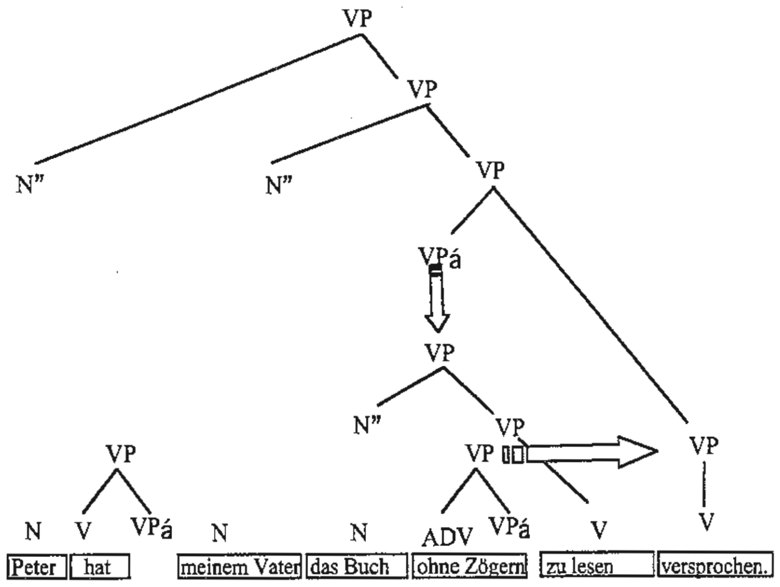


Figure 3: Adjunct scrambling is even worse

The only possibility of analyzing such a phrase in TAG is to combine the elementary tree for *singt* 'sings' and the one for the relativized modifier into one elementary tree of *auf etwas singen* 'sing on something', as if the balcony was an argument of the singing. This clearly violates the principle of non-compositionality.

In general, as soon as we want to move an element out of a substituted position, we are beyond the (derivative) generative capacity of TAG. Take example (4) of a German relative clause.

(4) der Film von dem ich nur die Musik kenne.
the movie of which I only the music know
'the movie I only know the music of.'

The predicate argument structure seems clear: *Musik* 'music' is the object of *kenne* 'know', and *von dem Film* 'of the movie' modifies *Musik.* The only way out for this case is to have separate *kenne* 'know' trees for the relativized modification of its arguments. Again, we do not obey the principle of non-compositionality and obtain severely messed-up derivation trees.

Extraposed relative clauses constitute another difficulty for the TAG formalism: The preferred placement for relative clauses in German is not directly behind the modified noun, but in the Nachfeld of the sentence's main domain. This preference is independent of the noun's position in the Vor- or Mittelfeld[3] and it is partic u-larly strong in two circumstances: First, when the right bracket is occupied only by a short and semantically weak element, like for example a so-called separable verbal prefix. Secondly, when the relative clause is rather long. In the following example (5), the separable prefix *ab* of the verb *abschließen* 'lock' has to take the right bracket of the main domain.[4] The Nachfeld position of the relative clause in (5) is clearly preferable.

(5) a.?? Peter schließt die Tür, die ich gestern bemalt habe, ab.
Peter closes the door, that I yesterday painted have, up.
b. Peter schließt die Tür ab, die ich gestern bemalt habe.
Peter closes the door up, that I yesterday painted have.
'Peter locks the door I painted yesterday.'

3. The agreement in number and gender of the relative pronoun with its head allows the reconstitution of the dependency relation. The placement behind the noun might be preferable in the rare case when neither agreement nor semantic plausibility allows associating the relative clause with its governor. Another rarity consists of two nouns of the same domain being modified by relative clauses. In this case again, one of the relative clauses has to be adjacent to its head.

4. The separable prefixes of some verbs behave syntactically just like bare infinitival arguments of the verb: They usually take the syntactic place the verbal dependant would take, and the more semantic weight they have, the easier is their independent placement into the Vorfeld. For a more detailed discussion see Gerdes & Kahane 2001.

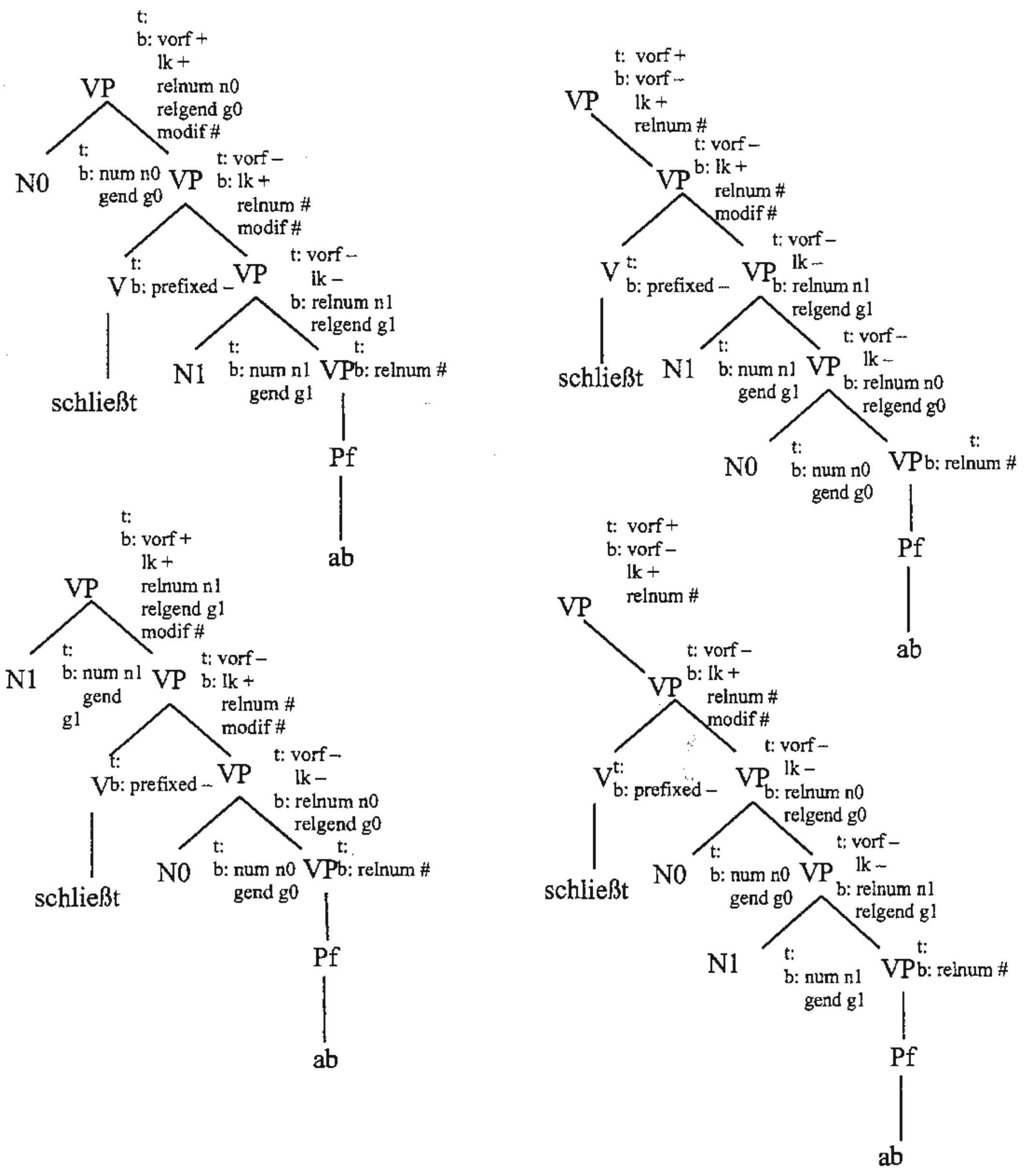


Figure 4: The simplified V2 trees for a transitive verb with separable prefix

This is another instance of an extraction of an element out of a substituted position. For the sake of the semantic consistency principle we have to realize the verbal prefix as a co-anchor of the verbal tree[5] and the verbal branch down to the prefix in the right bracket disallows the relative's adjunction to its head.

3. XTAG – FTAG – DTAG?

The question we have to address now is whether all this is any worse than the anglo-roman situation. English and French don't have scrambling, but it is clear that there are many cases in these languages, too, where the only possible analyses with a regular TAG will give us derivation trees with incorrect dependencies with our TAG principles falling by the wayside (Shieber & Schabes 1994, Rambow 1995, Candito & Kahane 1998). For example, the case of relative clauses is in fact very similar in the three languages in question: The gloss I give for sentence (3) is a correct English sentence (although the stranded preposition is preferable), parallel constructions to (3) and (4) exist in French (e.g. *Le film dont je ne connais que la musique* 'The movie I only know the

5. One sees easily that the situation is in fact identical even if we treated the verbal prefix as an argument or had it adjoin into the main verb's elementary tree.

music of'), and French has extraposed relatives just like English (in particular for parallel constructions to the separable prefix: *She threw the book away that I wanted to read.*)[6]

Several extension of the TAG formalism have been proposed attempting to remedy these shortcomings (VTAG, DTG, TDG, ...). The scrambling case of sentence (1), for example, could still be handled in tree local multi-component TAG, a little add-on to regular TAGs that does not change the weak generative power. However, if we added a 4th verb to the sentence, this will no longer be possible.

3.1. A generation grammar!

Leaving aside the theoretical and linguistic problems of these new formalisms, including multi-component TAG, their biggest flaw is the lack of implemented tools, a fact that creates the desire among researchers in language engineering for a 'real' German TAG. In particular the language generation task can live without the complete set of word orders. One could even go as far as saying that the limited word order possibilities are a feature not a bug: A formalism that allows all the possible orders would force the generation module to choose between the different orders. These choices are dependent mainly on the information structure of the sentence, a problem far beyond today's running generation systems. The snag is of course that the choice implied by the formalism is not necessarily the best choice in a given context. It has been shown though that a sentence in the so-called 'standard word order' can obtain most information structures by prosodic means (Lenerz 1977, Choi 99).[7]

For our German TAG, we could thus be content with the standard word order. However, even for a restricted generation task, it might be preferable in some cases to have access to verbal trees allowing different word orders. The difficulty of this work lies in the fact that one has to find a compromise between restrictions of the formalism (and the metagrammar description) and the concrete necessities for generation. We try to obtain a German TAG linguistically as good as possible and practically as useful as possible for the generation job. On the basis of corpora for quite simple generation tasks in German,[8] we decided to add all verb internal argument permutations to our grammar (in order, for example, to encode a lexically triggered preference of some verbs to place the accusative argument before the dative, or a preference to place temporal or local modifiers into the Vorfeld). We disallow inter-verbal argument exchange with two exceptions too common to leave aside even for a simple generation system: Topicalization into the Vorfeld and extraposition of relative clauses into the Nachfeld. Further we allow placement of embedded non-finite verbs only in the right bracket of its governor's domain and its extraposition into the Nachfeld.[9] We believe not to need any verbal positions like (partial) VP fronting or intraposition. The grammar ends up with many more trees than the standard word order, but we suppose that when these trees are inadequate, the tree description[10] allows sorting them out with no trouble.[11]

Of course such a grammar will be of limited use for parsing because in a corpus, we will encounter the other word orders as well. It seems that the phenomena that cause trouble to TAG are more frequent in German than in Anglo-Roman, where the 'pure' TAG grammars can obtain reasonable parsing results.[12]

4. The compromising trees

The trees presented here are product of a compromise between a minimal violation of the TAG principles (this comes down to a maximal semantic interpretability of the derivation tree), a maximal coverage of the grammar, a maximal usefulness for simple language generation systems, and a maximal simplicity in the meta-

6. Another awkward example on relatives is the fact that in the phrase *apples that Mary thinks John likes*, the tree of thinks has to adjoin into the likes tree, whereas in *the woman who thinks John likes apples*, the tree for thinks has an argument position (a substitution node) taken by the like tree. This XTAG analysis is triggered by the restrictions of the formalism and, to my knowledge, not by linguistic intuition.

7. This is in fact one of the definitions of 'standard word order'. The 'standard' order of a verb's nominal arguments can vary lexically (Müller 1999).

8. We actually looked at the requirements for the German generation systems in the MultiMétéo project (Coch 1998) and in the SmartKom project (http://smartkom.dfki.de).

9. Some matrix verbs like *scheinen* 'seems' prefer to construct incoherently, i.e. their verbal argument should go into the Nachfeld of the main domain.

10. Tree descriptions are finer grained than the notion of family and this should allow a smoother interface with the lexicon, the neglected but essential component of any grammar. Future work will show whether this approach resists when leaving the toy state.

11. This will have to be verified when actually using the grammar. The metagrammar setup easily allows generating limited subgrammars if desired.

12. It will be difficult to actually prove this point: One would have to compare an English and a German grammar with equivalent lexicons, equivalent grammatical coverage, and equivalent 'tricking' around the limits and principles of the formalism.

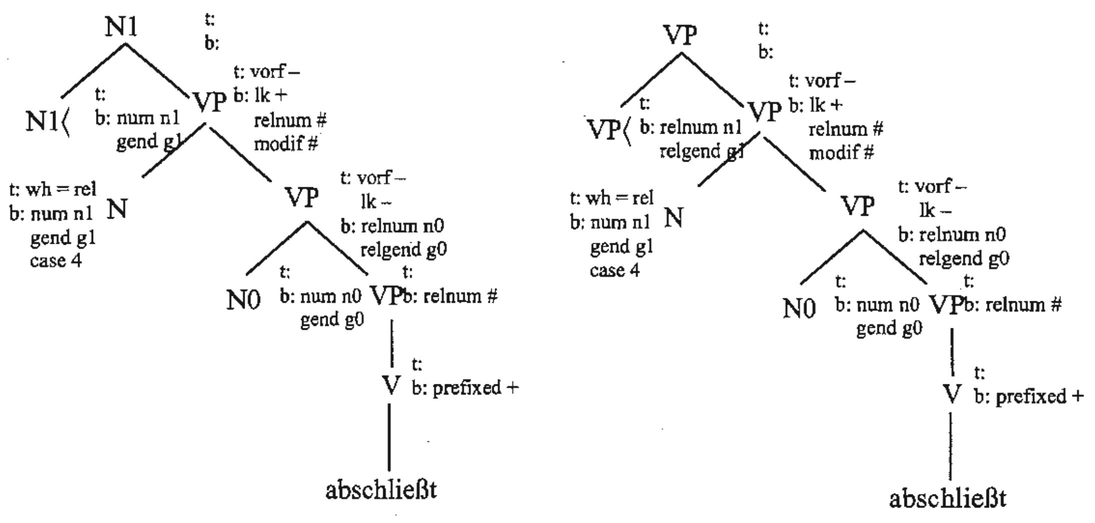


Figure 5: a relative elementary tree for adjacent modification (left), the other for long-distance modification (right)

grammar description. We do not expect syntactic expressiveness from the derived tree though.[13] The trees in Figure 4 illustrate some intermediate results on the way to this compromise.

We do not distinguish VP from S nodes. This simplifies the tree description in the metagrammar (see below) and additionally, it makes it easier to describe relative extraposition from the Vor- to the Nachfeld. Since the long-distance modification by the extraposed relative clause is beyond the generative capacity of TAGs fulfilling the strong cooccurrence constraint, we relax this constraint and push the lexical information of the noun's number and gender into the VP spine of the verbal elementary tree. Now, an adjoining relative phrase can check its agreement on the VP node. Of course, the corresponding derivation tree bares a relative clause modifying a verb and not a noun, and the generation module has to take this into account.[14] See Figure 5 as an example of an extraposed relative phrase and its corresponding elementary tree: The number and gender information of the argument is passed up into the *relnum* and *relgen* features of the verbal spine, where it can be read by the tree of the relative clause. The relative clause then passes it back down to its substitution node for the relative pronoun. Case is assigned directly by the verb.

So the treatment of extraposed relative phrases is one reason for the right branching structure we use. The other reason is that in a flat structure, the TAG formalism does not allow adjunction between sister nodes. As modifiers can appear at any position in the Mittelfeld, we need the VP nodes as landing points for the right adjoining modifiers we use.[15] A third reason is that the right branching tree is easily described in the metagra m-mar, as we will show below, as every nominal argument introduces its verbal spine element into the tree sketch, independently on the final order or realization. We do not stipulate however, that the VP-headed subtrees correspond to linguistic (functional, prosodic, semantic...) objects of their own; their justification is internal to the formalism.

The distribution of the German verb distinguishes different positions in what is usually called 'the topological model' (Drach 37, Bech 55): The finite verb can take the 2nd position of the sentence (V2, for main clauses[16]) or the final position (Vfin, for sentential complements and relative clauses). V2 means that one constituent of any nature has to be placed before the verb. This easily stated constraint is again beyond TAG's expressiveness; the translation into TAG becomes a highly non-trivial exercise of feature manipulation. The basic idea is the

13. Many TAG grammar writers seem to put the derived structure in second place. The end of establishing a correspondence between the word string and the semantically interpretable derivation tree justifies all syntactic means.

14. This boils down to some preprocessing of the derivation tree before the actual TAG generation comes in. The original derivation tree for an extraposed relative originally contains a verb whose nominal argument is again modified by a verb (the verb that opens the relative clause). In order to establish agreement, the preprocessing then not only has to move up the node of the relativized verb from nominal to verbal modification, it also has to compute the address of the node in the verbal matrix tree that will receive the relative adjunction. If this preprocessing comes out to be too costly, one could also put unique features in the tree to make sure that the tree of the relative clause only adjoins to the corresponding node in the verbal spine. We consider further that this preprocessor takes care of the uniqueness of the relative extraposition. An alternative choice would be to put a feature into the verbal spine that rules out double extraposition of relatives.

15. We have to avoid this modifying adjunction from the left into the root VP and into the VP of the verb in second position (V2).This is done with the non-unifying feature value # of the feature *modif.*

16. There are some cases of embedded V2, not taken care of in this grammar.

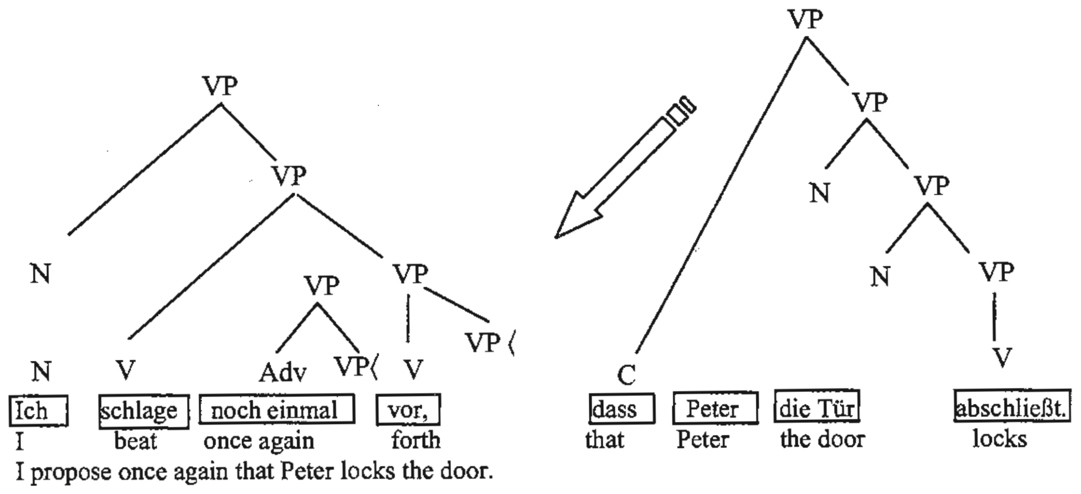


Figure 6: An analysis with sentential complement

following: The Vorfeld can be filled by any complement of the verb, argument or modifier. In both cases, the concerned verb has a VP node in its elementary tree. If this node dominates the Vorfeld-argument it has to prevent the adjunction of modifiers, since there is only one constituent before the finite verb (see the left trees of Figure 4). The second possibility is that the VP remains a unary node, which then forces the unique adjunction of modifier (see the right trees of Figure 4).[17]

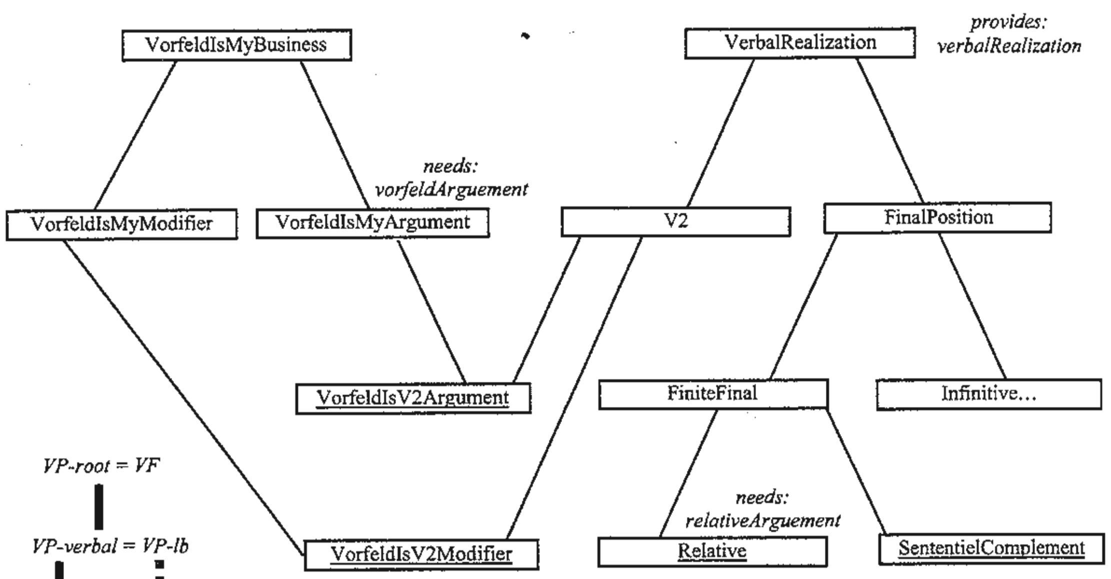
Relative clauses and sentential complements behave similarly: The complementizer or the relative pronoun takes the first place, topologically often called complementizer field. This field behaves similarly to the left verbal bracket, and there have been attempts to consider the complementizer and the finite verb in V2 to be appearing in the same position (see for example Kathol 1995, for a contrary view in the light of universal grammar see Rambow & Santorini 1995). For TAG, the semantically empty complementizers should appear either with the matrix verb or with the embedded verb. We consider the complementizers field as unification of Vorfeld and left bracket, as in inherits properties of both fields (Gerdes & Kahane 2001). In the metagrammar, this is easily translated as the union of the two quasi nodes Vorfeld and left bracket. The 'universal' interpretation of a grammar could then consist of saying that in languages where the complementizer and the verb in V2 position do not have a complementary distribution, this node union did simply not take place (see Rambow & Santorini 1995 for the example of Yiddish).

