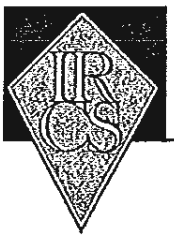


Institute for Research in Cognitive Science

Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+4)

University of Pennsylvania
3401 Walnut Street, Suite 400A
Philadelphia, PA 19104-6228

August 1998

Site of the NSF Science and Technology Center for Research in Cognitive Science

IRCS Report 98--12

Foreword

The papers in this volume were presented at the fourth workshop on Tree-Adjoining Grammar and related frameworks (TAG+4), held at the Institute for Research in Cognitive Science at the University of Pennsylvania in August 1998. Previous workshops were held at Schloß Dagstuhl in Germany (1990), at the University of Pennsylvania (1992), and at the University of Paris VII (1994).

As was the case with previous TAG+ workshops, the papers presented at TAG+4 represent a broad spectrum of interests, including linguistics, formal studies, parsing, and studies relating TAG to other frameworks. In addition, this collection includes a significant number of papers that address applications and engineering issues.

We would like to thank all authors for their work in preparing the papers, the Institute for Research in Cognitive Science at the University of Pennsylvania for financial support (including a supplementary grant from the NSF to IRCS for the tutorial preceding the workshop), Aravind Joshi for advice, and Jennifer MacDougall and the IRCS Staff for their invaluable help.

Anne Abeillé (Université de Paris VII)
Tilman Becker (DFKI GmbH)
Owen Rambow (CoGenTex, Inc.)
Giorgio Satta (Università di Padova)
K. Vijay-Shanker (University of Delaware)

Table of Contents

An Experiment on Synchronous TAGs for the Construction of a Transfer Module

Alexandre Agustini & Vera Lúcia Strube de Lima

PUCRS - Instituto de Informática
Av. Ipiranga, 6681 prédio 30 bloco 4
90619-900 Porto Alegre RS - BRASIL
{agustini,vera}@inf.pucrs.br

Abstract

This paper presents some considerations on the use of Synchronous TAGs for the design of a structural transfer module, which is the main component of transfer-based systems for Machine Translation. The transfer module establishes the correspondences between the structural representation of both the source and target languages. A study of a corpus from Economics was carried out in order to define structural divergences for the translation between the Portuguese and English languages.

1 Introduction

Machine translation (MT) has been a challenge for linguists and computer scientists over the last decades. During this period, plenty of progress was accomplished, though the results are not yet the ones expected.

Transfer based approaches to MT involve three main phases: analysis, transfer and generation. During analysis, the syntactic and semantic structure of a sentence is made explicit through a source language (SL) grammar and semantic processing modules. The result of the analysis is one or more syntactic and semantic representations which are used to construct a syntactic and/or semantic representation in the target language (TL) through a series of transfer rules and according to a bilingual lexicon. From this representation a TL sentence is generated based on some form of mapping procedure [Hutchins & Sommers 92; Trujillo 95].

In this paper we describe a prototype implementation of a transfer MT module based on the Synchronous Tree-Adjoining Grammars (STAGs) formalism. STAGs [Shieber & Schabes 90] are a variant of Tree-Adjoining Grammars (TAG) to express the related representations of semantics and syntax in natural-language description.

2 Corpus based development

Our basic approach is corpus-driven. We started by collecting a source-language corpus (Portuguese sentences) in a limited domain. The corpus made up by 200 sentences was created randomly from an economics headlines database. About 50% of them were discarded because they were ill-formed senteces. The database had previously been generated from a news broadcasting system.

An English version of the corpus was produced by a native translator with experience in the domain terminology. Finally, both corpuses were tagged and aligned in order to achieve:

- *virtual grammars*[1] for both the source

[1] In this context, *virtual grammar* refers to a syntactic structure subset necessary for parsing any input sentence and generating target structures occuring in the corpus.

and target corpuses: the subset of lexicalized trees necessary for syntactic/semantic analysis of source and target corpus was defined. These grammars are based on [Kipper 94] and [Becker et al. 94] technical reports.

- *lexicon coverage*: source and target lexical dictionaries were set. As we are working on a lexicalized model [Abeillé 90; Srinivas et al. 94], each lexical item anchors one or more syntactic structures.
- *translation discrepancies*: translation problems to be solved during transfer from source to target structures were addressed.

We found it helpful to divide translation problems into three different types: lexical, syntactic and lexical-semantic. These terms are used according to the following concept (according to [Dorr 94]): *Lexical problems* are concerned about finding correct choices for expressions that occur in the source and target languages. *Syntactic problems* feature syntactic properties associated with each language (i.e., properties that are independent of the actual lexical items that are used). Finally, *lexical-semantic problems* which feature properties that are lexically determined.

Some examples of divergences observed in the corpus are presented on Table 1. In case (1), problems originated from lexical gaps in the source and target languages are shown; the translation has to deal with structural problems and feature inheritance. Syntactic problems (2) usually have to do with word order, in the examples, adjectival phrase order. In the last one, a lexical-semantic problem, see (3), the Portuguese verb (*fazer*) and its complement (*leilão*) are translated into a verb (*to auction*) in English.

	Portuguese	English
(1)	*corretora*	firm of brokers
	parlamentares	members of parliament
	empreiteiras	Contract construction companies
	linhas aéreas	airlines
(2)	*peso Mexicano*	Mexican peso
	bolsa de Nyork	New York stock exchange
(3)	*fazer leilão*	to auction

Table 1: Example of some discrepancies

3 A Model Proposed and Implementation

Figure 1 illustrates the proposed model in which the structural divergence related information is modeled in two different dictionaries: a *structural dictionary* and a *bilingual dictionary*.

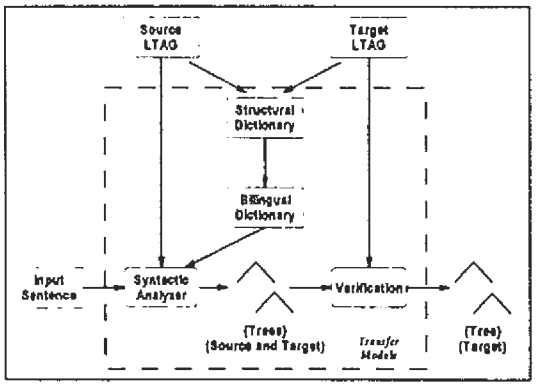


Figure 1: Overview of Proposed Model

The *structural dictionary* connects LTAGs structures that define both the source and the target languages. This structure maintains the node-to-node links between the source and the target grammars. All elementary trees in the source language are associated with trees in the target language. It has information about the inheritance of semantic attributes and also holds all the information for syntactic divergence resolution Table 2 illustrates some entries of the structural dictionary.

```
# format:
# ( source_id : target_id ) → [ links ]

# transitive verbs (NP object complement)
# S( N V N ) --> S( NP CV( V NP ) )
( 2:302) = [ $0:$0 , $1:$1 , $2:$3 , $3:$4 ] ;

# adjectives
# N( N Adj ) --> NP( Adj NP )
(100:400) = [ $0:$0 , $1:$2 , $2:$1 ] ;
# N( Adj N ) --> NP( Adj NP )
(101:400) = [ $0:$0 , $1:$1 , $2:$2 ] ;

# adjectival phrase
# +NPROP = Proper Noun
# N( N Prep N ) --> NP( Adj NP )
( 20:400) = [ $0:$0 , $1:$2 , $3[+NPROP]:$1 ] ;
```

Table 2: Selected Structural Dictionary Entries

The *bilingual dictionary* contains the rules for the resolution of lexical and lexical-semantic divergences. This dictionary manipulates the pairs of lexicalized items and points out one or more elementary structures of the structural dictionary to which the item is anchor. In this dictionary, derivation tree fragments can be defined, with the purpose of resolving lexical and lexical-semantic divergences. Furthermore, the dictionary can extend the rules contained in the structural dictionary to state the restrictions imposed by the accomplished lexical insertion. A fragment of the bilingual dictionary is presented in Table 3.

The transfer module receives a sequence of lexical items, generated by a lexicon-morphologic module. The output corresponds to one or more derivation trees in the target language with all structural modifications accomplished and *decorated* with the semantic features inherited from the source language.

Virtual grammars for source and target languages are described in an independent way and the notation introduced by [Kipper 94] for both grammars was used. The Portuguese grammar is a subset of the [Kipper 94] grammar and the English description was extracted from [Becker et al. 94].

```
# format:
# ( source_entry : translation ) = [ anchor list ];
(hoje : today)          = ;
(fazer : make)          = ; [2];

% (fazer : ##)  // redefinition of verb fazer
% S(N V[#lex=fazer] N(N[#lex=leilao] Prep N))
%                →
%   ( S ( NP CV ( V[#lex=auction] NP ) ) :
%     [ $0:$0, $1:$1, $2:$2, $6:$4];

# default
%(#lex : #lex)
```

Table 3: Selected Bilingual Dictionary Entries

Due to the incremental characteristic of the STAGs method, transfer functions were incorporated to syntactic analysis. The implementation involves two distinct steps: syntactic analysis (parser) and verification.

The *parser* uses a top-down algorithm for LTAG recognition. Each operation carried out by the parser in the SL enables one or more operations in the TL. The output is: for each SL syntactic structure a set of structures in the TL is generated.

During the process of analysis and translation, two types of attributes are manipulated: structural and semantic attributes. The structural attributes are inherent to each language and do not need to be transferred. On the other hand, semantic attributes are inherited by each one of the accomplished items of the pairs in lexicalized trees.

Finally, in the *verification step*, the unification of semantic features and the verification for structural consistency on the generated target trees is carried out. This process is based on the target LTAG grammar and inconsistent trees are discarded.

4 Conclusion and Future Work

This work investigated the use of the STAGs formalism for the treatment of lexical, syntactic and lexical-semantic divergences defined from a corpus in the field of Economics. Due to the extended domain of locality of LTAGs, it is possible to define regular correspondences among complex structures without the need of intermediary representations.

Although it was possible to set the translation rules for about 85% of the selected corpus (composed of 90 sentences), the model cannot yet be validated due to the short number of sample sentences.

Nowadays, we are starting to work on tagging and aligning tools for a bilingual corpus. These tools will allow us to set a more complex corpus of sentences to validate the work we have developed.

References

[Agustini 97] AGUSTINI, A. *Experiência de Utilização do Formalismo STAGs para a Construção de um Módulo de Transferência Estrutural*. Master's Degree in Informatics, PUCRS, RS, Brazil, Mater's Dissertation, 1997.

[Abeillé 90] ABEILLE, A. et. al. Using Lexicalized TAGs for Machine Translation. In *Proceedings of the 13th International Conference on Computational Linguistics*, Helsinki, Finland, 1990.

[Beckeret et al. 94] BECKER, T. et al. A Lexicalized TAG for English. University of Pennsylvania, *Technical Report*, 1994.

[Dorr 94] DORR, B. J. Machine Translation Divergences: A Formal Description and Proposed Solution. *Computational Linguistics*, v. 20, n. 4, 1994.

[Hutchins & Somers 92] HUTCHINS, W. J. & SOMERS, H. L. *An Introduction to Machine Translation*. Academic Press, Great Britain, 1992.

[Kipper 94] KIPPER, K. *Uma Experiência de Utilização do Formalismo de Gramáticas de Adjunção de Árvores para a Língua Portuguesa*. Master's Degree in Informatics, CPGCC-UFRGS, Porto Alegre, RS, Brazil, Mater's Dissertation, 1994.

[Rambow & Satta 96] RAMBOW, O. & SATTA, G. Synchronous Models of Language. In *Proceedings of the 19th International Conference on Computational Linguistics*, California, USA, 1996.

[Shieber & Shabes 90] SHIEBER, S. M. & SHABES, Y. Synchronous Tree-Adjoining Grammars. In *Proceedings of the 13th International Conference on Computational Linguistics*, Helsinki, Finland, 1990.

[Trujillo 95] TRUJILLO, A. Bi-Lexical Rules for Multi-Lexeme Translation in Lexicalist MT. *MI-95*, Leuven, Belgium, 1995.

Transplanting Supertags from English to Spanish

Srinivas Bangalore
AT&T Labs–Research
180 Park Avenue
Florham Park, NJ09732
srini@research.att.com

Abstract

In this paper, we present an approach to quickly develop supertags for a target language given supertags for another language (reference language), along with a sentence-aligned parallel corpus between reference language and target language pairs. Our method can be interpreted as composing the alignment relation with dependency relation of the reference sentence to obtain the dependency relation for the target sentence. This dependency relation is then used to induce the supertags for the target words.

1 Introduction

Supertags localize lexical and structural ambiguity by associating rich and complex descriptions to words of a language. This localization allows us to compute lexical and contextual distributional properties of supertags. In earlier work (JS94; Sri97a; Sri97b) we have shown that this distributional information can be used in a novel way to perform *almost parsing*. Trained on a million words of correctly supertagged Wall Street Journal Text, a simple trigram based supertagger assigns the same supertags to 92% of the words as they would have been assigned in the intended parse of a sentence. In subsequent work we have demonstrated the utility of supertags in a variety of applications including, Language Modeling (Sri96), Information Filtering (CS97b; CS97c), Information Extraction (DNB+97) and Sentence Simplification (CS97a).

2 An issue in Supertagging approach

However, constructing a rich repertoire of supertags for a language is a time consuming and tedious task as exemplified by the history of development of the English XTAG Grammar (XTA95) at University of Pennsylvania and the French XTAG Grammar at University of Paris.[1] In this paper, our attempt is to provide a solution to alleviate the task of building a supertag collection for a language (*target language*) based on the set of supertags of another language (*reference language*). In particular, we present a method of transplanting the set of supertags from the XTAG Grammar for English to Spanish using a parallel corpus of sentence-aligned English-Spanish sentences.

3 Grammar Induction vs Grammar Transplantation

Previous proposals (Res92; Sch92) for learning LTAG grammars involved inducing elementary trees from unannotated corpora. However, these proposals require training of a large number of parameters on even larger collections of corpora and yet the resulting structures may not be linguistically motivated. In contrast, our approach is based on the premise that elementary trees of natural language grammars are related and that these structures can be inherited *almost* as is, from the reference language to the target language. We use the term *grammar transplantation* as opposed to *grammar induction* in order to differentiate the amount effort involved in the development of supertags for the target language. However, a limitation of our approach is that the target language is imposed with structures that closely resemble the source language structure.

4 Methodology

Our approach to transplanting supertags involves applying the following steps to each sen-

[1] But this should not be regarded as a limitation exclusively of the supertag-based parsing paradigm. Treebank-based statistical parsing methods are limited by the effort involved in constructing a treebank.

tence pair in the reference-target parallel corpus. We have applied this method to an English-Spanish ATIS corpus.

- We first obtain a word alignment for each sentence pair using the alignment algorithm described in (ABD98). The alignment algorithm is completely unsupervised and only requires a sentence aligned corpus in two languages. It uses a correlation metric among reference-target word-pairs as a cost of reference-target word pairing and performs an alignment search that minimizes the sum of the costs of a set of pairings which map the reference sentence to its target sentence.
- The words of the English sentence are supertagged using a supertagger. The supertagger used for the ATIS domain was trained on 2000 word-supertag pairs and performs at 92% accuracy on a 500 word test set.
- The supertagged English sentence is further annotated with dependency links using the Lightweight Dependency Analyzer described in (Sri97b).
- The dependency links are then migrated to the target sentence as follows: if words w_i and w_j are linked in the reference sentence, w_i is aligned with v_p and w_j is aligned with v_q, then a dependency link is posited between v_p and v_q.
- Finally, the dependency structure migrated on to the target sentence is used to recover the correct ordering of arguments of each word. This information is used to construct the supertag for the word.

Our method can be interpreted as composing the alignment relation with dependency relation of the reference sentence to obtain the dependency relation for the target sentence. This dependency relation is then used to induce the supertags for the target words.

5 Example

Consider the following pair of sentences from the sentence-aligned English-Spanish ATIS corpus.

English: SHOW BUSINESS CLASS FARES ON U S AIR FROM BOSTON TO TORONTO

Spanish: MUESTRE LAS TARIFAS EN CLASE DE NEGOCIOS EN U S AIR DE BOSTON A TORONTO

The result of the alignment algorithm is shown below. Notice that the result contains alignments between one word in the source string (FARES) to two words in the target string (LAS:TARIFAS). Multi-word alignments are shown separated by a ":". The alignment algorithm allows mapping between at most two words in the source string to two words in the target string.

English: SHOW BUSINESS CLASS FARES ON U S AIR FROM BOSTON TO TORONTO

Spanish: MUESTRE LAS:TARIFAS EN CLASE DE NEGOCIOS EN U S AIR DE BOSTON A TORONTO

Target Position	Source Position
1	1
2 3	4
4	
5	3
6	
7	2
8	5
9	6
10	7
11	8
12	9
13	10
14	11
15	12

The output of the supertagger for the English string is in Table 1. The supertagger assigns to each word the part-of-speech and supertag information. The supertag information is used to assign dependency information among the words of the sentence.

The POS, supertags and dependency links are transplanted on to the target string using the

Position	Words	POS	Supertag	Dependency links
1	SHOW	VB	A_Inx0Vnx1	4.
2	BUSINESS	NN	B_Nn	3*
3	CLASS	NN	B_Nn	4*
4	FARES	NNS	A_NXN	
5	ON	IN	B_nxPnx	4* 8.
6	U	NNP	B_Nn	7*
7	S	NNP	B_Nn	8*
8	AIR	NNP	A_NXN	
9	FROM	IN	B_nxPnx	8* 10.
10	BOSTON	NNP	A_NXN	
11	TO	IN	B_nxPnx	8* 12.
12	TORONTO	NNP	A_NXN	

Table 1: Result of applying the supertagger and the LDA on the English string

Position	Words	POS	Supertag	Dependency links
1	MUESTRE	NN	A_Inx0Vnx1	2:3.
2:3	LAS:TARIFAS	NNS	A_NXN	
4	EN			
5	CLASE	NN	B_Nn	2:3*
6	DE			
7	NEGOCIOS	NN	B_Nn	4*
8	EN	IN	B_nxPnx	2:3* 11.
9	U	NNP	B_Nn	10*
10	S	NNP	B_Nn	11*
11	AIR	NNP	A_NXN	
12	DE	IN	B_nxPnx	11* 13.
13	BOSTON	NNP	A_NXN	
14	A	TO	B_nxPnx	11* 15.
15	TORONTO	NNP	A_NXN	

Table 2: Result of combining the alignment information with the dependency information

alignment information and the result is in Table 2.

The target string dependency structure is examined for *completeness* and *consistency*. Completeness requires that each word is assigned a supertag and its dependency requirements are satisfied. Consistency requires that the direction of the head/dependent of a given word matches the direction of its dependency requirement.

In our example, the words at positions 4 and 6 are not assigned any supertags and hence violate completeness constraint and the words at positions 5 and 7 violate consistency constraints since the supertag (B_Nn) requires the head to appear to its right while the head appears on the left.

We solve the consistency and completeness problems by assigning to a word the most frequent supertag it is associated with, given the entire corpus, which can fit into the dependency context of the target string and at the same time respect the dependency constraints imposed by the source language. The corrected POS, supertag and dependency structure for the target string is shown in Table 3.

6 Evaluation

The system can be evaluated in a number of ways: in the context of an application, in terms of the supertags assigned, in terms of the dependency links assigned or in terms of time reduced in developing a full-fledged domain independent grammar. We are in the process of evaluating the system on its performance in assigning

Position	Words	POS	Supertag	Dependency links
1	MUESTRE	NN	A_Inx0Vnx1	2:3.
2:3	LAS:TARIFAS	NNS	A_NXN	
4	EN	IN	B_nxPnx	2:3* 5.
5	CLASE	NN	A_NXN	
6	DE	IN	B_nxPnx	5* 7.
7	NEGOCIOS	NN	A_NXN	
8	EN	IN	B_nxPnx	2:3* 11.
9	U	NNP	B_Nn	10*
10	S	NNP	B_Nn	11*
11	AIR	NNP	A_NXN	
12	DE	IN	B_nxPnx	11* 13.
13	BOSTON	NNP	A_NXN	
14	A	TO	B_nxPnx	11* 15.
15	TORONTO	NNP	A_NXN	

Table 3: Result of correcting the dependency structure based on completeness and consistency constraints.

supertags and dependency links to 1000 words of annotated test corpus from the ATIS domain. Preliminary results suggest that the performance in assigning supertags is about 80% accurate.

References

Hiyan Alshawi, Srinivas Bangalore, and Shona Douglas. Automatic acquisition of hierarchical transduction models for machine translation. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics*, Montreal, Canada, 1998.

R. Chandrasekar and B. Srinivas. Automatic induction of rules for text simplification. *Knowledge-based Systems*, 10:183–190, 1997.

R. Chandrasekar and B. Srinivas. Gleaning information from the web: Using syntax to filter out irrelevant information. In *Proceedings of AAAI 1997 Spring Symposium on NLP on the World Wide Web*, 1997.

R. Chandrasekar and B. Srinivas. Using syntactic information in document filtering: A comparative study of part-of-speech tagging and supertagging. In *Proceedings of RIAO'97*, Montreal, June 1997.

Christine Doran, Michael Niv, Breckenridge Baldwin, Jeffrey Reynar, and B. Srinivas. Mother of Perl: A Multi-tier Pattern Description Language. In *Proceedings of the International Workshop on Lexically Driven Information Extraction*, Frascati, Italy, July 1997.

Aravind K. Joshi and B. Srinivas. Disambiguation of Super Parts of Speech (or Supertags): Almost Parsing. In *Proceedings of the 17th International Conference on Computational Linguistics (COLING '94)*, Kyoto, Japan, August 1994.

Philip Resnik. Probabilistic tree-adjoining grammar as a framework for statistical natural language processing. In *Proceedings of the Fourteenth International Conference on Computational Linguistics (COLING '92)*, Nantes, France, July 1992.

Yves Schabes. Stochastic lexicalized tree-adjoining grammars. In *Proceedings of the Fourteenth International Conference on Computational Linguistics (COLING '92)*, Nantes, France, July 1992.

B. Srinivas. "Almost Parsing" Technique for Language Modeling. In *Proceedings of ICSLP96 Conference*, Philadelphia, USA, 1996.

B. Srinivas. *Complexity of Lexical Descriptions and its Relevance to Partial Parsing.* PhD thesis, University of Pennsylvania, Philadelphia, PA, August 1997.

B. Srinivas. Performance Evaluation of Supertagging for Partial Parsing. In *Proceedings of Fifth International Workshop on Parsing Technology*, Boston, USA, September 1997.

The XTAG-Group. A Lexicalized Tree Adjoining Grammar for English. Technical Report IRCS 95-03, University of Pennsylvania, 1995. Updated version available at http://www.cis.upenn.edu/ xtag/tr/techreport.html.

Recursive Matrix Systems (RMS) and TAG

Tilman Becker
DFKI GmbH
Stuhlsatzenhausweg 3
D-66123 Saarbrücken
becker@dfki.de

Dominik Heckmann
Universität des Saarlandes
D-66123 Saarbrücken
dheck@studcs.uni-sb.de

Introduction

We define Recursive Matrix Systems (RMS), a highly parameterizable formalism that allows for a clear separation of various kinds of recursion. One instance of RMS, namely context–free RMS with two rows and a specific reading interpretation turns out to be weakly equivalent to TAG. This allows for the transfer of results from TAGs to this class of RMS. Furthermore, the equivalence proof is constructive and exhibits a very close relationship between the structures of the two formalism, namely trees and matrices. This allows to transfer interesting restrictions which can easily be defined in RMS to TAG. In particular, the obvious restriction of context–free RMS to regular RMS results in a restricted form of TAG which appears sufficient for natural language processing, albeit being less complex than regular TAG.

Recursive Matrix Systems

A *Recursive Matrix* is a finite matrix whose elements are either terminal symbols or again recursive matrices (see Figure 1). Recursive matrices are created by grammars (in particular by regular and context–free grammars) that have vectors as their terminal symbols. Strings are derived from a recursive matrix by a *reading interpretation* which reads the terminal symbols of a matrix line-by-line either from left-to-right or right-to-left and recursively descends for elements that are recursive matrices. In the following, we consider only Recursive Matrices with a constant number of rows in all (sub-) matrices. This number n is an important parameter. We denote the set of all recursive matrices as RM.

a	b	c	ε
a	b	c	d e f d e f d e f
d	ε	a	c

In this example the element in the second row, fourth column is the recursive (sub-) matrix

d	e	f
d	e	f
d	e	f

. The other elements are terminals.

Figure 1: A recursive matrix.

A *regular (context–free) Recursive Matrix System (reg-RMS, cf-RMS)* is a tuple $\langle G,I \rangle$ where G is a grammar that generates recursive matrices and I is an interpretation to read a string from each recursive matrix. $L(G)$ is the set of all recursive matrices derived by the grammar G. $L(G,I)$ is the set of all strings derived from the recursive matrices in $L(G)$ by the interpretation I.

A regular (context–free) grammar G that generates recursive matrices is a grammar with terminal symbols Vec[1], nonterminals N, a start symbol S from N and a set P of regular (context–free) rules. All vectors $v \in Vec$ have constant size n; the elements of v are either symbols from a set T (these are called the terminal symbols of the RMS) or non-terminals from N. T, V, N, P are finite but non–empty sets, $N \cap T = \emptyset$.

The derivation relation $\Rightarrow$ is defined over *Extended Recursive Matrices*, i.e., concatenations

[1] not to be confused with T, the terminal symbols of the RMS.

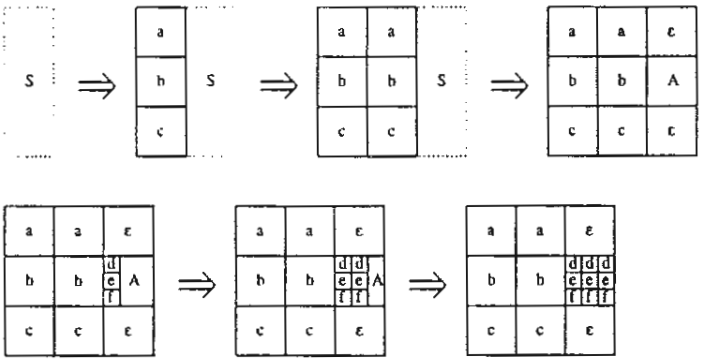


Figure 2: A derivation with RMS grammar G_1.

of vectors and non-terminals, where the elements of a vector are either terminal-symbols of the RMS, non-terminals of G or Extended Recursive Matrices. Each derivation step rewrites exactly one non-terminal according to a rule in P. The language $L(G)$ is defined as $L(G) := \{r | S \stackrel{*}{\Rightarrow} r, r \in RM\}$.