As it is shown in Figure 6, we consider for the moment the sentential complement to substitute into the matrix verb. This allows handling all finite verb final constructions to be described quite uniformly in the metagrammar, but has the important drawback not to be able to treat wh-extraction into the Vorfeld. Contrarily to infinitival construction where every argument can join the Vorfeld, only wh-elements can be extracted out of sentential complements (see sentences (6)). However, this extraction is not a frequent phenomenon, and many native speakers of German consider it as bad style, preferring parenthetical constructions (which are not part of my Grammar for the moment either).

(6) a. ? Was schlägst du vor, dass Peter abschließt?
What beat you forth, that Peter locks?
b. Was, schlägst du vor, schließt Peter ab?
What, beat you forth, closes Peter up?
What do you propose that Peter locks?

Non-finite verbs come in many flavors: Keeping up the XTAG/FTAG division of subject-carrying control verbs on one hand and subjectless auxiliaries and raising verbs on the other, we are obliged to distinguish infinitival trees carrying a nominative argument or not. The infinitive can further care for the Vorfeld position or leave this field for other verbs. Caring for the Vorfeld position means to either place an argument there or forcing a Vor-

17. The expletive *es* stands out in this description because it is neither a modifier nor an argument and it can only appear in the Vorfeld. Its function is to occupy the Vorfeld if the speaker wants to avoid emphasize (topicalization or focalization, see Choi 99) of any semantically full element. The resulting sentence is purely rhematic. Formally the tree sketch corresponding to the expletive *es* is just the same as an adverbial one with the restriction that it can only adjoin into the Vorfeld. However, giving the *es* an individual entry seems at odds with the semantic consistency principle, as the function of this word is precisely to take a topological position without adding semantic information.



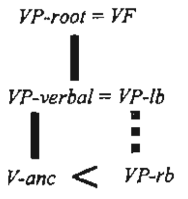


Figure 7: Fragment of Verbal Realizations

feld adjunction to its highest node. We leave aside all other crossing of arguments with other verbs. Even now, without cases of ellipsis we obtain 6 different trees for a simple transitive bare infinitive (or past participle or zu-infinitive that behave identically for the matter[18]).

5. Meta!

The freer word order of German arguments corresponds well to the idea of underspecified tree descriptions in the metagrammar: If we don't indicate the linear ordering between two elements, all the possible orders will be realized in the resulting tree sketches. On the other hand, the metagrammar becomes the actual linguistic description, and the tree sets obtain the status of intermediate products in the parsing process. For example, having two elementary trees for exactly the same word forms and the same sub categorization frame as in Figure 4, makes these cases indistinguishable on the tree level from actually different syntactic instances.[19]

5.1. Implementations

Our first attempt was to model the German verbal grammar around Candito 99's French metagrammar in order to be able to use her compiler. The main difficulty, except for the fact that this tool is no longer maintained, lies in the rigid definitions of the three dimensions she uses: initial subcategorization, redistribution, and realization of arguments. In order to capture the difference between V2 and Vfin positions, which is a general choice for all verbs (independently of their arguments), we have to introduce a realization module not only for arguments, but also for the head, the verb itself.[20] Although it was possible to hack Candito's tool to accept the verbal realization, the tree description for the V2 structure revealed a serious problem in this implementation:

At most one of the arguments of the verb has to be fronted, if none of its argument goes into the Vorfeld, it has to be a modifier, and the highest VP node has to force exactly one adjunction from the left. So we need leaf nodes whose function remains underspecified until the final class crossing and that eventually disappear if e.g. they can unify with their mother node. In Candito 99's implementation, those leaf nodes default to substitution nodes, which prevents them from unifying with their governor. In a word: It was not possible to express this linguistically appealing description of how the V2 elementary tree sketches came into being.

18. The infinitive extraposition to the Nachfeld, only possible for zu-infinitives, does not appear in the infinitival tree itself. It is the matrix verb subcategorizing for a zu-infinitive that has one more class crossing in (which can mean many more elementary trees depending on the arguments of the matrix verb).

19. This situation differs from French and English where, leaving aside some exceptions, different trees correspond to different syntactic constructions.

20. The possibilities described in this dimension correspond to the GB notion of head movement. We consider instead that the head is positioned out of an unordered dependency tree directly into its final position. Nevertheless, our V2-elementary trees without infinitival complement have an empty node at the verb-final position, which can be interpreted as the trace of the verb moved to the V2 position, or topologically (and preferably) as the right bracket that remained empty.

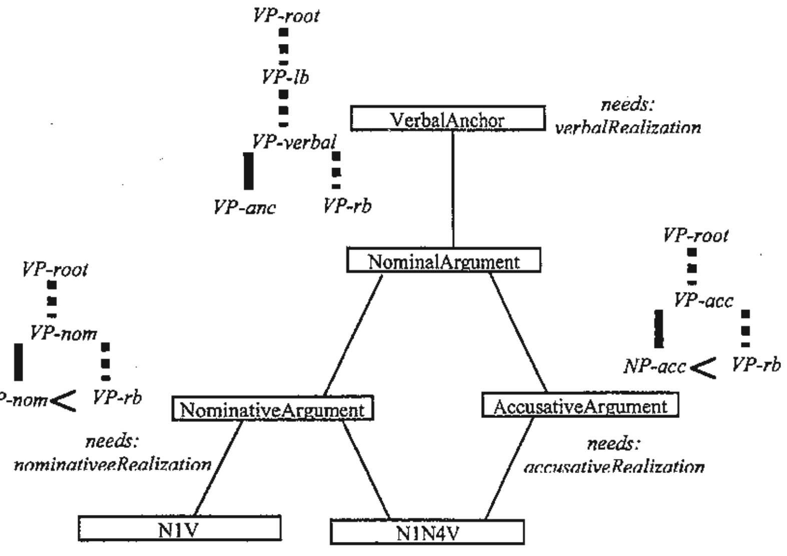


Figure 8: Fragment of the initial subcategorization

Help came from a reimplementation of the metagrammar compiler by B. Gaiffe (Gaiffe et al. 2002). This compiler allows without difficulty to obtain the desired minimal tree for the description[21]. The second a dvantage of the new compiler is that it is based on a more general approach than Candito's three dimensions: Each final class has a set of polar features, called needs and provisions, and all classes that can 'do' something for each other, i.e. mutually neutralize a polar feature, are crossed. Only when all needs and provisions are neutralized the minimal tree is calculated. This resolves the unclear status of the third dimension in the original approach that consisted of a number of quite independent hierarchies for each argument to realize. Moreover, for German at least two of the 'classical' redistribution for French, the passive and the causative in German, don't have a specific behavior different from verbal subcategorization. There is no syntactic reason to consider the passive auxiliary and the past participle as a unit, although they are, certainly, on a semantic level. If we see the metagrammar only as a generator of all necessary tree sketches, we have got already the necessary tree sketch among the trees for predicative adjectives.[22] A third advantage is the notion of tree description that replaces the notion of family: A feature structure, as fine grained as needed, replaces the name of the family. For details see (Gaiffe et al. 2002).

We conclude this section with a few extracts out of the complete German verbal metagrammar. We illustrate how the information on elementary tree sketches is distributed across the hierarchy.

In Figure 7 we depict a fragment of the verbal realization class hierarchy, indicating some partial tree description we use. Final classes are underlined. Each final class inherits the 'verbalRealization' provision and will thus be crossed with classes that need this feature. We just observe the combined tree description that the class VorfeldIsV2Modifier has inherited from all its superclasses: The quasi-node VP-root combines with the Vorfeld and it governs directly the combination of the VP-verbal node (the quasi node governing directly the verbal anchor) and the VP-lb (left bracket) node. Thus the desired V2 position is created. Moreover, we make sure that the right bracket (VP-rb) follows the left bracket. This partial tree description will cross with other classes that need or provide any feature for it. This crossing is completely incremental; no constraint can be erased.

Another subgrammar of the (not necessarily connected) metagrammar is the verbs' initial subcategorization frame (corresponding to Candito's first dimension). We observe in Figure 8 which constraints each class adds to the hierarchy: The verbal anchor class introduces 5 quasi nodes and their specific constraints. Each final verbal class inherits from this class and inherits thus these quasi nodes and their constraints. The final class N1N4V[23] for simple transitive nouns, for example, inherits the verbal anchor's tree description, the nominal argument's tree description, and the accusative argument's tree description. This means that the class's tree description has already the quasi nodes for a verbal anchor, a nominative argument, and an accusative argument, each with their

21. The calculation of the minimal representative of the tree description differs slightly from the original one by Rogers & Vijay-Shanker 1994.

22. The (rare) passives of intransitives may be a better reason to introduce the redistribution step, because these forms need a tree sketch without subject. In any case, if, for a given application, the notion of family containing all the passive trees as it is used in XTAG/FTAG proves to be useful, this 'redistribution step' can easily be added to the grammar.

23 These final class names recall the family names in the classical elementary tree arrangement. To avoid the discussion on whether the term subject corresponds to nominative (and indirect object to dative, etc) we name the arguments directly by their case (1 = nominative, 2 = genitive, 3 = dative, 4 = accusative).

corresponding VP-spine node. Their mutual order however will only be defined when the class's three needs will be satisfied, i.e. when the verb and the two arguments will get realized.

This should suffice to make clear the underlying principle of class inheritance and class crossing. The metagrammar is actually being worked on with the goal to include as many grammatical phenomena as possible. For real use, we will then choose the necessary grammatical phenomena and only generate the required trees.

6. Conclusion

We have seen the possibilities and limits of describing German with Tree Adjoining Grammars. As the formalism is very restricted (and for that computationally attractive), the design of the elementary trees is a constant trade-off between usefulness and descriptiveness. If however, we want a grammar for a well-restricted area, we can nearly always find a way of covering the desired string with a more or less good-looking tree.

One experience that should be shared with TAG writers of other new languages: It seems very difficult to start with the metagrammar as a descriptive tool, and see afterwards what trees fall out. It is much easier to decide first on the concrete tree sketches and then to distribute the information contained in the tree into a hierarchy, in a way to maximize sharing of information. The resulting metagrammar can then be seen as an interesting linguistic description, although its TAG-independent usefulness remains to be shown.

It was thought that the metagrammar would liberate the grammar writer from the actual messy mass of elementary trees, that one could easily extend a metagrammar by just adding some classes that will automatically be crossed with the existing classes. In reality, the grammar writer has to check all new resulting tree sketches, and most of the time lots of features have to be altered all around the existing metagrammar to fit in smoothly the new classes. The metagrammar should maybe better be thought of as a specific elementary tree editor. It is true however that it is easy to create limited subgrammars by simply commenting out the undesired classes in a more complex grammar.

In this paper we cannot go into greater detail on the proposed structure of a German metagrammar. We could only outline some of the problems we encountered on the way to a German TAG, problems which seem to us to be instructive for the general understanding of TAGs and of the idea of a metagrammar.

References

Abeillé Anne, 1991. *Une LTAG pour le français.* Ph.D. thesis. Univ. Paris 7.

Bech G. 1955. *Studien über das deutsche Verbum infinitum*, 2nd edition 1983, Linguistische Arbeiten, Nr. 139, Niemeyer, Tübingen.

Becker, T., A.K. Joshi, O. Rambow 1991. "Long distance scrambling and tree adjoining grammars" in *Proceedings of ACL-Europe.*

Becker, T., M. Niv, O. Rambow 1992. *The Derivational Generative Power of Formal Systems or Scrambling is Beyond LCFRS.* IRCS Report 92-38, IRCS, University of Pennsylvania.

Candito M.-H., Kahane S. 1998. 'Can the TAG derivation tree represent a semantic graph? An answer in the light of Meaning-Text Theory.' *Proc. Fourth Workshop on Tree-Adjoining Grammars and Related Frameworks (TAG+4).*

Candito, M.H., 1999. *Structuration d'une grammaire LTAG : Application au français et à l'italien.* Ph.D. thesis, University of Paris 7.

Choi, H-W. 1999.*Optimizing Structure in Context – Scrambling and Information Structure*, CSLI, Stanford.

Coch, J., 1998. "MultiMeteo: Interactive weather report generation" in: *Proceedings of the 9th International Natural Language Generation Workshop (INLG'98)* Niagara-on-the-Lake, Canada.

Drach, E. 1937. *Grundgedanken der deutschen Satzlehre*, Diesterweg, Frankfurt.

Gaiffe B., B Crabbé, A. Roussanaly 2002. "A New Metagrammar compiler" in *Proceedings of the Sixth International Workshop on Tree Adjoining Grammar and Related Frameworks (TAG+6).* Universitá di Venezia.

Gerdes, K., S. Kahane 2001. "Word Order in German: A Formal Dependency Grammar Using a Topological Hierarchy", in *Proceedings ACL 2001,* Toulouse.

Kathol, A. 1995. *Linearization-based German Syntax*, PhD thesis, Ohio State University.

Lenerz J. 1977. *Zur Abfolge nominaler Satzglieder im Deutschen*, Tübingen, Narr.

Müller St. 1999, *Deutsche Syntax deklarativ: Head-Driven Phrase Structure Grammar für das Deutsche*, Linguistische Arbeiten 394; Niemeyer: Tübingen.

Rambow, O. 1994. *Formal and Computational Aspects of Natural Language Syntax*, Institute For Research in Cognitive Science, PhD thesis, University of Pennsylvania, Philadelphia.

Rambow, O., K. Vijay-Shanker, D. Weir 1995. "D-Tree Grammars", in *Proceedings of ACL'95.*

Rambow, O, B. Santorini 1995 "Incremental Phrase Structure Generation and a Universal Theory of V2" in *Proceedings of NELS 25*, Amherst, MA.

Rogers, J., Vijay-Shanker, K., "Obtaining Trees from Their Descriptions : An Application to Tree-Adjoining Grammars", *Computational Intelligence*, 10, 4, 1994.

Schabes Y., S. Shieber 1994. "An alternative conception of tree-adjoining derivation". in *Computational Linguistics*, 20(1).

The XTAG Research group 1995. *A lexicalized Tree Adjoining Grammar for English.* IRCS-95-03, Institute of Research in Cognitive Science, Philadelphia.

Cross-Serial Dependencies in Tagalog

Anna Maclachlan and Owen Rambow
Discern Communications and University of Pennsylvania
anna@discerncomm.com *and* rambow@unagi.cis.upenn.edu

1. Cross-Serial Dependencies

There are two salient linguistic uses of adjunction: for analyzing long-distance *wh*-movement (and related movement types) in many languages and for analyzing cross-serial dependencies (CSD) in Dutch and Swiss German. While the need for and the adequacy of adjunction to model *wh*-type movement have been questioned (Rambow and Vijay-Shanker, 1998; Rambow et al., 2001), CSD seems ideally suited for a TAG analysis, since, as Shieber (1985) showed, CSD cannot be derived by a context-free grammar. In fact, some of the alternate tree rewriting systems proposed which do not include adjunction, such as the DSG of (Rambow et al., 2001), cannot provide a satisfactory analysis of CSD, either: it is specifically the definition of adjunction as an tree-rewriting operation that inserts one tree in its entirety into the center of another that is crucial for deriving CSD. What is somewhat troubling, however, is that the construction appears to be limited to two West Germanic languages/dialects, Dutch and Swiss German. In this paper, we show that the same construction, though with different syntactic characteristics, is found in a completely unrelated language, Tagalog. We show how the analysis of Kroch and Santorini (1991) for Dutch can be adapted for Tagalog, and we show furthermore that the syntactic analysis suggested by TAG is preferable to an analysis based on head movement and verb incorporation.

2. The Tagalog Data

Tagalog, a major Austronesian language spoken in the Philippines, is strongly verb first. Complements and the subject follow the verb with preferences for the agent to directly follow the verb and for the nominative argument to be last (preferences which can be in conflict). The nominals are case marked for nominative (NOM) and oblique (OBL), and another distinguished case is un-glossed in the examples (for a discussion of this case as both ergative and accusative see (Maclachlan, 1994)). The standard ordering in complex sentences is V1 Agent1 linker [V2 (Agent2) Theme2], as shown in example (1a). Phrases of various sorts are separated by a linker (LK) and Tagalog also has sentential conjunction (CONJ). A cross serial dependency ordering alternates with this basic ordering in which the agent of the matrix clause follows the embedded verb as in (1b): V1 linker V2 Agent1 (Agent2) Theme2.[1]

(1) Basic and CSD alternates

a. Nagisip si Pedro-ng bumili ng bulaklak
AT-thought NOM-Pedro-LK AT-buy flower

b. Nagisip na bumili si Pedro ng bulaklak
AT-thought LK AT-buy NOM-Pedro flower

'Pedro thought to buy (of buying) a flower.'

Let us note two further properties of the CSD for which we will account with a TAG analysis. First, the CSD process can be iterated:

(2) Iteration of CSD

1. Baldridge (1998) claims that Tagalog simply has long-distance scrambling and that a $V_1V_2N_2N_1$ ordering is also allowable. However, he gives only one example, and admits that the sentence may have a completely different interpretation (in which the two NPs form one NP). We have, in our work with native speaker informants, not found any evidence for generalized long-distance scrambling, and therefore will assume for the sake of this paper that we have a CSD, not a long-distance scrambling construction. If it were in fact a long-distance scrambling construction, TAG would not be powerful enough for an analysis – see (Rambow, 1994) for a discussion of German. Baldridge (1998) also discusses asymmetries in *wh*-extraction in Tagalog. We do not address the issue here.

a. Ipiniangako ni Maria-ng subuka-ng manalo sa karera
promised Maria-LK try-LK win OBL-race

b. Ipiniangako-ng subukan ni Maria-ng manalo sa karera
promised-LK try Maria-LK win OBL-race

c. Ipiniangako-ng subuka-ng manalo ni Maria sa karera
promised-LK try-LK win Maria OBL-race

'Maria promised to try to win the race.'

Second, the CSD sentence permits only one NOM nominal, while the basic complex sentence permits two. This can be seen when the theme is NOM in the embedded clause as in (3). In this essentially passive clause type, the verb is marked with Theme Topic morphology (TT) whereas in the essentially active clause type, as in both clauses in (1), the verb is also marked but with Agent Topic morphology (AT). While the matrix agent is NOM in (3a) it cannot be in the CSD equivalent in (3b) as long as the embedded theme is NOM

(3) Basic and CSD alternates with embedded passivization

a. umasa si Maria-ng sulatin ang kuwento
AT-hoped NOM-Maria-LK write-TT NOM-story

b. umasa-ng sulatin {*si Maria / ni Maria} ang kuwento
AT-hoped-LK write-TT {*NOM-Maria / Maria} NOM-story

'Maria hoped to write the story.'

3. A TAG Analysis

In TAG, we derive CSD by recursively adjoining elementary trees into each other at interior nodes. As is the case with all embedded clause constructions (be they CSD or not), each clause is adjoined into its immediately embedded clause, since the most deeply embedded clause does not have a linguistically meaningful footnode labeled with a clausal category (and hence its embedding clause must adjoin into it, rather than *vice versa*). When we adjoin an auxiliary elementary tree such as that shown schematically at the top left in Figure 1 (the superscript 1 indicates that this represents the matrix clause) to the initial tree at the top-right (the superscript 2 indicates that this is the embedded clause, we are only considering one level of embedding in this schematic discussion, though of course the process can iterate), the result is as shown below in Figure 1. The nodes labeled A, B, C, D represent either substitution or terminal nodes – in either case, these are positions below which terminals can be generated. Of course, we do do not expect terminal symbols to be generated below each of these symbols. In fact, if we restrict ourselves to the case in which we have one (overt) noun phrase and one verb in each elementary tree, two of the symbols will dominate the empty string. If we choose A and D to dominate the empty string, we obtain a center-embedded structure with the associated string $B^{(1)}B^{(2)}C^{(2)}C^{(1)}$, as desired. The derivation is essentially a context-free derivation and does not actually make use of the full power of adjunction, since no terminal nodes are generated above the adjunction site.

It is clear that to obtain CSD, we must choose as overt terminal nodes one above the adjunction site (A or D) and one below (B or C). If we choose, say, A and B as the overt nodes, we obtain a structure which is not derivable with a context-free grammar, but the string nonetheless represents center-embedding ($A^{(2)}A^{(1)}B^{(1)}B^{(2)}$). Thus, we must choose one overt terminal to the left of the spine, and one to the right. This leaves us with exactly two possibilities – A and C are overt, or B and D. Since in Dutch and Swiss German CSD, the first element is always a matrix noun phrase (and not an embedded one), we cannot use A and C as the overt elements: while adjoining the matrix clause into the embedded clause would result in $A^{(2)}A^{(1)}C^{(2)}C^{(1)}$, with cross-serial dependencies, the string starts with the wrong A: $A^{(2)}$ rather than $A^{(1)}$. Thus, we must leave A and C empty, with the overt material in B and D.

This is of course exactly the choice that Kroch and Santorini (1991) make. They propose that in Dutch,[2] the verb raises from its ordinary position as sister to the S footnode to a position above the adjunction site (which

2. Their analysis also applies to the relevant Swiss German data.

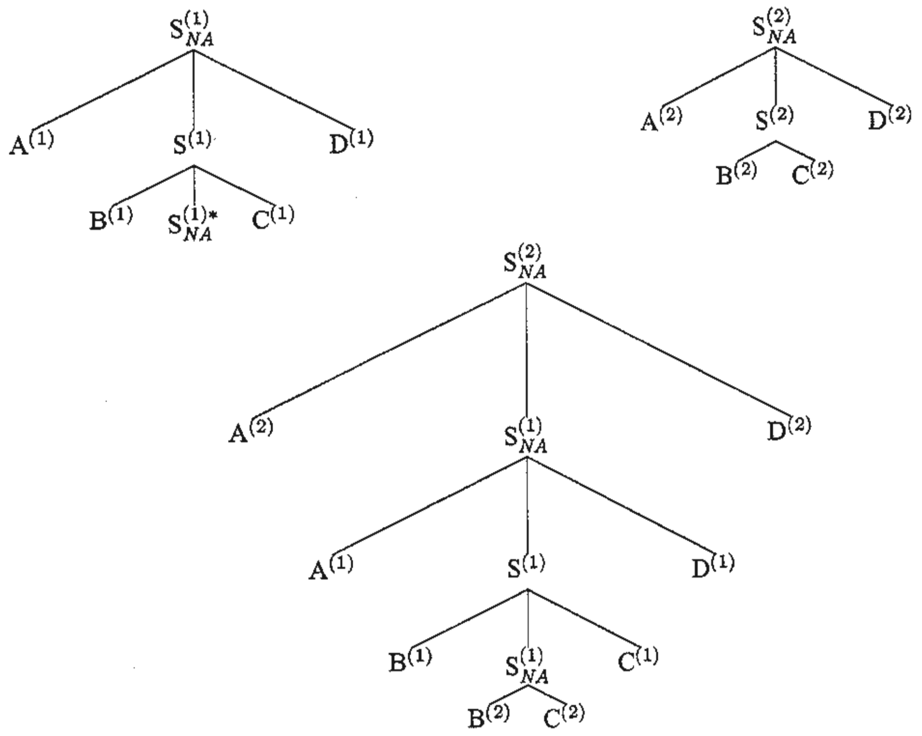


Figure 1: Elementary trees (auxiliary, above left,and initial, above right) and derived tree obtained by adjoining the auxiliary tree into the initial tree at the latter's interior S node(below)

can be interpreted as right-Chomsky-adjoining[3] to the regular maximal projection). What makes their analysis so compelling is that this analysis, in which the verb "raises" to a higher position in the tree, takes up some elements of the analysis suggested previously in the Germanic syntax literature. In this analysis, which dates to at least Bech (1955) and was expressed somewhat more formally in a transformational framework by Evers (1975), the verbs actually raise out of their clauses and form a single morphological unit. Such an analysis is impossible in TAG, since apart from the effect of adjunction, the elementary tree retain their structural integrity. Furthermore, Kroch and Santorini (1991) argue that there are empirical arguments against a morphological verb cluster, though not against verb raising itself. Thus, the analysis proposed by Kroch and Santorini (1991) is the closest possible TAG-based analysis which uses the independently proposed notion of verb raising (but not verb cluster formation), and it also corresponds to the only possible analysis considering the topology of trees and the definition of adjunction!

Let us now turn to Tagalog. In Tagalog, we have a verb-initial construction rather than a verb-final construction. However, the argument about possible analyses is exactly the same as in Dutch, and we conclude that B and D must be overt, not A and C. Because Tagalog is verb-initial, we must choose B to represent the verb, and D to represent the noun phrase. We thus are forced to adopt an analysis in which the NP is raised, and in which it is the raising of the NP which results in the CSD.[4] This is shown in Figure 2 (the subscripts indicate the relation between traces and moved elements within elementary trees, while the superscripts, as before, indicate which clause a terminal symbol belongs to). The trees in this figure derive the CSD version of (1), (1b), repeated here for convenience:

(4) Basic and CSD alternates (=(1))

a.	Nagisip	si Pedro-ng	bumili	ng bulaklak
	AT-thought	NOM-Pedro-LK	AT-buy	flower

3. we use "Chomsky-adjoining" to refer to derivation processes within elementary trees (following the general approach of (Frank, 2001)), while "adjoining" refers to the TAG operation that combines elementary trees.
4. This does not mean that the verb cannot also raise from a VP-internal position to a higher position on its own projection, as is customarily assumed for verb-initial languages. It just means that the landing site of the verb must be below the node at which adjunction of the matrix clause happens. This is in fact exactly the analysis in Figure 2.

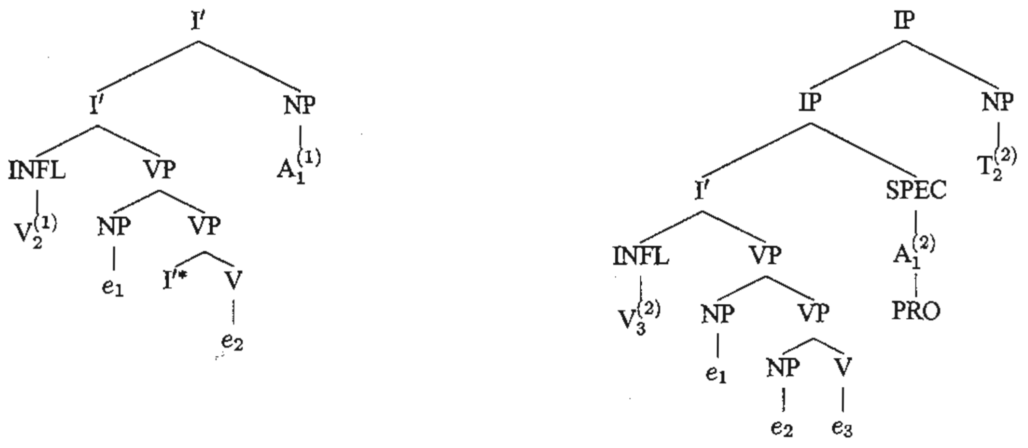


Figure 2: Elementary trees for Tagalog clauses in CSD order: matrix clause (left) and embedded clause (right)

b. Nagisip na bumili si Pedro ng bulaklak
AT-thought LK AT-buy NOM-Pedro flower

'Pedro thought to buy (of buying) a flower.'

We assume that Tagalog is underlyingly verb-final,[5] and in both trees the verb has raised from V to INFL. In the matrix clause on the left (with superscript (1)), the verb also subcategorizes for an I′ (the footnode) and an NP, the agent, which has raised to a position to the right of the spine, Chomsky-adjoining to I′. In the embedded clause on the right, the verb subcategorizes for two NPs, the first of which (the agent) is realized as PRO and occupies the SPEC of IP position, and the second of which (the theme) has scrambled out beyond the agent. The matrix clause adjoins to the matrix clause at its I′ node, and the result is the derived tree shown in Figure 3, with the word order in sentence (4b). We will postpone a discussion of the non-CSD case (4a) until after a discussion of case assignment.

The question arises whether this analysis, imposed on us as it is by the definition of adjunction. is independently motivated. As it turns out, in Tagalog there is independent linguistic evidence for NP raising (just as there is independent linguistic evidence for verb raising in Dutch). The evidence comes from the agreement facts in Tagalog. In the following, we will assume that NOM (a case marker which does not fully correspond to nominative case in other languages) is assigned through SPEC-head agreement, and that IP is SPEC-final.[6] Rather than being associated with tense or aspect marking, NOM is associated with topic morphology (TT and AT) in Tagalog. We will assume that other, non-NOM arguments (including the clausal argument) may optionally leave the VP and Chomsky-adjoin to the I′ or IP nodes (without being assigned NOM). This optional movement of arguments is at the heart of the availability of two analyses in Tagalog: we obtain the "basic" word order when the clausal argument has been moved beyond the NOM argument, as shown in Figure 4 on the left. We will assume that when the clausal argument has not moved out of the VP, it cannot project to more than I′, presumably for reasons related to case (or some extended notion of case). This will prevent us from deriving nested dependencies. At the same time, this asymmetry – between a node label of IP for the basic word order, and a node label of I′ for the CSD word order – reflect a widely held intuition that in the CSD order, the clauses are less "sepaarted" from one another than in the basic word order. Similar intuitions in Dutch (and other West Germanic languages) led Bech (1955) and Evers (1975) to postulate the existence of only one clause in certain constructions ("clause union").

We now turn to the question why in the CSD construction, there can be at most one NOM-marked nominal argument, while in the basic construction, each clause may have its own. It is clear that in our analysis of the basic construction, there are two IPs each with its own SPEC position, so that two NOM cases can be assigned independently, as in (3a), repeated here for convenience.

(5) Basic and CSD alternates with embedded passivization (=(3))

5. This assumption is actually irrelevant from the TAG point of view, but it is consistent with much recent work on the syntax of V1 languages.
6. There is cross-linguistic support for this claim from other Austronesian languages such as Malagasy.

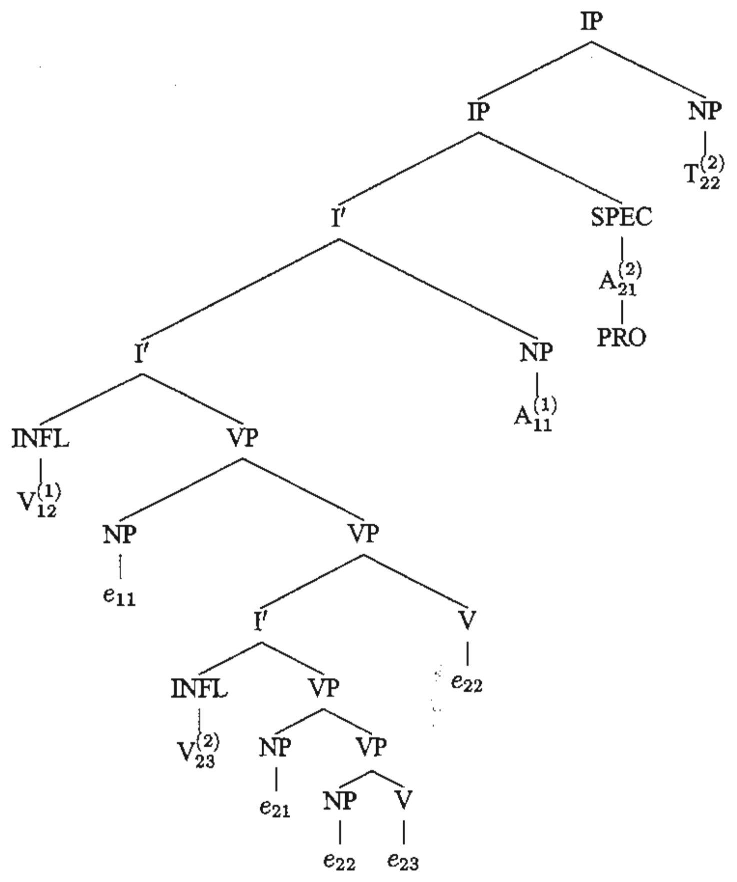


Figure 3: Derived tree for Tagalog clauses in CSD order

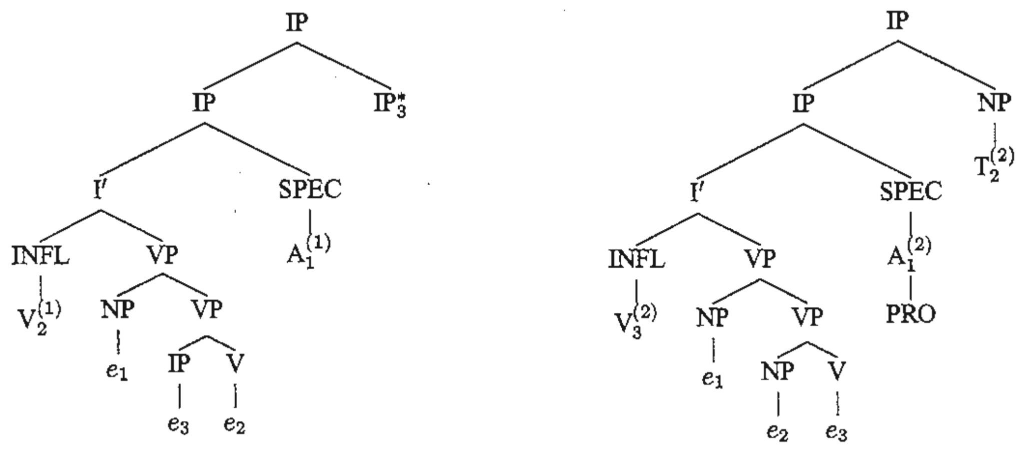


Figure 4: Elementary trees for Tagalog clauses in standard order: matrix clause (left) and embedded clause (right)

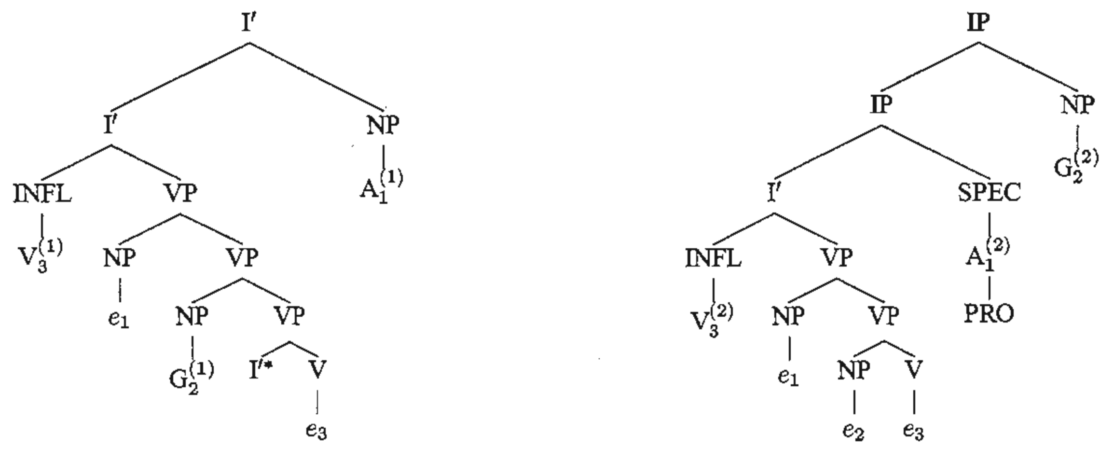


Figure 5: Elementary trees for Tagalog clauses in CSD order with interleaved nominal arguments, corresponding to sentence (7): matrix clause (left) and embedded clause (right)

a. umasa si Maria-ng sulatin ang kuwento
AT-hoped NOM-Maria-LK write-TT NOM-story

b. umasa-ng sulatin {*si Maria / ni Maria} ang kuwento
AT-hoped-LK write-TT {*NOM-Maria / Maria} NOM-story

'Maria hoped to write the story.'

We might be tempted to derive the fact that the matrix clause agent cannot receive NOM case along with the embedded verb theme (as shown in (5b)) from the fact that the matrix clause simply does not have a SPEC position for INFL, and therefore it cannot have a NOM-marked argument. However, this cannot be the whole story, since it is clearly wrong: in (4) we have a NOM-marked matrix argument in both the basic and CSD word orders. A different approach would be to postulate the existence of a feature shared across and between projections of different verbs, but not across IP nodes (again capturing the same intuition of "clause union"). This feature would ensure that only one constituent is marked NOM. However, this theory misses an additional complicating factor: in (5), it is altogether impossible for the matrix agent to be NOM-marked, whether or not the embedded theme is NOM-marked:

(6) * umasa-ng sulatin si Maria {ang kuwento / ng kuwento}
AT-hoped-LK write-TT NOM-Maria {NOM-story / story}

Intended meaning: 'Maria hoped to write the story.'

We therefore return to an analysis which exploits the node labels, but we make an additional assumption: case assignment can be shared between an overt argument and a PRO it controls (just as other features, such as referential indices, are shared). Thus, the matrix agent in (4b) gets its NOM not from the matrix SPEC of IP (since there is none), but rather through co-indexation with the embedded agent, which is PRO in SPEC of IP of the embedded verb. In fact, in all cases in which we have a NOM case in a matrix clause of a CSD, it is the controler of a PRO in the lower clause which must be in SPEC of IP, given the verbal morphology. While the notion of case being passed from PRO to its controler may seem at first strikingly odd, we note that in the derived tree (see Figure 3), the PRO actually c-commands its controler!

4. Evidence Against Verb Incorporation

We point out a series of cases that are readily handled in the TAG analysis which pose a problem for two other types of analyses of CSDs. As argued in (Maclachlan, 1991), analyses that assume a mechanism like head-to-head movement of an embedded verb head into the matrix verb head similar to a causative verb incorporation approach or a morphological verb complex (Evers, 1975) cannot account for these cases. It is possible for a phrase from the matrix clause to occur intermingled with embedded clause elements even while other matrix clause elements remain in place. Namely, negation, floated quantifiers and full phrases can intervene between the two verbs of a CSD. One such example is given in (7) where the matrix clause has an agent, a sentential and an oblique argument. The basic order is V1 Agent1 Goal1 [CONJ V2 Goal2] as in (7a) but a possible CSD order is V1 *Goal1* CONJ V2 Agent1 Goal2 as in (7b) where the matrix oblique phrase remains between the verbs.

(7) Basic and CSD alternates with intervening phrase

a. Sinabi ni Fe kay Juan kung kailan tatawag kay Maria
TT-said Fe OBL-Juan CONJ when AT-call OBL-Maria

b. Sinabi kay Juan kung kailan tatawag ni Fe kay Maria
TT-said OBL-Juan CONJ when AT-call Fe OBL-Maria

'Fe told Juan when to call Maria.'

We can derive such cases by assuming that in the analysis in (2), the matrix goal has remained in the VP, while the matrix agent is Chomsky-adjoined to I′. (Note that no NOM is assigned in this example.) This is shown in Figure 5. Crucial to our analysis is the fact that each verb has its own projection which enters into the derivation fully formed, thus allowing the intervening material.

5. Conclusion

In this paper, we have discussed cross-serial dependencies in Tagalog. As in the case of Dutch discussed by Kroch and Santorini (1991), the definition of adjunction leaves only one possible way of using TAG in the linguistic analysis. As in Dutch, but for completely different reasons, this analysis is independentlty motivated by other (non-TAG) linguistic analyses. Furthermore, as in the case of Dutch, analyses have been proposed for Tagalog which include processes such as clause union or verb complex formation/verb incoproration which cannot readily be modeled by TAG. Again, as in the case of Dutch, there is independent empirical evidence against such a process. The striking parallel betwen Dutch and Tagalog, despite the stark differences in syntax between the two languages, lends further credence to the claim that adjunction represents a linguistically meaningful operation.

References

Baldridge, Jason Michael (1998). Local scrambling and syntactic asymmetries in Tagalog. Master's thesis, University of Pennsylvania.

Bech, Gunnar (1955). *Studien über das deutsche Verbum infinitum.* Det Kongelige Danske videnskabernes selskab. Historisk-Filosofiske Meddelelser, bd. 35, nr.2 (1955) and bd. 36, nr.6 (1957). Munksgaard, Kopenhagen. 2nd unrevised edition published 1983 by Max Niemeyer Verlag, Tübingen (Linguistische Arbeiten 139).

Evers, Arnold (1975). *The Transformational Cycle in Dutch and German.* PhD thesis, University of Utrecht. Distributed by the Indiana University Linguistics Club.