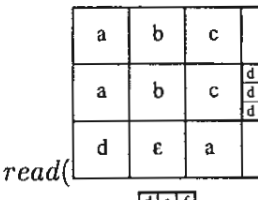
The following example grammar is used to show the derivation process:

$$G_1 = \langle\ T=\{a,b,c,d,e,f\}, N=\{S,A\}, S, P=\{S \rightarrow \begin{bmatrix} a \\ b \\ c \end{bmatrix} S\ ,\ S \rightarrow \begin{bmatrix} \epsilon \\ A \\ \epsilon \end{bmatrix},\ A \rightarrow \begin{bmatrix} d \\ e \\ f \end{bmatrix} A,\ A \rightarrow \begin{bmatrix} d \\ e \\ f \end{bmatrix} \}\rangle$$

All vectors have the size 3 and all rules are regular. G_1 is a reg-RMS. When applying the first or third rule, a vector is added to the matrix. When applying the second rule, a descend into the next recursive "matrix-level" takes place. Only the last rule is a terminating one. A possible derivation with the grammar G_1 is shown in figure 2. Note that the horizontal dimension of the recursive matrices is unbound.

The *reading interpretation* of a recursive matrix is derived from a vector of directions for each row of the matrix, i.e., an n-dimensional vector $I = \begin{bmatrix} i_1 \\ \vdots \\ i_k \end{bmatrix}$ of elements $i_j \in \{\rightarrow, \leftarrow\}$. It is recursively defined as shown in figure 3.

For example, with $I = \begin{bmatrix} \rightleftarrows \\ \rightleftarrows \end{bmatrix}$, we get



$read(\quad) =$

$abc \circ read(\quad) \circ cba \circ dac =$

$abc \circ def \circ fed \circ def \circ cba \circ dac =$

$abcdeffeddefcbadac.$

The Equivalence of CF-RMS$_{\rightleftarrows}$ and TAG

Although a TAG can be directly transformed into a weakly equivalent RMS, it is easier to demonstrate if we assume a normal form for TAG where no adjunction is possible into root and foot nodes, the root node has only one daughter, and there are no more than two inner nodes dominating the foot node. Figure 4 shows how such an auxiliary tree β can be directly mapped into a rule P of a context-free RMS$_{\rightleftarrows}$. The details for mapping the subtrees s, t, u, v to submatrices of the right-handside of P are omitted here.

Note the close resemblance of the notation of a TAG as an RMS to the notation of a TAG as a Linear Context-Free Rewriting System (LCFRS, Weir 1988). Even though in general, RMS can be captured as LCFRS, the particular structure of RMS which separates different dimensions of recursion has lead us to a number of observations which are not obvious

$$
\begin{aligned}
read(recursive\ matrix, I) &:= read(row_1, i_1) \circ \ldots \circ read(row_k, i_k) \\
read(row[1..m], \rightarrow) &:= read(row[1], I) \circ read(row[2..m], \rightarrow) \\
read(row[1..m], \leftarrow) &:= read(row[m], I) \circ read(row[1..m-1], \rightarrow) \\
read(terminal\ symbol, I) &:= terminal\ symbol
\end{aligned}
$$

Figure 3: Definition of the reading interpretation *read* for *recursive matrix* $= \begin{bmatrix} row_1 \\ \vdots \\ row_k \end{bmatrix}$.

when looking at TAGs or even at LCFRS.

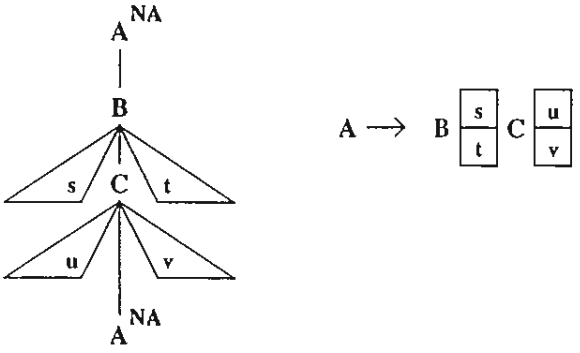


Figure 4: Tranforming a TAG into a weakly equivalent RMS.

Like contect–free grammars, context–free RMS can be transformed into a normal form resembling Chomsky normal form. In such a transformed cf-RMS$\rightleftarrows$, all rules are of the form shown in figure 5.

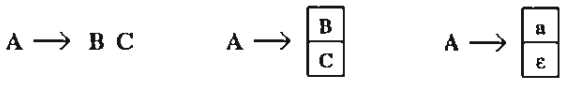


Figure 5: A normal form for cf-RMS$\rightleftarrows$.

Figure 6 sketches how a TAG grammar is constructed from such a cf-RMS that derives the same language.

Given this relation, the question arises whether a TAG can be transformed into a *regular* RMS, i.e., whether the non-terminal B in Figure 4 can be dropped. The answer is no, and it can be seen, e.g., by the fact that the normal form transformation cannot be tightend up to only one inner node dominating the foot node. This implies that regular RMS are a proper subset of context–free RMS[2].

[2]Actually, we found this relation when failing to show the equivalence of regular RMS and TAG, forcing us to extend RMS to context–free RMS.

On the other hand, this emphasizes a parameter of TAGs that was not obvious before: Even though the well known example grammars for deriving $L^4 = \{a^n b^n c^n d^n\}$ and $L^{copy} = \{ww | w \in \{a,b\}^*\}$ already exhibit non context–free properties and even cross–serial dependencies, they are restricted in the sense that their trees have only one node dominating the foot node that is available for adjunction. While it is not easy to give an example for the effects that can be achieved with two or more such nodes, when looking at RMS, this parameter becomes obvious (i.e. as the difference between regular or context–free RMS).

Looking at natural languages, it appears that in fact the restriction to TAG with only one adjunction node on the spine (an important restriction of regular RMS) are sufficient since recursive, unbounded dependencies are restricted to one type (e.g., either embedded or cross–serial), but don't occur intertwined with a second type of recursive, unbounded dependencies.

It remains unclear though, whether the second restriction of regular RMS, which in TAG terms means that *no* path from the root to a leaf can have more than one available adjunction node is too strong.

Current Work

We are currently exploring the consequences of the restrictions that reg-RMS have compared to CF-RMS. Exploiting the equivalence of TAGs and RMS allows us to adopt results for TAGs for RMS. A point of special interest is parsing and its time complexity. Taking any of the various known parsing algorithms for TAGs immediately gives us an $O(n^6)$ parsing algorithm

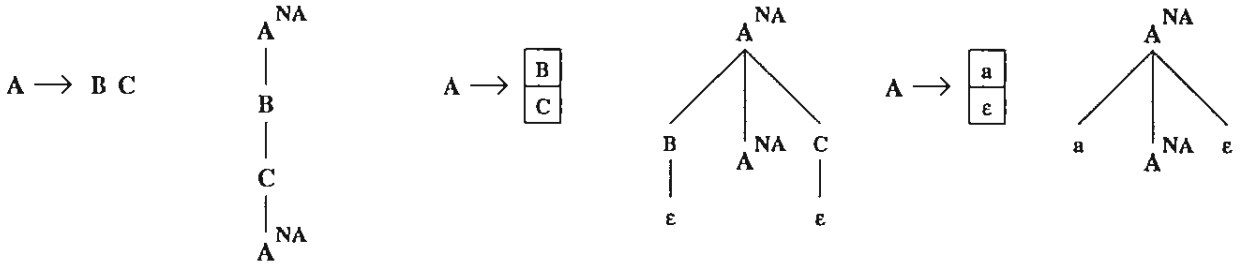


Figure 6: Elementary trees constructed for each rule of a cf-RMS$_{\rightleftarrows}$ in normal form.

for CF-RMS$_{\rightleftarrows}$. Moreover, given the tight coupling between the grammar rules of an RMS and the elementary trees of the equivalent TAG, we can find stronger restrictions on the steps of the TAG parser if the original RMS grammar is regular and not context-free. In particular, using the algorithm by (Nederhof 1997), we conjecture that reg-RMS can be parsed in at most $O(n^5)$ time.

A further avenue of research is the fact that the context-freeness of RMS is not necessary to construct grammars that exhibit cross-serial dependencies, one of the core arguments for TAGs. While 2-dimensional reg-RMS with a reading interpretation of $\rightrightarrows$ ($\rightleftarrows$) are sufficent to exhibit cross-serial dependencies (center-embedded dependencies resp.), they can't exhibit both. However, 3-dimensional reg-RMS are sufficient and therefore a candidate for a further restriction on TAGs for natural language processing which might result in a further reduction of the time complexity of parsing. While such a restriction might not be obvious when looking at TAG trees, the representation as an RMS allows for a very succint formulation.

Bibliography

Heckmann, Dominik. *Recursive Matrix Systems.* In Proceedings of the Student Session at the 11th ESSLLI, Saarbrücken, Germany, 1998.

Nederhof, Mark-Jan. *Solving the Correct-prefix Property for TAGs.* In Proceedings of MOL5, Saarbrücken, Germany, 1997. Available as DFKI document D-97-02.
`http://www.dfki.uni-kl.de/~dfkidok/publications/D/97/02/abstract.html`

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

Motion Verbs and Semantic Features in TAG*

Tonia Bleam, Martha Palmer, K. Vijay-Shanker
Institute for Research in Cognitive Science
University of Pennsylvania
400A, 3401 Walnut St.
Philadelphia, PA 19104
tbleam, mpalmer, vshanker@linc.cis.upenn.edu

1 Introduction

We show how the domain of locality in a TAG elementary tree, (Frank 1992), can be extended through adjunction to include optional arguments for a class of motion verbs and how the adjunctions can be restricted appropriately through the use of semantic features. Some examples of motion verbs we consider are shown in Table 1, which categorizes the verbs according to Levin classes (Levin 1993). Note that we are using a broader definition of "motion" verbs than Levin's class 51.

VIDMs	Roll	Run	Force	Carry
arrive	float	jump	press	carry
enter	roll	run	pull	lug
escape	slide	slide	push	pull
exit	rotate	walk		push
	turn			

Table 1: Levin Classes of Verbs Involving Motion

These verbs are classified according to their syntactic behavior, which is taken to be a reflection of their underlying semantic properties. Motion verbs are able to occur with path phrases, where the term "path" is used as a cover term for *source*, *goal*, *via* and directional modifiers (PPs and adverbs), along the lines of Jackendoff (1976, 1990). Examples of these are given in (1-4).

2 Manner of motion verbs: (Run and Roll classes)

(1) I ran to the store. (goal)
(2) I ran from the room. (source)
(3) I slid the sleeve over the valve. (via)
(4) I slid the coupling nut forward. (direction)

We analyze manner of motion verbs as having the feature [eventType: motion:+]. Path phrases are constrained to only adjoin onto motion-compatible VPs.

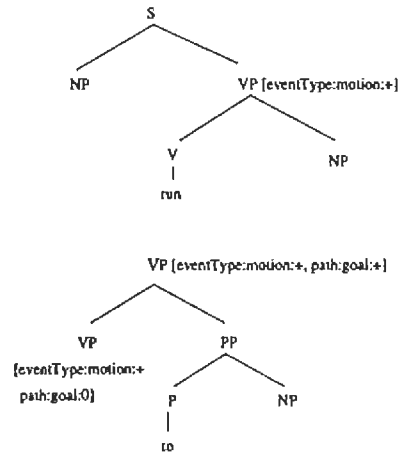


Figure 1: Elementary Tree for *run* and Auxiliary Tree for *to* PP

3 Explanation of Features

Within the feature structure that we propose there are several features whose values are not atomic, rather the feature contains another complex feature structure. For example, the feature [eventType] can be multi-valued. Possible features within [eventType] are [motion], [force], and [contact]. Similarly, path is composed of a complex feature structure which has the features [via], [direction], [source], and [goal]. The path features can take the values +,

* We would like to thank Hoa Trang Dang, Christy Doran, Aravind Joshi, Tony Kroch, Jeff Lidz, Joseph Rozenzweig, Matthew Stone, and two anonymous reviewers for helpful discussion and/or participation in this research.

0, or NONE. [1] A "+" value means that the feature has been specified. A "0" value means that it has not yet been specified, but that it is appropriate for this feature to have a value. The feature [path: goal:0] or [path: source:0] occurs in the foot node of adjoining trees that represent source or goal, to ensure that an element with that value has not already been adjoined. An example is shown in Figure 1 above. The value "NONE", on the other hand, means that it is not appropriate to specify this value.

EventType features are also atomically valued, taking the values + or −. Having the feature [eventType: motion:−] means that the event is unable to be interpreted as a motion event and entails that path phrases cannot adjoin on. An example of a verb with this feature might be *eat*. On the other hand, non-specification of the [eventType: motion] feature entails that path phrases can adjoin. If a path phrase does adjoin, the event becomes a motion event. Sound emission verbs are of this sort.

4 Verbs of Inherently Directed Motion

The class of verbs of inherently directed motion (VIDMs) have a path component built into the meaning of the verb. Usually the verb specifies a source, as in *leave* and *exit*, or a goal, as in *enter*, *arrive*.

One interesting property of VIDMs is that they have a more limited ability to take path PPs even though they are motion verbs. For example, *arrive* does not take a prototypical goal PP (with the preposition *to*), but instead takes a locative PP which represents the goal of motion.

(5) a. Mary arrived at the station.
b. *Mary arrived to the station.

(6) arrive = [GO [TO X]]
(where X=location)

Following Jackendoff 1990, we analyze the goal function "TO" as being incorporated in the LCS of *arrive*, shown in (6). The PP slot in the subcategorization frame is coindexed with the location argument slot X. Therefore, the PP that represents the goal must be a location. In TAG terms, we assume that the part of the path inherently specified in the verb semantics constitutes an (optional) argument. In order to constrain what kind of preposition can instantiate the goal, we will need to define a class of locative prepositions and impose a constraint on the P node so that only this class is allowed to occur there. For now, we show the feature [locative:+] on the P node of the elementary tree for *arrive* in Figure 2.

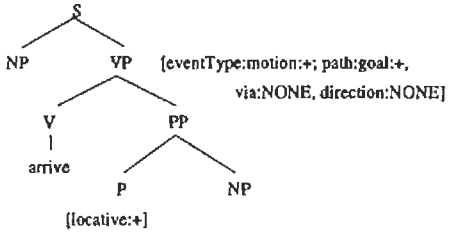


Figure 2: Elementary Tree for *arrive*

In addition, many VIDMs (like *arrive, enter, exit*) are achievement verbs; that is, they have no durative aspect. Because of this, they cannot take a path phrase that modifies durative motion.

(7) John arrived (*around the lake) at Mary's house.

The [via:NONE] and [direction:NONE] features in the VP node in Figure 2 represent a non-durative path. While via and direction PPs cannot occur with *arrive*, a source can be specified, as shown in (8), because this does not conflict with the lack of durativity of the event.

(8) John arrived in Chicago from Philadelphia.

5 Regular sense extensions

Path phrases can adjoin to a VP node which is unspecified for motion. Even verbs that are not inherently motion verbs can be modified by path phrases, augmenting their semantic representation to include explicit motion. For instance, verbs of sound emission such as *whistle* and *roar* can convey directed motion when they appear with path phrases, as in (9) and (10).

(9) The train whistled into the station.

[1] We use atomically valued features for source and goal rather than putting in the actual value of the goal (i.e. the referent of the goal) because the simple presence or absence of these features is what affects the derivation. That is, having a goal present means that another goal cannot adjoin on (but see footnote 3). For this purpose, the referent of the goal does not need to be represented.

(10) The truck roared past the weigh station.

Additionally, we see other cases where the syntactic frame in which a verb occurs determines the senses that a verb can have. For example, *push* can have the senses shown in (11–14). (See Dang et al. 1998 for discussion).

(11) Mary pushed the chair. [force:+, contact:+]
(12) Mary pushed the cart to the store. [motion:+, path:+]
(13) Mary pushed the branches apart. [motion:+, separation:+]
(14) Mary pushed at the boulder. [motion:–]

The transitive sentences (11), (12), and (13) will all be generated from a transitive elementary tree where the VP node has the features [force:+] and [contact:+], but is unspecified for [motion]. Adjoining in the modifiers *to the store* and *apart* will introduce the additional features listed in (12) and (13), respectively.

The conative construction (illustrated in (14)) is represented by the elementary tree given in Figure 3.

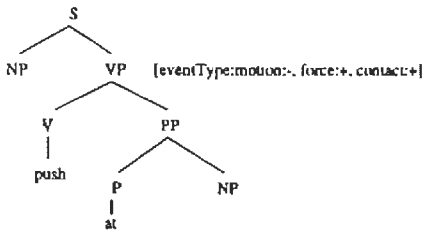


Figure 3: Elementary Tree for Conative Construction

6 Tree Families and Optional Arguments

Implicit in our discussion of VIDMs and regular sense extensions above is the assumption that some PPs are arguments of the verbs they occur with, and hence are present in the verb's elementary tree. The cases in question are (1) the PP which represents the inherently specified path of a VIDM; and (2) the *at* PP of the conative construction.

6.1 Optional arguments of Verbs of Inherently Directed Motion

The first case is represented by the following example, where *at the station* represents the goal that is implicit in the meaning of *arrive*.

(15) The train arrived at the station.

Note that the meaning of (15) is not compositional since *at the station* by itself or combined with a motion verb like *run* can only mean a location of the event.[2] It cannot represent the goal of motion in these cases.

(16) The athlete ran at the gym.
(17) I saw Mary at the station.

It is only with a verb whose meaning includes [goal:+], that an *at*-PP or any other locative PP can represent a goal. Thus, in this example, it is the head verb which determines the role that the PP phrase has in the clause. This kind of idiosyncracy is evidence that a constituent is an argument rather than an adjunct (see e.g., Pollard and Sag 1987; Marantz 1984). By this criterion, then, the PP representing an inherent role of a VIDM should be considered an argument, and thus, should be present in the elementary tree.

It has been noted that all source and goal PPs simultaneously show both argument and adjunct properties. Larson (1988) discusses the argument status of the source and goal phrases in sentences like (18) and (19).

(18) John walked to the store.
(19) Mary ran from the house.

They act like adjuncts in being optional, but like arguments in being non-iterable. (The following examples are Larson's.)[3]

(20) * John flew to New York to Kennedy Int'l Airport.
(21) * Max got a letter from Felix from his friend.

[2] It can also have the meaning of *towards*.

[3] Note that (20) isn't that bad if the second PP is interpreted as a further specification of the goal location.

(1) ? I ran to Philadelphia to IRCS.

We do not yet have an account of this phenomenon, but we do not take it as counterevidence to the generalization that only one goal may be given per event. This is unlike true modifiers like PPs of location, of which more than one can be given without any restriction:

(2) I hid in the building on the third floor in a classroom under a desk.

Jackendoff (1976) takes motion verbs to contain the abstract predicate GO which is a three-place relation, taking the arguments (x,y,z), where x is an element that moves from y (source) to z (goal).

For current purposes, however, we do not take all sources and goals to be present in the elementary tree. Only PPs whose meaning is implicit in the meaning of the verb itself are present in the elementary tree, whereas all other PPs are adjoined. This is in contrast with the analysis provided by Levin and Rappaport Hovav (1995) in which all sources and goals are treated as arguments as a result of a lexical rule that applies to verbs of motion.

6.2 The Conative Construction and Elementary Trees

The other case to consider is the conative *at* construction, shown in (22).

(22) The child hit at the ball.

We assume that the conative *at* PP is present in the elementary tree. If we took the *at* PP to be adjoined in, then an intranstive elementary tree for *hit* is required. However, *hit* can only occur transitively, and so we would need additional mechanisms for blocking the intransitive tree from ever occurring outside of the conative construction.

On the other hand, we could take the position that the noun phrase (*the ball* in (22)) is an argument of the verb, and *at* adjoins in. However, it is not possible for a PP to adjoin at this point.[4]

The conative is properly analyzed as a lexical process of object demotion – an operation that applies to the lexical representation of the verb, affecting its argument-structure. It demotes a direct object to be an oblique element with the effect that the object is interpreted as not affected by the action of the verb. However, in TAG, there is no level of representation independent of the elementary trees in which demotion operations of this sort could take place. Therefore, the best TAG analysis of the conative treats the PP as an argument, and hence, present in the elementary tree.

[4]It would only be possible for an NP to adjoin, requiring an analysis of the *at* PP as an NP, which is linguistically unmotivated.

7 Conclusion

The goal of our work is to capture lexical semantic properties that we hope will be helpful in reducing the search space in parsing, as well as aid in generation (SPUD; see Stone and Doran 1997; Stone and Webber 1998) and machine translation (in the transfer of lexical semantic properties) (see Palmer, et al. (to appear)).

We have examined several subclasses of motion verbs, and posited features to capture their semantic properties. These features not only allow us to place restrictions on the verbs to constrain possible derivations, but also allow us to account for regular sense extensions through the underspecification of certain features and by having modifiers introduce these features in the course of the derivation.

References

Dang, H. Trang, K. Kipper, M. Palmer, J. Rosenzweig. (1998). Investigating regular sense extensions based on intersective Levin classes. ACL

Frank, R., (1992). *Syntactic locality and Tree Adjoining Grammar: grammatical, acquisition and processing perspectives.* PhD Thesis, University of Pennsylvania, Technical Report IRCS-92-47.

Jackendoff, R. (1976). Toward an Explanatory Semantic Representation. *Linguistic Inquiry* 7, 89-150.

Jackendoff, R. (1990). *Semantic Structures.* MIT Press.

Larson, R. (1988). Implicit Arguments in Situation Semantics. *Linguistics and Philosophy* 11:169-201.

Levin, B. (1993). *English Verb Classes and Alternations.* University of Chicago Press.

Levin, B., and M. Rappaport Hovav. (1995). *Unaccusativity: At the Syntax-Lexical Semantics Interface.* MIT Press.

Palmer, M., J. Rosenzweig, W. Schuler. (to appear, 1998). Capturing Motion Verb Generalizations with Synchronous TAGs. in St. Dizier, Patrick, *Predicative Forms in NLP*, Kluwer Press.

Stone, M. and C. Doran. (1997). Sentence planning as description using tree-adjoining grammar. *Proceedings of ACL*, 198–205.

Stone, M. and B. Webber. (1998). Textual economy through close coupling of syntax and semantics. *International Natural Language Generation Workshop*

A Generalization of Mildly Context-Sensitive Formalisms

Pierre Boullier
INRIA-Rocquencourt
BP 105
78153 Le Chesnay Cedex, France
Pierre.Boullier@inria.fr

1 Introduction

In (Boullier 98), we presented range concatenation grammars (RCGs), a syntactic formalism which is a variant of literal movement grammars (LMGs), described in (Groenink 97), and which is also related to the framework of LFP developed by (Rounds 88). In fact it may be considered to lie halfway between their respective *string* and *integer* versions; RCGs retain from the string version of LMGs or LFPs the notion of concatenation, applying it to ranges rather than strings, and from their integer version the ability to handle only (part of) the source text. The basis of RCGs is the notion of *range*, a couple of integers $(i \,..\, j)$ which denotes the occurrence of some substring $a_{i+1} \ldots a_j$ in an input string $a_1 \ldots a_n$. Of course, only consecutive ranges can be concatenated into a new range[1]. This formalism, which extends CFGs, aims at being a convincing challenger as a syntactic base for various tasks, especially in natural language processing. We have shown that the positive version of RCGs, as simple LMGs or integer indexing LFPs, exactly covers the class *PTIME* of languages recognizable in deterministic polynomial time. Since the composition operations of RCGs are not restricted to be linear and non-erasing, its languages (RCLs) are not semi-linear. Therefore, RCGs are *not* mildly context-sensitive (Joshi, Vijay-Shanker, and Weir 91) and are more powerful than linear context-free rewriting systems (LCFRS) (Vijay-Shanker, Weir, and Joshi 87), while staying computationally tractable: its sentences can be parsed in polynomial time. However, our formalism shares with LCFRS the fact that derivations are context-free (i.e. the choice of the operation performed at each step only depends on the object to be derived from). As in the CF case, its derived trees can be packed into parse forests (Lang 94). Let ρ be a range. The nodes of a CFG parse forest are couples (A, ρ) while for an RCG they have the form $(A, \vec{\rho})$ where $\vec{\rho}$ is a vector (list) of ranges. Besides its power and efficiency, this formalism possesses many other attractive properties. RCLs are closed under intersection and complementation[2]. Since this closure property can be reached without changing the structure (grammar) of the constituents (i.e. we can get the intersection of two grammars G_1 and G_2 without changing neither G_1 nor G_2), it allows for a form of modularity which may lead to the design of libraries of reusable generic grammatical components. Moreover, like CFGs, this formalism can act as a syntactic backbone upon which decorations from other domains (probabilities, logical terms, feature structures) can be grafted, and last, in our opinion, it is very elegant and understandable.

2 RCGs

The rewrite rules $\psi_0 \rightarrow \psi_1 \ldots \psi_m$ of an RCG are called *clauses*. Each component $\psi_i = A(\alpha_1, \ldots, \alpha_p)$ is a *predicate*. Each argument α_i of a predicate is a string of terminal symbols

[1] Ranges can be generalized to denote couples of states in some FSA representing ill-formed, incomplete or ambiguous (multi tagged/multi part of speech or word lattice) input.

[2] The set $T^* - L$, complementary of L, is defined on the basis of "negation by failure" rules.

and *variables*. Variables and arguments in a clause are supposed to be bound to ranges by a substitution mechanism. An *instantiated clause* is a clause in which arguments and variables are consistently replaced by ranges; its components are *instantiated predicates*. For example, $A(\langle g \,..\, h\rangle, \langle i \,..\, j\rangle, \langle k \,..\, l\rangle) \rightarrow B(\langle g{+}1 \,..\, h\rangle, \langle i{+}1 \,..\, j{-}1\rangle, \langle k \,..\, l{-}1\rangle)$ is an instantiation of the clause $A(aX, bYc, Zd) \rightarrow B(X, Y, Z)$ if the source text $a_1 \ldots a_n$ is such that $a_{g+1} = a, a_{i+1} = b, a_j = c$ and $a_l = d$. A *derive* relation is defined on strings of instantiated predicates. If an instantiated predicate is the LHS of some instantiated clause, it can be replaced by the RHS of that instantiated clause. An input string $a_1 \ldots a_n$ is a sentence iff the empty string (of instantiated predicates) can be derived from $S(\langle 0 \,..\, n\rangle)$ where S is the start symbol. The arguments of predicates may denote discontinuous or even overlapping ranges. Fundamentally, a predicate A defines a notion (property, structure, dependency, ...) between its arguments whose ranges may be scattered over the source text. What is "between" its arguments is *not* the responsibility of A, and is described (if at all) somewhere else. RCGs are therefore well suited to describe long distance dependencies. Overlapping ranges are due to the non-linearity of the formalism. For example, the same variable may occur in different arguments in the RHS of some clause, expressing different views (properties) of the same portion of the source text.

As an example of an RCG, the following set of clauses describes the three-copy language $\{www \mid w \in \{a,b\}^*\}$ which is known to be beyond the formal power of TAGs.

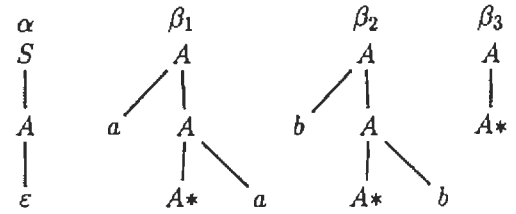
$$\text{(I)} \quad \begin{cases} S(XYZ) & \rightarrow & A(X,Y,Z) \\ A(aX,aY,aZ) & \rightarrow & A(X,Y,Z) \\ A(bX,bY,bZ) & \rightarrow & A(X,Y,Z) \\ A(\varepsilon,\varepsilon,\varepsilon) & \rightarrow & \varepsilon \end{cases}$$

3 RCGs & TAGs

Within the TAG formalism, if we consider an auxiliary tree τ and the way it evolves until no more adjunction/substitution is possible, we realize that some properties of the final tree are already known on τ. The yield derived by the part of τ to the left (resp. to the right) of its spine are contiguous and, the *left yield* (produced by the left part) lies to the left of the *right yield* in the input string. Thus, for any tree τ (initial or auxiliary) consider its m internal nodes where adjunction is allowed[3]. We decorate each such node with two variables L_i and R_i $(1 \leq i \leq m)$ which are supposed to capture respectively the left and right yield of this i^{th} node. The root and foot of auxiliary trees have no decoration. Each terminal leaf has a single decoration which is its terminal symbol or ε. Afterwards, we collect into a string d_τ the decorations gathered during a top-down left-to-right walk in τ. If τ is an auxiliary tree, let d^l_τ and d^r_τ be the part of d_τ gathered before and after the foot of τ has been hit. With each tree, we associate an RCG clause constructed as follows:

- Its LHS is the predicate $S(d_\tau)$ if τ is an initial tree (S is the start predicate).
- Its LHS is the predicate $A(d^l_\tau, d^r_\tau)$ if τ is an auxiliary A-tree.
- Its RHS is $\psi_1 \ldots \psi_m$ with $\psi_i = A_i(L_i, R_i)$ if A_i is the label of the i^{th} inside node.

For example, the following TAG



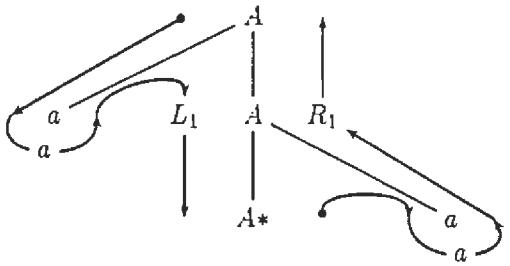
where α is the initial tree and β_1, β_2 and β_3 are the auxiliary trees[4], defines the language $\{ww \mid w \in \{a,b\}^*\}$, which is translated into the strongly equivalent RCG

$$\begin{array}{lcl} S(L_1R_1) & \rightarrow & A(L_1,R_1) \\ A(aL_1,aR_1) & \rightarrow & A(L_1,R_1) \\ A(bL_1,bR_1) & \rightarrow & A(L_1,R_1) \\ A(\varepsilon,\varepsilon) & \rightarrow & \varepsilon \end{array}$$

[3] In TAGs, we assumed that initial trees are all labeled by a unique start symbol, say S, which is not used somewhere else, that adjunction is not allowed at the root or at the foot of any auxiliary tree but is mandatory on inside nodes.

[4] Each foot is marked by an $*$.

As an example, the arguments of the LHS predicate of the second clause have been gathered during the following walk in β_1



We know (Vijay-Shanker and Weir 94) that TAGs, LIGs and HGs are three weakly equivalent formalisms though they appear to have quite different external forms. Groenink has shown that HGs can be translated into equivalent LMGs. We have shown that transformation from TAGs to RCGs also exists. In (Boullier 98) we have proposed a transformation from LIGs into equivalent RCGs. While the process involved to get an equivalent RCG for a TAG or an HG is rather straightforward, the equivalence proof for LIG is much more complex and relies upon our work described in (Boullier 96). This is due to the fact that an RCG is a purely syntactic formalism in the sense that it only handles (part of) the source text, exclusive of any other symbol. Therefore the stack symbols of LIGs have no direct equivalent in RCGs and the translation process needs to understand what the structural properties induced by these stack symbols are. An interesting property of all these translations is that the power of RCGs comes for free. In particular, if the input TAG or LIG is in some normal form[5], the corresponding RCG can be parsed in $\mathcal{O}(n^6)$ time at worst. Moreover, in RCGs, the incidence of each clause on the total parsing time can be isolated. Of course, complicated clauses induce high polynomial exponents. If we look at the clauses generated by the translation, some are simple, and few (if any) are complicated (and therefore induce an exponent of 6). In fact these translations bring new insight and help to understand why and at which point the maximum complexity is introduced.

[5]Auxiliary trees in TAGS are such that there are at most two internal nodes where the adjunct operation can take place or the number of objects in the right-hand side of LIG rules is at most two.

4 RCGs & RNRGs

Ranked node rewriting grammars (RNRGs) (Abe and Mamitsuka 97) are an extension of TAGs. They are used to predict the protein secondary structure from their amino acid sequence patterns. These secondary structures, the so-called β-sheet regions in particular, form a kind of long distance dependency which can be captured by RNRGs. More precisely, it is a stochastic version of RNRGs which is used in this application[6]. The probability of each rewrite rule is set by training over a protein whose structure is known (corpus) and then used to analyze other proteins. RNRGs form a strictly growing hierarchy of grammars and languages (RNRLs) which is characterized by an integer called its *rank*. For any $k \geq 1$, $RNRL(k)$ properly contains $RNRL(k\text{-}1)$. $RNRL(0)$ are the CFLs and $RNRL(1)$ are the TALs.

An RNRG is a labeled tree rewriting system that consists of a *starting tree* and a finite set of rewriting rules, $A \rightarrow \alpha$, where A is a nonterminal symbol and α is a tree structure, which specifies how a node ν, labeled A, can be rewritten. Some leaves in α, called *empty leaves*, are labeled by a $\sharp$ sign. Empty leaves are placeholders which indicate where the children of ν must be grafted. The number of children of ν and the number of empty leaves in α must be equal. This number is the *rank*. After rewriting, the children of a node are attached to these empty leaves in the same order as before rewriting. A tree whose nodes are only labeled by terminal symbols is a *terminal tree*. The *tree language* of an RNRG is the set of terminal trees which can be derived from the starting tree after a finite number of applications of its rewriting rules. Its *string language* is the set of yields of its tree language. Note that if an internal node is labeled by a terminal symbol, this node cannot be rewritten and its label does not contribute

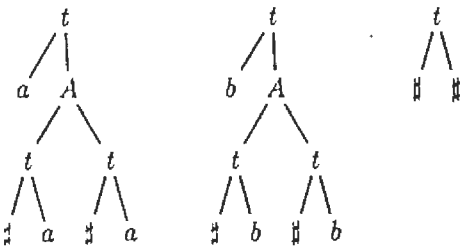
[6]In fact, for computational considerations, only a subclass of RNRGs is processed.

to the string language.

It is not difficult to transform an RNRG of any rank into an equivalent RCG. In fact the algorithm is a generalization of the one used for TAGs. Once again, no complexity penalty is induced by this transformation.

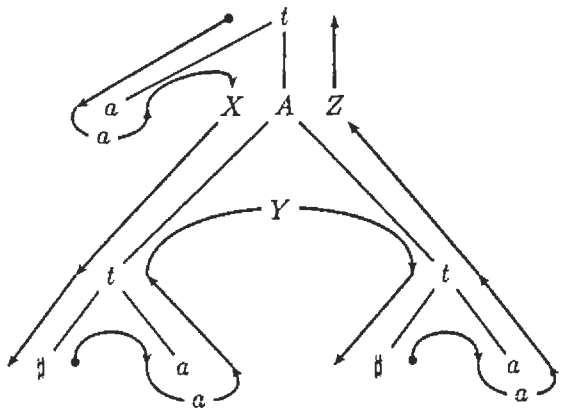
The previous three-copy language can be described by an RNRG of rank 2 whose initial tree is

and the set of rewrite rules for the node A is



where t stands for an anonymous terminal symbol which labels non leaf nodes.

Our algorithm exactly yields the RCG labeled (I). As an example, the arguments of the LHS predicate of the second clause have been gathered during the following walk on the tree structure of the first rewrite rule for A. The variables X, Y and Z denote the left, bottom[7] and right environment of A.



The corresponding parser has a cubic time complexity. This global parsing time can be reduced to linear if we remark that the ranges substituted to the variables X, Y and Z in the first clause are of equal sizes. Such a property can be automatically discovered or explicitly specified.

[7] For a node with $l+1$ sons, there will be $Y_1, \ldots Y_l$ "bottom" variables.

References

Naoki Abe, Hiroshi Mamitsuka. 1997. Predicting Protein Secondary Structure Using Stochastic Tree Grammars. In *Machine Learning*, 29, Dec. 1997, pages 275–301.

Pierre Boullier. 1996. Another Facet of LIG Parsing In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL96)*, University of California Santa Cruz, California, USA, pages 87–94. See also *Research Report No 2858* at `http://www.inria.fr/RRRT/RR-2858.html`, INRIA-Rocquencourt, France, Apr. 1996, 22 pages.

Pierre Boullier. 1998. Proposal for a Natural Language Processing Syntactic Backbone. In *Research Report No 3342* at `http://www.inria.fr/RRRT/RR-3342.html`, INRIA-Rocquencourt, France, Jan. 1998, 41 pages.

Annius V. Groenink. 1997. Surface without Structure, word order and tractability in natural language analysis. PhD thesis, Utrecht University, The Nederlands, Nov. 1977, 250 pages.

Aravind K. Joshi, K. Vijay-Shanker, David Weir. 1991. The convergence of mildly context-sensitive grammatical formalisms. In *Foundational Issues in Natural Language Processing*, P. Sells, S. Shieber, and T. Wasow editors, MIT Press, Cambridge, Mass.

Bernard Lang. 1994. Recognition can be harder than parsing. In *Computational Intelligence*, Vol. 10, No. 4, pages 486–494.

William C. Rounds. 1988. LFP: A Logic for Linguistic Descriptions and an Analysis of its Complexity. In *ACL Computational Linguistics*, Vol. 14, No. 4, pages 1–9.

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

K. Vijay-Shanker, David J. Weir. 1994. The equivalence of four extensions of context-free grammars. In *Math. Systems Theory*, Vol. 27, pages 511–546.

Can the TAG derivation tree represent a semantic graph ? An answer in the light of Meaning-Text Theory.

Marie-Hélène Candito & Sylvain Kahane
TALANA, Université Paris 7 2, place Jussieu, case 7003, 75251 Paris cedex 05
marie-helene.candito@linguist.jussieu.fr sk@ccr.jussieu.fr

Introduction

From the parsing point of view, the derivation tree in TAG [hereafter DT] is seen as the "history" of the derivation but also as a linguistic representation, closer to semantics, that can be the basis of a further analysis.
Because in TAG the elementary trees are lexicalized and localize the predicate-arguments relations, several works have compared the DT to a structure involving dependencies between lexical items (RJ92; RVW95).[1] We agree with these authors that there are divergences between the DT and syntactic dependencies, but we show here that the DT — in the sense of (SS94) — can be viewed as a semantic dependency graph, namely a SemS for Meaning-Text Theory [MTT] (ZM67; M88). This requires the predicate-argument cooccurrence principle and also constraints on the adjunction of predicative auxiliary trees. We briefly introduce the representation levels in MTT before studying the dependencies shown by the DT.[2]

1. Representation levels in MTT

MTT distinguishes between linguistic representations and correspondance rules to go from a representation to another, at an adjacent level. For a written sentence, there are 5 representations, each with a central structure : semantic [SemS], deep and surface syntactic [DSyntS and SSyntS], deep and surface morphological [DMorphS and SMorphS]. At each level, additional structures may supplement the central structure.
A key feature of MTT is that it distinguishes between semantic and syntactic dependencies. The SemS is a graph showing *semantic dependencies* between *semantemes* (= semantic units). The dependencies are numbered to distinguish between the different arguments of a predicative semanteme. An additional structure (the Sem-CommS) indicates communicative features (theme-rheme, focus ...). Figure 1 shows an example of SemS for :
(1) The new library owns the book that Peter thinks Mary needs

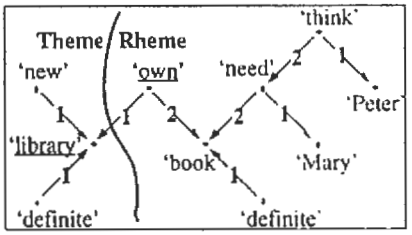


Figure 1 : SemS + Sem-CommS for

The DSyntS (Figure 2) is a dependency tree whose nodes are *generalized lexemes* (= lemma or set of lemmas corresponding to a semantic unit). Its arcs are *deep syntactic dependencies*, that are language independent (6 actancy relations I, II, ...VI, plus ATTR, COORD and APPEND). The SSyntS is a dependency tree showing grammatical relations — language dependent — between lexemes, that may be semantically void. Word order is defined at the deep morphological level.

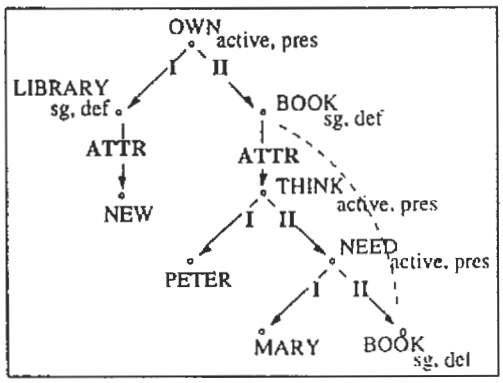


Figure 2 : DSyntS

The dictionary encodes for each generalized lexeme the associated semanteme along with the correspondence between Sem arguments and DSynt arguments.
Notation : the word *library* is a form of the lexeme LIBRARY whose semanteme is 'library'.

2. The DT nodes as semantemes

We assume the following linguistic properties for

[1] (RJ92) relate the DT to the deep syntactic structure (DSyntS) of MTT, namely a syntactic dependency tree, but they note that this correspondence DT / DSyntS is not direct, because the interpretation of adjunction arcs in terms of dependencies is not constant. (RVW95) take this divergence between DT and dependency tree as one of the motivations for defining D-Tree Grammars.

[2] We are thankful to Anne Abeillé, Laurence Danlos and Owen Rambow for valuable comments on earlier versions of this work.

elementary trees. The elementary trees correspond to exactly one semantic unit (A91)[3], and respect the predicate-argument co-occurrence principle (PACP), though with a semantic interpretation : semantic predicates anchor trees with positions for the syntactic expression of *all and only* their semantic arguments.[4] These positions are typed as substitution nodes and foot nodes. For instance in the tree for an attributive adjective, the adjective semantically governs the semanteme represented by the foot node.[5] Traditionally auxiliary trees are used for recursive structures. If syntactic structure is considered though, another dichotomy cuts across the distinction initial/auxiliary: the syntactic head is either the main anchor (for *predicative* trees) or the foot node (for *modifier* trees) ((K89), (SS94)).[6] All initial trees are predicative. Typical predicative auxiliary trees are the trees for bridge verbs. [7]

Let us now compare DT nodes with SemS nodes. The DT refer to lexicalized elementary trees, which correspond to a semantic unit (cf supra). Therefore, a DT node can be conceived as a semanteme, plus information for a particular lexicalization of that semanteme and for a particular syntactic construction. Yet with respect to SemS nodes, two differences appear. First, in the DT, there can be several nodes in coreference (though this coreference is not handled by the TAG formalism), that would be represented by a single node in the SemS. And second, semantic units realized in the language as inflections (eg. number, tense ...) are represented as features in TAG and, thus do not appear as nodes of the DT. So provided inflectional semantemes are not taken into account and coreferent nodes in the DT are considered a single node, there is a one-to-one relation between the SemS nodes and the DT nodes.

3. The DT arcs as semantic dependencies

As we said previously, several works have noted divergences between syntactic dependencies and DT arcs. Our claim is that a constant interpretation of the DT arcs can be found, though in terms of *semantic* and not syntactic dependencies : substitution and adjunction arcs both represent semantic dependencies, though in the opposite direction (Fig. 3).[8] For illustration see Fig. 4 the DT and SemS for sentence (1).[9]

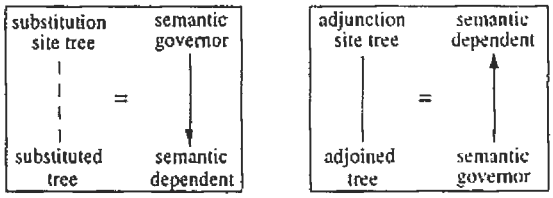


Figure 3 : Interpretation of DT arcs in terms of semantic dependencies

This result is a direct consequence of the linguistic properties we have assumed for the elementary trees. It can be noted that it is true for any type of adjunction arc (either predicative or modifier), with the definition of TAG derivation of (SS94), where multiple modifier adjunctions are allowed at the same address.[10]

---

[3] Thus elementary trees can have several lexical anchors, either because some are semantically empty (empty prepositions, complementizers ...), or because the several anchors form an idiom, whose semantic is not compositional.

[4] This counts for expressed semantic arguments only, so not for the agent in agentless passive constructions for instance.

[5] The notion of semantic governor must not be confused with the notion of *semantic head*. In « white car » white semantically governs car, yet car is the semantic head (a white car is a car). Following (P90) we define the semantic head as the semanteme that summarizes a semantic sub-graph. Not all sub-graphs can be summarized. In general a semantic graph for a whole sentence does not have a single semantic head, but one for its theme and one for its rheme.

[6] We follow the terminology of (SS94). Here *predicative* is used with its syntactic meaning.

[7] Another example is the tree for *glass of* in a *glass of wine*. The anchor *glass* is the syntactic head of the whole tree (A93). Yet the semantic interpretation of the trees for a bridge verb and for *glass-of* differ crucially: from the semantic point of view *glass of* behaves as a modifier and is not the semantic head of *glass of wine*. In *want to stay*, which expresses a will, the syntactic head is *want*.

[8] The fact that the DT should represent semantics is not new. See for example (A93) who distinguishes between *glass* in *a wine glass* and in *a glass of wine* on purely semantic grounds; (K89) who mentions that TAG should "preserve a straightforward compositional semantics"; (D98) who describes G-TAG, a generation system based on TAG where a derivation tree is built by lexicalizing a conceptual structure.

[9] The TAG analysis is from (X95), except that determiners are not considered as nominal complements and are thus adjoined.

[10] In case of adjunction, the interpretation in terms of semantic dependency is valid only if adjunction occurs on the spine of the tree receiving adjunction. This is the case most of the time. Yet we thank Martine Smets for pointing to us a problematic case: in *Paul gives flowers only to Mary*, *to* is semantically empty and appears as co-head in the *give* tree. The adverb *only* adjoins on the PP node of the *give* tree though it semantically governs *Mary*.

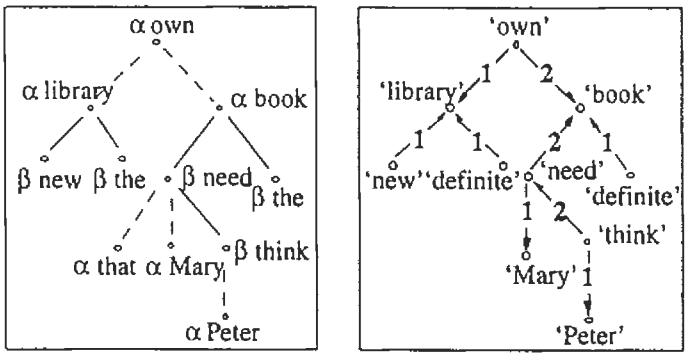


Figure 4 : DT (left) and SemS (right), with a different lay out to facilitate comparison

But obviously, the predicative adjunction arcs and the modifier adjunction arcs do not behave in the same way with respect to *syntactic* dependencies. Typically modifiers show a semantic and syntactic dependency in the opposite direction, while complement auxiliary tree preserve the direction of dependency in the semantic-syntax interface. The interaction of the various links can cause differences between the DT and the DSyntS.

Another example of mismatch is shown Fig. 4. The DT for sentence (1) shows the right chain of semantic dependencies for the sequence think-need-book, as the SemS shows. The only difference is the extra node for *that* in the DT, which does not count as a semantic unit. On the contrary in the DSyntS (Fig. 2), a syntactic dependency appears between BOOK and THINK, without a corresponding semantic dependency.

So, we have seen that in the general case, a DT induces a SemS. Further, the DT contains an additional information since it defines a partial order on its nodes, so that it form a tree. Thus the DT defines a path to cover all nodes once. The TAG procedure, from a generation point of view, is equivalent to fixing a starting node, the DT root. From that root, semantic dependencies gone through from the governor to the dependent (= positively) give substitution arcs, and semantic dependencies gone through in the opposite direction (= negatively) give adjunction arcs. It can be noted that it types the elementary trees involved as initial/auxiliary. For example, in Fig. 1, if we want to represent 'own' as a verb with two nominal arguments extended by substitution, the structure for 'think' will necessarily be an auxiliary tree, since one of its leaving arc has to be gone through negatively. Thus this gives another proof that bridge verbs have to be represented by auxiliary trees in relative clauses (or embedded interrogative clauses).

For the same reasons, to derive (4) *John knows the city in which Mary met Peter* and read the DT as a semantic graph (see the corresponding SemS Fig. 5), if the arguments of know are to be substituted, then *in* has to adjoin on *city* and *met* to substitute in *in*, though *met* is the syntactic governor of *in*.

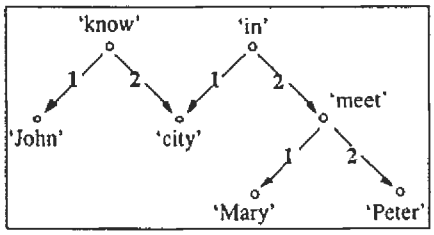


Figure 5 : SemS of (4)

4. Problematic derivations

It remains to study cases where there exists a SemS but no satisfactory DT. First TAG imposes a formal constraint that the DT be a tree. This implies in the case of cycles in the SemS, either to discard some dependency, or to cut the cycle at some node and to split that node into several coreferent ones (cf Section 2). And second, even provided a tree-like path exists for a given SemS, there are well-known cases where pure TAG fails to derive the correct word order (eg. clitic climbing in Romance (B98), or Kashmiri wh-extraction (cf RVW95)). To get the right word order a less restrictive formalism must be used.

More problematic are cases of TAG derivations showing the wrong dependencies. While adjunction of bridge verbs gives the right semantic dependencies in case of extraction, these adjunctions may be problematic when the bridge verb serves as argument for another predicate. Consider the following sentences, where a clause containing an embedded clause serves as argument for the main verb:

(5a) *Paul claims Mary said Peter left.*

(5b) *Paul claims Mary seems to adore hotdogs* (RVW95)

(5c) *That Paul wanted to stay surprised Mary.*

For (5a), in the classic TAG analysis (X95), the two bridge verbs adjoin recursively, and the DT is perfect (with the interpretation of adjunction arcs defined in Fig. 3). Yet for (5b) *Mary seems to adore hotdogs* serves as argument for *claims*, but here *seems* adjoins

on VP, and thus *claims* has to adjoin on *adore*.[11] Thus the DT does not show the right dependencies (either semantic or syntactic, cf (RVW95)). For (5c), the verb *surprised* traditionnally receives its subject via substitution (to block extraction), thus if the bridge verb *wanted* is still adjoined, the DT is different from the SemS (Fig. 6) (apart from the splitting of the 'Paul' node into 2 coreferent nodes; we show the coreference with a curved dashed line). The problem arises because the tree αstay substitutes in αsurprise, but when the predicative tree βwant adjoins on αstay, it becomes the semantic head of the whole subtree.[12]

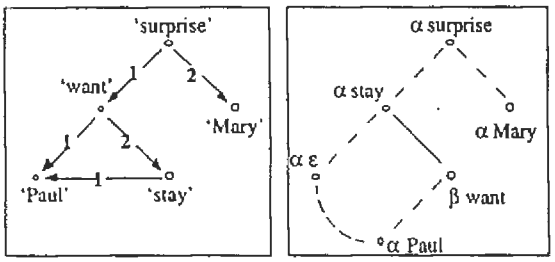


Figure 6 : Problematic derivation (SemS and DT) for
That Paul wanted to stay surprised Mary

So to read a DT as a SemS, we need not only the PACP, but also a control over the combination of the elementary trees : it must be checked that the argumental positions in a tree are actually filled by the right arguments.[13]
It can be noted that for sentence (5b) and (5c), ruling out adjunctions of complement trees (as in DTG (RVW95)) solves the problem. Yet it might be problematic for sentence (1), for which we have seen that the TAG DT shows the right semantic dependencies. And it also rules out the adjunction of an athematic complement tree (such as the one for *glass-of*). This is investigated in (CK98).

Conclusion

We have shown that in the general case the DT can be viewed as a semantic representation, in the sense of MTT, provided coreference is not taken into account. We have given a characterization of problematic derivations. This result is of crucial importance for any further processing based on the TAG derivation tree.
We have also provided a new characterization of adjunction and substitution arcs depending on the direction of the semantic dependency they represent.

[11] (SS94) already noted that multiple adjunctions of bridge verbs at one node should be ruled out, here we find that this holds for a whole tree.

[12] (K89) already noted that « derivations under which thematic roles, once established, are altered by further adjunctions » should be ruled out.

[13] Another case where positions « are not filled by the right arguments » is for instance pied-piping. The XTAG derivation for *the woman whose daughter Peter talks to* does not show the right semantic dependencies, since a link appears between *talks-to* and *woman*.

References

(A91) A. Abeillé, 1991: *Une grammaire lexicalisée d'arbres adjoints pour le français*. Ph.D. Thesis. Univ. Paris 7.

(A93) A. Abeillé, 1993: Interactions Syntaxe-Sémantique dans une TAG. Colloque ILN'93. Nantes.

(B98) T. Bleam, forthcoming: Clitic Climbing and the power of TAG, in Abeillé, A., Rambow, O. (eds.), *Tree-adjoining Grammars*, CSLI. Stanford.

(CK98) M-H. Candito, S. Kahane, 1998: Defining DTG derivations to get semantic graphs. This volume.

(D98) L. Danlos, forthcoming, G-TAG : a formalism for text generation, in Abeillé, A., Rambow, O. (eds.), *Tree-adjoining Grammars*, CSLI, Stanford.

(F92) R. Frank, 1992: *Syntactic locality and Tree Adjoining Grammar: Grammatical, Acquisition and Processing Perpectives*. Ph.D. Thesis. Univ. of Pennsylvania.