Frank, Robert (2001). *Phrase Structure Composition and Syntactic Dependencies.* MIT Press, Cambridge, Mass.

Kroch, Anthony and Santorini, Beatrice (1991). The derived constituent structure of the West Germanic Verb Raising construction. In Freidin, R., editor, *Principles and parameters in comparative grammar*, pages 269–338. MIT Press, Cambridge, Mass.

Maclachlan, Anna (1991). Implications of a Tagalog construction for incorporation theory constraints. In Wilson, Tom, editor, *Proceedings of the Annual Conference of the Canadian Linguistic Association*, pages 209–220, Toronto.

Maclachlan, Anna (1994). Conjunction reduction and the syntax of case in Tagalog. In Ode, Cecilia and Stokhof, Wim, editors, *Proceedings of the Seventh International Conference on Austronesian Linguistics*, pages 443–460, Leiden, the Netherlands.

Rambow, Owen (1994). *Formal and Computational Aspects of Natural Language Syntax.* PhD thesis, Department of Computer and Information Science, University of Pennsylvania, Philadelphia. Available as Technical Report 94-08 from the Institute for Research in Cognitive Science (IRCS) and also at ftp://ftp.cis.upenn.edu/pub/rambow/thesis.ps.Z .

Rambow, Owen and Vijay-Shanker, K. (1998). *Wh*-islands in TAG and related formalisms. In *Proceedings of the Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+4)*, number 98–12 in IRCS Report, pages 147–150. Institute for Research in Cognitive Science, University of Pennsylvania.

Rambow, Owen; Vijay-Shanker, K.; and Weir, David (2001). D-Tree Substitution Grammars. *Computational Linguistics*, 27(1).

Shieber, Stuart B. (1985). Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8:333–343.

Reranking an N-Gram Supertagger

John Chen*, Srinivas Bangalore*, Michael Collins*, and Owen Rambow†
**AT&T Labs–Research, † University of Pennsylvania*
{jchen, srini, mcollins}@research.att.com, rambow@unagi.cis.upenn.edu

1. Introduction

As shown by Srinivas (1997), standard n-gram modeling may be used to perform supertag disambiguation with accuracy that is adequate for partial parsing, but in general not sufficient for full parsing. A serious problem is that n-gram modeling usually considers a very small, fixed context and does not perform well with large tag sets, such as those generated by automatic grammar extraction (Xia, 1999; Chen and Vijay-Shanker, 2000; Chiang, 2000). As an alternative, Chen, Bangalore and Vijay-Shanker (1999) introduce class-based supertagging. An example of class tagging is n-best trigram-based supertagging, which assigns to each word the top n most likely supertags as determined by an n-gram supertagging model. Class-based supertagging can be performed much more accurately than supertagging with only a small increase in ambiguity. In a second phase, the most likely candidate from the class is chosen.

In this paper, we investigate an approach to such a choice based on reranking a set of candidate supertags and their confidence scores. RankBoost (Freund *et al.*, 1998) is the boosting algorithm that we use in order to learn to rerank outputs. It also has been used with good effect in reranking outputs of a statistical parser (Collins, 2000) and ranking sentence plans (Walker, Rambow and Rogati, 2001). RankBoost may learn to correct biases that are inherent in n-gram modeling which lead to systematic errors in supertagging (cf. (van Halteren, 1996)). RankBoost can also use a variety of local and long distance features more easily than n-gram-based approaches (cf. (Chen, Bangalore and Vijay-Shanker, 1999)) because it makes sparse data less of an issue.

The outline of this paper is as follows. First, we develop the background and motivations behind the task of reranking the output of an n-best trigram supertagger. Second, we introduce RankBoost as the approach that we adopt in order to train the reranker. Third, we perform an initial set of experiments where the reranker is trained with different feature subsets. Fourth, we perform an in-depth analysis of several reranking models. Fifth, after pointing out causes that at times render the reranker ineffective, we develop and test some new models that attempt to sidestep these limitations. Lastly, after some significance testing results, we state our conclusions and remark on potential future directions.

2. Background and Motivation

In this section, we motivate the desirability of exploring the use of n-best reranking of supertags. Although we give multiple motivations, we focus on justifying our approach as a promising alternative in improving the performance of a full parser. First, we review the supertagging task and its applications. Because supertagging requires the existence of a particular TAG, we subsequently introduce automatically extracted TAGs and motivate their use. Although extracted grammars have their advantages, supertagging using automatically extracted TAGs runs into damaging sparse data problems. We review n-best supertagging as one means of alleviating these problems. Lastly, we run experiments that show supertagging is potentially a viable option in order to speed up a full parser. Throughout this section, we describe the kinds of linguistic resources that we use in all of our experiments and the kinds of notation that we will employ in the rest of this paper.

2.1. Supertagging

Supertagging (Bangalore and Joshi, 1999) is the process of assigning the best TAG elementary tree, or supertag, to each word in the input sentence. It performs the task of parsing disambiguation to such an extent that it may be characterized as providing an almost parse. There exist linear time approaches to supertagging, providing one promising route to linear time parsing disambiguation. However, Srinivas (1997) shows that standard n-gram modeling may be used to perform supertagging with accuracy that is adequate for partial parsing, but not for full parsing. On the other hand, n-gram modeling of supertagging has been found to be useful in other applications such as information retrieval (Chandrasekhar and Srinivas, 1997b) and text simplification (Chandrasekhar and Srinivas, 1997a).

2.2. Automatically Extracted Grammars

Recently, procedures have been developed that automatically extract TAGs from broad coverage treebanks (Xia, 1999; Chen and Vijay-Shanker, 2000; Chiang, 2000). They have the advantage that linguistically motivated TAGs can be extracted from widely available treebanks without a huge investment in manual labor. Furthermore, because of their direct extraction from a treebank, parameters can be easily and accurately estimated for building statistical TAG models for parsing (Chiang, 2000; Sarkar, 2001) or generation (Bangalore, Chen and Rambow, 2001).

In our experiments, we use an automatically extracted TAG grammar similar to the ones described by Chen and Vijay-Shanker (2000). This grammar has been extracted from Sections 02-21 of the Penn Treebank (Marcus, Santorini and Marcinkiewicz, 1993). It contains 3964 tree frames (non-lexicalized elementary trees). The parameters of extraction are set as follows. Each tree frame contains nodes that are labeled using a label set similar to the XTAG (XTAG-Group, 2001) label set. Furthermore, tree frames are extracted corresponding to a "moderate" domain of locality. Also, only those empty elements in the Penn Treebank that are usually found in TAG (subject and object trace, for example) are included in this grammar.

2.3. N-best Supertagging

The efficacy of n-gram modeling of supertagging is limited by sparse data problems of very large TAGs, such as those that are automatically extracted from broad coverage treebanks. Chen and Vijay-Shanker (2000) show that supertagging using extracted TAGs is performed at a lower accuracy (around 80%) than accuracies that have been published for supertagging using hand-written TAGs (around 90%). Faced with this evidence, it might seem that it is a hopeless task to use supertagging using extracted TAGs as a preprocessing step to accelerate full parsing. On the other hand, Chen, Bangalore and Vijay-Shanker (1999) investigate class-based supertagging, a variant of supertagging where a small set of supertags are assigned to each word instead of a single supertag. The idea is to make the sets small enough to represent a significant reduction in ambiguity so as to speed up a full parser, but to construct the sets so that class-based supertagging is much more accurate than supertagging.

One such promising class-based supertagging model is n-best supertagging, where a trigram model assigns up to n supertags for each word of the input sentence. Let $W = w_1, \ldots, w_n$ represent the sequence of words that is the input to a supertagger. Let $T_{tri} = t_{1,1}, \ldots, t_{n,1}$ be the output of the (1-best) trigram supertagger. The output of the n-best supertagger is a sequence of n-best supertags NBEST$(i) = t_{i,1}, \ldots, t_{i,n(i)}$ for each word w_i such that each supertag $t_{i,j}$ has an associated confidence score $c_{i,j}$. Assume that each sequence NBEST(i) is sorted in descending order according to these confidence scores.

The n-best supertagger is obtained by a modification of the (1-best) trigram model of supertagging. Both supertaggers first use the Viterbi algorithm to find T_{tri} by computing the most likely path $p(T_{tri})$ through a lattice of words and pairs of supertags. In the trigram supertagger, each node k along the path $p(T_{tri})$ is associated with exactly one prefix probability (the highest). In contrast, the corresponding node k in the n-best supertagger is associated with the n highest prefix probabilities. This difference allows the n-best supertagger to associate up to n supertags for each word w_i. The confidence score $c_{i,j}$ of supertag $t_{i,j}$ is the jth-best prefix probability of a node k divided by the least best prefix probability of the same node.

2.4. Parsing with N-best Supertagger Output

We claim that supertagging is a viable option to explore for use as a preprocessing step in order to speed up full parsing. In order to substantiate this claim, we perform exploratory experiments that show the relationship between n-best supertagging and parsing performance. Using the grammar that is described in Section 2.2, we train n-best supertaggers on Sections 02-21 of the Penn Treebank. For each supertagger, we supertag Section 22, which consists of about 40,100 words in 1,700 sentences. We then feed the resulting output through the LEM parser, a head-driven TAG chart parser (Sarkar, 2000). Given an input sentence and a grammar, this parser either outputs nothing, or a packed derivation forest of every parse that can be assigned to the sentence by the grammar. It does not return partial parses.

The results of these experiments are shown in Table 1. The input to the parser can be the output of either a 1, 2, or 4-best supertagger. It can also be sentences where each word is associated with all of the supertags with that word's part of speech, as determined by a trigram part of speech tagger. This is labeled as "POS-tag" in the table. Lastly, it can simply be sentences where each word is associated with the correct supertag. This is labeled as "Key." The table shows the supertagging accuracy of each corpus that is input to the parser. It also shows each

Table 1: Relationships between n-best supertagging and parsing

Input to Parser	% Supertagging Accuracy	Ambiguity (supertags/word)	% Sentences Receiving Some Parse	Time to Parse Corpus
1-best	81.47	1.0	28.2	< 3 hours
2-best	88.36	1.9	53.6	< 2 days
4-best	91.41	3.6	76.7	2-3 weeks
8-best	92.77	6.3	-	-
POS-tag	97.30	441.3	-	-
Key	100.00	1.0	97.0	< 5 hours

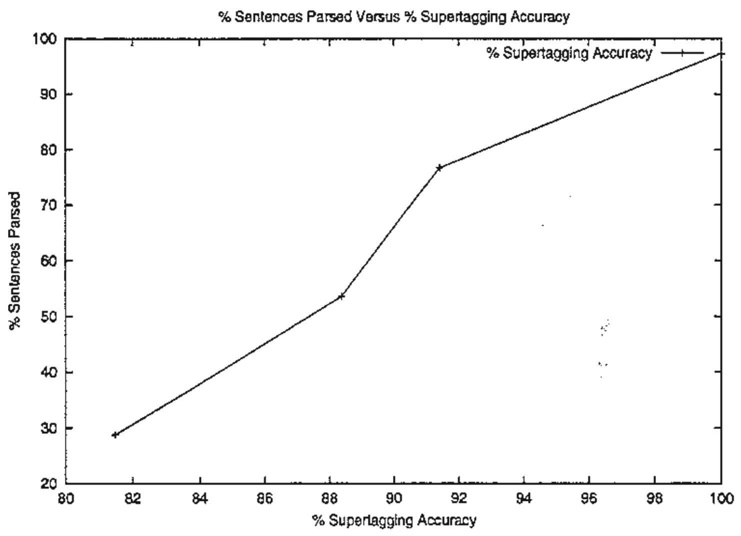


Figure 1: Percentage of Sentences That Were Parsed Versus Percent Supertagging Accuracy

corpus's ambiguity in supertags per word, the percentage of sentences in the corpus which the parser successfully found a parse, and also the time to parse the corpus. Parsing results are not available for "8-best" and "POS-tag" because of the unreasonable amount of time the parser takes for those kinds of corpora.

Table 1 reveals some interesting aspects of the relationship between supertagging and parsing. For example, it shows that merely doing part of speech tagging is inadequate as a preprocessing step if the purpose is to significantly speed up full parsing. In contrast, it also shows that the 1-best supertagger does speed up full parsing, but at the cost of missing many parses of sentences. Row "Key" shows that if supertagging works accurately enough, then it would indeed fulfill the promise of speeding up a full parser.

The second column of Table 1 is plotted against its fourth column in Figure 1. It shows how the percentage of parsed sentences in the test corpus increases as the supertagging accuracy on the test corpus increases. There is the obvious result that a higher supertagging accuracy always leads to a greater percentage of sentences being able to be parsed. There is apparently a less obvious result that this relationship is non-linear; the steepest increase in percentage of parseable sentences occurs for supertagging accuracies between 88% and 92%.

We have seen that full parsing of automatically extracted TAG grammars is apparently quite slow. We have also seen that simply part of speech tagging the input sentences as a preprocessing step does not seem to reduce ambiguity to a sufficient degree in order to speed up full parsing to a desirable extent. On the other hand, we have shown that 1-best supertagging does indeed speed up full parsing considerably—at least more than tenfold. However, in order for supertagging to fully parse a considerable portion of a corpus, it is necessary to achieve sufficiently high supertagging accuracies. Regarding the use of n-best supertagged input to a parser, we have seen that it is best to keep $n \leq 3$ in order to prevent extreme degradation in parsing performance.

2.5. Summary

We have seen that reranking the output of an n-best supertagger based on a TAG extracted from a treebank is attractive for a variety of reasons. Use of such a TAG is justified because parameters for stochastic models can be estimated easily and accurately. Use of an n-best supertagger is justified because of the considerable potential error reduction and its implications. In particular, it can be clearly seen from Table 1 that an optimal reranking of the output of an 8-best supertagger would achieve a more than 50% reduction in supertagging error. It is not unreasonable to believe that this would greatly improve the performance of applications based on supertagging, such as information retrieval and text simplification. Furthermore, Figure 1 shows that this error reduction would greatly increase the viability of using supertagging as a preprocessing step to speed up parsing.

3. Reranking an N-Best Supertagger

Our reranker takes as input a set of sentences that has been supertagged by an 8-best supertagger, including a confidence score for each selected supertag. It then ranks them according to its model. This model is trained using the machine learning program RankBoost (Freund *et al.*, 1998) which learns from sets of correctly supertagged sentences the same sentences that have been supertagged using an 8-best supertagger.

We use the variant of RankBoost introduced by (Collins, 2000). Further information about RankBoost is found in (Schapire, 1999). RankBoost learns from a set of examples. For our purpose, an example is an occurrence of a word w_i in a particular sentence along with its supertag $t_{i,j}$ selected by an n-best supertagger and its confidence score $c_{i,j}$. Each example is associated with a set of m binary indicator functions $h_s(t_{i,j})$ for $1 \leq s \leq m$. For example, UNI(w,s) is a two-argument feature template that states that the current word w has supertag s. When this template is instantiated with w_i =book and $t_{i,j}$ = αNXN, we obtain the following indicator function: function might be

$$h_{1234}(t_{i,j}) = \begin{cases} 1 & if\ t_{i,j} = \alpha NXN\ and\ w_i = book \\ 0 & otherwise \end{cases} \tag{1}$$

Each indicator function h_s is associated with its own parameter α_s. There is also a parameter α_0 associated with the confidence score. Training is a process of setting the parameters α to minimize the loss function:

$$loss(\alpha) = \sum_{i,j} e^{-\left(\alpha_0(ln(c_{i,1})-ln(c_{i,j}))+\sum_s \alpha_s(h_s(t_{i,1})-h_s(t_{i,j}))\right)} \tag{2}$$

At the start of training, no features are selected, i.e., all of the α_s's are set to zero. The optimization method that is used in training is greedy; at each iteration it picks a feature h_s which has the most impact on the loss function. The result is a set of indicator functions whose output on a given candidate is summed. These sums are used to rerank a set of candidates. Another set of examples—tuning data—is used to choose when to stop.

4. Initial Experiments

A set of features is required in order to train RankBoost to rerank supertags. As pointed out by Srinivas (1997), the traditional n-gram modeling of supertagging suffers from the flaw of only considering local dependencies when deciding how to supertag a given word. This is counter to one of the attractions of the TAG formalism, namely that even long distance dependencies are localized within a given TAG (Schabes, Abeillé and Joshi, 1988). Chen, Bangalore and Vijay-Shanker (1999) provide an example sentence where non-local context is needed to determine the correct supertag: "Many Indians *feared* their country *might* split again." Here, the supertag for the word *feared* is partially determined by the proximity of the word *might*. Chen, Bangalore and Vijay-Shanker (1999) introduce the notion of *head supertag* context which they show increases supertagging accuracy when suitably folded into a stochastic model. While the notion of head supertags can be useful, it cannot be straightforwardly applied to our current situation; determining head supertags was feasible in (Chen, Bangalore and Vijay-Shanker, 1999) because they used the XTAG grammar, whereas it is not immediately clear which supertags should be head supertags in our extracted grammar, which is an order of magnitude larger than the XTAG grammar (3964 tree frames in the extracted grammar versus 500 tree frames in the XTAG grammar).

Chen, Bangalore and Vijay-Shanker (1999) make it clear, however, that both local and long distance features are important. In that spirit, we have designed an initial set of feature templates that is shown in Table 2. For example, UNI is a two-argument feature template that states that the current word w_0 has the supertag $t_{0,1}$. Feature

Table 2: Feature Templates Used In Initial Experiments

w_i	ith word in input sentence relative to current word which is w_0
t_i	supertag of ith word in input sentence relative to current word which is w_0

Name	Parameter List	Example of Instantiation
UNI	$w_0, t_{0,j}$	w_0 =book, $t_{0,j} = \alpha$NXN
BI	$w_0, t_{-1,1}, t_{0,j}$	w_0 =book, $t_{-1,1} = \beta$Nn, $t_{0,j} = \alpha$NXN
TRI	$w_0, t_{-2,1}, t_{-1,1}, t_{0,j}$	w_0 =book, $t_{-2,1} = \beta$Dnx, $t_{-1,1} = \beta$Nn, $t_{0,j} = \alpha$NXN
FORWARD-BI	$w_0, t_{0,j}, t_{1,1}$	w_0 =book, $t_{0,j} = \alpha$NXN, $t_{1,1} = \alpha$nx0V
FORWARD-TRI	$w_0, t_{0,j}, t_{1,1}, t_{2,1}$	w_0 =book, $t_{0,j} = \alpha$NXN, $t_{1,1} = \alpha$nx0V, $t_{2,1} = \beta$vxN
LEFT-FAR-BI$_x$ ($3 \leq x \leq 8$)	$t_{-x,1}, t_{0,j}$	$t_{-x,1} = \beta$Dnx, $t_{0,j} = \alpha$NXN
RIGHT-FAR-BI$_x$ ($3 \leq x \leq 8$)	$t_{0,j}, t_{x,1}$	$t_{0,j} = \alpha$NXN, $t_{x,1} = \beta$nxPnx
LEFT-WIN$_x$ ($x \in \{ 4, 8, 16 \}$)	$t_{-y,1}, t_{0,j}$	$t_{-y} = \alpha$nx0Vnx1, $0 < y \leq x$, $t_{0,j} = \alpha$NXN
RIGHT-WIN$_x$ ($x \in \{ 4, 8, 16 \}$)	$t_{0,j}, t_{y,1}$	$t_{0,j} = \alpha$NXN, $t_{y,1} = \beta$nxPnx, $0 < y \leq x$

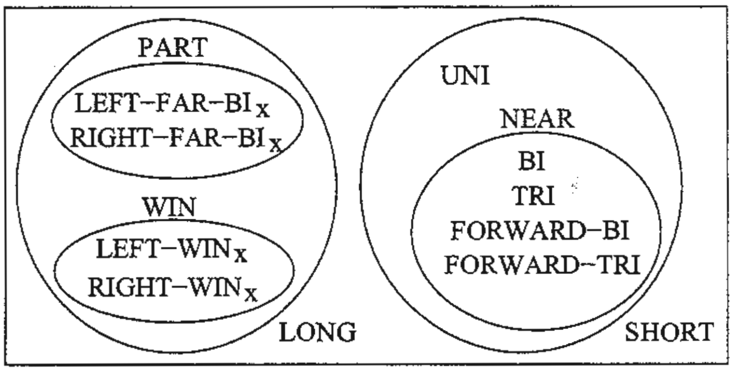


Figure 2: Sets of Features That Are Used In Various Experiments

templates exist that take into account local context and others that take into account long distance context. Local feature templates basically take into consideration the same context that a trigram model considers. They are UNI, BI, TRI, FORWARD-BI, and FORWARD-TRI. Long distance feature templates take into consideration extra-trigram context. There are two kinds of long distance feature templates: *-FAR-BI$_x$ and *-WIN$_x$. The *-FAR-BI$_x$ kind states that the current word has the supertag $t_{0,j}$ and there exists a supertag a *fixed* distance x away from the current word having supertag $t_{x,1}$. The *-WIN$_x$ kind of feature template states that the current word has the supertag $t_{0,j}$ and there exists a supertag $t_{y,1}$ which lies *within some distance* y, $0 < y \leq x$, of the current word.