(K89) A. Kroch, 1989: Asymmetries in Long-Distance Extraction in a Tree-Adjoining Grammar. In *Alternative Conceptions of phrase structure*. M. Baltin and A. Kroch (eds), Univ. of Chicago Press, Chicago.

(M88) I. Mel'cuk, 1988: *Dependency Syntax : Theory and Practice*. Albany. State Univ. of NY Press.

(P90) A. Polguère, 1990 : *Structuration et mise en jeu procédurale d'un modèle linguistique déclaratif dans un cadre de génération de texte*. Ph.D. Thesis. Univ. of Montreal.

(RJ92) O. Rambow, A. Joshi, 1992: A Formal Look at Dependency Grammars and Phrase-Structure Grammars, with Special Consideration of Word-Order Phenomena. in Leo Wanner (ed.) *Recent Trends in Meaning-Text Theory*

(T59) L. Tesnière, 1959: *Eléments de syntaxe structurale*, Klincksieck, Paris.

(RVW95) O. Rambow, K. Vijay-Shanker, D. Weir. 1995: D-Tree Grammars, ACL'95.

(SS94) S. Shieber, Y. Schabes, 1994: An alternative conception of tree-adjoining derivation. *Computational Linguistics*, 20.1.

(XTAG95) XTAG research group: A Lexicalized TAG for English. Technical report IRCS 95-03. Univ. of Pennsylvania. (Online updated version).

(ZM67) A. Zolkovskij, I. Mel'cuk, 1967 : O semanticeskom sinteze [On semantic synthesis]. *Problemy kybernetiki*, v. 19, 177-238.

Defining DTG derivations to get semantic graphs

Marie-Hélène Candito & Sylvain Kahane
TALANA, Université Paris 7, 2, place Jussieu, case 7003, 75251 Paris Cedex 05
marie-helene.candito@linguist.jussieu.fr, sk@ccr.jussieu.fr

Introduction

The aim of this paper is to find a formalism of the TAG family, where the derivation controller can be interpreted as a semantic dependency graph, in the sense of Meaning-Text Theory (ZM67; M88).

In a previous paper (CK98), we study this interpretation of the derivation tree (DT) in the case of standard TAG. We prove that, in the general case, if the predicate-argument cooccurence principle[1] [= PACP] holds and if elementary trees correspond to a semantic unit (A91), substitution arcs can be read as semantic dependencies where the dependent is the anchor of the substituted tree, and adjunction arcs — of any type— can be read as semantic dependencies in the opposite direction.

Yet we also characterized cases where the DT shows wrong (semantic) dependencies (cf also (RVW95)). A problem may occur when, in the same sentence, clausal complementation is handled both with substitution of an embedded clause and with adjunction of a main verb.[2]

Further, there are well-known cases that TAG cannot handle if the PACP holds (e.g. clitic climbing in Romance (B98), Kashmiri wh-extraction (RVW95), extraction out of NP in French (A98). Finally, in some cases, the argumental positions in a tree are not filled by the right arguments, and thus the derivation tree does not show the right semantic dependencies (pied-piping (CK98)).

(RVW95) have defined D-tree Grammars (DTG) by ruling out predicative adjunction (e.g. adjunction of bridge verbs). Thus, DTG seems a good candidate for our goal.[3] In Section 1, we recall DTG operations and study the case of relative clause interacting with a bridge verb, which has proved to be correctly handled by TAG, as far as semantic dependencies are concerned (CK98). This leads us to propose an extension of DTG, called GAG for Graph-driven Adjunction Grammar, whose derivation controllers are graphs (Section 2). Finally, in Section 3, we develop an original analysis of wh-words in GAG.[4]

1. Generalized substitution and generalized adjunction

DTG (RVW95) handles both clausal and nominal complementation with the same operation, a generalized substitution, called subsertion, and thus avoids the use of predicative adjunction. In order to cover the long-distance dependency data (including cases not handled in TAG), this operation allows pieces of the substituted element to come in between elements of the tree receiving substitution.

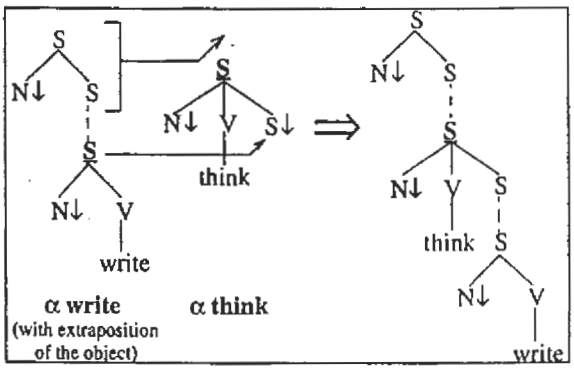


Figure 1 : Subsertion (= generalized substitution)

A DTG elementary structure is essentially a TAG elementary tree, but it can contain d-edges, namely underspecified paths between two nodes (represented by dotted lines). An elementary structure in DTG is called a d-tree and is made of one or several components which are ordinary trees, related by d-edges. When a d-tree α is subserted at a substitution node of another d-tree γ, a component of α is substituted at a substitution node of γ, and all components of α that are above the substituted component are inserted into d-edges of γ, above the

[1] A tree anchored by a predicate must contain positions for all and only its arguments.

[2] For a sentence such as *That Paul wanted to stay surprised Mary*, the DT shows the wrong dependencies if the tree for *surprise* has a substitution node for its subject, and the one for *want* has a foot node for its embedded clause (CK98). Another problem occurs with a raising verb that serves as semantic argument to a bridge verb as in *Paul claims Mary seems to adore hotdogs* (adapted from (RVW95)). To get the correct semantic dependencies, the trees for *claims* and *seems* should combine together (either via substitution of *seems* or adjunction of *claims*) but this is impossible in TAG since *seems* is represented by a VP-rooted tree.

[3] In (RVW95), one motivation was to get (deep) syntactic dependencies. Though in most cases semantic and deep syntactic dependency structures induce the same non-oriented graph, we will study a case of mismatch in Section 1 and 3.

[4] We are thankful to Owen Rambow and David Weir for valuable discussions about this work.

substituted node or placed above the root node. Fig. 1 shows an example of subsertion.[5]

Now, ruling out predicative adjunctions implies to reconsider cases that were correctly handled by TAG as far as semantic dependencies are concerned.

For example, in order to handle extraction out of a modifier (e.g. preposition stranding) and « extraction of a modifier », we define a parallel generalization of the adjunction operation.[6] We will thus refer to generalized substitution and generalized adjunction. Fig. 2 shows the generalized adjunction of *in [this bed]* for the sentence:[7]

(1) In this bed, I think I have slept twice.

To get the semantic dependency between *in* and *slept*, we want βin to adjoin in αslept, still allowing a piece of the modifier (here the whole modifier) to be inserted higher.

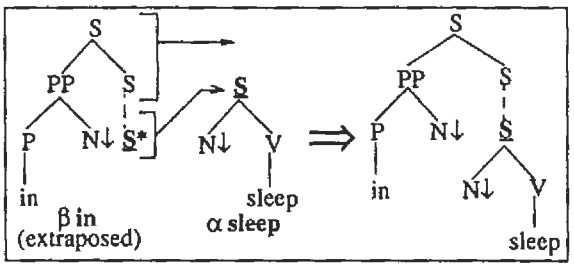


Figure 2: Generalized adjunction

Now consider the sentence:

(2) I bought the books which Peter thinks Mary wrote.

In TAG, the relation between a verb and a relativized complement is localized. So for instance to handle (2), *wrote* anchors an NP modifier tree (thus an auxiliary tree) in which the bridge verb *thinks* adjoins (K87). In DTG, bridge verbs receive their clausal complement via substitution. Thus in order to keep the semantic dependency between a verb and a relativized complement, we propose to allow a d-tree to substitute in a d-tree and adjoin in another one.[8] We will call this extension GAG.

2. GAG: a multi-rope DTG

GAG is an extension of DTG that uses the same elementary structures, namely d-trees. But in GAG, some nodes of an elementary d-tree must be marked as being ropes (underlined in the figures). Only roots of components can be ropes. Any component containing a foot node has a root which is a rope. A component without foot node is subsitutable if and only if its root is a rope. In (RVW95), all the components of an elementary d-tree are considered substitutable, namely each component's root is a rope, but a d-tree can be subserted only once, namely all ropes are mutually exclusive. In our extension, d-trees with n mutually exclusive ropes are expanded in n d-trees with a single rope. Further, a d-tree may have several ropes which are not mutually exclusive, that is, that can each be combined with a separate d-tree. To sum up, GAG is a multi-rope DTG.

From the linguistic point of view, we foresee the use of one-rope and two-rope d-trees only. Examples of two-rope d-trees will be given in Section 3.

Let us now define the derivation graph (DG), which is a structure that partially encodes a GAG derivation (and that we will interpret as a semantic graph).[9]

If a two-rope d-tree substitutes in a one-rope d-tree, we obtain a two rope derived d-tree and nothing in the DG tells us from which elementary d-tree each rope comes from. Thus in GAG, the original elementary d-tree for each node of a derived d-tree is memorized.

We thus have to specify what happens in the case of node unification during substitution or adjunction. In case of substitution, a rope unifies with a substitution site. We then consider that the resulting node comes from the elementary d-tree that is substituted. In case of generalized adjunction, the node receiving adjunction is replaced by a component of the adjoined tree. In the derived tree, we consider that the root of that adjoined component belongs to the tree receiving adjunction.

The DG can now be defined as follows: let γ be an elementary d-tree. Let φ be a derived tree, and ψ the corresponding derivation graph (DG). If φ substitutes (resp. adjoins) in γ, one of its ropes is used up. Let α be the name of the d-tree from which this rope originates. The resulting DG ψ' is the DG ψ plus a

[5] Figure 1 shows d-trees that are inspired from the d-trees proposed by (RVW95) to handle a sentence such as *Children's books Peter thinks Mary wrote*. For sake of simplicity VP nodes are omitted.

[6] In (RVW95), modifiers are handled by sister-adjunction, an operation that is equivalent to adding at a given node a left-most or right-most daughter node. We prefer to maintain adjunction (as in TAG), notably because we want to be able to adjoin the tree for *glass-of* for instance (CK98).

[7] In this example, the bottom component of βin is reduced to the foot node, which is also the rope (see definition in Section 2).

[8] As a referee pointed out to us, there is an alternate derivation of (3) in DTG in which *thinks* is adjoined to *book* (creating the syntactic attachment) and *wrote* subserted into *thinks* with the relative pronoun being inserted into the right place and receiving co-reference with *books* through features (thus creating the semantic attachment).

[9] The equivalent in DTG is called a SA-tree. In GAG, it is a graph due to multi-rope d-trees.

substitution (resp. adjunction) arc between α and γ (γ being the mother node).[10] Consequently to this definition, a d-tree has as many mother nodes in the final DG as it has used ropes.

As in DTG, the derivation succeeds if the d-edges of the derived tree can be collapsed (forgetting the fact that some nodes can be rope nodes). From the computational point of view it can be noted that the ropes of a multi-rope d-tree can combine in whatever order with other d-trees.

3. Taking advantage of GAG to analyse extraction

As we said, the main motivation for GAG is to have a formalism inspired by TAG whose derivation controllers induces semantic dependency graphs. In order to achieve that, we have relaxed the constraint that these controllers be trees.
As linguistic constraints for elementary structures, in addition to the PACP, we type the argumental positions as foot nodes and substitution nodes on purely linguistic grounds.[11] Generalized substitution is used for elements that are subcategorized and to which a thematic role is assigned, while generalized adjunction is used for modifiers.

In the following, we concentrate on examples of GAG analysis involving wh-words. Consider:

(3a) *Children books Peter thinks Mary wrote.*
(3b) *I bought the books which Peter thinks Mary wrote.*
(3c) *I wonder which books Peter thinks Mary wrote.*
(3d) *Which books does Peter think Mary wrote ?*

In these four examples we have a clause of the form *Peter thinks Mary wrote [book]*. The distribution of this clause depends on the extracted element: for example, in (3b) that clause is an NP modifier because of the relative wh-word *which* and in (3c) that clause can be the syntactic argument of *wonder* because of the interrogative wh-word *which*. The (T59) analysis of relative and (indirect) interrogative clauses is that the wh-word plays two roles: on one hand, it fills a position in the clause as pronoun and on the other hand it controls the distribution of the clause and is thus its syntactic head.

We claim that it is possible (though not mandatory) to have an analysis where the particular distribution of wh-clauses is completely assumed by the wh-word. To do this, we represent wh-words with two-rope elementary d-trees: the first rope will be linked to the main clause and the second one will be linked to the phrase showing extraction. So for a relative wh-word, the first rope is a foot node which adjoins on the antecedent and the second rope substitutes or adjoins in the phrase showing extraction, depending on the complement/modifier nature of the extracted element. For an interrogative wh-word, the first rope substitutes in the verb which subcategorizes for the interrogative clause. We give example of relative clauses only:

(4a) *I know the books which Mary wrote.*
(4b) *I know the bed in which Peter slept.*
(4c) *I know the books whose authors are famous.*
(4d) *I know the man whose car Peter borrowed.*
(4e) *I know the place where Peter was born.*

Fig. 3 shows the two-rope d-trees for the wh-words involved (the ropes are underlined).

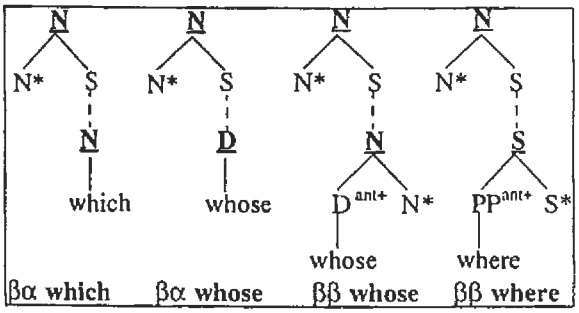


Figure 3: some two-rope elementary trees (for relative wh-words)

The analysis for (3b), (4a) and (4b) use the same d-tree for *which*, βαwhich, which substitutes respectively in the d-trees for *wrote* and *in* (Fig. 4).

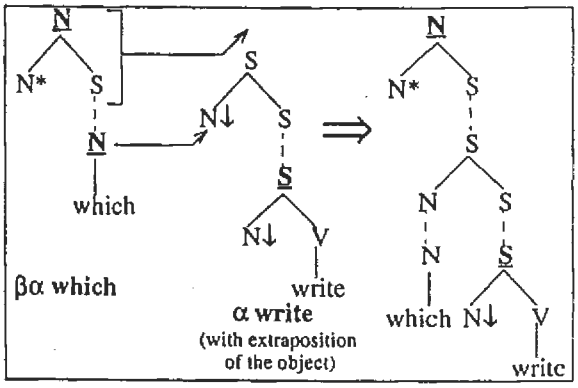


Figure 4: Substitution of a two-rope d-tree

Fig. 5 shows the DG for (3b). To interpret a DG as a semantic graph, one needs to:

- translate d-trees names into semantemes
- read substitution arcs as semantic dependencies from the site of substitution to the substituted tree;
- read adjunction arcs as semantic dependencies from the adjoined tree to the tree receiving adjunction;
- collapse some arcs that link coreferent nodes.

[10] It can be noted that because we remember the origin of each node of a derived tree, a derivation need not be bottom-up.
[11] This is possible because we allow the derivation controller to be a graph and use the generalized substitution.

This last operation arises typically for some relative pronouns. In the βα d-trees of Fig. 3, the foot node does not represent a semantic argument of the anchor, but a duplication of the anchor itself (the antecedent in syntax). Thus tne correspondant adjunction arc in the derivation controller (eg. the adjunction arc in Fig. 5) has to be collapsed in order to get the semantic graph (Fig. 6).

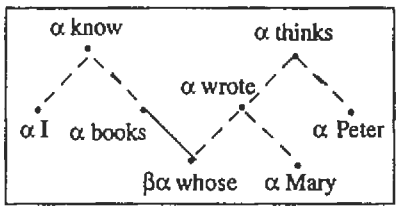


Figure 5: GAG derivation graph
I know the books which Peter thinks Mary wrote

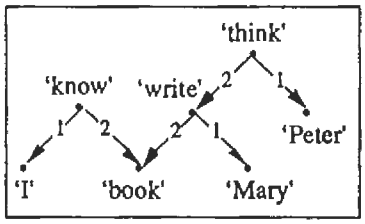


Figure 6: MTT semantic graph
I know the books which Peter thinks Mary wrote

In (4c), *whose* is an argument of *authors*, thus βαwhose substitutes in the *authors* tree. In (4d), we consider that *whose* is a lexicalization of the two-place semanteme 'own'. Its d-tree ββwhose[12] adjoins twice, on the trees for both its arguments (here lexicalized by *man* and *car*). Fig 7 shows the DG for (4d).

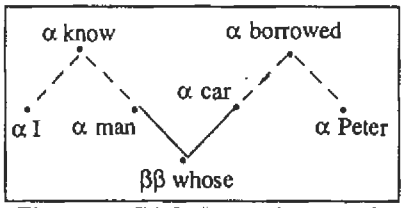


Figure 7: GAG derivation graph
I know the man whose car Peter borrowed

To get the semantic graph, the two adjunction arcs of ββwhose are interpreted as semantic dependencies from the adjoined tree to the tree receiving adjunction (Fig. 8). Similarly, ββwhere corresponds to a semanteme 'location' (= 'is located in') with two arguments.

In the anaysis we have shown, features must be added to control which components can be inserted in a d-edge (cf. the subsertion insertion constraints in DTG). They are needed for instance to block extraction in the case of non-bridge verbs or to express constraints on double extractions and topicalization.

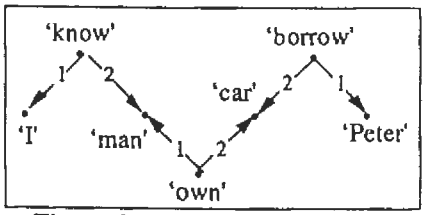


Figure 8: MTT semantic graph
I know the man whose car Peter borrowed

Conclusion

Building on DTG and TAG, we have defined a formalism, GAG, where the derivation controller can be seen as a semantic dependency graph, with the reading defined in (CK98). This allows us to propose an analysis in which the distribution of clauses containing wh-words is totally controlled by the d-trees associated with the wh-words themselves. Thus topicalization, relativization, (direct or indirect) interrogation and cleft clauses can be handled with the same elementary d-trees for verbs. Computational properties of GAG need a further study.

References

(A91) A. Abeillé, 1991 : Une LTAG pour le français. Ph.D. thesis. Univ. Paris 7.

(A98) A. Abeillé, forthcoming : Extraction out of NP and clitic-noun dependencies in French, in Abeillé, A., Rambow, O. (eds.), *Tree-adjoining Grammars*. CSLI, Stanford.

(B98) T. Bleam, forthcoming : Clitic Climbing and the power of TAG, in Abeillé, A., Rambow, O. (eds.). *Tree-adjoining Grammars*, CSLI, Stanford.

(CK98) M.-H. Candito, S. Kahane, 1998 : Can the TAG derivation tree represent a semantic graph ? An answer in the light of the Meaning-Text Theory. This volume.

(F92) R. Frank, 1992 : Syntactic locality and Tree Adjoining Grammar: Grammatical, Acquisition and Processing Perpectives. Ph.D. thesis. Univ. of Pennsylvania.

(K87) A. Kroch, 1987 : Subjacency in a Tree-Adjoining Grammar. In A. Manaster-Ramer, ed. *Mathematics of Language*.

(M88) I. Mel'cuk, 1988 : Dependency Syntax : Theory and Practice. Albany. State Univ. of New York Press.

(RVW95) O. Rambow, K. Vijay-Shanker, D. Weir, 1995 : D-tree Grammars, ACL'95.

(ZM67) A. Zolkovskij, I. Mel'cuk, 1967 : O semanticeskom sinteze [On semantic synthesis]. Problemy kybernetiki, v. 19, 177-238. [Fr. Transl. In T.A. Informations, 1970, #2, 1-85.]

[12] We advocate that a determiner which is not an argument is adjoined. It is the case for the possessive when it refers to a possessor.

The LEXSYS Project

John Carroll, Nicolas Nicolov, Olga Shaumyan, Martine Smets & David Weir
University of Sussex
Brighton, BN1 9QH, UK

1 Introduction

We present an overview of the ongoing LEXSYS project[1]. The aim is to bring together, and evaluate, a variety of current NLP techniques, including the organisation of grammars into inheritance hierarchies for compact representation, exploitation of diverse precompilation techniques for efficient parsing, and use of statistical analysis to disambiguate parse results. In conjunction with this we are using several existing tools and resources, such as the lexicon developed in the Alvey Natural Language Tools project (Briscoe et al., 1987), lexical frequency information from the SPARKLE project[2], and an established lexical knowledge representation language DATR (Evans and Gazdar, 1996a) to represent the grammar. The overall architecture of LEXSYS is shown in Figure 1 and the following sections discuss each of the system's main components.

2 The morphological analyser

The text is first *tokenised* and then a *sentence-splitter* is applied to it to determine likely sentence boundaries. The resulting sentences are tagged with extended part-of-speech (PoS) labels using a first-order HMM *tagger* (Elworthy, 1994) trained on the SUSANNE corpus (Sampson, 1995). The SUSANNE lexicon is augmented with open-class words from the LOB corpus and the tagger incorporates a part-of-speech guesser that empirically achieves around 85% label assignment accuracy for unknown words. For each word the tagger returns multiple-label hypotheses, but filters out any whose probabilities are below a preset factor of the most probable. The thresholding technique allows us to fine-tune the trade-off between the costs of incorrect tagging and processing complexity due to lexical ambiguity.

After tagging, a *lemmatiser* finds the lemma, or base form, corresponding to each word-label pair, using an enhanced version of the GATE project stemmer (Cunningham et al., 1995). Finally, the lemma and PoS label are combined with syntactic information associated with the word's morphological form (e.g. number for nouns).

3 The grammar

Lexicalized D-Tree Grammar (LDTG) (Rambow et al., 1995) is a variant of LTAG. The primitive elements of LDTG are called elementary d-trees and are combined together to form larger structures during a derivation. Although, for convenience, we present d-trees graphically as though they were conventional trees, they are more correctly thought of as expressions in a tree description logic (Rogers and Vijay-Shanker, 1992). These expressions *partially* describe trees by asserting various relationships between nodes: parenthood, domination, precedence (indicating that one node is to the left of another), equality and inequality.

There are two substitution-like operations for composing d-trees, both of which involve combining two descriptions while equating exactly one node from each description. One of the operations is always used to add complements and involves equating a frontier node (in the d-tree that is getting the complement) with the root of some component (in the d-tree that is providing the complement), such that the two nodes

[1]This work is supported by UK EPSRC project GR/K97400 and by an EPSRC Advanced Fellowship to Carroll. Thanks to Roger Evans, Gerald Gazdar & K. Vijay-Shanker for helpful discussions.

[2]CEC Telematics Applications Programme project LE1-2111 "SPARKLE: Shallow PARsing and Knowledge extraction for Language Engineering".

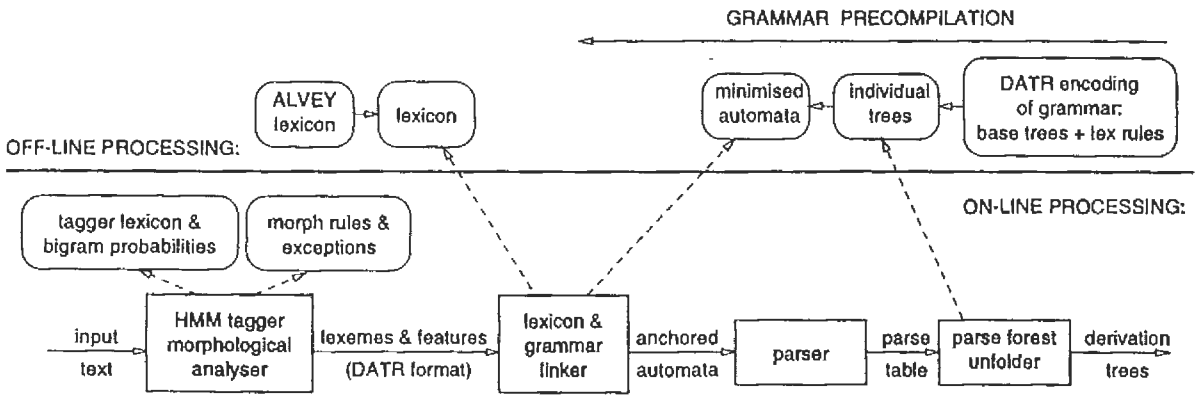


Figure 1: System architecture

being equated are compatible. Two schematic examples of this operation are shown at the top of Figure 2. These are the two cases that appear in our grammar for English[3]: at the top left is the case in which the entire complement d-tree appears below the point of substitution; the top right gives the case in which the complement involves extraction where the extracted component is placed at the top of the d-tree.

A second operation is used to add modifiers. In terms of tree descriptions, this operation is similar to the complement-adding operation since it also involves combining two d-trees while equating a pair of nodes. In this case, however, it involves equating an *internal* node (in the d-tree that is getting the modifier) with the root of some component (in the d-tree that is providing the modifier), such that the two nodes being equated are compatible. Two schematic examples are shown at the bottom of Figure 2. As in the case of the complement-adding operation, these are the two cases that appear in our grammar for English: at the bottom left is the case in which the entire modifying d-tree appears below the point of modification; the bottom right gives the case in which the modifier involves extraction, where the extracted component is placed at the top of the d-tree[4].

We are in the process of developing a wide-coverage LDTG based on the XTAG grammar. There are a number of differences between the formalisms and the analyses they allow. One of the main differences is that the LDTG formalism allows the existence of VPcomplements for main verbs, and this has a number of consequences: e.g. the grammar does not assume the existence of *PRO*, auxiliary and main verbs anchor the same type of tree, there are no predicative trees, passive participles anchor VPtrees [5]. See Smets (1998) for more details.

As in the XTAG system, ES'S are grouped into families. Currently we have 44 families with around 60 families expected in total. The total number of (unanchored) ES'S in the current grammar is 650 with approximately 1000 ES'S expected. The grammar is encoded using the lexical knowledge representation language DATR (Evans and Gazdar, 1996b), based on the scheme proposed for LTAG by Evans, Gazdar and Weir (1995). Encoding is compacted through the use of 36 lexical rules and non-monotonic inheritance. Details are presented in Smets and Evans (1998).

4 The lexicon

The lexicon is a reworked version of the Alvey Natural Language Tools (ANLT) lexicon (Carroll and Grover, 1989) where category and feature assignments are expressed in DATR notation to conform to the encoding used for the grammar and the results of morphological anal-

[3]The general case is explained in Rambow et al. (1995).