The list of feature templates in Table 2 is somewhat long and unwieldy. In order to simplify our exposition of different reranking models, we have given names to various subsets of these feature templates. These are shown in Figure 2. The set of all *-FAR-BI$_x$ feature templates is called PART. The set of all *-WIN$_x$ feature templates is called WIN. PART$\cup$WIN yields LONG. SHORT is the set of all trigram-context feature templates. NEAR is SHORT – UNI.

Training RankBoost for reranking supertags requires n-best supertagged data. This is obtained by first extracting a TAG from the Penn Treebank as described in Section 2.2. 8-best supertaggers are then used to derive training, tuning, and test data. Ten-fold cross validation of Sections 02-11 and part of 12 provides the training data (475197 words). 8-best supertagged versions of the rest of Section 12 and Sections 13-14 serve as tuning data (94975 words). Test data is derived from the output of an 8-best supertagger trained on Sections 02-14 on Section 22 (40117 words). Note that for these experiments, a truncated version of the usual Penn Treebank training data–Sections 02-21, are used. This is done merely to expedite the training and testing of different reranking models.

Table 3 shows the supertagging accuracy results for the n-best supertagger, before and after reranking by

Table 3: N-best supertagger results and Reranker results using different feature sets on Section 22.

	% Supertag Accuracy						
n-best	Before Rerank	SHORT	LONG	LONG ∪ SHORT	LONG ∪ UNI	WIN ∪ UNI	PART ∪ UNI
1	80.20	80.77	80.13	81.73	81.39	81.63	81.04
2	87.13	87.67	87.13	88.59	88.38	88.55	88.09
3	89.24	89.73	89.24	90.24	90.16	90.25	89.88
4	90.28	90.63	90.28	90.95	90.88	90.98	90.77
5	90.84	91.07	90.83	91.33	91.27	91.33	91.19
6	91.22	91.38	91.20	91.54	91.50	91.54	91.44
7	91.52	91.57	91.52	91.66	91.64	91.65	91.62
8	91.73	91.73	91.73	91.73	91.73	91.73	91.73

RankBoost. The n-best results for $1 \leq n < 8$ are derived by considering only the top n supertags proposed by the 8-best supertagger. The left half of the table shows three different models are trained using RankBoost, one that uses SHORT features only, one that uses LONG features only, and another that uses both LONG and SHORT features. The rules that are learned by RankBoost are then applied to the 8-best supertags to rerank them.

The results are encouraging. The 1-best supertagger achieves an accuracy of only 80.20%. Nevertheless, the 8-best accuracy is 91.73% which shows that an optimal reranking procedure would halve the error rate. Reranking using SHORT features results in a statistically significant error reduction (p <0.05) of 2.9% for 1-best. Reranking also using LONG features results in an error reduction of 7.7% for 1-best (and an error reduction of 13.3% with respect to the RankBoost topline of 91.73%). Therefore RankBoost is obviously able to use LONG features effectively in conjunction with the SHORT features, despite a big increase in the number of parameters of the model. Note also that reranking improves the accuracy for all n-best results, $1 \leq n < 8$.

Apparently, there is some interaction between LONG and SHORT features which makes model LONG∪SHORT effective whereas model LONG is useless. In order to study this interaction, and also to determine what kinds of LONG features help the most, we have tested models LONG∪UNI, WIN∪UNI, and PART∪UNI. The results are shown in the right half of Table 3. Model LONG∪UNI achieves much of the performance of model LONG∪SHORT, even though it only considers the unigram feature. One possible explanation for this phenomenon is that SHORT features aid LONG features not because the local trigram context that is modeled by SHORT is so much more important, but instead it is lexicalization that is important, SHORT features being lexicalized whereas LONG features are not. Also note that model WIN∪UNI outperforms model PART∪UNI. This seems to indicate that PART feature templates are less useful in supertag disambiguation than WIN feature templates.

5. Analysis of Some Initial Experiments

At first glance, there does not seem to be much of a difference between model LONG∪SHORT and model SHORT. The difference between them in terms of accuracy of 1-best supertagging reranking is slightly less than one percent, about five percent in terms of reduction in error. On the other hand, as Table 6 shows, this small difference is still statistically significant. In order to get a better grasp on the differences in behavior of model LONG∪SHORT and model SHORT, and also to get a feeling about how one might improve reranking models for supertagging, we perform a semi-qualitative analysis of the 1-best reranked output of these two models.

The ten most frequently mistagged supertags (i.e. those supertags that were most misclassified by the reranker), sorted by frequency, for model SHORT and model LONG∪SHORT are shown in Table 4. At first glance, there is not much difference between the two models; they both mistag mostly the same kinds of supertags, and the supertags' rankings are about the same. However, certain differences can be discerned. Notably, the frequency of mistagging αNXN is 25% less in LONG∪SHORT than it is in SHORT. Also, there is somewhat less of a PP attachment problem in LONG∪SHORT than there is in SHORT, as can be seen by the frequencies of the PP attachment supertags βnxPnx and βvxPnx. The fact that the frequency of mistaggings of αnx0Vnx1 drops from 168 in SHORT to 130 in LONG∪SHORT is also noteworthy; apparently LONG∪SHORT is performing better at resolving NP versus S subcat ambiguity.

For each of several supertags in Table 4, we proceed to determine the most important features that

Table 4: Ten Most Frequently Mistagged Supertags, By Frequency, for SHORT and LONG∪SHORT

SHORT				LONG∪SHORT			
Frequency	Supertag	Frequency	Supertag	Frequency	Supertag	Frequency	Supertag
650	αNXN	167	βnxN	474	αNXN	155	βVnx
410	βNnx	162	βnxPunct	356	βNnx	151	βnxPnx
303	βvxPnx	148	βnxPnx	289	βvxPnx	147	αN
216	βAnx	130	βucpPunct	203	βAnx	144	βnxPunct
168	αnx0Vnx1	117	αN	166	βnxN	130	αnx0Vnx1

LONG∪SHORT uses in order to choose the correct supertag. Our methodology is as follows. Given a supertag γ, we determine the set of instances in the test corpus where LONG∪SHORT reranked γ to first place from an originally lower ranking. For each instance, we determine the features that caused LONG∪SHORT to rank γ more highly, tabulating the number of times each feature is used. We also record the multiset $\phi(\gamma)$ of supertags $\gamma' \neq \gamma$ such that LONG∪SHORT replaced γ' with γ as the first ranked supertag.

Consider supertag αnx0Vnx1. Most frequently occurring members of ϕ(αnx0Vnx1) include βVvx, βnx0Vs1, αINnx0Vnx1 (declarative transitive supertag with complementizer), and βvxINnx0Vnx1. The most frequently used features that are used to rank αnx0Vnx1 more highly are LEFT-WIN$_{16}$(EOS,αnx0Vnx1) and LEFT-WIN$_8$(EOS,αnx0Vnx1), where EOS is a sentence delimiter, in this case the left sentence delimiter. Intuitively, these features seem to suggest that αnx0Vnx1 should appear nearer to the beginning of the sentence than for example, βvxINnx0Vnx1, being a verbal postmodifier, should. Another frequently used feature is LEFT-WIN$_4$(αNXN,αnx0Vnx1). It is apparently used to make sure there exists an NP to the left of the current word that would fit in the subject slot of αnx0Vnx1. The existence of the frequently used feature LEFT-WIN$_{16}$(βMDvx,αnx0Vnx1) is also of interest. Apparently, this feature occurs because αnx0Vnx1 often serves as the sentential complement of another verb to its left. This verb can take a variety of supertags, including βnx0Vs1 and βN0nx0Vs1 for example. Having a separate feature for each of these supertags would possibly lead to suboptimal reranking performance because of sparse data. Instead, apparently based on the generalization that these supertags are usually modified by a modal verb βMDvx, RankBoost chooses the feature LEFT-WIN$_{16}$(βMDvx,αnx0Vnx1).

All of the features that we have discussed are LONG. In fact, there is a preponderance of LONG features used to rank αnx0Vnx1: the ten most frequent features are LONG. There are however, some SHORT features that are heavily weighted, although they are not used quite as often. One notable SHORT feature is FORWARD-BI(has,βVvx,βDnx). Intuitively, it resolves the ambiguity between βVvx and αnx0Vnx1 by seeing whether an NP (prefixed by a determiner) immediately follows the current word.

Supertag αNXN presents another interesting case. The most frequently occurring members of ϕ(αNXN) include αnx0N, βnxN, and βvxN. The most frequently used features that are used to prefer αNXN include LEFT-WIN$_{16}$(αNXN,αNXN), RIGHT-WIN$_{16}$(αNXN,βsPeriod), RIGHT-WIN$_{16}$(αNXN,βnxPnx), LEFT-WIN$_4$(βNn,αNXN). These features seem to encode the context that is likely to surround αNXN. Of course, these features also seem likely to surround other members of ϕ(αNXN). Perhaps these features are chosen because of a general bias that the n-best supertagger has against supertagging head nouns appropriately.

6. Further, Exploratory Experiments

Based on our experience with reranking of n-best supertags, we have drawn some possible avenues for improvement of the reranking procedure. In the following, we list some common reasons for lack of optimum reranking performance and discuss how they might be eliminated.

- The feature that would perform the appropriate reranking is not chosen because of sparse data. Note that the supertags that do instantiate feature templates tend to be very common. It is not surprising, therefore, that there exists a feature such as LEFT-WIN$_4$(αNXN,αnx0Vnx1). Recall that this appears to ensure that αnx0Vnx1 has an NP subject to the left. An analogous feature is not likely to appear for an infrequently occurring supertag, such as βN0nx0Vs1. One possible solution would be to instantiate feature templates with certain aspects of supertags instead of entire supertags. Along this line, we perform some exploratory experiments in Section 6.1.

- The correct supertag for word w_0 does not exist in the n-best supertagged output. One way to ameliorate this problem is to improve the performance of the first stage n-best trigram supertagger. Along this line, we perform some exploratory experiments in Section 6.2.
- For words other than the current word, the feature template is instantiated only from the 1-best supertag output, which is not always correct. For example, the feature LEFT-WIN$_4$(αNXN,αnx0Vnx1), depends on the fact that the supertags to the immediate left of the current word are, in fact, correctly supertagged, whereas they are only correctly supertagged about 80% of the time. Now, the training process should compensate for this somewhat because the inputs to the training process are (flawed) supertagged sentences. On the other hand, perhaps a different approach would be more effective in tackling this problem. One avenue would be to rerank n-best *paths* of supertags, instead of n-best per word supertagged output. Along this line, we have implemented an n-best paths supertagger, based on a trigram model, but employing a search strategy similar to (Ratnaparkhi, 1996). Trained on Sections 02-21 of the Penn Treebank, this supertagger achieves about 89% supertag accuracy only when the top 100 paths are chosen. It remains to be seen whether this will cause difficulties in terms of memory or space resources for training the reranker.

6.1. Training the Reranker with Part of Speech Features

Having features consider entire supertags is a limiting factor in contributing to the performance of the reranker not in the least because of sparse data. One possible solution is to base features on aspects of supertags instead of entire supertags. For example, one might take the approach of breaking down each supertag into a feature vector (Srinivas, 1997; Xia *et al.*, 1998; Xia *et al.*, 2000; Chen, 2001), and to base RankBoost features on elements of that vector. Another approach would be to consider each supertag as generated by a Markov process (Collins, 1997; Collins, 1999). In this case, one would base RankBoost features on individual steps in that process. Here, we consider using part of speech as a component in feature space.

As implied by Section 2.2, the preterminal tag set for our extracted grammar is similar to the XTAG part of speech tag set. For our features, we can either choose to retain the XTAG parts of speech or use the more detailed Penn Treebank part of speech tagset. This choice displays the usual tradeoff between assuaging sparse data (the former) and having detailed enough features to make appropriate decisions (the latter). We have chosen the latter because the Penn Treebank part of speech tagset (about 45 tags) is already an order of magnitude smaller than the supertag tagset (about 3900 tags), although we believe that it would also be interesting to repeat our experiments using the XTAG part of speech tagset (about 20 tags).

For each feature template in LONG∪SHORT, an analogous feature template is created with supertag parameters other than the current word replaced with part of speech parameters. For example, LEFT-WIN$_4$-POS(p_y,$t_{0,cur}$) is a feature template that states that the current word is supertagged with $t_{0,cur}$ and there exists a word to the left that has part of speech p_y within a distance of four words of the current word. Furthermore, we give the same name to these new subsets of feature templates as is given to the previous subsets, affixed with -POS. For example, WIN-POS is the set of feature templates consisting of LEFT-WIN$_x$-POS and RIGHT-WIN$_x$-POS.

After the RankBoost training, tuning, and test corpora were suitably annotated using a trigram model Penn Treebank part of speech tagger, models NEAR-POS∪SHORT and LONG-POS∪LONG∪SHORT were trained and tested. The results are shown in the left half of Table 5. Although the 1-best reranking accuracies are not significantly higher for the *-POS models than for the corresponding non-POS models (Table 6), it is important to keep in mind that these are preliminary results. We believe that the higher accuracies for the *-POS models indicate that there may exist other, untried models which use part of speech information more effectively.

6.2. Reranking of Smoothed N-Best Supertagging

There are many cases where the reranker cannot give the correct supertag the top ranking because it does not exist in the n-best output. One possible solution to this problem is to enhance the n-best supertagger by smoothing its emit probability $p(w|t)$, and then run the reranker on the resulting output. Here, we perform such an experiment.

Our experiment proceeds as follows. We choose to smooth $p(w|t)$ using the approach mentioned in (Chen, 2001). It accounts especially for the fairly large set of cases (about 5%) in which the word and the correct supertag have both been seen in the training data, but not in combination. These cases would normally be assigned a probability of zero by the supertagging model. Using this approach, we prepared training, tuning, and test data using the smoothed version of the n-best supertagger as appropriate. We subsequently trained model LONG∪SHORT on this training and tuning data, and then tested the reranker as usual.

Table 5: N-best supertagger results, smoothing, and smoothing plus LONG ∪ SHORT reranker results

	% Supertag Accuracy				
	Before	NEAR-POS	LONG-POS ∪	Smoothed	Smoothed and
n-best	Rerank	∪ SHORT	LONG∪SHORT		LONG∪SHORT
1	80.20	80.97	82.04	81.64	82.99
2	87.13	87.77	88.83	89.02	90.42
3	89.24	89.77	90.38	91.24	92.31
4	90.28	90.63	91.04	92.37	93.14
5	90.84	91.07	91.34	93.07	93.59
6	91.22	91.37	91.53	93.54	93.88
7	91.52	91.57	91.65	93.84	94.05
8	91.73	91.73	91.73	94.14	94.14

Table 6: Differences in 1-best supertagging accuracy for all pairs of reranking models. Significant differences ($p < 0.05$) are marked with "*"

	Before Rerank	S	L	L ∪ S	L ∪ U	W ∪ U	P ∪ U	SM	SM & L ∪ S	NPOS ∪ S
S	+0.57									
L	-0.07	-0.64								
L ∪ S	+1.53*	+0.96*	+1.60*							
L ∪ U	+1.19*	+0.62	+1.26*	-0.34						
W ∪ U	+1.43*	+0.86	+1.50*	-0.10*	+0.24					
P ∪ U	+0.84	+0.27	+0.91*	-0.69	-0.35	-0.59				
SM	+1.44*	+0.87	+1.51*	-0.09	+0.25	+0.01	+0.60			
SM & L ∪ S	+2.79*	+2.22*	+2.86*	+1.26*	+1.60*	+1.36*	+1.95*	+1.35*		
NPOS ∪ S	+0.77	+0.20	+0.84	-0.76	-0.42	-0.66	-0.07	-0.67	-2.02*	
LPOS ∪ L ∪ S	+1.84*	+1.27*	+1.91*	+0.31	+0.65	+0.41	+1.00*	+0.40	-0.95*	+1.07*

The smoothing technique was successful in raising the 8-best supertagging accuracy to 94.14% from 91.73%. And, as can be seen in Table 5 RankBoost can still improve on the output, though to a slightly lesser extent. Overall, the error reduction increases to 14.1% over the unsmoothed, non-reranked 1-best supertags (of which RankBoost contributes 6.9% absolute). As far as we know, these are the currently best results for supertagging using large supertag sets.

7. Significance Testing

We performed a one-way analysis of variance on the 1-best supertagging results of all of the reranking models that are mentioned in this paper. Table 6 tabulates the differences between 1-best supertagging accuracies of the various models and marks significant differences, $p < 0.05$, with "*." The F-value is 18.11; the critical value for the Tukey test is 0.89.

8. Conclusions and Future Work

This paper has explored the use of RankBoost in order to rerank an n-gram supertagger. We have seen that such a reranking, performed effectively, is potentially useful in a variety of applications, including speeding up a parser. We have performed experiments that show that RankBoost can indeed produce models that perform reranking well, to a statistically significant degree. We have identified specific features that explain why the reranker performs effectively. We have also identified causes that limit the reranker's performance. Finally, we have performed other, exploratory experiments that ameliorate these limitations.

An advantage of using RankBoost is that numerous candidate features can be added robustly because RankBoost learns to choose only the relevant ones. This invites the possibility of investigating kinds of features for reranking other than the ones mentioned in this paper. Bilexical features may be useful, along with features that take into account tree families, different kinds of parts of speech, punctuation, or the results of chunkers or even parsers. It is also important to keep in mind that the performance of the reranker is limited by the performance of the n-best supertagger. Thus, novel means to increase the n-best supertagger's accuracy should also be explored. We also intend to investigate other ways of obtaining candidate supertag sets using other notions of class-based supertagging presented in (Chen, Bangalore and Vijay-Shanker, 1999).

References

Bangalore, Srinivas, John Chen and Owen Rambow. 2001. Impact of Quality and Quantity of Corpora on Stochastic Generation. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Langauge Processing*, Pittsburgh, PA.

Bangalore, Srinivas and A. K. Joshi. 1999. Supertagging: An Approach to Almost Parsing. *Computational Linguistics*, 25(2).

Chandrasekhar, R. and B. Srinivas. 1997a. Automatic Induction of Rules for Text Simplification. *Knowledge-Based Systems*, 10:183–190.

Chandrasekhar, R. and B. Srinivas. 1997b. Using Supertags in Document Filtering: The Effect of Increased Context on Information Retrieval. In *Proceedings of Recent Advances in NLP '97.*

Chen, John. 2001. *Towards Efficient Statistical Parsing Using Lexicalized Grammatical Information.* Ph.D. thesis, University of Delaware.

Chen, John, Srinivas Bangalore and K. Vijay-Shanker. 1999. New Models for Improving Supertag Disambiguation. In *Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics*, Bergen, Norway.

Chen, John and K. Vijay-Shanker. 2000. Automated Extraction of TAGs from the Penn Treebank. In *Proceedings of the Sixth International Workshop on Parsing Technologies*, pages 65–76.

Chiang, David. 2000. Statistical Parsing with an Automatically-Extracted Tree Adjoining Grammar. In *Proceedings of the the 38th Annual Meeting of the Association for Computational Linguistics*, pages 456–463, Hong Kong.

Collins, Michael. 1997. Three Generative Lexicalized Models for Statistical Parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics.*

Collins, Michael. 1999. *Head-Driven Statistical Models for Natural Language Parsing.* Ph.D. thesis, University of Pennsylvania.

Collins, Michael. 2000. Discriminative Reranking for Natural Language Parsing. In *Proceedings of the 17th International Conference on Machine Learning.*

Freund, Yoav, Raj Iyer, Robert E. Schapire and Yoram Singer. 1998. An Efficient Boosting Algorithm for Combining Preferences. In *Machine Learning: Proceedings of the Fifteenth International Conferece.*

Marcus, Mitchell, Beatrice Santorini and Mary Ann Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Ratnaparkhi, Adwait. 1996. A Maximum Entropy Model for Part-of-Speech Tagging. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 133–142, Somerset, NJ.

Sarkar, Anoop. 2000. Practical Experiments in Parsing using Tree Adjoining Grammars. In *Proceedings of the Fifth International Workshop on Tree Adjoining Grammars and Related Frameworks*, Paris, France.

Sarkar, Anoop. 2001. Applying Co-Training Methods to Statistical Parsing. In *Proceedings of Second Annual Meeting of the North American Chapter of the Association for Computational Linguistics*, Pittsburgh, PA.

Schabes, Yves, Anne Abeillé and Aravind K. Joshi. 1988. Parsing Strategies with 'Lexicalized' Grammars: Application to Tree Adjoining Grammars. In *Proceedings of the 12th International Conference on Computational Linguistics*, Budapest, Hungary.

Schapire, Robert E. 1999. A Brief Introduction to Boosting. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence.*

Srinivas, B. 1997. Performance Evaluation of Supertagging for Partial Parsing. In *Proceedings of the Fifth International Workshop on Parsing Technologies*, pages 187–198, Cambridge, MA.

van Halteren, H. 1996. Comparison of Tagging Strategies: A Prelude to Democratic Tagging. In *Research in Humanities Computing 4.* Clarendon Press, Oxford, England.

Walker, Marilyn A., Owen Rambow and Monica Rogati. 2001. SPoT: A Trainable Sentence Planner. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 17–24.