[4]In the examples shown at the bottom of Figure 2 the modifier d-tree is placed to the left of the subtree it modifies. It is also possible for modification to take place on the right.

[5]The analyses that we are able to implement are also adopted in a number of theories: GPSG, HPSG, LFG, CG.

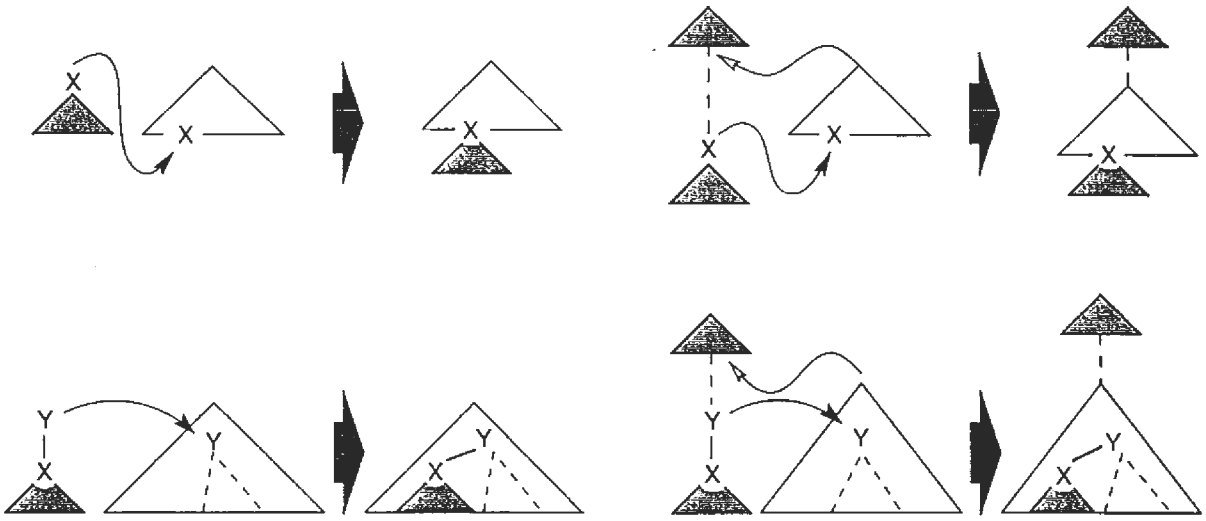


Figure 2: Composition operations

ysis. Although not currently exploited, this uniform notation would permit the lexicon to form the leaf nodes in the grammar hierarchy and so inherit automatically any of the syntactic information (such as default feature assignment) contained there. The lexicon contains only lemmas, with wordform information supplied by the morphological analyser. It should be noted that the morphological form of a linguistic datum affects how much of a family is selected: so the *ing* form of the verb will not inherit all of the ES's associated with the verb, but only the forms stipulated as *ing* or *non-finite*.

In separate but related work (Briscoe and Carroll, 1997), we are acquiring the complementation possibilities for predicates from large amounts of text information about. In that work we distinguish 160 verbal subcategorisation classes—a superset of those found in the ANLT and COMLEX Syntax dictionaries—and we acquire relative frequencies for each class found for each verb. The approach uses a previously-existing phrase-structure parser which yields 'shallow' parses, a subcategorisation class classifier, and *a priori* estimates of the probability of membership of these classes. Carroll et al. (1998a) demonstrate that adding this frequency information to a (non-lexicalised) statistical parser significantly increases its disambiguation accuracy. We intend also to incorporate this information into the system described in this paper, at the point where lemmas are associated with tree families: each lemma / family combination would have a separate probability. Carroll and Weir (1997) outline other alternative probabilistic models, some of which we also intend to investigate.

The same shallow phrase-structure parser is also providing data for the acquisition of selectional preferences, at present again just for verbs, and only for NP and PP subject, direct and indirect verbal complements (McCarthy, 1997). The technique uses the WordNet hypernym hierarchy (Fellbaum, 1998) in tandem with Minimum Description Length learning (Rissanen, 1978) to induce semantic classes of nominal heads at an appropriate level of abstraction. We have results of acquisition from a 10 million word extract from the British National Corpus, and will augment the lexicon with the acquired selectional frequencies and use them during parsing as a further source of disambiguation information.

5 The parser

We have implemented a simple bottom-up parsing algorithm which is being used for grammar development. The parser simulates anchor-up traversal of ES's. This traversal begins at the anchor node with the parser working outwards as it moves upwards towards the root of the ES. When visiting nodes during this traversal,

the parser must perform various actions. Which particular action is required at each node is determined by the type of node (e.g. whether it is a frontier or internal node) and its position relative to the anchor (whether it is to the right or left of the anchor). We refer to each step in this sequence as a parser action and to a sequence of parser actions associated with a ES, as an elementary computation (EC) of that ES.

Prior to parsing, each word of the input is associated with a set of ES'S that it can anchor. Each ES in the grammar can be pre-compiled into a (flat) sequence of parser actions. These sequences, rather than the ES'S themselves, are the objects that the parser manipulates during parsing.

The parser fills a 2-dimensional table (where each cell corresponds to a substring of the input) by advancing through these parser action sequences as actions are executed. In addition to action sequences, the items in cells contain multisets that hold suspended action sequences. In LTAG, adjunction has the effect of embedding one tree within another, where a stack can be used by a parser to control the unbounded nesting of ES'S that can occur in derivations. LDTG also allows embedding of ES'S; however, multisets rather than stacks are used to control this embedding. This difference is due to the limited control provided by LDTG over the relative positioning of the components of two composed ES'S. Each entry in the parse table contains a list of pointers to the entries that caused it to be added. Once the table is complete, top-down pruning is performed to remove entries that do not form part of a complete parse. This produces a parse forest from which phrase structure trees are derived.

Building an efficient parser for a *wide-coverage* LDTG or LTAG grammar represents a challenge. Each word in the input string introduces a large number of ES'S into the parse table: one for each of its possible alternative readings. In the current grammar the words *come*, *break* and *give* anchor around 130, 180 and 340 ES'S, respectively. In fact, if we include ES'S for all alternative feature values, these figures rise by an order of magnitude. There can be substantial overlap in structure among the ES'S associated with a given input word. Existing LTAG parsing algorithms treat each ES as independent, which results in considerable duplication of processing of common structure during parsing. Evans and Weir (1997; 1998) propose that a significant amount of overlapping among EC'S can be pre-compiled out by performing the following steps: (1) compile each ES into a finite state automaton; (2) for each set of ES'S that a single word can anchor, merge the corresponding automata into a single automaton; (3) minimise the number of states in the merged automaton (using standard techniques); and (4) rather than associating each input word with a set of d-trees, associate it with a minimized automaton and parse as usual. A preliminary indication of how the Evans-Weir proposal will work in practise on the LEXSYS grammar is discussed in (Carroll et al., 1998b) where we show that using minimized automata leads to a several-hundred-fold reduction in the number of automata states. Even greater savings are achieved when all feature information from the lexicon is included. In fact, the use of minimized automata appears to provide an efficient solution to processing ES'S whose node labels involve feature structures that might normally be encoded with disjunctive feature values (but which we encode with multiple instances of the ES). We are in the process of implementing a parser that exploits this technique in order to more fully evaluate its practical value.

6 Summary

LEXSYS is being developed as a wide-coverage parsing system using a lexicalized grammar formalism. We are employing two techniques to keep the scope of the task under control: (1) encoding grammar nsing DATR to achieve compact representation, and (2) parsing with minimized automata to achieve computation sharing. We feel that this approach allows us to maintain a separation between the issues of linguistic adequacy and processing pragmatics (grammar storage, parsing efficiency, etc.). The future work will also incorporate a stochastic component for parse disambiguation.

References

Ted Briscoe and John Carroll. 1997. Automatic extraction of subcategorization from corpora. In *Proceedings of the 5th ACL Conference on*

Applied Natural Language Processing, pages 356–363, Washington, DC.

Edward Briscoe, Claire Grover, Branimir Boguraev, and John Carroll. 1987. A formalism and environment for the development of a large grammar of English. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, pages 703–708, Milan, Italy.

John Carroll and Claire Grover. 1989. The derivation of a large computational lexicon of English from LDOCE. In B. Boguraev and E. Briscoe, editors, *Computational Lexicography for Natural Language Processing*. Longman.

John Carroll and David Weir. 1997. Encoding frequency information in lexicalized grammars. In *Proceedings of the 5th International Workshop on Parsing Technologies*, pages 8–17.

John Carroll, Guido Minnen, and Ted Briscoe. 1998a. Can subcategorisation probabilities help a statistical parser? In *Proceedings of the 6th ACL/SIGDAT Workshop on Very Large Corpora*, Montreal, Canada.

John Carroll, Nicolas Nicolov, Olga Shaumyan, Martine Smets, and David Weir. 1998b. Grammar compaction and computation sharing in automaton-based parsing. In *Proceedings of the First Workshop on Tabulation in Parsing and Deduction*, pages 16–25.

Hamish Cunningham, Robert Gaizauskas, and Yorick Wilks. 1995. A general architecture for text engineering (GATE) — A new approach to language R&D. Research memo CS-95-21, Department of Computer Science, University of Sheffield, UK.

David Elworthy. 1994. Does Baum-Welch re-estimation help taggers? In *Proceedings of the 4th ACL Conference on Applied Natural Language Processing (ANLP'94)*, Stuttgart, Germany.

Roger Evans and Gerald Gazdar. 1996a. DATR: A Language for Lexical Knowledge Representation. *Computational Linguistics*, 22(2):167–247.

Roger Evans and Gerald Gazdar. 1996b. DATR: A language for lexical knowledge representation. *Computational Linguistics*.

Roger Evans and David Weir. 1997. Automaton-based parsing for lexicalized grammars. In *Proceedings of the 5th International Workshop on Parsing Technologies*, pages 66–76.

Roger Evans 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 16th International Conference on Computational Linguistics*.

Roger Evans, 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 (ACL'95)*, pages 77–84.

C. Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

Diana McCarthy. 1997. Word sense disambiguation for acquisition of selectional preferences. In *Proceedings of the Proceedings of the ACL/EACL 97 Workshop Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*, pages 52–61, Madrid, Spain.

Owen Rambow, K. Vijay-Shanker, and David Weir. 1995. D-Tree Grammars. In *Proceedings of the 33rd Meeting of the Association for Computational Linguistics*, pages 151–158.

J. Rissanen. 1978. Modeling by shortest data description. *Automatica*, 14:465–471.

James Rogers and K. Vijay-Shanker. 1992. Reasoning with descriptions of trees. In *Proceedings of the 30th Meeting of the Association for Computational Linguistics*, pages 72–80.

Geoffrey Sampson. 1995. *English for the Computer*. Oxford University Press, Oxford, UK.

Martine Smets and Roger Evans. 1998. A compact encoding of a DTG grammar. In *Proceedings of the TAG+ Workshop*.

Martine Smets. 1998. Comparison of XTAG and DTG. In *Proceedings of the TAG+ Workshop*.

An Integrated Parser for TFG with Explicit Tree Typing

Marc Cavazza
EIMC Department, University of Bradford
Bradford, BD7 1DP, United Kingdom
M.Cavazza@Bradford.ac.uk
http://www.eimc.brad.ac.uk/~mcavazza

1. Introduction

One of the main conditions for the development of successful NLP applications is the usability of the syntactic formalisms adopted and the degree to which they facilitate syntax-semantics integration. TAG+ formalisms show a real potential for NLP applications, due to their linguistic descriptive capabilities. However, both the standard formalisms and parsing strategies are often quite complex and cannot easily be used as such for the development of NLP systems. We have thus investigated a simplified TAG+ formalism, which sacrifices some of TAG's descriptive and formal properties for the sake of usability. This is especially relevant considering the recent success of empirical approaches to NLP which tend to be based on very simple techniques and/or discard linguistically-motivated formalisms [Basili et al., 1996] [Appelt et al., 1993]. We report the implementation of a parser for a simplified TAG+ formalism, Tree Furcating Grammars (TFG), which integrates semantic processing, performing both syntactic disambiguation and the construction of a semantic representation for the sentence parsed. The parser has been developed for the purpose of real-time speech understanding of sublanguages (i.e., application-dependent vocabularies of 500-1000 words with specific, sometimes quite simplified, syntactic constructs). TAG+ formalisms were initially investigated because of their potential for syntax-semantics integration (see e.g., Abeille [1994]). We will successively describe the rationale for the TFG formalism, the principles underlying the algorithm used and a first assessment of its performance.

2. The Tree Furcating Grammars (TFG) Formalism

Tree Furcating Grammars are a lexicalised TAG+ formalism, in which adjunction is replaced by the *furcation* operation that essentially adds an additional branch to the target node in the initial tree, instead of copying the auxiliary tree under it. The furcation operation was originally introduced in segment grammars [De Smedt & Kempen, 1990]. A detailed comparison of furcation and adjunction has been given by Abeille [1991]. Though some syntactic phenomena are not properly handled by furcation, the fact that it introduces modifiers without embedding them into the tree structure is a definite advantage for syntax-semantics integration, and was the rationale for choosing it[1]. Successive furcations do not increase tree depth and complexity, producing derived trees that retain some properties of dependency trees. These can support the integrated construction of a semantic structure, based on the appropriate association of semantic functions to the tree structures (see below).

We have adapted our tree representations accordingly, by distinguishing between left auxiliary trees (which have a *X root node)[2] and right auxiliary trees (X* root node). The auxiliary symbol is on the root node, as these trees do not have a foot node. Also, in our implementation trees are explicitly typed as left or right auxiliary (l-aux, r-aux), initial and left or right substituable (l-subst, r-subst). Trees can have multiple types, for instance being both right and left substituable or, in the case of some PP trees, both left auxiliary and right substituable (e.g., fig 2, *V-with-N0).

Left (resp. right) auxiliary trees are combined through right (resp. left) furcation. Left and right furcations, as described by De Smedt & Kempen [1990] produce "flat" structures and in that sense differ from left and right adjunction in Tree Insertion Grammars [Schabes & Waters, 1994]. They tend to be closer to the operations described by Nasr [1995] for his dependency-based TAG variant. Also, furcations are not allowed to take place at substituable nodes prior to their substitution, but are allowed on auxiliary nodes (as compared with Schabes & Waters [1994]).

Another goal, which was the result of early experimentation, was to minimise tree traversal operations that can prove computationally expensive. These are minimised due to the representation itself and to the explicit recording of substituable leaves within tree representations. Only the determination of target nodes for furcation still requires tree

[1] We do not make a direct use of the properties of the derived tree, like dominance relations.
[2] With X in {P, N, V, A}.

traversal, but is made easier by the relatively flat structure of the derived trees.

Finally, a set of atomic semantic features, corresponding to the semantic description of the anchor is associated to the root node as well. These semantic features are used for semantic representations as well as selectional restrictions, in the spirit of preference semantics [Wilks, 1975]. Substituable nodes in initial and some auxiliary trees are associated semantic relations, which also constitute an explicit typing. The definition of these semantic relations can be quite specific, as it derives from the specific distributions of the lexicalised trees themselves [Cavazza, 1997].

3. The Integrated Parsing Algorithm

Several parsing algorithms have been described for TAG+, including CKY [Vijay-Shanker & Joshi, 1985] and Earley-type parsers [Schabes & Joshi, 1988] [Schabes et al., 1988] and a deterministic parser [Schabes & Vijay-Shanker, 1990], which was developed for reasons of efficiency (Schabes & Joshi, 1990). The latter has been recently revisited by Kinyon [1997], who proposed an improved LR(0) algorithm. Recently, Nederhof [1998] has described a new LR parsing method and a new recogniser based on Linear Indexed Automata. Specific approaches have also been developed for partial parsing of potentially ungrammatical sentences [Issac, 1994]. Another major source of innovation in parsing has been the many TAG+ variants developed in recent years, such as the "supertagging" approach [Joshi & Srinivas, 1994], dependency formalisms inspired by TAG [Nasr, 1995] and Tree Insertion Grammars [Schabes & Waters, 1994].

Due to the interleaving of syntactic and semantic processing in our system, we have opted for an *ad hoc* strategy, which eventually resulted quite similar to the one described by Nasr [1995]. The main idea is to make the syntactic part of the algorithm as simple as possible and to avoid "hidden" integration of syntax and semantics through contextual constraints on syntactic operations. Rather, keeping the parsing algorithm elementary would offer more space for experimentation and the integration of semantic processing.

The first step, which corresponds to a lexical filtering of the grammar, consists in generating all the possible set of trees (often termed *forests*) compatible with the input string. This step is very similar to the construction of a pushdown stack for trees as described in [Nasr, 1995]. The parsing algorithm consists in scanning the forest left-to-right and determining possible tree fusions from the explicit typing of the trees considered. The process is iterated until the forest is reduced to a single tree or no further operations are possible [Cavazza & Constant, 1996]. All the forests not reduced to a single tree are discarded as unsuccessful parses. Adjacent trees in a forest are considered for a possible fusion on a pairwise basis. From their explicit categories, the corresponding operation is given by a compatibility table. This table specifies the nature of the operation (substitution, furcation, or nil) as a function of the types of the adjacent trees. However, successful operations also depend on the existence of an appropriate target node as well as semantic compatibility (when applicable). In that sense, tree operations are not fully determined by the compatibility table. The target node for substitution is directly recorded in the representation for substituable trees, while target node for furcation is dynamically computed as being the rightmost/leftmost compatible node, including nodes internal to the tree. The forest is scanned left to right without look-ahead and the "cursor" backtracks one position after a successful fusion has been completed. The forest may have to be scanned several times, due to the conjunction of a strict left-to-right scanning with the restrictions imposed on tree operations. For instance, in the parsing of the forest on fig. 2, the first pass essentially assembles the nominal descriptions through furcation, and substitution at the N1 node takes place at the second pass only.

Additional heuristics are used as a declarative control strategy. For instance, whenever a PP tree is both of type l-aux and r-subst (like e.g., *V-with-N0), substitution has to be performed first, thus enabling correct semantic feature propagation, which will be subsequently needed for selectional restriction at (right) furcation time. This can be achieved by attributing precedence to some types; as a result some operations are postponed until proper conditions are met. It should be noted that PP attachments are a major requirement for the processing of definite descriptions, spatial expressions and instrumental actions, which constitute a significant fraction of the requirements for speech-based multimedia applications.

Throughout parsing, there is a full integration of semantic processing[3], which consists both in semantic features propagation and establishment of semantic/functional links for actants and various modifiers. Semantic features for a lexical entry are associated to the tree root and are transferred through furcation operations to the root node of the target tree (fig. 1 and 2). This ensures proper propagation of semantic features to constitute complete semantic frames.

[3] In that sense, our implementation would fall under the "Parallel" + "Generate-and-Test" paradigm for Syntax-Semantics integration [Dahl et al., 1992].

Furcation is responsible for semantic aggregation, while substitution establishes semantic relations between meaning units, essentially through the structure of initial trees of root S. However, furcation can also result in the establishment of semantic relations, for instance instrumental cases, as with *V-with-N0 trees. Figures 1 and 2 illustrate two different cases of selectional restriction, implementing the PP-attachment rules described above[4].

4. Results

The system is implemented in Common LISP and runs on a SGI O2 workstation with an R10000 processor at 150 MHz. Processing of a single forest corresponding to a 10-15 word sentence is regularly carried in 10-20 ms CPU time. The important point is that, even when parsing several forests for a sentence, the user time remains below 200 ms. Though this was measured with small vocabularies (typically less than 300 words), it is expected to remain roughly unchanged with the target application vocabulary being approx. 500 words in size. The reason is that global response times depend on the number of forests to parse, which is a function of the trees/word ratio. This ratio tends to remain stable within small sublanguages and is certainly much smaller than the generic ratio of 7 mentioned in [Schabes & Waters, 1994]. It is interesting to compare these results to the requirements proposed by Goerz and Kessler [1994] for anytime algorithms to be used in speech understanding. They give Result Production Granularity (RPG) values in the range of 10-100 ms, which means that in most cases our parser, developed for similar applications, could fit into that range.

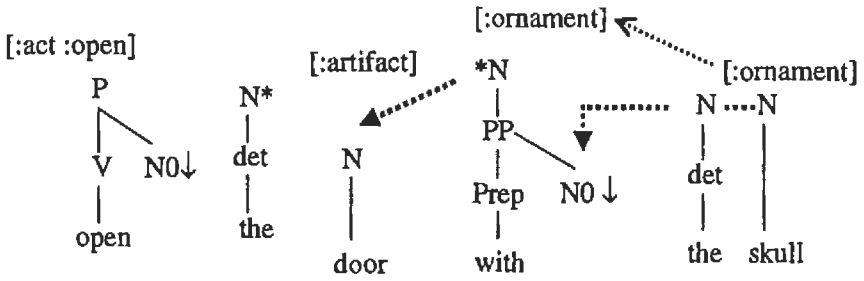


Fig. 1. Semantic propagation through substitution ("N0" node of the PP group) enables selection of right furcation on "N", because of compatibility between :artifact and :ornament features.

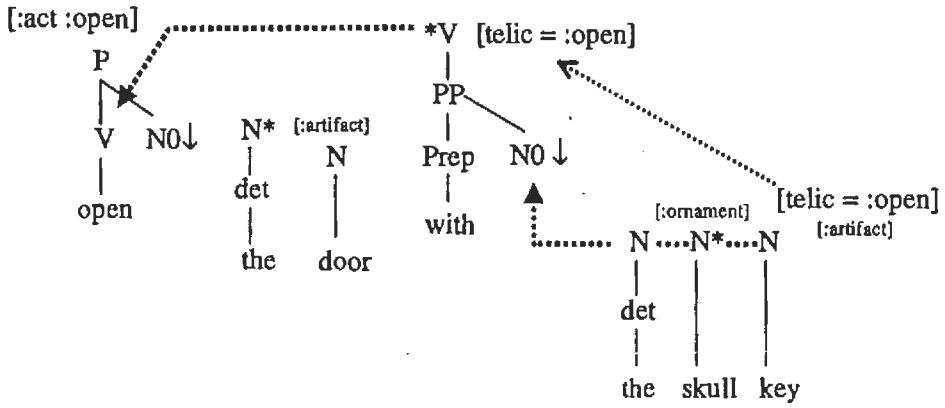


Fig. 2. Semantic propagation through substitution ("N0" node of the PP group) enables selection of right furcation on "V", because of compatibility between "telic" features *and* feature precedence rules.

[4] These refer to situations encountered in the popular "DOOM" video game (trademark of ID Software).

References

Abeillé, A., 1991. Une grammaire lexicalisée d'arbres adjoints pour le francais: application a l'analyse automatique. These de Doctorat de l'Université Paris 7.

Abeillé, A., 1994. Syntaxe et Sémantique: Interactions dans une grammaire d'unification. In: F. Rastier, M. Cavazza, A. Abeille, Sémantique pour l'Analyse. Paris, Masson. (English translation to appear, CSLI, University of Stanford Press).

Appelt, D. E., Hobbs, J. R., Bear, J., Israel, D. & Tyson M. (1993). FASTUS: A Finite-state Processor for Information Extraction from Real-World Text. In *Proceedings of the IJCAI'93 Conference.*

Basili, R., M.T. Pazienza, & Velardi, P., 1996. An Empirical Symbolic Approach to Natural Language Processing, *Artificial Intelligence*, vol. 85.

Cavazza, M. & Constant, P., 1996. Le Traitement Automatique du Langage Naturel et ses Applications. In: J. Caelen (Ed.), Nouvelles Interfaces Homme-Machine, Paris: Tec & Doc.

Cavazza, M., 1997. Sémiotique textuelle et contenu linguistique. *Intellectica*, vol. 23, pp. 53-78.

Dahl, D., Weir, C., Norton, L.M. & Linebarger, M., 1992. Integration of Syntax and Semantics in the Pundit System. *Proceedings of the ANLP'92 Workshop on Fully-Implemented NLP Systems*, IBM Technical Report, TR-80.92-033.

De Smedt, K. & Kempen, G., 1990. Segment Grammars: a Formalism for Incremental Sentence Generation. In: C. Paris (Ed.) *Natural Language Generation and Computational Linguistics*, Dordrecht, Kluwer.

Goerz, G. & Kesseler, M., 1994. Anytime Algorithms for Speech Parsing? *Proceedings of COLING'94*, Kyoto.

Issac, F., 1994. Un algorithme d'analyse pour les grammaires d'arbres adjoints. In: Proceedings of the Third International TAG Workshop, Paris. Technical report TALANA-RT-94-01, Universite Paris 7.

Joshi, A.K, & Srinivas, B., 1994. Disambiguation of Super Parts of Speech (or Supertags): Almost Parsing. *Proceedings of COLING'94*, Kyoto.

Kinyon, A., 1997. Un algorithme d'analyse LR(0) pour les Grammaires d'Arbres Adjoints Lexicalisees. *Proceedings of TALN'97.*

Nasr, A., 1995, A Formalism and a Parser for Lexicalized Dependency Grammars, *Proceedings of the Fourth International Workshop on Parsing Technologies*, Prague.

Nederhof, M.-J., 1998. Linear Indexed Automata and Tabulation of TAG Parsing. *Proceedings of the First Workshop on Tabulation in Parsing and Deduction*, Paris.

Rambow, O., 1994. Multiset-Valued Linear Index Grammars: Imposing Dominance Constraints on Derivations. *Proceedings of the COLING'94 Conference*, Kyoto.

Schabes, Y., Abeillé, A. & Joshi, A.K., 1988. Parsing Strategies with "Lexicalized" Grammars: Applications to Tree-Adjoining Grammars. *Proceedings of COLING'88*, Budapest.

Schabes, Y., & Joshi, A.K., 1988. An Earley-Type Parsing Algorithm for Tree Adjoining Grammars. *Proceedings of ACL'88*, Buffalo.

Schabes, Y. & Joshi, A.K., 1990. Two Recent Developments in Tree Adjoining Grammars: Semantics and Efficient processing. *Proceedings of the Third DARPA Speech and Natural Language Workshop.*

Schabes, Y. & Vijay-Shanker, K, 1990. Deterministic Left to Right Parsing of Tree Adjoining Languages. *Proceedings of ACL'90*, Pittsburgh.

Schabes, Y. & Waters, R.C., 1994. *Tree Insertion Grammar: A Cubic-Time Parsable Formalism That Lexicalizes Context-Free Grammar Without Changing the Trees Produced*, Technical Report TR-94-13, Mitsubishi Electric Research Laboratories, Cambridge (MA).

Wilks, Y., 1975. A Preferential Pattern-seeking Semantics for Natural Language Inference. *Artificial Intelligence*, vol. 6, pp. 53-74.