Xia, Fei. 1999. Extracting Tree Adjoining Grammars from Bracketed Corpora. In *Fifth Natural Language Processing Pacific Rim Symposium (NLPRS-99)*, Beijing, China.

Xia, Fei, Chung hye Han, Martha Palmer and Aravind Joshi. 2000. Comparing Lexicalized Treebank Grammars Extracted from Chinese, Korean, and English Corpora. In *Proceedings of the Second Chinese Language Processing Workshop (CLP-2000)*, Hong Kong, China.

Xia, Fei, Martha Palmer, K. Vijay-Shanker and Joseph Rosenzweig. 1998. Consistent Grammar Development Using Partial-Tree Descriptions for Lexicalized Tree Adjoining Grammars. In *Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks*, pages 180–183.

XTAG-Group, The. 2001. A Lexicalized Tree Adjoining Grammar for English. Technical report, University of Pennsylvania. Updated version available at http://www.cis.upenn.edu/~xtag.

Hidden Markov model–based Supertagging in a user–initiative dialogue system

Jens Bäcker and Karin Harbusch
University of Koblenz–Landau, Computer Science Department
Universitätsstr. 1, D–56070 Koblenz, Germany
E–mail: {jbaecker|harbusch}@informatik.uni-koblenz.de

Abstract

In this paper we outline the advantages of deploying a shallow parser based on Supertagging in an automatic dialogue system in a call center that basically leaves the initiative with the user as far as (s)he wants (in the literature called *user–initiative* or *adaptive* in contrast to *system–initiative* dialogue systems). The Supertagger relies on a *Hidden Markov model* and is trained with German input texts. The entire design of a Hidden Markov–based Supertagger with trigrams builds the central issue of this paper. The evaluation of our German Supertagger lags behind the English one. Some of the reasons will be addressed later on. Nevertheless shallow parsing with the Supertags increases the accuracy compared to a basic version of KoHDaS that only relies on recurrent plausibility networks.

1. Introduction

Wizard–of–Oz experiments show that users of automatic dialogue systems would preferentially take the initiative in many dialogues instead of being asked a long list of tiny little questions by the system (cf. (Boje *et al.*, 1999)). Empirical evaluations demonstrate that adaptation to the user's dialogue preference leads to significantly higher user satisfaction and task success (cf. (Strachan *et al.*, 1997) or (Litman, Pan and Walker, 1998)). In contrast to these results, it can also be observed that in such *user–initiated dialogue systems* the user is sometimes left without a clear understanding of his/her options at a given point in the dialogue. This can cause frustration or even breakdowns of the communication. Consequently, an *adaptive system* which reacts to the user's preferred mode, i.e. is able to ask explicit questions when the user doesn't take the initiative and to react to user–provided complex turns adequately as well at any particular state of the dialogue, serves as a user–friendly dialogue system.

The criticised strict dialogue structure with an explicit and inevitable initiative by the system (henceforth called *system–initiative* in contrast to *user–initiative*) results from the crucial fact that with any of these questions by the system a particular sub–grammar and sub–lexicon of the speech analysis system (e.g. a simple number or yes/no grammar and lexicon, respectively) can be associated to analyse the user's answer more reliably. Clarification dialogues caused by incorrectly analysed words can be circumvented by this method. Hence it is essential for a user–initiative or adaptive (or also called *mixed-initiative*) system to remedy the shortcomings resulting from the less reliable analysis of the user's spoken turn with a general grammar and lexicon, respectively. Furthermore, the *task parameters*, i.e. the information provided in the user's turn to perform the user–intended task by the system, have to be extracted without knowing exactly where in the user's turn or whether at all they have been uttered yet. In the case that not all task parameters are provided even a user–initiative system has to ask questions — similar to a system–initiative system.

KoHDaS–NN[1] is an *automatic help desk system in a call center* that basically leaves the dialogue initiative with the user as far as (s)he wants. The user's turn circumscribing the problem as a whole is handed to a hierarchy of recurrent plausibility networks which classify the according problem. In the next step the system extracts even only implicitly mentioned task parameters of this problem class from the turn by a graph-matching technique. Remaining or unidentified task parameters required to solve the problem are asked by the system in an ordinary question–answering manner. The results of KoHDaS–NN where the classification and the extraction of the task parameters is performed only on the basis of simple words are promising. However the number of wrong classifications and questions for yet uttered task parameters has to be further decreased in order to provide a user–friendly dialogue with the customers. Hence we investigate in the following whether deploying a shallow parser based on Supertagging increases the performance of the system — both with respect to the classification and the extraction of the task parameters.

The Supertagger in KoHDaS–ST relies on a *Hidden Markov model* and is trained with German input texts. The main section of this paper is devoted to the description of this method (cf. Section 3; it also comprises some

1. The acronym KoHDaS stands for **Ko**blenzer **H**elp **D**esk with automatic **S**peech recognition. In the following the basic version is called KoHDaS–NN (NN stands for Neural Networks). Later on we investigate KoHDaS–ST where ST stands for SuperTagging.

implementation details to gain efficiency). With respect to accuracy our German Supertagger lags behind the English one. Some of the reasons will be addressed later on. Nevertheless shallow parsing with the Supertags increases the accuracy compared to a basic version of KoHDaS that only relies on recurrent plausibility networks.

The paper is organized as follows. In the next Section KoHDaS and its functionality is described. In Section 3 the Supertagger based on Hidden Markov models is outlined. Section 4 is devoted to the description of KoHDaS–ST, i.e. how Supertagging used in shallow parsing is integrated into the application of KoHDaS. Furthermore, it depicts the results of KoHDaS–ST compared to KoHDaS–NN. In Section 5 related work is portrayed. Here further methods that favorably compare to Supertagging are outlined on the one hand. On the other, different Supertagging methods and their application domains are delineated. The paper ends addressing future work and open problems.

2. KoHDaS — an adaptive dialogue system for First–Level Support via telephon

KoHDaS (see, e.g., (Harbusch, Knapp and Laumann, 2001)) explores methods which provide automatic user–initiative dialogues in call centers, i.e. the initiative should basically be left with the user as far as (s)he wants. Compared to system–initiative dialogue systems, user–initiative systems cannot rely on restricted dictionaries and grammars during the speech recognition process to gain more reliable results. Hence, methods to remedy the reduction of understanding in the first phase have to be impinged on the system.

KoHDaS–NN deploys the following two techniques towards a natural dialogue behaviour. First, a hierarchy of neural networks classifies the user's whole turn (on average 25 words) according to a list of problem classes. After a confirmation dialogue the *task parameters* mentioned in the user's turn are extracted to avoid redundant questions by the system for information to perform the task the user wants the system to perform — in our case a data base look–up for a solution of the user's hardware problem.

As for classification, a *hierarchy of recurrent plausibility networks* (cf. (Wermter, 1995)) accomplishes the consideration of arbitrary previous contexts in current decision making in KoHDaS. After a confirmation that the problem class is correctly recognized, the task parameters, i.e. the necessary information to enable the system to perform the task the user intended the system to do, are extracted from the user's initial turn by a graph matching technique imposed on the dialogue graphs, i.e. the specification of all possibly asked questions by the system and the according task parameters provided by the user (note that these graphs have to be specified anyhow to model a user who does not take the initiative at all).

KoHDaS is currently customized to the domain of first level support for computer hardware but it can easily be adapted to new domains.

A drawback of KoHDaS–NN is the absence of syntactic information in the processing of the user's turn as KoHDaS–NN works simply word–based. With respect to the classification task, a word such as *"monitor"* remains active by the context–layer independent whether the word in mentioned in a subordinate clause or as a key concept to characterize the problem (cf. *"My new monitor flickers since I have ..."* vs. *"My SCSI hard drive which I bought together with my new monitor two weeks ago from your customer's service cannot be formatted ..."*). With respect to the extraction of task parameters negations are completely ignored in KoHDaS–NN. For instance, *"My monitor flickers although the boxes are not activated"* takes for granted that the monitor only occasionally flickers when the boxes are activated. Hence the wrong conclusion is drawn from the customer's utterance. In order to remedy this shortcoming, KoHDaS–ST deploys a robust syntactic analysis based on a Supertagger (Joshi and Srinivas, 1994) and a Lightweight Dependency Analyzer (LDA) (Srinivas, 1997a) in the analysis of the user's turns. Particularly for the correct interpretation of the scope of negations we have decided to use Supertagging instead of chunking (cf. Section 5). The structural information — which possibly remains partial — provided by this analysis is used in the classification step as well as in the information extraction step of KoHDaS. The use of structural information in classification can prevent words occurring in deeply nested sub–sentences from having high impact on the determination of the problem class. In the information extraction step, the use of structural information allows to detect dependencies between structures in the turn, that can't be detected in the word–based version of KoHDaS–NN (e.g., negations; cf. Section 4).

3. Hidden Markov model–based Supertagging

Our German Supertagger uses a *trigram model* and is based on *Hidden Markov models (HMMs)* enabling the use of the well known algorithms on HMMs (see, e.g., (Rabiner, 1989)) to guess the most likely syntactic structure of a sentence. We use a trigram model as it has shown to achieve good results in Supertagging (cf. (Srinivas, 1997a)).

In this section the basic concepts of Hidden-Markov models are briefly introduced. Thereafter a Hidden Markov model–based Supertagger is portrayed. Finally, aspects of the implementation —particularly with respect to time and space efficiency — are highlighted.

3.1. Basic concepts of Hidden Markov models

Let us assume the following notational conventions (adapted from (Rabiner and Juang, 1986) and (Charniak, 1993) or see, e.g., (Rabiner, 1989) for a good introduction):

- T = length of the sequence of observations (training set),
- N = number of states (either known or guessed),
- M = number of possible observations (from the training set),
- $\Omega_X = \{q_1, ...q_N\}$ (finite set of possible states),
- $\Omega_O = \{v_1, ..., v_M\}$ (finite set of possible observations),
- X_t random variable denoting the state at time t (state variable),
- O_t random variable denoting the observation at time t (output variable),
- $\sigma = o_1, ..., o_T$ (sequence of actual observations)

and distributional parameters:

- $A = \{a_{ij}\}$ with $a_{ij} = Pr(X_{t+1} = q_j | X_t = q_i)$ (transition probabilities),
- $B = \{b_i\}$ with $b_i(k) = Pr(O_t = v_k | X_t = q_i t)$ (observation probabilities),
- $\pi = \{\pi_i\}$ with $\pi_i = Pr(X_0 = q_i)$ (initial state distribution).

A *Hidden Markov model (HMM)* is a five-tuple $(\Omega_X, \Omega_O, A, B, \pi)$. Let $\lambda = \{A, B, \pi\}$ denote the parameters for a given HMM with fixed Ω_X and Ω_O. This means, a discrete–time, discrete–space dynamical system governed by a Markov chain emits a sequence of observable outputs: one output (observation) for each state in a trajectory of such states. From the observable sequence of outputs, the most likely dynamical system can be inferred. The result is a model for the underlying process. Alternatively, given a sequence of outputs, the most likely sequence of states can be inferred. The model can also be used to predict the next observation or more generally a continuation of the sequence of observations. Three basic problems can be formulated for HMMs:

1. Find $Pr(\sigma|\lambda)$, i.e. the probability of the observations given the model.
2. Find the most likely state trajectory given the model and observations.
3. Adjust $\lambda = \{A, B, \pi\}$ to maximize $Pr(\sigma|\lambda)$.

For any of these questions efficient algorithms are known (see, e.g., (Rabiner, 1989)). The *Forward–Backward algorithm* (Baum and Eagon, 1967) solves the first problem, problem 2 is yielded by the *Viterbi algorithm* (Viterbi, 1967) und problem 3 can be dealt with by the *Baum–Welch algorithm* (Baum, 1972).

3.2. Hidden Markov models for Supertagging

Many variants of Supertagging use models similar to POS–Tagging (cf. Section 5 for a brief description of the variants Trigram Supertagging, Head Supertagging and Transformation–based Supertagging). Here, we underlay the Supertagger with *Hidden Markov models (HMMs)*.

In this framework, the Supertags are encoded as states and the words as symbols of the output alphabet of the HMM. Assuming a bigram model (i.e. *n–Gram* with $n = 2$), the realization is easy and straightforward. Any Supertag becomes an individual state and any terminal an individual output symbol. The according Tagger can be trained with the Baum–Welch algorithm. The observation sequence is provided by the sentences of the training set (unsupervised learning). However, this method lacks behind supervised learning methods (see, e.g., (Merialdo, 1994)). Such a corpus can be gained with the *Viterbi algorithm* (cf. problem 2 mentioned above), as this algorithm denotes the optimal sequence of states for a given observation sequence — in our case the optimal sequence of Supertags.

The Supertagger we report on in this paper uses a trigram model (cf. (Bäcker, 2001) for an evaluation of the HMM–based Supertagger using bigrams). According to the trigram model, two previous states (i.e. Supertags) are encoded in the HMM in a well–known manner (see, e.g., (El-Beze and Merialdo, 1999)):

- The states of the HMM correspond to pairs of Supertags (t_{i-1}, t_i).
- The transition probability $Pr[(t_{i-1}, t_i)|(t_{i-2}, t_{i-1})]$ is denoted by the trigram probability $Pr(t_i|t_{i-2}t_{i-1})$.
- The output symbols are provided by the words which are tagged with t_i and which are emitted in states $(\_, t_i)$.

At the beginning of a sentence pseudo states $(\emptyset, t_j)$ with $\emptyset$ a pseudo category are assumed.

In general, the Baum–Welch algorithm (cf. problem 3) can be applied to optimize the model parameters in order to maximize $Pr(\text{training set}|\lambda)$. Our results are gained on the basis of a labeled corpus. Hence we don't impose the Baum–Welch algorithm on our Supertagger. On the basis of the labeled corpus we directly estimate the model parameters according to the *Maximum Likelihood Estimation (MLE)* method. For trigrams this means:

$$Pr_{MLE}(t_i|t_k, t_j) = \frac{c(t_k, t_j, t_i)}{c(t_k, t_j)}, \quad 1 \leq i \leq N, \quad 1 \leq j \leq N, \tag{1}$$

$$Pr_{MLE}(w_k|t_j) = \frac{c(w_k, t_j)}{c(t_j)}, \quad 1 \leq j \leq N, \quad 1 \leq k \leq M \tag{2}$$

with:

$c(t_j)$	$\widehat{=}$	number of occurrences of t_j in the training set,
$c(t_j, t_k)$	$\widehat{=}$	number of occurrences of t_j followed by t_k,
$c(t_k, t_j, t_i)$	$\widehat{=}$	number of occurrences of t_k followed by t_j, which itself is followed by t_i, and
$c(w_k, t_j)$	$\widehat{=}$	number of occurences of w_k labelled as t_j.

In order to overcome problems with *sparse data*, i.e. not all trigrams occur in the training set, *smoothing techniques* (for a good introduction see, e.g., (Jurafsky and Martin, 2000) or (Manning and Schütze, 2000); in (Chen and Goodman, 1996) the performance of various smoothing techniques are evaluated) are applied in our system. Furthermore the treatment of unknown words is described in the following.

In our system we employ *Good–Turing Discounting* (Good, 1953). This means, the equation (1) of the MLE estimation which relies on the absolute frequency $c(t_k, t_j, t_i)$ of a trigram is replaced by the following equation:

$$Pr_{GT}(t_i|t_k, t_j) = \frac{c^*(t_k, t_j, t_i)}{c(t_k, t_j)}, \quad 1 \leq i \leq N, \quad 1 \leq j \leq N, \tag{3}$$

with c^* the modified number of trigrams. The Good–Turing Discounting computes c^* according to:

$$c^* = c \quad \text{für } c > k \tag{4}$$

and according to (Katz, 1987):

$$c^* = \frac{(c+1)\frac{N_{c+1}}{N_c} - c\frac{(k+1)N_{k+1}}{N_1}}{1 - \frac{(k+1)N_{k+1}}{N_1}} \quad \text{for } 1 \leq c \leq k. \tag{5}$$

Here the constant k denotes a threshold to prevent the system from re–estimating rather accurate results (according to high frequency).

The discounting method can be further improved by a method that differentiates between unseen trigrams. The *Backoff method* (Katz, 1987) uses the frequencies of $(n-1)$–grams in the following manner:

$$Pr_{BO}(t_i|t_{i-2}t_{i-1}) = \begin{cases} \tilde{P}(t_i|t_{i-2}t_{i-1}), & \text{if } c(t_{i-2}t_{i-1}t_i) > 0, \\ \alpha_1 \tilde{P}(t_i|t_{i-1}), & \text{if } c(t_{i-2}t_{i-1}t_i) = 0 \text{ and } c(t_{i-1}t_i) > 0, \\ \alpha_2 \tilde{P}(t_i), & \text{otherwise.} \end{cases} \tag{6}$$

If the frequency of a trigram (bigram, resp.) is zero, the frequency of the bigram (unigram, resp.) is considered. However, a factor α_1 (α_2, resp.) is supposed to normalize the resulting probability. This means, for any given t_n the following holds:

$$\sum_{i,j} Pr(t_n|t_i t_j) = 1. \tag{7}$$

In formula (6) $\tilde{P}$ denotes the probability which results form a discounting method (otherwise the values don't fulfil equation (7)). We use again the Good–Turing discounting method here. The according formula looks as follows:

$$Pr_{BO}(t_i|t_{i-2}t_{i-1}) = \begin{cases} Pr_{GT}(t_i|t_{i-2}t_{i-1}), & \text{if } c(t_{i-2}t_{i-1}t_i) > 0, \\ \alpha(t_{n-2}^{n-1})Pr_{GT}(t_i|t_{i-1}), & \text{if } c(t_{i-2}t_{i-1}t_i) = 0 \text{ and } c(t_{i-1}t_i) > 0, \\ \alpha(t_{n-1})Pr_{GT}(t_i), & \text{otherwise.} \end{cases} \tag{8}$$

Unknown words are treated in our system in the following manner. The probability $Pr(w_k|t_j)$ is computed by the Backoff method. In case w_k is an unknown word we adapt the method by (Weischedel *et al.*, 1993) which deals with *features* of words. The prefixes and suffixes of words are considered to estimate the probabilities according to the following formula:

$$Pr(w_k|t_j) = \begin{cases} Pr_{MLE}(w_k|t_j) & \text{if } c(w_k, t_j) > 0, \\ Pr(unknown|t_j) * Pr(features|t_j) & \text{otherwise.} \end{cases} \tag{9}$$

The probability of the occurrence of an unknown word $Pr(unknown|t_j)$ for the currently considered Supertag is estimated according to:

$$Pr(unknown|t_j) = \frac{N_1(t_j)}{c(t_j)}, \tag{10}$$

where $N_1(t_j)$ is the number of words which occur in the training set exactly once with the Supertag t_j; $Pr(features|t_j)$ denotes the probability whether a word with the same prefix or suffix as w_k, respectively, occurs together with the Supertag t_j.

Now we face the task of tagging itself. The tagging is performed by the Viterbi algorithm. For a given observation sequence $O = \{O_1, O_2, \ldots, O_T\}$ the most likely sequence of states $Q = \{q_1, q_2, \ldots, q_T\}$ is computed in four steps (Initialization, Recursion, Termination and Reconstruction (Path Backtracking); the time complexity of the Viterbi algorithm ist $O(N^2 n)$ where n is the length of the input). For a German test set of 30 sentences, 78.3% of the words were assigned the correct Supertag. In the conclusions we compare this result with English Supertagging.

In general the above described HMM–bases Supertagger was tested with a German corpus of 250 tagged sentences (cf. the evaluation in Section 4). The German training and test corpus has been constructed in the following manner. Basically we looked at written German dialogues in news groups in the area of first level support for computer hardware. We developed an LTAG with 127 elementary trees covering the domain of the KoHDaS system (cf. (Bäcker, 2002)) and automatically parsed these dialogues. The reviewed results of all parses constitute the tagged corpus. We trained our Supertagger using 250 tagged sentences. For the estimation of the HMM's model parameters we used Good–Turing discounting combined with Katz's Backoff model to smooth the parameters resulting in better estimations of unseen trigrams. We use word features similar to (Weischedel *et al.*, 1993) (i.e. prefixes and suffixes of words) to estimate the observation probability of unknown words.

3.3. Implementation of the HMM–based Supertagger

The overall system is implemented in Java. In this paragraph we highlight some implementational details which reduce space and time complexity of our system (cf. (Cutting *et al.*, 1992) for a discussion of efficiency matters for POS-Taggers).

Let us first beer in mind which complexity a HMM comes along with. The model parameters of a HMM consist of N states and M output symbols from a $N \times N$ matrix A of transition probabilities, a $N \times M$ matrix B

of observation probabilities and a N–dimensional vector π (initial state distribution). All these parameters have to be yielded, i.e. the space complexity is $O(N^2 + MN)$.

The states of our HMM comprise pairs of Supertags. Hence the number of states equals the square of the number of Supertags T. Consequently the space complexity is $O(T^4 + MT^2)$, und the run time of the Viterbi algorithm is $O(T^4 n)$. From this fact directly follows that the model parameters cannot be represented by a two–dimensional array (for the 127 Supertags in our system, the two–dimensional array of 64–bit digits for the transition probabilities requires 2 GB space). As a consequence, all model parameters are stored in an *associative* manner in our system [2].