Synchronous TFG for Speech Translation

Marc Cavazza
EIMC Department, University of Bradford
Bradford, BD7 1DP, United Kingdom
E-mail: M.Cavazza@bradford.ac.uk
http://www.eimc.brad.ac.uk/~mcavazza

1. Introduction: Synchronous TAG+ for Machine Translation

The use of synchronous TAG for Machine Translation has been described by Abeille et al. [1990] and has resulted in several implementations [Prigent, 1994] [Egedi et al., 1994], mainly developed using the XTAG system [Paroubek et al., 1992]. While we subscribe to the general arguments in favour of the use of TAG+ for Machine Translation, it appears that speech translation could constitute an ideal application of these ideas [Harbusch & Poller, 1994]. It is actually easier to select specific areas where speech translation is both feasible and of practical impact (see e.g. the CSTAR, VERBMOBIL and SRI Speech Translation projects). In this paper, we report the implementation of a minimal speech translation prototype based on synchronous TAG+ (more exactly, synchronous Tree Furcating Grammars or STFG), which has been developed as a direct extension of our TFG parser [Cavazza, 1998].

Sheiber & Schabes [1990] originally coined the term "synchronous TAG". They described synchronous derivation of semantic structures from tree operations carried on lexicalised trees. A synchronous TAG is thus a pair of two elementary trees, one representing the source language and the other a logical formula, which is also represented as a variant of TAG (and is lexicalised as well). However, the term of synchronous TAG, when used for machine translation, actually subsumes different approaches and deserves some clarification. The initial presentation of TAG for Machine Translation by Abeille et al. [1990] referred to synchronous TAG, though in fact it directly mapped lexicalised trees to one another, without making recourse to the "semantic" trees described by Shieber & Schabes [1990]. In that sense, it could be considered as a transfer formalism or a structural correspondence system [Kaplan et al., 1989]. Further implementations by Prigent [1994] within the XTAG system [Paroubek et al., 1992] have been based on an extended transfer paradigm, mapping between *derivation* trees in the source and target languages, thus introducing an intermediate representation.

On the other hand, direct mapping between lexicalised trees has also been adopted in the STAG project [Egedi et al., 1994] [Egedi & Palmer, 1994]. We would like to suggest, adopting a terminology from Prigent [1994], that approaches based on the direct mapping between lexicalised trees should be renamed iso-synchronous. This would clearly indicate that the synchronous trees on which adjunction (resp. substitution) operations are carried are of the same kind. Our own implementation follows the iso-synchronous approach, and is based on paired elementary trees in the TFG formalism [Cavazza, 1998].

2. Synchronous Processing of Source and Target Forests

The overall prototype aims at demonstrating real-time speech translation of average 10-15 word sentences in spoken sublanguage areas. It relies on off-the-shelf software both for speech recognition and text-to-speech synthesis. The speech recognition system used in our experiments is the Nuance system (from Nuance Communications), with a British English database. Speech synthesis is based on a Text-To-Speech system, in our case TTS-SDK for French (from Learnout & Hauspie). Our system takes as input an ASCII string in the source language (English), as produced by the speech recognition system, and outputs an

ASCII string in the target language (French), which is passed to the TTS system. This is not to say that our system could be equally applied to the translation of written sublanguages, as the size and syntactic complexity of spoken and written sublanguages differ significantly[1].

The first step consists in selecting the relevant trees from the source language input. This corresponds to the lexical filtering step of the source grammar, and is equivalent to the construction of a set of tree stacks [Cavazza, 1998]. Each tree in the source language is associated a tree in the target language with appropriate mappings from source to target trees at roots, anchors and leaves (see below). The result is a set of candidate forests in the source language to be parsed. For each source forest, there exists an associated forest in the target language. However, it is the processing of the source language forest that fully determines the operations to be carried on the target forest. Parsing a forest involves tree fusion on the basis of adjacent categories, as described in [Cavazza, 1998]. Whenever a pair of adjacent trees (t1, t2) in the source forest undergoes a fusion operation (substitution or furcation), a synchronous operation is carried between the target pair (t1', t2'). In this way, the construction of the target sentence directly proceeds from the analysis of the source sentence. We do not resort to incremental generation of the target sentence, but delay output until the source forest has been entirely and successfully parsed.

Simple difference in constructs between French and English, like those described in Abeille et al. [1990] are handled by linking arguments in the source and target node. Processing arguments in the source forest will then lead to the correct attribution of arguments in the target forest (even though their order might differ, as the parsing algorithm only relies on the source forest order). This also applies to the translation of idioms or when a simple word in the source (resp. target) language does correspond to an idiomatic construction in the target (resp. source) language. Differences in word order for adjectives, like in *la clef bleue vs. the blue key*, are directly reflected in the tree representations, where "bleue" is a left auxiliary tree *N and "blue" a right auxiliary tree. As a result the (N, *N) pair in the source language is matched to a (N, N*) pair for which there would be no fusion. But, because the tree operation is determined by the source forest pair, it is sufficient to adapt the fusion procedure to detect this and perform the correct operation. There are several differences in our formalisation and our implementation with respect to the original description of Abeille et al. [1990]. We establish links only between root nodes and between leaves, hence not relating nodes which are internal to the source and target trees (e.g., "VP" nodes in [Abeille et al. 1990]). This is partly due to the fact that we do not make use of internal categories such as VP and NP, following in that sense both the description given by Abeille (1994) for French and the TFG philosophy, which aims at limiting tree depths. Another difference is that we restrict links between the source and target trees to nodes bearing the same syntactic category. This currently limits our ability to process some structural discrepancies, as in the example *John gave a weak cough / John toussa faiblement*, where an N*-based (left) furcation in the English tree (N*-weak) would correspond to a *V-based (right) furcation in the French tree (*V-faiblement) [Abeille et al., 1990]. However, the system is currently able to process a subset of structural discrepancies. This is illustrated by figure 1., where parsing the source forest for the sentence *the right door lacks a handle* produces as an output *il manque une poignée à la porte de droite*. Adopting the terminology of Dorr [1994] for translation divergences, we should be able to take into account mainly thematic, structural (e.g. "shoot-N0" *vs.* "tirer-sur-N0") and some lexical divergences. However, these points would necessitate further investigation due to the small size of our experiments.

Though the synchronous TAG approach to machine translation is essentially a kind of transfer formalism, we have augmented it with the inclusion of semantic features in order to perform some form of syntactic disambiguation, mainly dealing with PP-attachments. These ambiguities are amenable to selectional restrictions, based on semantic features matching. It could be argued that syntactic disambiguation is not strictly needed for French-to-English translation, as even incorrect attachments might generate correct translations (with similar ambiguities in the source and target languages). However, this would not be fully satisfactory and furthermore, accepting incorrect attachments would result in several forests being fully parsed before a result is produced.

[1] i.e., both in the average length of sentences and in the complexity of syntactic constructs.

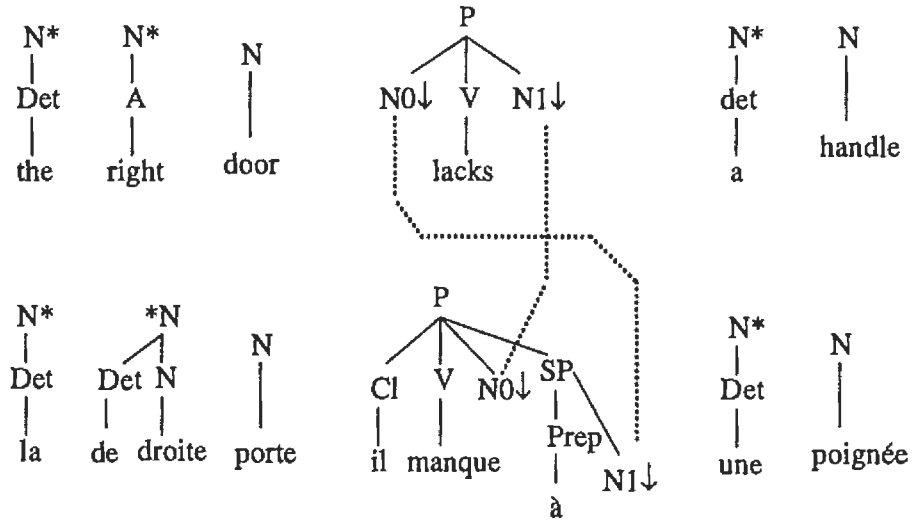


Fig. 1. Source and Target Forests with Synchronous Trees Aligned

3. Preliminary Evaluation

A first version of the system has been developed and tested with a small vocabulary of less than 200 lexical entries. Constructs dealt with include idiomatic expressions, transitive/intransitive constructs, differences in word order, and a subset of translation divergences. The system is written in Common LISP and runs on a SGI O2 with a R10000 processor at 150 MHz. The translation of a 10-15 word sentence is carried in 10-100 ms CPU time, depending on sentence complexity, essentially the number of PP-attachments. Performance of the system is not related to the size of the lexicon but rather to the tree/word ratio, which determines the number of forests to be parsed during the analysis of a given sentence (see [Cavazza, 1998]). This would make possible speech translation in user real-time (i.e., total time < 1 s), considering the time required by the speech recognition and speech synthesis components.This approach has been mainly developed for the translation of constrained languages or application-related sublanguages.

We do not claim it to be appropriate for written language translation, which requires the ability to process much longer sentences and a larger range of syntactic constructs. Further work would explore the usability of such a system in collaborative multimedia applications.

References

Abeillé, A. & Schabes, Y., 1990. Non compositional discontinous constituents in Tree Adjoining Grammars. *Proceedings of the International Symposium on Discontinous Constituency*, Tilburg.

Abeillé, A., 1993. *Les Nouvelles Syntaxes: Grammaires d'unification et analyse du francais*. Paris: Armand Colin.

Abeillé, A., Schabes, Y. & Joshi, A.K., 1990. Using Lexicalized Tags for Machine Translation. *Proceedings of COLING'90*, Helsinki.

Cavazza, M., 1998. An Integrated Parser for TFG with Explicit Tree Typing. *Proceedings of the Fourth TAG+ Workshop,* University of Pennsylvania (this volume).

Dorr, B., 1994. Machine Translation Divergences: A Formal Description and Proposed Solution. *Computational Linguistics*, 20:4, pp. 597-633.

Egedi, D. & Palmer, M., 1994. Constraining lexical selection across languages using TAG. *Proceedings of the Third International TAG Workshop*, Paris. Technical report TALANA-RT-94-01, Universite Paris 7.

Egedi, D., Palmer, M., Park, H.S. & Joshi, A.K., 1994. Korean to English Translation Using Synchronous TAGs. *Proceedings of the First Conference of the association for Machine Translation in the Americas*, Columbia.

Harbusch, K. & Poller, P., 1994. Structural translation with synchronous rewriting systems. *Proceedings of the Third International TAG Workshop*, Paris. Technical report TALANA-RT-94-01, Universite Paris 7.

Kaplan, R.M., Netter, K., Wedekind, J. & Zaenen, A., 1989. Translation by Structural Correspondences. *Proceedings of EACL'89*, Manchester.

Paroubek, P., Schabes, Y. & Joshi, A.K., 1992. XTAG – A Graphical Workbench for Developing Tree-Adjoining Grammars. *Proceedings of ANLP'92*, Trento.

Prigent, G., 1994. Synchronous TAGs and Machine Translation. *Proceedings of the Third International TAG Workshop*, Paris. Technical report TALANA-RT-94-01, Universite Paris 7.

Schieber, S.M. & Schabes, Y., 1990. Synchronous Tree-Adjoining Grammars. *Proceedings of COLING'90*, Helsinki.

A Tabular Interpretation of Bottom-up Automata for TAG

Eric de la Clergerie
INRIA
Domaine de Voluceau
Rocquencourt, B.P. 105
78153 Le Chesnay Cedex
France
Eric.Clergerie@inria.fr

Miguel A. Alonso Pardo
Depto. de Computación
Universidad de La Coruña
Campus de Elviña s/n
15071 La Coruña
Spain
alonso@dc.fi.udc.es

David Cabrero Souto
Centro Ramón Piñeiro para a
Investigación en Humanidades
Estrada Santiago-Noia km 3
15896 Santiago de Compostela
Spain
dcabrero@cirp.es

Abstract

We present a tabular interpretation for a class of 2–Stack Automata that may be used to describe bottom-up parsing strategies for TAGs. The results are also useful for tabulating other existing bottom-up automata models for this kind of languages.

1 Introduction

Several extensions of push-down automata has been proposed as operational devices for describing parsing strategies for TAGs. Embedded Push-Down Automata [EPDA] (Vijay-Shanker, 1988) and 2–Stack Automata [2–SA] (Becker, 1994) are suitable operational devices for top-down strategies. For bottom-up strategies, Bottom-up EPDA [BEPDA] (Schabes and Vijay-Shanker, 1990; Rambow, 1994) and Linear Indexed Automata [LIA] (Nederhof, 1998) have been proposed.

We classify parsing strategies for TAGs w.r.t. the way adjoining is recognized and regardless of how elementary trees are traversed. In *Top-Down* strategies, the auxiliary tree to be adjoined is predicted once the adjoining node has been reached. Examples are the Earley-like parsing algorithms which preserve the correct prefix property (Nederhof, 1997). Conversely, in *Bottom-Up* strategies, adjoining is considered only when a candidate auxiliary tree has been completely traversed. Examples are the popular CYK-like (Vijay-Shanker and Joshi, 1985) and Earley-like parsing algorithms without the valid prefix property (Schabes, 1991).

A TAG parser must handle elementary tree traversing as well as adjoining processing and keep some information about these two kinds of task. Then, a 2-stack automata is adequate to implement parsing algorithms for TAG.

Polynomial time complexity can be lost for a non deterministic grammar if redundant computations are not discarded using some kind of dynamic programming (tabular) techniques. For the above mentioned automata models, systematic tabulation is only available for LIA.

The automata model proposed in this paper for bottom-up parsing strategies presents the following characteristics: separation of the tree traversal and adjunction information by using two stacks; systematic tabulation, achieving $O(n^6)$ time complexity and $O(n^4)$ space complexity; and results comparable with existing tabular algorithms for TAGs.

2 (Strongly-driven) bottom-up 2–Stack Automata

Strongly Driven 2–Stack Automata [SD 2–SA] has been introduced in (de la Clergerie and Alonso Pardo, 1998) to describe arbitrary parsing strategies for TAGs. They work on 2 stacks with some restrictions added to make them equivalent, w.r.t. the recognized languages, to the class of tree adjoining languages.

A SD 2–SA uses the **Master Stack MS** to drive the evaluation and the **Auxiliary Stack AS** for restricted bookkeeping. Actually, AS should be considered as a stack of stacks, each of them representing a session. Typically, in TAG parsing, a session contains a sequence of adjunctions done along the spines of auxiliary trees. A session starts in mode w (write) where pop action are forbidden on MS and switches at some point to mode e (erase) where push actions are forbidden on MS. The actions on AS in mode e should faithfully retrace the actions done in mode w. Exiting a session is only possible when reaching back (in e mode) the MS element that initiated the session and when the session stack on AS is empty.

The bottom-up "projection" of SD 2–SA, henceforth BU 2–SA, imposes an additional restriction: AS must remain empty in mode w. That means that adjunction can be only recognized when a complete auxiliary tree has been constructed. The different behaviors of SD 2–SA and BU 2–SA are obvious when comparing the shape of derivations as illustrated in Fig. 1, where the axis display the stack sizes.

More formally, a BU 2–SA $\mathcal{A}$ is specified by a 6-tuple $(\Sigma, \mathcal{M}, \mathcal{X}, \$_0, \$_f, \Theta)$ where Σ denotes the finite set of terminals, $\mathcal{M}$ the finite set of master stack elements and $\mathcal{X}$ the finite set of auxiliary stack elements. The init symbol $\$_0$ and final symbol $\$_f$

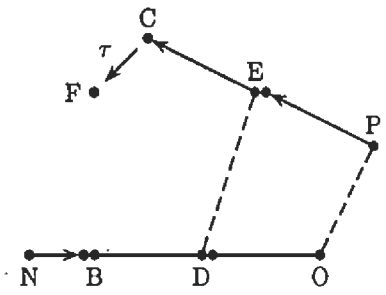


Figure 4: Application of Rule 1

Space complexity of the tabular technique for BU 2–SA is obviously $\mathcal{O}(n^4)$ as at most 4 indices are stored in buXCF items.

5 Related work

Our tabular interpretation may be used to re-interpret other existing tabular algorithms for TAGs, based on some automata model or not.

Linear Indexed Automata [LIA] (Nederhof, 1998) is the only other automata model we are aware of that has an associated tabular algorithm. This algorithm considers items $((B,C,i,j),(\diamond,\square,\square,0,0))$ corresponding to buCF items $B\delta Cm$, as well as items $((B,C,i,j),(c,D,E,p,q))$ corresponding to buXCF items $B\triangleright[DE]\bar{C}e$. Because LIAs work on a stack of stacks, the empty stack markers we use are useless, the $\models$ mark being implicit when the second part of an item is equal to $(\diamond,\square,\square,0,0)$.

If we now consider the tabular algorithm of (Vijay-Shanker and Weir, 1994), which is not based on an automata model, we find that, using their terminology, our buXCF items $B\triangleright[DE]\bar{C}e$ correspond to a *head* $B\bar{C}$ with a *terminator pointer* $[DE]$ and buCF items to a head, without terminator pointer.

In both cases, marks and modes (w and e) are absent from the proposed items, but one may show that they are actually implicitly present. They may be also be discarded from our items when considering specific parsing strategies, but are needed if one wishes to exploit the full potentiality of BU-2SA, for instance for more complex parsing strategies.

6 Conclusion

Bottom-up 2–SA may be seen as the projection of a subclass of strongly-driven 2–SA, specialized to describe parsing strategies for TAG where adjunction is recognized in a bottom-up way (i.e. when being in mode erase). A tabular interpretation of BU 2–SA is straightforwardly derived by "projecting" the tabular interpretation for SD 2–SA. So, a buXCF item $B\triangleright[DE]\bar{C}e$ is the projection of a XCF item $AB\delta[\bar{D}E]\bar{C}e$ and a buCF item $B\delta Cm$ is the projection of a CF item $AB\delta Cm$. For SD 2–SA, A is needed to handle popping on AS in w mode, but it may be safely removed for BU 2–SA because of the extra condition on the emptiness of **AS** in w mode. While the worst case time complexity remains $\mathcal{O}(n^6)$, the worst case space complexity decreases from $\mathcal{O}(n^5)$ for 2–SA to $\mathcal{O}(n^4)$ for BU 2–SA. Of course, the drawback is the violation of the valid-prefix property and it remains to investigate whether or not this is a good thing for TAG grammars used in Natural Language Processing.

7 Acknowledgements

This work has been partially supported by the European Union (1FD97-0047-C04-02), Government of Spain (HF97-223) and Xunta de Galicia (XUGA10505B96 and XUGA20402B97).

References

Tilman Becker. 1994. A new automaton model for TAGs: 2-SA. *Computational Intelligence*, 10(4):422–430.

Eric de la Clergerie and Miguel A. Alonso Pardo. 1998. A tabular interpretation of a class of 2-Stack Automata. In *Proc. COLING/ACl'98*, Montreal, Canada, August.

Mark-Jan Nederhof. 1997. Solving the correct-prefix property for TAGs. In T. Becker and H.-V. Krieger, editors, *Proc. of the Fifth Meeting on Mathematics of Language*, pages 124–130, Schloss Dagstuhl, Saarbruecken, Germany, August.

Mark-Jan Nederhof. 1998. Linear indexed automata and tabulation of TAG parsing. In *Proc. of First Workshop on Tabulation in Parsing and Deduction (TAPD'98)*, pages 1–9, Paris, France, April.

Owen Rambow. 1994. *Formal and Computational Aspects of Natural Languege Syntax*. Ph.D. thesis, University of Pennsylvania.

Yves Schabes and K. Vijay-Shanker. 1990. Deterministic left to right parsing of tree adjoining languages. In *Proc. of 28th Annual Meeting of the Association for Computational Linguistics*, pages 276–283, Oittsburgh, Pennsylvania, USA, June.

Yves Schabes. 1991. The valid prefix property and left to right parsing of tree-adjoining grammar. In *Proc. of II International Workshop on Parsing Technologies, IWPT'91*, pages 21–30, Cancún, Mexico.

K. Vijay-Shanker and Aravind K. Joshi. 1985. Some computational properties of tree adjoining grammars. In *23rd Annual Meeting of the Association for Computational Linguistics*, pages 82–93, Chicago, IL, USA, July.

K. Vijay-Shanker and David J. Weir. 1994. Parsing some constrained grammar formalisms. *Computational Linguistics*, 19(4):591–636.

K. Vijay-Shanker. 1988. *A Study of Tree Adjoining Grammars*. Ph.D. thesis, University of Pennsylvania, January.

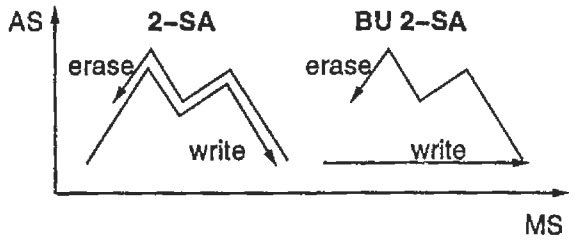


Figure 1: Derivation shapes for SD and BU 2–SA

are distinguished elements of $\mathcal{M}$. Θ is a finite set of transitions.

MS is a word in $(\mathcal{D}\mathcal{M})^*$ where $\mathcal{D}$ denotes the set $\{\triangleright, \models\}$ of action marks, projection of the larger action mark set $\{\nearrow, \rightarrow, \searrow, \models\}$ used for SD-2SA. Pushing an element on MS is either marked with $\models$ if a "new session" starts at the same time, or by $\triangleright$ otherwise.

AS is a word of $(\mathcal{K}\mathcal{X}^*)^*$ where symbols in $\mathcal{K} = \{\models^{\mathrm{w}}, \models^{\mathrm{e}}\}$ are used to delimit session stacks and remember the mode of the previous session.

Given some input string $x_1 \ldots x_n \in \Sigma^*$, a configuration of $\mathcal{A}$ is a tuple (m, i, Ξ, ξ) where $m \in \{\mathrm{w}, \mathrm{e}\}$ denotes the current mode, i the current string position in $[0, n]$, Ξ the master stack and ξ the auxiliary stack. The initial configuration of $\mathcal{A}$ is $(\mathrm{w}, 0, \models\$_0, \models^{\mathrm{w}})$ and the final one $(\mathrm{e}, n, \models\$_f, \models^{\mathrm{w}})$.

A transition τ is represented by a pair $(m, \Xi, \xi) \xmapsto{z} (m', \Theta, \theta)$ where $m, m' \in \{\mathrm{w}, \mathrm{e}\}$, z in Σ^*, Ξ and Θ are suffixes of master stacks in $\mathcal{M}(\mathcal{D}\mathcal{M})^*$, and ξ, θ suffixes of auxiliary stacks in $(\mathcal{X} \cup \mathcal{K})^*$. We denote $(m, i, \Psi\Xi, \psi\xi) \vdash (m', j, \Psi\Theta, \psi\theta)$ a valid derivation step using τ with $z = x_{i+1} \ldots x_i$, and by $\vdash^*$ the reflexive and transitive closure of $\vdash$. A string $a_1 \ldots x_n$ is accepted by $\mathcal{A}$ if $(\mathrm{w}, 0, \models\$_0, \models^{\mathrm{w}}) \vdash^* (\mathrm{e}, n, \models\$_f, \models^{\mathrm{w}})$.

For BU 2–SA, we consider the following kinds of transitions (which enforce that the AS topmost session remains empty in w mode), namely SWAP to change the top element of the MS; $\models$–WRITE and $\models$–ERASE to start and end sessions; and $\triangleright$–WRITE and δ–ERASE ($\delta \in \{\nearrow, \rightarrow, \searrow\}$) to push to and pop from MS while acting on AS:

SWAP1 $(p, A, \epsilon) \xmapsto{z} (p, B, \epsilon)$

SWAP2 $(\mathrm{w}, A, \models^{o}) \xmapsto{z} (\mathrm{e}, B, \models^{o})$

$\models$–WRITE $(m, A, \epsilon) \xmapsto{z} (\mathrm{w}, A \models B, \models^{m})$

$\models$–ERASE $(\mathrm{e}, A \models B, \models^{m}) \xmapsto{z} (m, C, \epsilon)$

$\triangleright$–WRITE $(\mathrm{w}, A, \epsilon) \xmapsto{z} (\mathrm{w}, A \triangleright B, \epsilon)$

δ–ERASE $(\mathrm{e}, A \triangleright B, c) \xmapsto{z} (\mathrm{e}, C, c')$ with $(\delta = \rightarrow$ and $c = c' = \epsilon)$ or $(\delta = \nearrow$ and $c = \epsilon)$ or $(\delta = \searrow$ and $c' = \epsilon)$.

3 TAG parsing with BU 2–SA

We present a BU 2–SA that simulates a Earley-like parsing algorithm without the valid-prefix property (Schabes, 1991). The automata performs full prediction on the context-free backbone but no prediction on the adjunctions during the descent phase.

Each elementary tree is represented by a set of context free productions of the form $\nu_{k,0} \rightarrow \nu_{k,1} \ldots \nu_{k,n_k}$, where $\nu_{k,0}$ denotes some non-leaf node k and $\nu_{k,i}$ the i^{th} son of k, and a set of terminal productions $\nu_{k,0} \rightarrow a_k$, where $\nu_{k,0}$ denotes some leaf node k with terminal label a_k.

The 6-tuple $(V_T, \mathcal{M}, \mathcal{X}, \nu_{0,0}, \nu_{0,0}', \Theta)$ defines the automata $\mathcal{A}$, with $\mathcal{M} = \{\nabla^{\alpha}_{k,i}\} \cup \{\nu^{\alpha}_{k,i}\} \cup \{\nu^{\alpha}_{k,i}{}'\}$ and $\mathcal{X} = \{\nabla^{\alpha}_{k,i}\}$, where symbols $\nabla_{k,i}$ denote dotted productions and $\nu^{\alpha}_{k,i}$ (resp. $\nu^{\alpha}_{k,i}{}'$) denote the prediction (resp. successful recognition) of a node. The transitions are given by the following rules:

- Call / Return for a node not on a spine. The call starts a new session, exited at return.

CALL : $(m, \nabla_{k,i}, \epsilon) \longmapsto (\mathrm{w}, \nabla_{k,i} \models \nu_{k,i+1}, \models^{m})$
RET : $(\mathrm{e}, \nabla_{k,i} \models \nu_{k,i+1}{}', \models^{m}) \longmapsto (m, \nabla_{k,i+1}, \epsilon)$

- Call / Return for an adjunction on node $\nu_{k,0}$. The computation is diverted to parse some acceptable auxiliary tree β with root node r_β. At return we check if the subtree attached to the foot node of β corresponds to the subtree rooted by $\nu_{k,0}$.