A reasonable space reduction results from only storing probabilities greater than zero[3]. With respect to the transition probabilities the following holds. These probabilities are computed during the training phase where they don't become smoothed. Smoothing is performed during tagging. During that process the trigram, bigram and unigram models are determined. Furthermore, the factors of the Backoff method are computed. A smoothed probability is only computed on demand (`getA(`(t_{i-2}, t_{i-1})`)`). Consequently the overall space complexity depends on the actually deployed training set (unknown trigrams are not stored).

With respect to the run time the following improvements can be performed to gain more efficiency in Supertagging. The Viterbi algorithm iterates over all words and all states in the following manner:

```
for each word $w_i$ in sentence {
    for each state $m$ {
        for each state $n$ {
            ⋮
        }
        ⋮
    }
}
```

Shortcuts for states with an observation probability zero and unique POS can reduce the run time reasonably. The associative hash tables allow to access all states of the currently considered word occurring in the training phase. These sets computed for the current word and its predecessor build the basis to collect the set of *relevant states* of the current word (Backoff method for the observation probabilities). Only for these states the iteration needs to be performed instead of the nested iteration over all states. More formally speaking:

$$\texttt{relevantStates}(i,j) = \{(t_k, t_l) \mid Pr(w_i|t_k) > 0 \wedge Pr(w_j|t_l) > 0\}$$

is supposed to be regarded in the two nested loops mentioned above. For $i < 0$ and $j < 0$, respectively, w_i and w_j, respectively, denote the pseudo words at the beginning of a sentence. Assuming only relevant states decreases the average run time reasonably. Our Supertagger requires approximately 28 ms for the tagging of a sentence only conducting relevant states whereas it runs at least a second if all states are considered.

4. Application of Supertagging in the user–initiative Help Desk system KoHDaS

The results of the Supertagger described in the previous section allow a LDA (Srinivas, 1997a) to discover dependencies between the Supertags in the user's turn. The dependency structure accomplished by this method is used in classification and in information extraction in the following manner.

In KoHDaS–NN, the user's turn is classified according to *significance vectors* of the form:

$$v(w, c_i) = \frac{\text{occurrences of a word from } w \text{ within class } c_i}{\sum_{j=1}^{n} \text{occurrence of words from } w \text{ within class } c_j}$$

2. The associative storing in Java is realized by the class HashMap, which provides a Hash table (see, e.g., (Flanagan, 2002)).
3. It is important to notice here that due to the fact that the real–digit arithmetics cannot differentiate between zero an very small values (as holds for the products of probabilities computed in the Viterbi algorithm) we deal with the logarithms of the probabilities in the hash table of the probabilities of Supertags. This states a suitable method here because not probabilities themselves but the arguments of such probabilities are maximized, i.e. the Viterbi algorithm computes the maximum sum of the logarithms of the probabilities instead of the maximum product of the probabilities.

where only 616 words are actually regarded and matched with a reduced vocabulary with 131 word groups c_i, i.e. general concepts in our domain (such as *'hard disk'*, *'monitor'* and *'capacity'*) containing words, which can be considered to be synonymous (e.g., words in class *'hard disk'* are *'disk'* or *'harddrive'*). Generally, significance vectors account for the importance of the word in a specific problem class.

In KoHDaS–ST, these significance vectors are adjusted using the results of the structural information collected by the Supertagger and the LDA. The adjusted significance vectors v^* are computed by:

$$v^*(w, c_i) = \begin{cases} \alpha^d\, v(w, c_i) & \text{if } v(w, c_i) > \frac{1}{n}\sum_{j=1}^{n} c_j, \\ (1 + \beta d) * 0.1 & \text{if } v(w, c_i) = 0 \text{ and } d > 0, \\ (1 + \beta d)\, v(w, c_i) & \text{otherwise.} \end{cases}$$

where d represents the syntactic depth of the sentence in that the current word occurs and α and β are constant values. Tests have shown that suitable values for α and β are $\alpha = 0.8$ and $\beta = 0.6$.

The results of this approach compared to the pure neural net–based version of KoHDaS are outlined in Table 1. The table shows that the Mean Squared Error (MSE) of three sub–networks of KoHDaS–NN is decreased in the top level net as well as in the local net for disk problems but increased in the local net for monitor problems. The reasons why the monitor problems behave in this unexpected manner are topic of future investigations (cf. Section 6).

Table 1: Mean Squared Error (MSE).

Net	MSE in Test	
	KoHDaS–NN	KoHDaS–ST
NN differentiating monitor and disk probs.	3.85	3.45
Local NN - disk probs.	10.33	9.65
Local NN - monitor probs.	4.72	4.91

In the graph–based information extraction step, each node of the graph corresponds to the information already extracted from the turn. Nodes can be associated with questions to be asked by the system. Edges are labeled with sets of word groups enabling a transition if an appropriate word occurs in the user's input. See Figure 1 for a partial dialogue graph of KoHDaS–NN.

In KoHDaS–ST the results of the dependency analysis together with features in the lexicon are used to create a kind of *semantic representation* of the user's turn (cf. (Bäcker, 2002)). Edges in the new dialogue graphs are labeled with this representation (see Figure 2) resulting in improved processing of the turn. For example the turn *"Sometimes my computer drives me mad. My monitor started glimmering 3 days ago."* would enable the transition from node 5 to 51, as *'sometimes'* is found in the input. In KoHDaS–ST this will not happen, because *'sometimes'* is not related to *'glimmering'*.

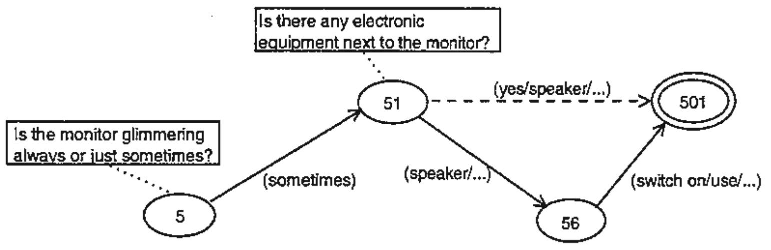

Is there any electronic equipment next to the monitor?
Is the monitor glimmering always or just sometimes?
5
51
501
56
(sometimes)
(speaker/...)
(yes/speaker/...)
(switch on/use/...)

Figure 1: Part of a dialogue graph in the problem class of glimmering monitors in KoHDaS.

5. Related work

A well studied method to extract relevant information from potentially ill–formed (as is the case for spoken utterances) or not completely mastered (as is the case for automatically analysed spoken utterances) input is *chunk*

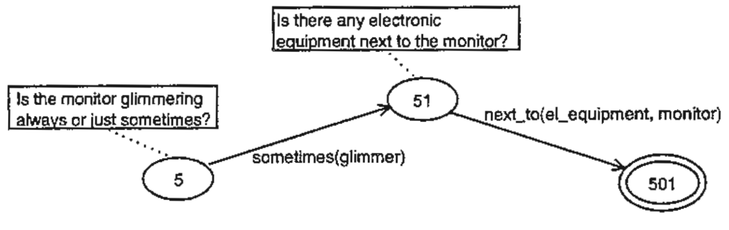

Is there any electronic equipment next to the monitor?
Is the monitor glimmering always or just sometimes?
51
next_to(el_equipment, monitor)
sometimes(glimmer)
5
501

Figure 2: Part of a dialogue graph in the problem class of glimmering monitors in KoHDaS–ST.

parsing (also called *chunking*; see, e.g., (Abney, 1991), (Appelt *et al.*, 1993), (Grishman, 1995) or (Hobbs *et al.*, 1997))). Cascaded finite state automata analyse various phrase types expressed in terms of regular expressions. The main advantage of this approach is its robustness and the fast run time. The main obstacle of chunking issues from the restricted formal power of finite state automata. Cascades of several levels (cf. the system FASTUS (Appelt *et al.*, 1993), (Hobbs *et al.*, 1997) with five levels) allow for the analysis of recursive structures to some extend. The basic level accepts smaller linguistic constructions, whereas in the next levels these elements become grouped into larger linguistic units. Accordingly, FASTUS can basically recognize 'complex words' such as proper nouns consisting of several individual words. However the coverage remains restricted to the static number of cascaded phases.

Another robust and fast parsing method offers *statistical parsing*. According to a labeled corpus generalization rules can be extracted (cf. *Treebank*, (Marcus, Santorini and Marcinkiewicz, 1993)). These rules can be *grammar rules* (see, e.g., (Charniak, 1997), (Collins, 1996)) or *decision trees* (see, e.g., (Magerman, 1995)). Each rule becomes associated with a probability, which is determined in the training phase with respect to the corpus. Parsing means to find the most likely derivation according to these rules. The term statistical parsing subsumes a further variant where the rule set is also determined beforehand (see, e.g., (Black *et al.*, 1993)). In the training phase probabilities for these rules are computed according to the corpus (cf. *Probabilistic Context–Free Grammars* (*PCFG*, (Booth, 1969)). The goal of this method is primarily disambiguation whereas robustness is a not so relevant here.

Let us finally mention recent work in the area of Supertagging. Supertagging (Joshi and Srinivas, 1994) is a disambiguation method which extends *Part–Of–Speech Tagging* (*POS–Tagging*). A *Lexicalized Tree Adjoining Grammar*, i.e. any rule has at least one lexical anchor (cf. (Schabes, 1990)) underpins the system. As is the case for Part–Of–Speech Tagging, the model is trained with a labeled corpus. In the training phase, each word of each input sentence becomes associated with a lexicalized rule according to the model (*Supertag*). On the basis of this relation, a Lightweight Dependency Analyzer (LDA, (Srinivas, 1997a)) identifies relations between the Supertags of the individual lexical entries of a input sentence, i.e. the grammar rules the Supertagger gives the highest probabilities. As far as possible a complete parse is computed. Particularly the ability to produce complete parses (if possible) compared to individual phrases in chunking led to the choice of the latter method for our application domain. Here, the user's utterance should be best analysed particularly in the step of extracting task parameters. Our conjecture is that Supertagging allows for a more elaborate identification of complex constructions such as negations and their scope in the user's utterance.

In the area of Supertagging various approaches for the model have been proposed in the literature (e.g., *Trigram Supertagging* by (Srinivas, 1997a) with Good–Turing Discounting (Good, 1953) and the Backoff method by Katz (Katz, 1987)). On the basis of a training corpus of 1 000 000 English words the Supertagger provides an accuracy of 92,2%. *Head Trigram Supertagging* (Srinivas, 1997a) is a similar method based on trigrams. However not the two previous Supertags are used to compute the current Supertag but the two previous *Head Supertags*. A Head Supertag is a previously computed Supertag which influences the choice of the current Supertag. The method works in two phases. In the first one all Head Supertags are determined. In the second phase, the Head Supertags are used to compute the probabilities of all Supertags. This method assigns in 91,2% of the cases the correct Head Supertags for a training corpus of 1 000 000 words. Its accuracy is 87%.

Transformation–based Supertagging (Srinivas, 1997a), (Srinivas and Joshi, 1999) adapts the central idea of transformation–based POS–Tagging (Brill, 1993), (Brill, 1995). For this method any word in the corpus is labeled with its most frequent tag. During Tagging these tags can be changed by a list of transformations. Such a transformation consists of a pattern, which activates the rule and a rewriting rule. In order to train this Supertagger a set of transformation patterns and a labeled reference corpus has to be provided. The training algorithm determines the best order of rule applications by minimizing the error rate of the Tagger compared to the reference corpus. The Supertagger based on this model has been trained with 200 000 words and reaches an accuracy of 90%.

A Supertagger for German (Bäcker, 2001) based on Hidden Markov models and a bigram model was trained and tested with word classes instead of individual words. Notice that from this fact a loss of accuracy results. Furthermore only basic smoothing techniques were imposed. Accordingly, first results lack far behind the previously mentioned ones. On the basis of 5 460 sentences of the NEGRA corpus (Brants *et al.*, 1997) the Supertagger has an accuracy of 65,5%. This was basically the reason to impose trigrams to the Supertagger we describe here.

Supertagging and Lightweight Dependency Analyzers exhibit high robustness and efficiency[4] and are deployed for shallow parsing in various contexts (e.g., *text chunking* with Supertags (Srinivas, 1997b)). Text chunking (Abney, 1991) means that a sentence is divided into several non-overlapping segments. By this method individual types of phrases (e.g., identification of noun phrases *Noun Chunking*) can be identified in a text. A respective LDA reaches high precision (91,8%) and recall (93%) (cf. (Srinivas, 1997b)).

6. Conclusions

In order to conclude, the application of Hidden Markov model–based Supertagging in the user–initiative dialogue system KoHDaS–ST helps to remedy the lack of accuracy resulting from speaker–independent speech recognition on the basis of general dictionaries and grammars as required in a user–initiative dialogue system.

Comparing the results of German Supertagging (78.3%) to English (92.2%), two different reasons lead to less good results. First, our training set (1 973 words) is small compared to the English one (1 000 000 words). Accordingly, many unseen trigrams are imposed on the system. Second, German is a language with free word order. This fact amplifies the effects of sparse data (cf. Spanish Supertagging has an accuracy of about 80% (Srinivas, 1998)). In the future the training set of KoHDaS–ST will be extended. Furthermore, unsupervised learning methods integrated with supervised methods (cf. (Montoya, Suárez and Palomar, 2002)) will be deployed in our system. How far we can get with a free word order language like German is currently an open problem.

A further open problem is why the adjustment of significance vectors according to the identified sentence structure decreases the number of correctly classified problems with respect to the class of monitor problems. Perhaps our conjecture that concepts mentioned in subordinate clauses have a lower impact to the decision making than those in the main clause is too strict.

References

Abney, Steven. 1991. Parsing by chunks. In Robert Berwick, Steven Abney and Carol Tenny, editors, *Principle-based parsing: Computation and Psycholinguistics*. Kluwer Academic Publishers, Dortrecht, The Netherlands, pages 257–278.

Appelt, Douglas E., Jerry R. Hobbs, John Bear, David Israel and Mabry Tyson. 1993. FASTUS: A Finite-state Processor for Information Extraction from Real-world Text. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI–93)*, pages 1172–1178, Chambéry, France.

Bäcker, Jens. 2001. Entwicklung eines Supertaggers für das Deutsche. Studienarbeit, Universität Koblenz-Landau, Institut für Computerlinguistik, Koblenz, Germany.

Bäcker, Jens. 2002. KoHDaS–ST — Supertagging in dem automatischen Dialogsystem KoHDaS. Diplomarbeit, Universität Koblenz-Landau, Institut für Computerlinguistik, Koblenz, Germany.

Baum, Leonard E. 1972. An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. *Inequalities*, 3:1–8.

Baum, Leonard E. and J. A. Eagon. 1967. An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology. *Bulletin of American Mathematical Society*, 73:360–363.

Black, Ezra, Frederick Jelinek, John Lafferty, David M. Magerman, Robert Mercer and Salim Roukos. 1993. Towards History–based Frammars: Using Richer Models for Probabilistic Parsing. In *Proceedings of the 31st Conference of the Association of Computational Linguistics (ACL–93)*, pages 31–37, Columbus, Ohio, USA.

Boje, Johan, Mats Wiren, Manny Rayner, Ian Lewin, David Carter and Ralph Becker. 1999. Language–Processing Strategies and Mixed–Initiative Dialogues. In J. Alexanderson, L. Ahrenberg, K. Jokinen and Jönsson, editors, *Special Issue on Intelligent Dialogue Systems. ETAI (Electronic Transactions on Artificial Intelligence).*

Booth, Taylor L. 1969. Probabilistic representation of formal languages. In *IEEE Conference Record of the 10th Annual Symposium on Switching and Automata Theory*, pages 74–81.

Brants, Thorsten, Roland Hendriks, Sabine Kramp, Brigitte Krenn, Cordula Preis, Wojciech Skut and Hans Uszkoreit. 1997. Das NEGRA-Annotationsschema. Negra project report, Universität des Saarlandes, Computerlinguistik, Saarbrücken, Germany.

Brill, Eric. 1993. Automatic grammar induction and parsing free text: A transformation-based approach. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics (ACL–93)*, pages 259–265, Columbus, Ohio, USA.

4. The run time of a Supertagger–based LDA is $O(n)$ where n denotes the length of the input.

Brill, Eric. 1995. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–566.

Charniak, Eugene. 1993. *Statistical Language Learning*. Cambridge, Massachusetts: MIT Press.

Charniak, Eugene. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI–97)*, pages 47–66, Menlo Park, CA, USA.

Chen, Stanley F. and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL–96)*, pages 310–318.

Collins, Michael John. 1996. A New Statistical Parser Based on Bigram Lexical Dependencies. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL–96)*, pages 184–191, San Francisco, CA, USA.

Cutting, Douglas, Julian Kupiec, Jan O. Pedersen and Penelope Sibun. 1992. A practical part-of-speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing (ANLP)*, pages 133–140, Trento, Italy.

El-Beze, Marc and Bernard Merialdo, 1999. "Hidden Markov Models". In Hans van Halteren, editor, *Syntactic wordclass tagging*, chapter 16, pages 263–284. Dordrecht, the Netherlands: Kluwer Academic Publishers.

Flanagan, David. 2002. *Java in a Nutshell*. 4th edition. Cambridge, MA, USA: O'Reilly.

Good, Irving J. 1953. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40:237–264.

Grishman, Ralph. 1995. The NYU System for MUC-6 or Where's the Syntax? In *Proceedings of the 6th Message Understanding Conference (MUC-6)*, pages 167–175, San Francisco, CA, USA.

Harbusch, Karin, Melanie Knapp and Christoph Laumann. 2001. Modelling user-initiative in an automatic help desk system. In Hitoshi Isahara and Qing Ma, editors, *Proceedings of the Second Workshop on Natural Language Processing and Neural Networks (NLPNN2001)*, pages 69–76, Tokyo, Japan.

Hobbs, Jerry R., Douglas E. Appelt, John Bear, David Israel, Megumi Kameyama, Mark Stickel and Mabry Tyson. 1997. FASTUS: A cascaded finite-state transducer for extracting information from natural-language text. In Emmanuel Roche and Yves Schabes, editors, *Finite State Devices for Natural Language Processing*. MIT Press, Cambridge, MA, USA, pages 383–406.

Joshi, Aravind K. and Bangalore Srinivas. 1994. Disambiguation of Super Parts of Speech (or Supertags): Almost Parsing. In *Proceedings of the 17th International Conference on Computational Linguistics (COLING–94)*, pages 154–160, Kyoto, Japan.

Jurafsky, Daniel and James H. Martin. 2000. *Speech and Language Processing*. Upper Saddle River, NJ, USA: Prentice Hall.

Katz, Slava M. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 35(3):400–401.

Litman, Diane, Shimei Pan and Marilyn Walker. 1998. Evaluating response strategies in a web–based soken dialogue agent. In *Proceedings of the 36th Annual Meeting of the ACL and the 17th International Conference on Computational Linguistics (COLING–ACL–98)*, pages 780–786, Montreal, Canada.

Magerman, David M. 1995. Statistical Decision-Tree Models for Parsing. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL–95)*, pages 276–283, Cambridge, MA, USA.

Manning, Christopher D. and Hinrich Schütze. 2000. *Foundations of statistical language processing*. Cambridge, MA, USA: MIT Press.

Marcus, Mitchell P., Beatrice Santorini and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19:313–330.

Merialdo, Bernard. 1994. Tagging English text with a probabilistic model. *Computational Linguistics*, 20(2):155–172.

Montoya, Andrés, Armando Suárez and Manuel Palomar. 2002. Combining Supervised–Unsupervised Methods for Word Sense Disambiguation. In Alexander Gelbukh, editor, *Proceedings of the 3rd International Conferences on Intelligent Text Processing and Computational Linguistics (CICLING)*, pages 156–164, Mexico City, Mexico. Springer.

Rabiner, Lawrence R. 1989. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Rabiner, Lawrence R. and Biing-Hwang Juang. 1986. An introduction to hidden Markov models. *IEEE ASSP Magazine*, pages 4–15, January.

Schabes, Yves. 1990. *Mathematical and Computational Aspects of Lexicalized Grammars*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, USA.

Srinivas, Bangalore. 1997a. *Complexity of Lexical Descriptions and its Relevance to Partial Parsing*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, USA.

Srinivas, Bangalore. 1997b. Performance evaluation of Supertagging for partial parsing. In *Proceedings of Fifth International Workshop on Parsing Technology (IWPT–97)*, Boston, USA.

Srinivas, Bangalore. 1998. Transplanting Supertags from English to Spanish. In *Proceedings of Fourth International Workshop on Tree-Adjoining Grammars (TAG+4)*, pages 5–8, Philadelphia, PA, USA.

Srinivas, Bangalore and Aravind K. Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2):237–265.

Strachan, Linda, John Anderson, Murrey Sneesby and Mark Evans. 1997. Pragmatic user modelling in a commercial software system. In *Proceedings of the 6th International Conference on User Modeling*, pages 189–200, Chia Laguna, Italy.

Viterbi, Andrew J. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, IT-13:260–269.