ACALL : $(\mathrm{w}, \nu_{k,0}, \epsilon) \longmapsto (\mathrm{w}, \nu_{k,0} \triangleright r_\beta, \epsilon)$
ARET : $(\mathrm{e}, \nu_{k,0} \triangleright r_\beta{}', \nabla_{k,n_k}) \longmapsto (\mathrm{e}, \nu_{k,0}{}', \epsilon)$

- Call / Return for a node $\nu_{k,i+1}$ on a spine. The adjunction stack is propagated bottom-up along the spine.

SCALL : $(\mathrm{w}, \nabla_{k,i}, \epsilon) \longmapsto (\mathrm{w}, \nabla_{k,i} \triangleright \nu_{k,i+1}, \epsilon)$
SRET : $(\mathrm{e}, \nabla_{k,i} \triangleright \nu_{k,i+1}{}', \epsilon) \longmapsto (\mathrm{e}, \nabla_{k,i+1}, \epsilon)$

- Call / Return for a foot node f_β. A candidate adjunction node for β is predicted. At return we remember what node was considered.

FCALL : $(\mathrm{w}, f_\beta, \epsilon) \longmapsto (\mathrm{w}, f_\beta \triangleright \nabla_{k,0}, \epsilon)$
FRET : $(\mathrm{e}, f_\beta \triangleright \nabla_{k,n_k}, \epsilon) \longmapsto (\mathrm{e}, f_\beta{}', \nabla_{k,n_k})$

- Production Selection
  SEL : $(\mathrm{w}, \nu_{k,0}, \epsilon) \longmapsto (\mathrm{w}, \nabla_{k,0}, \epsilon)$
- Production Publishing
  PUB : $(m, \nabla_{k,n_k}, \epsilon) \longmapsto (\mathrm{e}, \nu_{k,0}{}', \epsilon)$
- Scanning
  SCAN : $(\mathrm{w}, \nu_{k,0}, \models^{m}) \xmapsto{a_k} (\mathrm{e}, \nu_{k,0}{}', \models^{m})$

4 Tabulation

In a tabular framework, items store essential information about characteristics "points" of elementary derivations. Tabulation of SD 2-SA (de la Clergerie and Alonso Pardo, 1998), that achieves

$\mathcal{O}(n^6)$ time and $\mathcal{O}(n^5)$ space complexity, needs two kinds of items, namely 3-point Context-Free [CF] items and 5-point escaped Context-free [XCF] items. Each point is either a *mini configuration* $\langle i, A, a\rangle$ or a *micro configuration* $\langle i, A\rangle$ that stores some relevant information about a configuration, namely the position i in the input string, the top MS element A, and optionally the top AS element a. The uppermost curve of Fig. 2 illustrates a 3-point CF item $[\langle h, A, -\rangle, \langle i, B, -\rangle, \delta, \langle j, C, c\rangle]$, also denoted $B\delta\bar{C}\mathrm{w}$ where A and B are micro configurations and $\bar{C}$ is a mini configuration. The uppermost curve of Fig. 3 illustrates a 5-point XCF item $[\langle h, A, -\rangle, \langle i, B, -\rangle, \delta, \langle p, D, d\rangle, \langle q, E, -\rangle, \langle j, C, c\rangle]$, also denoted $AB\delta[\bar{D}E]\bar{C}\mathrm{e}$ where A, B, E (resp. $\bar{D}$, $\bar{C}$) are micro (resp. mini) configurations.

BU 2-SA restrictions imply that AS remains empty in w mode, so the points A, B and $\bar{C}$ of a CF item and the points A, B and $\bar{D}$ of a XCF item are "projected" w.r.t. the top element of the AS. Furthermore, it may be shown that point A is actually redundant and can be discarded. The bottom curve of Fig. 2 illustrates a BU 2-SA CF item $[\langle i, B, -\rangle, \triangleright, \langle j, C, c\rangle]$, also denoted as $B\triangleright\bar{C}\mathrm{w}$ The bottom curve of Fig. 3 illustrates a BU 2-SA XCF item $[\langle i, B, -\rangle, \triangleright, \langle p, D, \models^o\rangle, \langle q, E, -\rangle, \langle j, C, c\rangle]$, also denoted as $B\triangleright[DE]\bar{C}\mathrm{e}$. In both figures, the projection is materialized by the dashed arrows.

Formally, we identify two kinds of items for BU 2-SA, associated to two different kinds of derivations:

Bottom-up CF [buCF] items correspond to context-free derivations that depend only on the topmost element of MS

$$\begin{array}{c}(\mathrm{w}, i, \Xi B, \xi\models^o) \vdash^* (m, j, \Xi B\triangleright C, \xi\models^o)\\ \text{or}\\ (o, i, \Xi B, \xi) \vdash^* (\mathrm{w}, j, \Xi B\models C, \xi\models^o)\end{array}$$

and are denoted by $B\delta\bar{C}m$, where $B = \langle i, B\rangle$, $\bar{C} = \langle j, C, \models^o\rangle$, and $\delta \in \mathcal{D}$.

Bottom-up Escaped CF [buXCF] items correspond to escaped context-free derivations of the form:

$$\begin{array}{rcl}(\mathrm{w}, i, \Xi B, \xi\models^o) & \vdash^* & (\mathrm{w}, p, \Xi\Phi D, \xi\models^o)\\ & \vdash^* & (\mathrm{e}, q, \Xi\Phi D\triangleright E, \xi\models^o\phi)\\ & \vdash^* & (\mathrm{e}, j, \Xi B\triangleright C, \xi\models^o\phi c)\end{array}$$

and are denoted by $B\triangleright[DE]\bar{C}\mathrm{e}$, where $B = \langle i, B\rangle$, $D = \langle p, D\rangle$, $E = \langle q, E\rangle$, $\bar{C} = \langle j, C, c\rangle$.

A set of rules combines items and transitions in order to retrieve all possible derivations. Due to space limitations, we only describe the most complex rule (see Fig. 4), used to apply a transition $\tau = (\mathrm{e}, B\triangleright C, c) \xmapsto{z} (\mathrm{e}, F, \epsilon)$, omitting the scanning constraint z on the input string:

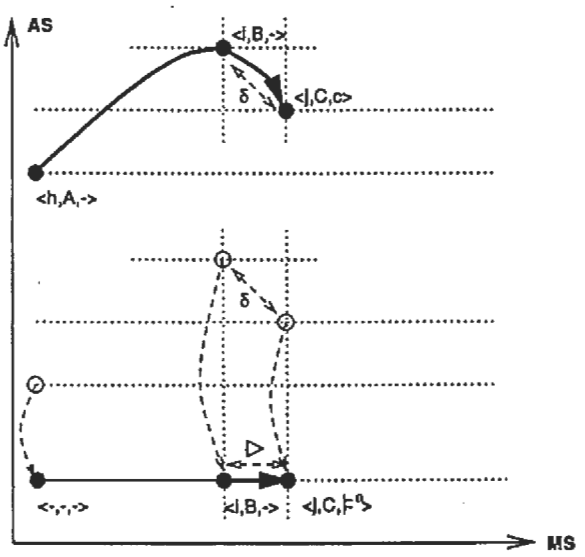


Figure 2: CF items for SD 2-SA and BU 2-SA

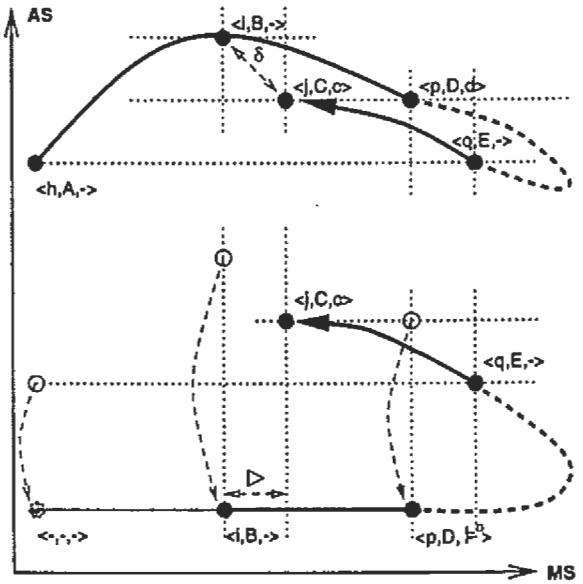


Figure 3: XCF items for SD 2-SA and BU 2-SA

$$\left.\begin{array}{l}\bar{B}^\circ\triangleright[D\bar{E}^\circ]\bar{C}\mathrm{e}\\ N\delta\bar{B}\mathrm{w}\\ D\triangleright[OP]\bar{E}\mathrm{e}\end{array}\right\} \xRightarrow{\tau} N\delta[OP]\bar{F}\mathrm{e} \qquad (1)$$

where $\bar{C} = \langle j, C, c\rangle$, $B = \langle i, B, \models^o\rangle$, $\bar{F} = \langle k, F, b\rangle$, and $\bar{B}^\circ = \langle i, B\rangle$ the projection of $\bar{B}$ to a micro configuration.

The time complexity of this rule is $\mathcal{O}(n^7)$ but may be reduced to $\mathcal{O}(n^6)$ by partially applying the rule on the first two items to build an intermediary structure where B is discarded.

TAG Derivation as Monotonic C-Command*

Robert Frank
Dept. of Cognitive Science
Johns Hopkins University
rfrank@cogsci.jhu.edu

K. Vijay-Shanker
Dept. of Computer and Information Sciences
University of Delaware
vijay@cis.udel.edu

The TAG adjunction operation operates by splitting a tree at one node, which we will call the adjunction site. In the resulting structure, the subtrees above and below the adjunction site are separated by, and connected with, the auxiliary tree used in the composition. As the adjunction site is thus split into two nodes, with a copy in each subtree, a natural way of formalizing the adjunction operation posits that each potential adjunction site is in fact represented by two distinct nodes. In the FTAG formalism (Vijay-Shanker, 1988) each potential adjunction site is associated with two feature structures, one for each copy. As an alternative to this operationally defined rewriting view of adjunction, Vijay-Shanker (1992) suggests that TAG derivations instead be viewed as a monotonic growth of structural assertions that characterize the structures being composed. This proposal rests crucially on the assumption that the elementary trees are characterized in terms of a domination relation among nodes, and that each potential adjunction site is represented by two nodes standing in a domination relation. Under this proposal, the structures α and β in Figure 1 would be used to derive long-distance wh-movement. To adjoin β into α, the root and foot nodes of β are identified with the two C′ nodes standing in a domination relation in α (represented by the dotted line). This domination relation still holds after adjunction, as do all the other domination relations stated in defining α and β. (In sentences in which there is no adjoining at the C′ node, e.g., 'I wonder what Mary saw,' these C' nodes could collapse, preserving domination under the assumption that it is a reflexive relation.) Domination has also been argued to play a role in multi-component structures, where there is assumed to be a domination relationship between a frontier node of one component and the root of the other.

While the use of domination relationships is attractive in allowing us to view TAG derivations as monotonic additions to a set of domination relations, the linguistic motivation for such domination statements among duplicated nodes is not very clear. Instead, from the point of view of the grammar, what seems to be crucial in defining the relevant portion of the structure of α is not that there should be two C′ nodes standing in a domination relation, but rather that the moved element 'what' must stand in a certain structural relation with its trace, namely c-command, both in the the elementary tree and throughout the derivation. Given the way in which adjunction is defined and the manner in which domination statements have been utilized, it turns out that this c-command relation is always preserved by the application of adjunction. In this work, we take this preservation of c-command under adjunction to be the central property of the operation, and not a residual effect of some specific use of dominance relations and their interaction with adjunction. Thus, what was previously seen as the central preservation of dominance relations will turn out to arise as a side effect of the preservation of c-command relations on our proposal. This leads us to postulate that TAG elementary structures are defined in terms of their c-command relations, and that TAG derivations constitute monotonic additions to a set of c-command relations. That is, instead of viewing TAG structures being defined in terms of domination relations, we consider any domination relations that will be preserved to arise or be inferred from the c-command relations used in defining TAG structures.

In characterizing TAG elementary trees, we make use of independently motivated assumptions concerning the c-command relations that exist among structural elements. Thus, we assume that the c-command relations within elementary trees will be determined by (at least) the following principles (cf. the definitions in Kayne (1994)):[1]

(1)a. A moved element c-commands its trace.

b. A head and its complement c-command one an-

*Thanks to Tony Kroch, Seth Kulick, and two anonymous reviewers for helpful comments and discussion. We gratefully acknowledge the financial support of NSF grants SBR-97-10247 and SBR-97-10411.

[1]We leave for the moment the question of the relationship between specifiers and the X′ projections they specify.

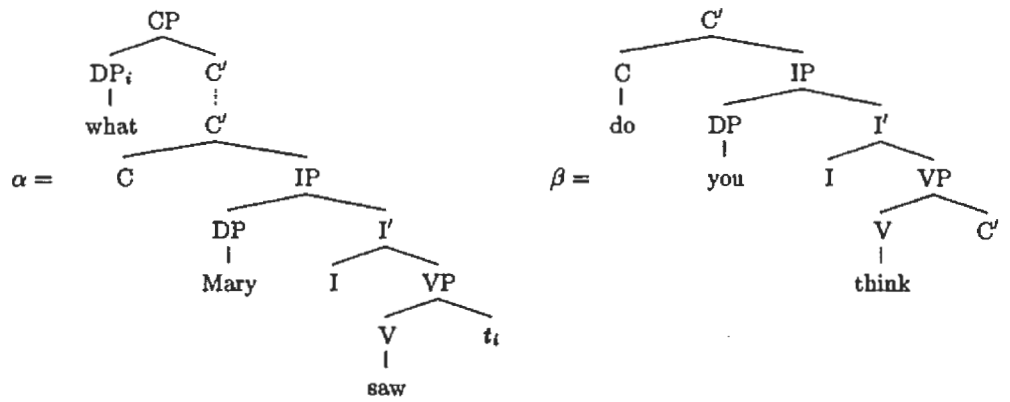


Figure 1: Preservation of Domination in TAG Derivation

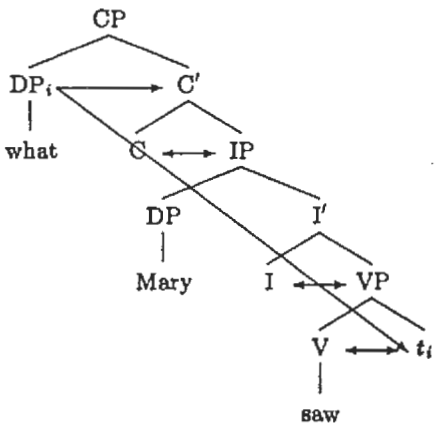


Figure 2: Defining elementary trees with c-command

other.

c. A modifier c-commands the phrase it modifies.

Following these principles leads us to the structure in Figure 2 for the elementary tree α from Figure 1 (where arrows indicate c-command relations).[2] There are two crucial c-command relations to observe in this structure: the first between the fronted wh-phrase and its trace, and the second between the wh-phrase and the C′ node, which serves as the target of movement within the elementary tree. Let us suppose that derivations proceed as monotonic combinations of structures like this one defined in terms of c-command. This means that we can perform an operation analogous to adjunction, inserting a structure like β in Figure 1 between the fronted wh-element and the C′, by identifying this C′ with the foot node of the auxiliary structure. In the structure that results, all of the c-command relations stated in the elementary trees are preserved, most notably those between the fronted wh-element and both the C′ and its trace. From this perspective, we can now understand why it was necessary in the framework of Vijay-Shanker (1992) to posit a domination relation between the two C′ nodes in α in Figure 1: as an indirect representation of (at least) the principle requiring that moved elements c-command their traces.

This proposal allows us to explain many previously stipulated properties of TAG elementary trees and constraints on the adjunction operation. Consider, first of all, the structural differences between two classes of auxiliary trees noted by Kroch (1989) and Schabes and Shieber (1994): *complement* auxiliary trees on the one hand and *modifier* or *athematic* auxiliaries on the other. Recall that modifier auxiliaries have the distinctive property that their foot node is the sister of a modifying phrase and is the daughter of the root node. Following the principles in (1), it follows that the foot of a modifier auxiliary will c-command its XP sister, i.e., the adjunction site, though not vice versa. In contrast, the foot node of a complement auxiliary must be the sister of some head of which it is a complement. Thus, this foot node will both c-command and be c-commanded by its sister node. From this structural difference, we can derive certain contrasts in the use of these classes of auxiliaries during TAG derivations. Since modifier auxiliary trees introduce an asymmetrical c-command relation with their foot node, it follows that their adjunction will not disrupt any c-command relations that the modified phrase already enters into. Thus, it follows the adjunction of mod-

[2]The linkages of direct domination in Figure 2 are not intended as part of the representation, but rather as aid to the reader in comparing our proposed structure to that standardly assumed. Note that certain implicit c-command relations, such as that between C and the subconstituents of IP are suppressed in this figure, but we assume that they are present. See Frank and Vijay-Shanker (1998) for extensive discussion of the properties of structures defined in terms of c-command and the relationship between such structures and those defined in terms of dominance.

ifier auxiliaries should be quite free and indeed may occur at any node in an elementary tree. In fact, if the root and foot of the auxiliary tree are considered segments of the same category (which explains the asymmetrical c-command relation between the modifier and modifiee), this would explain the possibility of multiple adjunction by modifier auxiliary trees at a single node considered by Shieber and Schabes. On the other hand, it has sometime been stipulated that adjunction of predicative auxiliaries is blocked at the foot node of predicative auxiliary trees. As just noted, since the foot of a predicative auxiliary is a complement, this node c-commands the lexical head of the auxiliary. Adjoining to this foot node by another predicative auxiliary tree will have the effect of lowering it, so that it no longer c-commands the head. This would violate the monotonicity requirement on c-command relations during the derivation, and we could therefore reduce the stipulation often used in TAG to a more general condition on monotonicity. In contrast, adjunction at the foot node of a modifier auxiliary will not be ruled out, as the modification relation does not entail mutual c-command, and such lowering of the foot does not force the retraction of any c-command relations.

Now that we have seen that complement auxiliary trees may not adjoin at a complement node, the obvious question is where they may adjoin. Clearly, adjoining at the root of a structure would not require any statements of c-command relations to be retracted, and thus is permissible. But this is not an interesting situation as it can also be considered to be substitution. Saying that this derivation step is a case of adjunction is merely an artifact of the TAG formalism which, quite possibly, has no significant implications. The interesting cases correspond to adjoining complement auxiliary trees to internal nodes (i.e., non-root nodes). Suppose that we follow Kayne's (1994) suggestion that specifier positions should be assimilated to adjuncts, specifically with respect to their c-command relations (i.e., they c-command but are not c-commanded by their X′ sister).[3] This will mean that we must add the following additional principle of elementary tree formation to those in (1):

(2) A specifier c-commands the phrase to which it attaches.

¿From this, we are able to derive the result that the only internal (non-root) nodes where predicative auxiliary trees can adjoin are X′ nodes that are sister to a specifier. The reason for this is exactly as

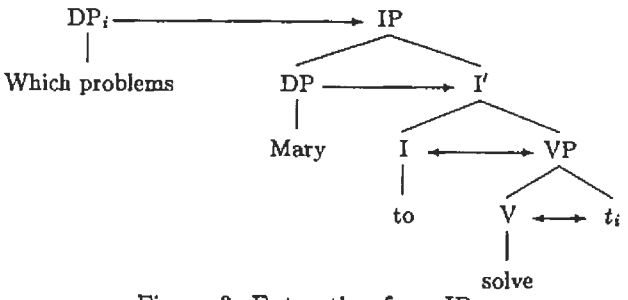


Figure 3: Extraction from IP

in our discussion of the tree in Figure 2, namely that it is only in the context of unidirectional c-command from the specifier to the X′ node that it is possible to insert a complement auxiliary that will have the effect of lowering the X′ node. Interestingly, this view matches quite well what has been assumed in previous TAG analyses, where successive cyclic A′-movement is accomplished by adjunction at C′ as discussed earlier, and successive cyclic A-movement by adjunction at I′. Indeed, we believe that this proposal provides a means of explaining why unbounded movement uniformly proceeds through specifier positions.

One potentially problematic case of complement adjunction at an internal XP node involves wh-extraction from an ECM verb as in an example like 'Which problems (do) you expect Mary to solve?' The most straightforward TAG analysis of such a case would adjoin an IP auxiliary tree representing the matrix clause, i.e., *you expect IP* into a CP initial tree representing the embedded clause from which extraction has taken place, i.e., *which problems Mary to solve.* It is possible, however, that this extraction involves a more complex multi-component derivation. Thus, the representation of the embedded clause may not include a CP projection at all, but rather could perhaps simply represent the fronted wh-element as c-commanding the IP node, as in Figure 3. This c-command relation would be preserved if the embedded IP substituted into the complement position of a CP-rooted matrix tree and the wh-phrase substituted into the specifier of CP position of the same tree.[4,5] This kind of multi-component tree set, in which there is no dominance link between

[3]This raises the interesting possibility that specifiers could be adjoined in the TAG sense as well. Although this would have certain benefits with respect to the treatment of subject islands, we believe at present that it is not immediately compatible with our proposal to derive the possible loci of adjunction from c-command monotonicity.

[4]It should be noted that this version of adjoining does not remove the restrictive character of adjoining that is crucial in deriving island effects. It is in fact fairly straigbtforward to provide a simple view of possible elementary tree domains, analogous to the CETM of Frank (1992), so that the standard effects are derived.

[5]Other analyses of this case are, of course, possible, some reminiscent of ideas presented in a TAG framework by Rambow and Kroch (1994), in which ECM is taken to involve raising to a specifier position of a higher clause. Space presents us from exploring this alternative here.

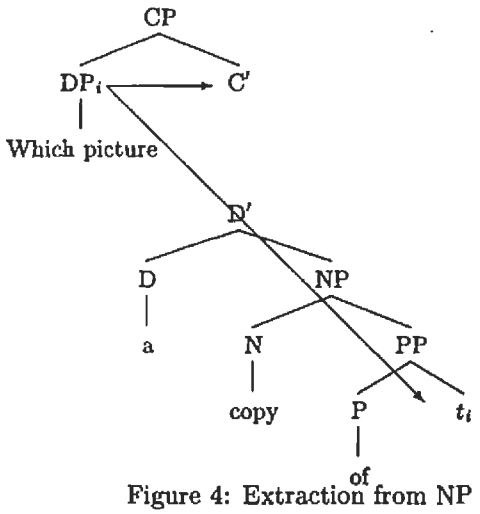


Figure 4: Extraction from NP

the two components, but instead a c-command link, has in fact been exploited in previous TAG analyses of wh-movement (Frank, 1992). Under our proposal, dominance links as they have been exploited in multi-component sets can effectively be replaced by c-command links, as these more effectively convey the grammatically relevant structural relations. Moreover, our proposal allows us to understand why no dominance links were previously posited between certain components of a multi-component set: there is no relevant structural relation linking them, so their hierarchical order is free.

It is a well-known that extraction from NP must be handled in a different fashion in TAG from extraction from clausal complements, as the adjoining operation allows only the insertion of recursive structure. However, using c-command to define the elementary structures allows us to generalize the adjoining operation so as to capture both cases. Specifically, a derivation of a sentence like 'Which picture did you buy a copy of?', could proceed by inserting a non-recursive structure, with root C' and foot D' between the two components of the set in Figure 4.[6] What would previously have been assumed to be a domination relation between the C' node and the D' node now can be seen to follow from the c-command relation between the moved element and the trace. In the derived structure, this c-command relation, and therefore as a side effect the domination relation, continues to hold. Note that our hypothesis that c-command relations should be preserved during derivation would rule out a possible TAG analysis where the structure for *a copy of* is considered to be an auxiliary tree. Adjunction of such an auxiliary tree would violate the requirement of preservation of c-command as it would have to be adjoined at the complement NP node of the verb *buy*.

Finally, we suggest that our recasting of TAG derivations as manipulations of c-command relations leads to a resolution of thorny issues for the TAG framework posed by examples such as 'Does Gabriel appear to like gnocchi?'. The relevant property of this example and others like it (e.g., involving clitic climbing) is that the lexical material associated with the matrix clause (i.e., *does* and *appear*) is intermingled with that of the embedded clause in such a way that there is no natural way of localizing it in a single auxiliary tree. Consequently, this example seems to require a derivation that is considerably more complex than a simple instances of raising. Supposing instead that the elementary tree headed by *appear* consists of the usual I' raising auxiliary (stated in c-command terms) together with the verb *does* which is stated to c-command the root I', as a result of its having raised, in a spirit similar to the structure in 3, but applied to head movement. When this auxiliary combines with the subordinate clause elementary tree, *does* is free to float above the subject, as this will preserve the c-command relation.[7]

References

Frank, R. 1992. *Syntactic Locality and Tree Adjoining Grammar: Grammatical, Acquisition and Processing Perspectives.* Ph.D. thesis, University of Pennsylvania, Philadelphia, PA.

Frank, R. and K. Vijay-Shanker. 1998. Primitive c-command. Manuscript, Johns Hopkins University and University of Delaware.

Kayne, R. 1994. *The Antisymmetry of Syntax.* MIT Press, Cambridge, MA.

Kroch, A. 1989. Asymmetries in long distance extraction in a tree adjoining grammar. In M. Baltin and A. Kroch, editors, *Alternative Conceptions of Phrase Structure.* University of Chicago Press, Chicago, IL, pages 66–98.

Schabes, Y. and S. Shieber. 1994. An alternative conception of tree adjoining derivation. *Computational Linguistics*, 20:91–124.

Vijay-Shanker, K. 1988. Feature-structure based tree-adjoining grammars. In *Proceedings of the 12th International Conference on Computational Linguistics*, Budapest.

Vijay-Shanker, K. 1992. Using descriptions of trees in a tree adjoining grammar. *Computational Linguistics*, 18:481–518.

[6]The derivation shares a good deal in common with the proposal of Kulick (this volume). Detailed comparison of these two analyses awaits future work.

[7]Of course, it is also possible for *does* to remain below the subject, deriving 'Gabriel does appear to like gnocchi.' It is unclear to us at present whether this is a problem, and if it is, what the nature of the solution would be.

Describing Discourse Semantics

Claire Gardent
Computational Linguistics
University of the Saarland
Saarbrücken, Germany
claire@coli.uni-sb.de

Bonnie Webber
Computer and Information Science
University of Pennsylvania
Philadelphia PA 19104-6389 USA
bonnie@central.cis.upenn.edu

Descriptions. In recent years, both formal and computational linguistics have been exploiting *descriptions* of structures where previously the *structures* themselves were used.

The practice started with (Marcus et al., 1983), who demonstrated the value of (syntactic) tree descriptions for near-deterministic incremental parsing. Vijay-Shankar (Vijay-Shankar and Joshi, 1988; Vijay-Shankar, 1992) used descriptions to maintain the monotonicity of syntactic derivations in the framework of Feature-Based Tree Adjoining Grammar. In semantics, both (Muskens, 1997) and (Egg et al., 1997) have shown the value of descriptions as an underspecified representation of scope ambiguities.