Weischedel, Ralph, Marie Meteer, Richard Schwartz, Lance Ramshaw and Jeff Palmucci. 1993. Coping with ambiguity and unknown words through probabilistic models. *Computational Linguistics*, 19(2):359–382.

Wermter, Stefan. 1995. *Hybrid connectionist natural language processing*. London, Great Britain: Chapman and Hall, International Thomson Computer Press.

Exploiting Semantic Knowledge in LTAG-based controlled indexing of technical data

Patrice Lopez
LORIA Labs

David Roussel
EADS Suresnes

1. Introduction

The work presented in this abstract follows the first experiments presented in (Lopez and Roussel, 2000) on the robust modeling of terms in the LTAG framework to index spoken annotation transcriptions. We continue to experiment the LTAG workbench (Lopez, 2000), and integrate it with on the shelf tools (term extractor, taggers, terminological model) that embed and manage different kind of linguistic resource. The key advantages of using the LTAG formalism in this context is a precise linguistic modeling useful for the representation of term variants, the exploitation of semantic constraints and the ability to specialize the terminological resources to several specific tasks. To illustrate the last point, we first present another application that motivates this work, the exploitation of technical documentation. In this particular application, the semantic disambiguisation can help to improve the accuracy of the documents and their reuse to design checking procedures. We then present more precisely the LTAG modeling and the implemented system TERESA based on a POS tagger, finite-state transducers encoding LTAG trees and a semantic tagger.

2. Application

When documentation is an important part of a company activity, there are always some existing resources which formalize semantic information available for technical words. For example, currently, in the EADS context, the design of an ontology that gives the semantic categories of specific terms is considered as an important starting point. During the document life cycle within a project, an ontology facilitates also intra-operation between different kind of document and so, a mandatory part of the work being done by the project community is to standardize the terms, acronyms and abbreviations. This task is an EADS directive and procedure.

Since these terms are already defined, their identification for the purpose of classifying and accessing documents is called *controlled indexation* in opposition to free indexation where the index terms are automatically defined. Controlled indexation allows us to exploit existing resource to achieve a better precision in the indexation and to link old and new information in a more coherent and comprehensive way for documentalists.

The experiments in this work have been made with XML elements called WARNINGS extracted from an aircraft documentation. The correct identification of a particular term and its variants The use of controlled indexing on these elements is twofold : first, help the navigation into those elements in order ot control the coherence of the content of these element, second, to be able to disambiguate semantically sequence of words.

An expected enhancement of robust controlled indexing is to derive more easily a procedure from the description of the warning in the whole documentation, or at least to take more easily into account the important warning in the procedures. The identification of a particular operation benefit from a disambiguation of certain sequence of words.

For instance, engine operation concern the motor intervention (table 1) or the system intervention (table 1). To avoid engine damaged, it is sometime necessary to access both the cockpit and the motor. One interest of semantic knowledge exploitation in controlled indexing is to extract directly the sentences that concern one type of engine intervention.

You must not operate the engine with the fan cowl doors open.
During engine operation, the bleed valve can open.
Operation of the engine can cause fuel to go overboard.
Ear protection must be worn by all person who operate the engine while engine operates.

Figure 1: Example of motor intervention.

The engine must operate 9 hours at idle with the lubrication system filled.
Do not motor, start or operate engine unless a positive fuel inlet pressure is indicated.
The exhaust gas is hot when the engines operates at idle or higher power.
To maintain satisfactory engine oil temperature do an engine start and operate the engine at idle.

Figure 2: Example of engine intervention that need a control from the cockpit.

In the first case (motor intervention, table 1), the operation implies the filtering of sentences that gather a person as an agent or implicit agent. A syntactic analysis is enough to disambiguate this case from the next case (table 2), that implies that the engine is the subject of the operation. However, different elementary trees are concern, and don't provide an easy interface to the integration of the syntactic analysis of the terms within an application.

To consider a unique semantic feature instead of different kind of derived tree, we extend the syntactic categories of elementary trees with semantic constraints and compile them as FST. This allows us to keep abstraction in the description of linguistic resource and to benefit from other linguitic tool, namely the semantic tagger Tropes in order to study the dependance between semantic classes in a corpora. Tropes is a semantic analyser (Ghiglione et al., 1998). It embed morphosyntatic and semantic analyser that i) segment a text in linguistic proposition, ii) extract homogeneous category according to their thematic content, iii) export the result in a XML coding, iv) to count the frequence and the dependency between semantic classes. The Tropes environment facilitate the adaptation of the default semantic classes hierarchy in order to take into account specific semantic knowledge. A set of heuristics are applied to disambiguate the semenatic categorie of a lexical unit. They consist in finding isotopies of a same semantic classe and exploiting statistical coocurrences between complementary concepts inside a grammatical proposition.

3. LTAG-based Terminological Processing

3.1. LTAG representation of a terminology

A given term can be represented as a partial parsing tree, i.e. a derived tree, as represented figure 3. After removing all lexical information in this tree, we obtain a elementary tree schema in the sense of (Candito, 1996) that can be used to represent syntactically similar terms.

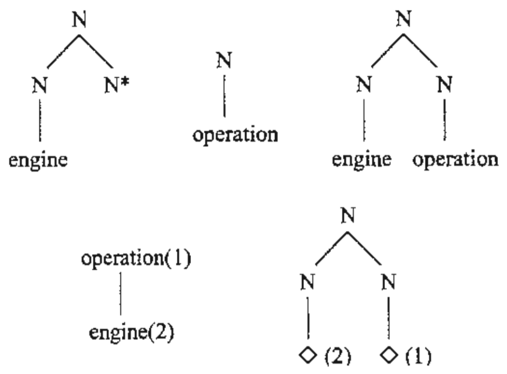
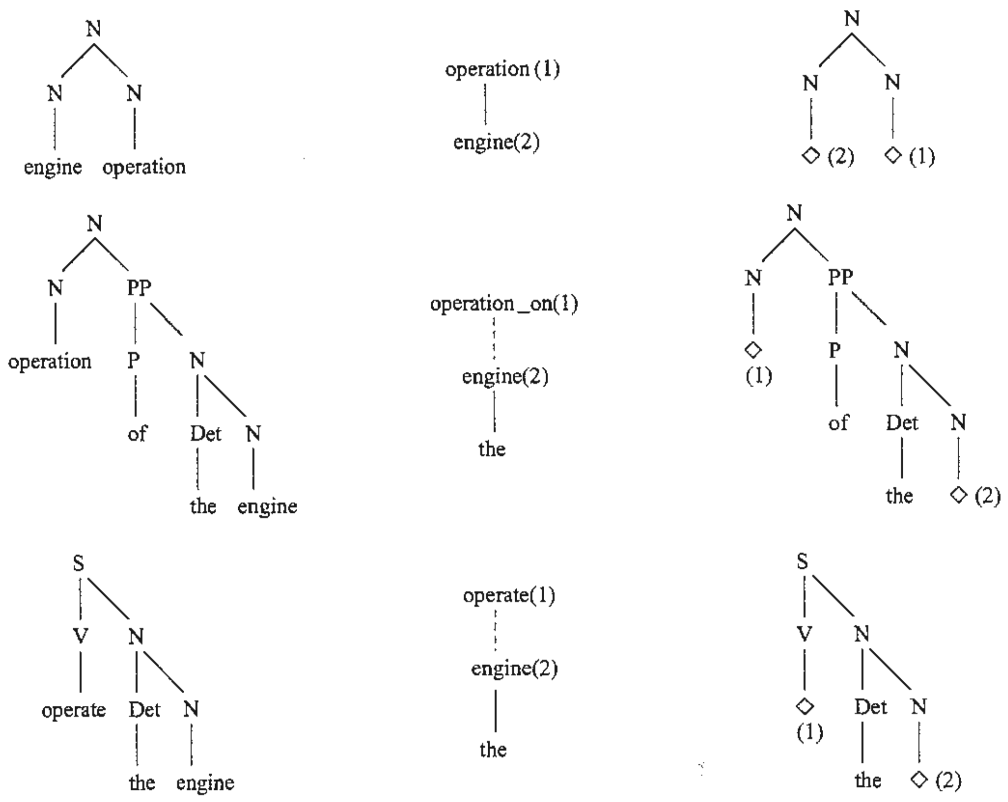


Figure 3: Anchored elementary trees, derived and derivation tree and the corresponding term schema for N-N terms.

This principle can be used to represent a complete terminology by parsing the list of terms with a LTAG grammar which coverage is limited or with existing term trees. For each term we obtain one or several derived and derivation trees. We have used the LTAG Workbench presented in (Lopez, 2000) for this purpose. Practically for English and French, a LTAG grammar covering only NP structures and basic verb and relatives is enough to cover



N
N N
engine operation
operation (1)
engine(2)
◇ (2) ◇ (1)
operation P N of Det N the engine
operation_on(1)
engine(2)
the
◇ (1) P of Det N the ◇ (2)
S V N operate Det N the engine
operate(1)
engine(2)
the
S V N ◇ (1) Det N the ◇ (2)

Figure 4: A basic term aligned with two of its variants.

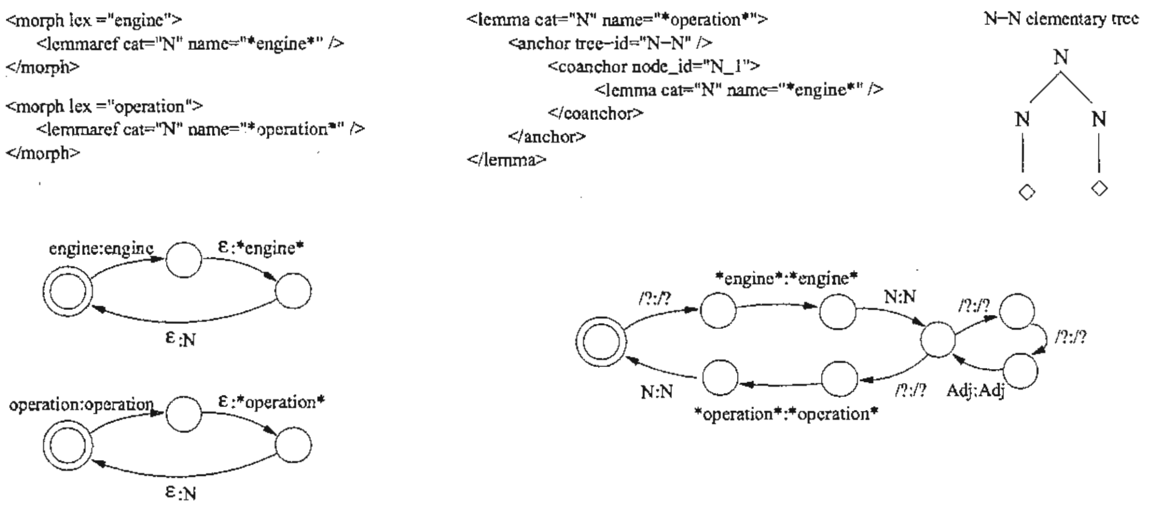
nearly all terms and their variants. The resulting set of elementary tree schema can be reused and be anchored by new terminologies. This linguistic representation allows us to extend very easily the coverage of a list of terms to their variants without the use of specific complex meta-rules specifically developed for the terminological purpose as in (Jacquemin and Tzoukermann, 1999).

3.2. Term variance

Term variants are very important in terminology analysis and extraction. Variance is can be caused by inflection, morphological derivation, adjectival modification, optional or variable preposition, ellipsis, coordination or the use of copulative or support verbs. Experiments have shown that usually approximately 30% of terms occur as term variants of basic terms (Jacquemin and Tzoukermann, 1999). In our model a basic term template and its variants are gathered in a tree family, i.e. the possible variance is a linguistic knowledge encoded in a family. The lexical information are then removed from these trees templates and their correspondances in term of *morphological root* are directly annotated as shown in figure 4. Classically candidate terms are validated by an occurence in a corpus.

3.3. Finite-state Compilation

Using a classical LTAG parser would be too expensive and too powerful since the identification of a term is limited to the lexical anchoring of a LTAG tree taking into account possible variances. Consequently we compile the LTAG model into another structures more relevant for computational processing. Finite-State Transducers-based processing is particularly well suited for processing large scale and lexicalized grammars.



```
<morph lex ="engine">
    <lemmaref cat="N" name="*engine*" />
</morph>

<morph lex ="operation">
    <lemmaref cat="N" name="*operation*" />
</morph>
```

```
<lemma cat="N" name="*operation*">
    <anchor tree-id="N-N" />
        <coanchor node_id="N_1">
            <lemma cat="N" name="*engine*" />
        </coanchor>
    </anchor>
</lemma>
```

Figure 5: A term represented as an elementary tree schema encoded in TAGML and its compilation in Layered FST (without the morpho-syntactic features and the morphology root for more readability).

All the transducers used in this work are Layered Finite State Tranducers (LFST). LFST have been described in (Adant, 2000). LFST extends usual FST by constraining each sequence of transition to be divided in layers. The alphabets associated to these layers can be different. Traditionally the character /? is the default symbol, and ϵ the symbole for empty string. LFST allow the combination of different levels of information while keeping an important sharing of states. Figure 5 gives an exemple of two morphosyntatic transducers and a syntactic transducer compiled from a LTAG grammar initially in the TAGML format. The convertion algorithm extends certain transition by possible categories which can be introduced by modifiers.

All the resulting transducers are then combined into a morphosyntactic transducer and a *terminological* transducer which are both determinized and minimized as possible thanks to standard FST algorithms (Mohri, 1997). The lexicalization step consists of representing the text as an identity transducer that is combined first with the *morpho-syntactic* transducer and then with the *terminological* transducer, resulting in a transducer where all possible terms have been identified.

3.4. Adding Semantic Constraints

A basic assumption concerning the use of semantic knowledge in NLP applications is that it improves the customization of the final results. On the other hand, the amount of ambiguities the application have to deal with grows up and perturb the result interpretation.

The idea is to add semantic class categories in the node label of LTAG trees similarly as presented in (Lopez and Roussel, 2000). The semantic consistency principle is exploited in order to localize the semantic constraints of the predicate represented by the term and the tree. When compiled into a LFST, the semantic category introduce a new layer as shown in figure 6.

Practically the semantic class are provided by the Tropes semantic tagger based on a training corpus.

4. The TERESA System

4.1. Bootstrapping the system resources

The resources for a given terminological domain are obtain thanks to two training corpora. The first one validates term variants allowed given a list of terms as explained in section 3.2. The second one is used to obtain a list of relevant semantic class thanks to the Tropes semantic tagger. For instance, in the application presented in

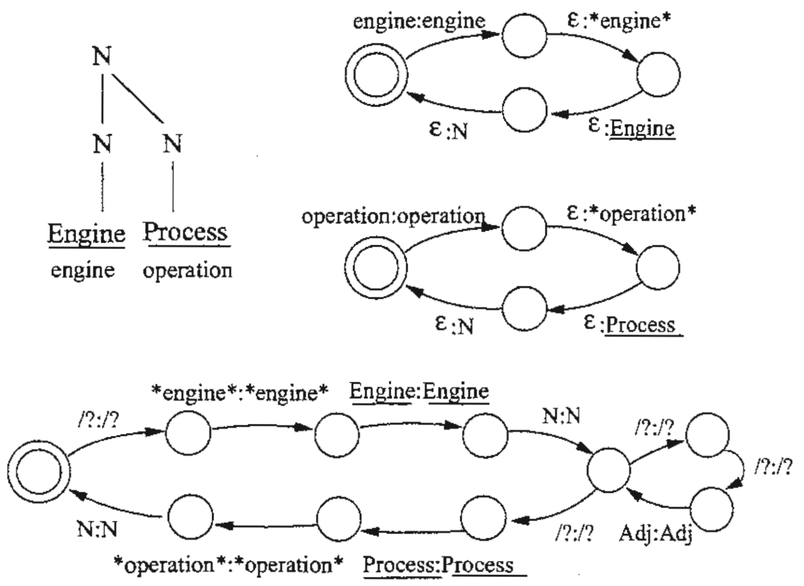


Figure 6: Semantic information integrated in elementary tree schema and in its compiled LFST.

section 2, the semantic tagging is based on 125 collocations that have extracted from a short extract of 350 caution documents. The semantic tag must match the semantic categories given by the term hypothesis. If the semantic tag differs, the corresponding term hypothesis is pruned.

4.2. Term analysis with TERESA

The TERESA system (TERminological Extraction and Statistical Analysis) allows us to analyse or extract terms in textual or semi-structured documents. Textual data is first tokenized and the morphology is processed thanks to the combination of the input string represented as a FST and a morpho-syntactic FST. The result is a FST that encoded all possible lexical analysis of the text.

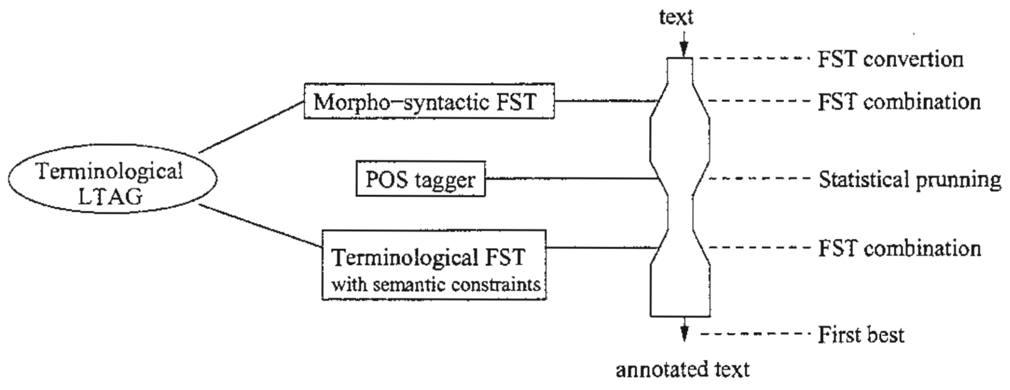


Figure 7: Overview of the TERESA system.

We apply then on this FST a POS tagger specially developed for this purpose. While the vast majority of POS tagger are limited to a linear tagging of text and a fully disambiguated tag assignation. The tagger used for this experiment is able to process efficiently word graphs coming from an Automatic Speech Recognizer for instance, and to give, if necessary, the list of ambiguous tags with their corresponding probabilities. This tagger is based on a classical trigram model with a viterbi search algorithm, it uses the linear interpolation algorithm for sparse data, implements a suffix based statistical models for unknown words. Classically a beam is used to speed up significantly the viterbi search with a negligible impact on the accuracy result. This pure statistical process is combined with a deterministic step based on the application of negative rules. This rules are compiled into a FST that is combined to the input represented as a FST, preliminarly to the statistical process, similarly to (Tzoukermann and Radev, 1999). The ability to deliver ambiguous results is very important since we know that accuracy of POS taggers is limited.

The terminological LTAG model that encodes semantic class category constraints is then combined to the resulting ambiguous tagged LFST. After this step all possible terms are identified according to the morphosyntactic, the POS tagging and a semantic match.

Finally only the best path of the final structure is considered thanks to a classical Dijkstra shortest path algorithm implementation.

5. Conclusion

We have presented a LTAG-based terminological system able to identify very precisely a given list of terms in text. The same LTAG-based terminology can be specialized to spoken application and can exploit other relevant parsing techniques. This specialization illustrates the benefit of using a linguistically motivated formalism as a generic resource. The interest of LTAG for our indexing application is the ability to exploit semantic knowledge in this process thanks to the precise semantic interface and the use of a semantic tagger.

This work fit into a serie of experiments using LTAG formalism in applications in order to :

- manage grammars because it's easier to control and design one lexicalized grammar than several small grammars
- design of a robust LTAG parser that cope with the analysis of a graph of speech recognition hypothesis.
- to detect certain ambiguities in the procedures and prevent misunderstandings.

A major feature is the possible integration with existing NLP tool thanks to the XML framework adopted.

References

Arnaud Adant. 2000. Study and Implementation of a weighted finite-state library - application to speech synthesis. M.sc., Faculté Polytechnique de Mons.
Marie-Hélène Candito. 1996. A principle-based hierarchical representation of LTAGs. In *COLING'96*, Copenhagen, Denmark.
R. Gbiglione, A. Landré, M. Bromberg, and P. Molette. 1998. *L'analyse automatique des contenus*. Dunod, Paris.
C. Jacquemin and E. Tzoukermann. 1999. NLP for Term Variant Extraction: A Synergy of Morphology, Lexicon and Syntax. In T. Strzalkowski, editor, *Natural Language Information Retrieval*. Kluwer, Boston, MA.
Patrice Lopez and David Roussel. 2000. Predicative LTAG grammars for Term Analysis. In *TAG+5*, Paris, France.
Patrice Lopez. 2000. LTAG Workbench: A General Framework for LTAG. In *TAG+5*, Paris, France.
Mehryar Mohri. 1997. Finite-State Transducers in Language and Speech Processing. *Computational Linguistics*, 23:269–312.
Evelyne Tzoukermann and Dragomir Radev. 1999. Use of weighted finite state transducers in part of speech tagging. In Andras Kornai, editor, *Extended Finite State Models of Language*. Cambridge University Press.