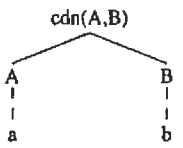
The current paper further extends the use of descriptions, from individual sentences to discourse, showing their benefit for incremental, near-deterministic discourse processing. In particular, we show that using descriptions to describe the semantic representation of discourse permits: (1) a monotone treatment of local ambiguity; (2) a deterministic treatment of global ambiguity; and (3) a distinction to be made between "simple" local ambiguity and "garden-path" local ambiguity.

Discourse descriptions. Suppose we have the discourse:

(1) a. Jon and Mary only go to the cinema
b. when an Islandic film is playing

On hearing the second sentence, the hearer infers a CONDITIONAL relation (CDN) to hold between the event partially specified in (1a) and the event partially specified in (1b). We associate this with the following description of structure and semantics:

(2)



The dashed lines indicate domination, the plain lines immediate domination. Labels on the nodes are first-order terms abbreviating their associated semantic information. Capital letters indicate variables, lower letters indicate constants, and shared variables indicate re-entrancy. Whenever two node descriptions are identified and taken to refer to the same node, their labels must unify.

The description licenses a local tree whose root semantics is CDN(A,B), where A and B are the semantics of nodes dominating the nodes whose semantics is a and b, respectively. Intuitively, A and B represent the *final* arguments of the CONDITIONAL relation, whereas a and b stand for its *current* arguments.[1] Formally, A/a and B/b nodes are quasi-nodes in the sense of Vijay-Shankar: they are related by dominance and therefore can (but need not) be identical.

Local ambiguity. As (Marcus et al., 1983) has noted, descriptions facilitate a near-deterministic treatment of local attachment ambiguities in incremental parsing. This is also true at the discourse level. For instance (1) can be continued in two ways: additional discourse material can "close off" the scope of the relation

(3) a. so they rarely go.
b. Semantics: *cause(cdn(a,b),c)*

[1] Descriptions can be formulated more precisely, using tree logic (Vijay-Shankar, 1992). For this paper however, we will use a graphic presentation, as in (2) above, which is easier to read than conjunctions of logical formulae.

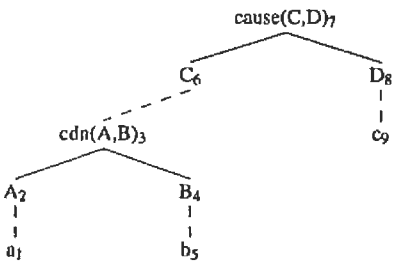
or it can extend it:

(4) a. or the film got a good review in *The Nation.*

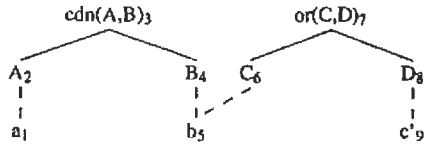
b. Semantics: *cdn(a,or(b,c))*

By using descriptions of trees rather than the trees themselves, we have a representation which is compatible with both continuations: In the first case (continuation 3), addition of the third clause will lead the hearer to infer a CAUSE relation to hold between (1) and (3). This extends the description in (2) to:

(5)

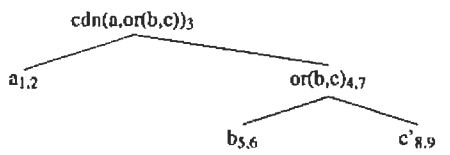


By contrast, if (1) is continued with (4), the initial description is expanded to:



where OR stands for disjunction. Both descriptions are compatible with the initial description (2) and both descriptions can be further constrained to yield the appropriate discourse semantics.

Suppose that no further material is added: now the scope of the discourse relation becomes known. This in turn licenses node identifications which conflate final and current arguments. So if the discourse ends in (4), then node 6 is identified with node 5, fixing to b the left-hand argument of the OR relation. Similarly, nodes 8 and 9 can be identified, thereby fixing the right-hand argument to c'. Given these additional constraints, the minimal tree structure which satisfies the resulting description is:



In summary, dominance permits underspecifying the *syntactic* link between nodes, while *semantically*, quasi-nodes permits underspecifying the arguments of discourse relations. In both cases, monotonicity is preserved by manipulating descriptions of trees rather than the trees themselves.

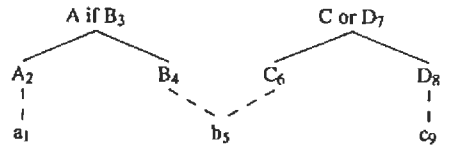
Global ambiguity Discourse exhibits global scope ambiguities in much the same way sentences do:

(6) a. I try to read a novel

b. if I feel bored

c. or I am unhappy.

This discourse means either that the speaker tries to read a novel under one of two conditions (boredom or unhappiness), or that the speaker is unhappy if s/he can't read a novel when bored. Discourse-level scope ambiguities can be captured as in (Muskens, 1997) by leaving the structural relations holding between scope bearing elements underspecified. For example, the (ambiguous) structure and semantics of (6) can be captured in the description:



In the absence of additional information (i.e. when the respective scope of the discourse relations remains unspecified), no additional constraints come into play, so that not one but two trees satisfy the description: one with root semantics *a if (b or c)* and the other with root semantics *(a if b) or c.*

Defaults, underspecification and preferences. We assume that a cognitive model of incremental discourse processing should distinguish between those cases of "simple" local ambiguity which do not trigger repair when they are resolved by information later in the discourse and those cases of "garden path" ambiguity that do.

Now there is a continuum of ways to deal "economically" with local ambiguity, without generating all the possible readings. At one end is a pure *default* approach, commiting to one reading and discarding the others. At the other end is a pure *underspecification* approach, with a single compact representation of all possible readings but no indication of the reading of the text so far.

Neither of these "pure" approaches suffices to distinguish simple local ambiguity from garden path ambiguity in either sentence-level processing or discourse. While defaults can be subsequently overridden, there is no difference between overriding a simple local ambiguity and overriding a garden path. On the other hand, underspecification, which does not "commit" to any specific choice, provides no indication of the reading of the text so far and thus again, no way of distinguishing simple local ambiguity from cases where the reader garden-paths.

However, in between these polls are approaches that combine features of both. One that seems able to meet the cognitive criteria given above is an approach that combines underspecification with a preference system that highlights a specific reading corresponding to the hearer's currently preferred interpretation. Such a proposal was suggested in (Marcus et al., 1983), and is the one we are currently exploring for discourse. The two aspects of the approach we want to discuss here are: (1) partial underspecification, and (2) biases in choosing a preferred reading.

Partial Underspecification. The degree of underspecification in a description is usually only partial: there is always something that it commits to. For example, while underspecifying domination, the structural descriptions used above still rigidly distinguish each branch of a tree from its sisters. Similarly, while allowing underspecification in each individual argument to a predicate, the descriptions used here still rigidly distinguish one argument from another. We take this to be a "feature" with respect to making a cognitive distinction between simple local ambiguity and local ambiguity that leads to garden paths.

In particular, we associate simple local ambiguity with domination underspecification, whether it be at the sentence-level or in discourse: the local ambiguity associated with "my aunt" after processing "I saw my aunt ..." – whether it continues

(7) a. I saw my aunt.
    b. I saw my aunt's cat.
    c. I saw my aunt's cat's litter box.

– is purely a matter of how the domination relation eventually resolves itself.

On the other hand, the ambiguity associated with "raced" after processing "The horse raced ..." – whether it continues

(8) a. The horse raced past the barn.
    b. The horse raced past the barn fell

– is a matter of choosing whether "raced" takes "the horse" as its argument or whether it acts as a modifier of "the horse" (in distinguishing this horse from other ones, cf. (Crain and Steedman, 1985)). This ambiguity cannot be captured by domination underspecification. As such, it can only be represented as a (disjuctive) alternative, a matter of non-deterministic or preferential choice. If the choice is incorrect, revision is required, thus providing a way of making the desired distinction between simple and garden-path local ambiguity.

Biases in choosing a preferred reading. In any abductive process, there are many ways of explaining the given data, and *biases* are used to identify one that is preferred. For example, in plan recognition (identifying the structure of goals and subgoals that give rise to what is usually taken to be a sequence of observed actions), Kautz (Kautz, 1990) suggested a "goal minimization" bias that preferred a tree with the fewest goals (non-terminal nodes) able to "explain" the sequence of actions. Where goal minimization is known to produce the wrong explanation, some other bias is needed to yield the one that is preferred (Gertner and Webber, 1996).

Similarly, in associating a preferred reading with a compact underspecified representation, (Marcus et al., 1983) proposed a bias towards a tree that minimised the dominance relation. That is, if two node names stand in a dominance relation, they are taken to refer to one and the same node, provided nothing rules it out. Of course, such a "min_dom" bias might yield several trees, each of which are equally minimal. Typically, this is true of global ambiguities as in (6) above, where dominance can be minimised by identifying node 5 either with node 4 or with node 6, each move resulting in an equally minimal tree.

An alternative bias combines "min_dom" with "right-association" (Frazier, 1995; Chen and Vijay-Shankar, 1997), yielding a preference for a structure in which the incoming unit attaches "low in the tree" and can be obtained by minimising the most recent dominance link. In example 6, this means identifying node 6 with node 5 first so that the default reading in this case is the reading where *if* scopes over *or*.

Other biases are possible: Crain and Steedman (Crain and Steedman, 1985) argue for a preference for referring forms that distinguish one

already-evoked (discourse) entity from possible alternatives. We believe it is worth exploring what bias best models the preferences people have in discourse interpretation, and how it resembles their preference at the sentence level.

Comparison with related work. A related approach to discourse structure and semantics is presented in (Webber and Joshi, 1998), where Lexicalised Tree Adjoining Grammar (LTAG) is used to construct the compositional semantics of discourse. Although the basic structures used here are different, we foresee no difficulty in modifiying them in order to integrate the additional information included in the LTAG discourse trees. Essentially, the atomic labels representing the relations should be mapped into the feature structures used in (Webber and Joshi, 1998) and this information used to label not the root node of a local tree but its anchor. Second, the LTAG approach has focussed on describing the *compositional* semantics of discourse – that is, the semantics explicitely given by the text (as opposed to what can be inferred). In contrast, the present approach does not differentiate between compositional and inferred semantics, though again, the difference does not seem essential as the description based approach could be either *extended* to explicitely distinguish (e.g. by means of features) between compositional and inferential information, or *restricted* to describe those aspects of discourse semantics that are compositional. Third, the LTAG proposal does not address the focus of the current approach – incrementality and underspecification. On the whole then, the two systems are complementary rather than antagonistic.

Conclusion. We have argued that a technique developed to handle well-known problems in sentence processing can also benefit the processing of monologic discourse. First, it provides a well-defined framework for monotonically describing the incremental construction of discourse semantics. This departs from approaches in the discourse literature which give up either monotonicity (Asher, 1993) or incrementality (Hob90; MT87). Second, it has a well-understood formal basis in tree logic. Third, it permits a clear-cut distinction between local ambiguities that lead the hearer down the garden path and those that don't.

Acknowledgements. The authors would like to thank Mark Steedman and Aravind Joshi for comments and suggestions. An early draft of this paper was presented at the *Workshop on Underspecification*, Bad Teinach, Germany, May 1998. Claire Gardent is grateful to the Deutsche Forschungsgesellschaft for financial support within the SFB 378, C2.

References

N. Asher. 1993. *Reference to abstract objects in discourse*. Kluwer, Dordrecht.

J. Chen and K. Vijay-Shankar. 1997. Towards a Reduced Commitment, D-Theory Style TAG Parser *Proc. Int'l Workshop on Parsing Technologies*.

S. Crain and M. Steedman. 1985. On not being led up the garden path: The use of context by the psychological parser. In D. Dowty, L. Kartunnen and A. Zwicky (eds.), *Natural Language Parsing: Psychological, computational and Theoretical Perspectives*, Cambridge University Press.

M. Egg, J. Niehren, P. Ruhrberg and F. Xu 1998. Constraints over Lambda Structures in Semantic Underspecification *Proc. COLING/ACL*, Montreal CA.

L. Frazier. 1995. Issues of Representation in Psycholinguistics. In J.L. Miller and P.D. Eimas, eds., *Speech, Language and Communication*. Academic Press, New York.

A. Gertner and B. Webber. 1996. A Bias towards Relevance: Recognizing plans where goal minimization fails. *Proc. 1996 National Conference on Artificial Intelligence (AAAI-96)*, Portland OR, pp. 1133-1138.

J. Hobbs. *Literature and Cognition*. CSLI Lecture Notes, Number 21. 1990.

H. Kautz. A circumscriptive theory of plan recognition. In Jerry Morgan Philip R. Cohen and Martha E. Pollack, editors, *Intentions in Communication*. Bradford Books, 1990.

W. C. Mann and Sandra A. Thompson. Rhetorical structure theory. Technical Report RS-87-190, USC/ISI, 1987.

M. P. Marcus, D. Hindle, and M.M. Fleck. 1983. Talking about talking about trees. In *Proceedings of ACL*, Cambridge, MA.

R. Muskens. 1997. Order-independence and underspecification. University of Tilburg.

L. Vijay-Shankar and A. Joshi. 1988. Feature based tags. *Proceedings of ACL*, Budapest.

K. Vijay-Shankar. 1992. Using descriptions of trees in a tree-adjoining grammar. *Computational Linguistics*.

B. Webber and A. Joshi 1998. Anchoring a lexicalized Tree-Adjoining Grammar for discourse. *Proceedings of COLING/ACL workshop on Discourse Relations and Discourse Markers*, Montreal CA.

Tree-Grammar Linear Typing for unified Super-Tagging/Probabilistic Parsing Models

Ariane Halber
halber@enst.fr
AI group, dept of Computer-Science, ENST-Paris*
Man-Machine Interaction lab, Corporate Research Laboratory, Thomson-CSF†

Abstract

We integrate super-tagging, guided-parsing and probabilistic parsing in the framework of an item-based LTAG chart parser. Items are based on a linear-typing of trees that encodes their expanding path, starting from their anchor.

1 Introduction

Practical implementations of LTAG parsing have to face heavy lexical ambiguity and parsing combinatorial ambiguity. Main techniques to address these issues are **super-tagging** (Joshi and Srinivas, 1994), which consists in disambiguating elementary trees before parsing; **guided-parsing**, like head-driven parsing (van Noord, 1994) or anchor driven parsing (Lavelli and Satta, 1991; Lopez, 1998); and **probabilistic parsing** (Schabes, 1992; Caroll and Weir, 1997).

All of these approaches exploit specific properties of LTAG to improve parsing efficiency, but none is totally satisfactory.

Guided-parsing is a very nsefull means to limit overgeneration of spurious items in the chart, but it does not provide a new ambiguity bound. Besides, lexical ambiguity remains the main factor of computational load and this problem is only undirectly addressed by such techniques.

Super-tagging strength is to discard elementary trees while avoiding to go through actual tree combinations. It exploits instead local models of Well-Formedness (WF), as those used for tagging, where parse depencies remain implicit or underspecified. The problem though is that if a single tree is incorrect the parse will fail. To be robust, parsing must thus take several hypothesis into account. This leaves one with two regrets: first, parsing has still to find some way to tackle combinatorial ambiguity, and second, there is a lack of synergy between super-tagging and parsing , while they seem to share a kuowledge about tree potential-combinations.

Probabilistic parsing offers a way to tune the compromize between accuracy and speed, by thresholding partial parsing paths according to their current Inside probability. This incurs a well-known bias (Goodman, 1998): At a given derivation step, the Inside-probabilities of parse constituents estimate the relevance of the derivation *past*, but do not tell anything about its *future*. This can be corrected by A^* cost functions, or Outside-probability estimates.

To meet the weak points mentionned above, at least partialy, we develop a unified framework for the three techniques, and push their interactions further.

Sharing a parsing framework We propose an item-based chart-parser, where the parsing scheme is expressed as a deduction system (Shieber, Schabes, and Pereira, 1994). This framework is also amenable for expressing probabilistic parsing (Goodman, 1998).

Sharing models for super-tagging and item-pruning. Super-tagging can be seen as a tree-sequence WF-model, and extended to score derived item-sequences in the chart, wrt their likelihood of completing a parse. This yields a sound thresholding technique (Rayner and Carter, 1996; Goodman, 1998).

Sharing tree-types for item-pruning and guided-parsing. To support the WF parametric model, trees and items are abstracted by their linear type, which consists in a list of *connectors* that represent combination properties. Guided-parsing relies on a specific ordering of these connectors, so that a single type drives the parsing deduction and

*ENST Paris, 46 rue Barrault, 75634 Paris Cedex 13
†Thomson-CSF, LCR, Domaine de Corbeville, 91404 Orsay Cedex, FRANCE

Item form: $<\Gamma_l|\Gamma_r>_{\alpha[i,j,f_l,f_r]}$

Goal: $<\mathcal{S}\downarrow \mid \downarrow\mathcal{S}>_{\alpha[0,n,-,-]}$

Axioms:

Anchor $<\Gamma_l|\Gamma_r>_{\alpha[i,i+1,-,-]}$ $\quad Anchor(\alpha) = w_i$ $\quad \Gamma_l, \Gamma_r$ connected walks of α

co-Anchor $<w_i\downarrow \mid \downarrow w_i>_{unrooted[i,i+1,-,-]}$

Rules: (for left expansion, right is symetrical)

Substitution
$$\frac{<\mathcal{N}\downarrow \mid \downarrow\mathcal{X}>_{\alpha'[i,j,f_l',f_r']} <\downarrow\mathcal{X}\Gamma_l|\Gamma_r>_{\alpha[j,k,f_l,f_r]}}{<\Gamma_l|\Gamma_r>_{\alpha}\,[i,k,\ f_l\oplus f_l',\ f_r\oplus f_r']}$$ wrap-3 and co-Anchor recognition

Right Adjunction
$$\frac{<{}^*r\mathcal{X}_{(foot)}\Gamma_l'|r\mathbf{X}^*_{(foot)}>_{\beta[i,j,-,-]} <r\mathbf{X}^*_{(\eta)}\gamma\mathcal{X}^*_{(\eta)}\Gamma_l|\Gamma_r>_{\alpha[j,k,f_l,f_r]}}{<\gamma\Gamma_l'\Gamma_l|\Gamma_r>_{\alpha[i,k,f_l,f_r]}} \quad \Gamma_l' \in \{\mathcal{X}^*,\$\}$$

Left Adjunction
$$\frac{<\ell\mathcal{X}_{foot}|\mathbf{X}^*_{foot}>_{\beta[i,j_l,-,-]} <\mathbf{X}^*_{(\eta)}\Gamma_l|\Gamma_r>_{\alpha[j_l,k,f_l,f_r]}}{<\Gamma_l|\Gamma_r>_{\alpha[i,k,f_l,f_r]}} \quad \eta \notin \text{sl spine}(\alpha)$$

Left Adj on spine
$$\frac{<\mathcal{X}_{foot}|\Gamma_r'\mathbf{X}^*_{foot}>_{\beta[i,j,-,-]} <\mathbf{X}^*_{(\eta)}\Gamma_l|\Gamma_r^*r\mathbf{X}_{(\eta)}>_{\alpha[j,k,f_l,f_r]}}{<\Gamma_l|\Gamma_r\Gamma_r'>_{\alpha[j,k,f_l,f_r]}} \quad \Gamma_r' \in \{{}^*r\mathcal{X},\$\}$$

Sub-tree extraction
$$\frac{<\mathcal{X}|r\mathbf{X}^*>_{\beta[i,j,j_l,j_r]} <r\mathbf{X}^*_{(\eta)}\gamma\mathcal{X}^*_{(\eta)}\Gamma_l|\Gamma_r>_{\alpha[j,k,f_r,f_l]}}{<\gamma\mathcal{X}^*_{(\eta)}.\mathcal{X}\eta\downarrow \mid \downarrow\mathcal{X}\eta r\mathcal{X}^*_{(\eta)}>_{\alpha[j_r,j_r,-,-]} <\downarrow\mathcal{X}\eta\Gamma_l|\Gamma_r>_{\alpha[j,k,f_r,f_l]}>}$$ wrap-1

Wrap Adj on spine
$$\frac{<{}^*\mathbf{X}|r\mathbf{X}^*>_{\beta[i,j,j_l,j_r]} <r\mathbf{X}^*_{(\eta)}\Gamma_l|\Gamma_r{}^*\mathbf{X}_{(\eta)}>_{\alpha[j_l,j_r,f_l',f_r']}}{<\Gamma_l|\Gamma_r>_{\alpha[i,j,f_l',f_r']}}$$ wrap-2 adjunction on sub-tree

No Left Adjunction
$$\frac{<\mathbf{X}^*_{(\eta)}\Gamma_l|\Gamma_r>_{\alpha[i,j,f_l,f_r]}}{<\Gamma_l|\Gamma_r>_{[i,j,f_l,f_r]}}$$

Gap creation
$$\frac{<{}^*\mathbf{X}_{(foot)}\Gamma_l|\Gamma_r>_{\beta[i,f_r,-,-]}}{<\Gamma_l\,{}^*r\mathcal{X}_{(foot)}\not\downarrow\mathcal{X}^*_{(foot)}\Gamma_r>_{,j]_{\beta[i,j,f_r,j]}}} \quad \Gamma_l \notin \{{}^*\mathcal{X},\$\}$$

Table 1: Deductive system for an LTAG bidirectional chart-parser, lexically guided and based majoritarily on trees, thanks to a precompilation of their nodes into left and right walks.

The active connector is poped on extreme left (resp. right) of its stack Γ_l (resp. Γ_r). Each connector is associated with its node η, though we do not always mark it. The *spine* is the path from anchor to root.**wrap-1, wrap-2, wrap-3** identifie the three steps of a wrapping adjunction on an internal node, cf figure 1.

estimates the pruning model. Types are described in section 2, their use in the deduction system, in section 3, their use for item-pruning in section 4.

2 Linear Typing

Guiding the Tree expansion We guide the parsing by independent left and right *connected-walks*, inspired from (Lavelli and Satta, 1991) bidirectional parser and (Lopez, 1998) connected routes. Left and right connected walks follow respectively left- and right- monotonic expansion, outward, from the anchor to the root, as displayed in figure 2. They list node operations considered as *connectors*.

Link-Grammar expression To express linear types, we exploit the Link-Grammar formalism (Lafferty, Sleator, and Temperley, 1992), which is close to Categorial Grammar. Left and right walks are expressed as stacks of connectors, so that the extreme connector is the one to connect the closest to the anchor[1] An illustration is given in table 2 for the tree in figure 2.

Typing strategy. In its own walk, the foot bears the adjunction, with type l or r inversly to the foot side. In the opposit walk, the foot-node may be reached as well, provided that there is a direct path from root to foot. In the deduction system, in table 1, the foot-node of a left or right auxiliary tree achieves adjunction, but the foot-node of a wrapping auxiliary tree creates a gap and passes its adjunction

[1]The derivation is represented as a fully connected and oriented graph of trees whose edge labels are connector names (provided that a sub-tree is extracted to decompose wrapping adjunction, cf. figure 1.

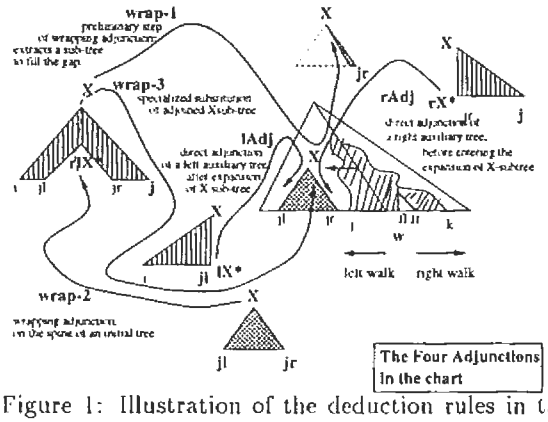


Figure 1: Illustration of the deduction rules in table 1.

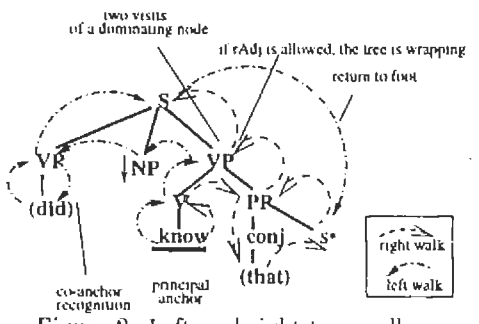


Figure 2: Left and right tree walks.

capacities to the root-node, with an opposit type for the opposit sides.

It can be noted that each node that can receive adjunction yields two linked connectors, which bound the sub-list of connectors of their sub-tree.

3 Deductive Chart Parser

We wish to get elementary-like types on derived structure, so as to use a super-tagging-like model to prune derived paths. We try thus to keep as close as possible to trees when driving the parsing. But we are not aiming at top-down parsing, since we wish to evaluate derived paths that span the input. This leads to isolating wrapping adjunction from left- and right-, adjunctions, since it is the only case where sub-tree extracting is unavoidable (cf. figure 1). Actually this emphasis on wrapping auxiliary trees is not surprising, since they account for LTAG context-sensitiveness (Schabes and Shieber, 1994).

The full deductive parsing system is defined in table 1. for the LTAG bidirectional-chart parser.

Our approach advantage is threefold: first, it considers only operations that are lexically sound, according to the input string sequence; second, it keeps the number of spurious items very low, by creating very few *sub-tree-* (or *node-*) items; third, it isolates

left walk meta-rule:
$^*\mathcal{B}_{\text{foot}}\ \mathcal{B}^*\ \mathcal{N}P^*\ \downarrow$did $\ ^*\mathcal{V}P^*\ NP\downarrow\ \mathcal{N}P^*\ \mathcal{N}^* \leftarrow\bullet$ know
right walk meta-rule:
know $\bullet\rightarrow\ ^*\mathcal{V}$ that$\downarrow\ \mathcal{B}^*_{\text{foot}}$ (NA$^*\mathcal{V}$P) $^*\mathcal{S}$
left and right connector stacks:
$\langle \mathcal{N}^* \cdots \mathcal{B}^*\ ^*\mathcal{B}_{\text{foot}} \mid\ ^*\mathcal{S} \cdots\ ^*\mathcal{V}\rangle$
type: abstraction on connector stacks, removes specialized substitutions:
co-Anchor: $w\downarrow\rightarrow$ LEX$\downarrow$ sub-tree: $X\eta\downarrow\rightarrow X\downarrow$

Table 2: Typing the tree in figure 2.
In a right walk, $\downarrow X^*$ expresses an auxiliary root-node and $^*X$, a node expecting adjunction, $X\downarrow$ expresses a substitution site and $\downarrow X$, the root of an initial tree. In a left walk they work the other way around.

clearly the CF-component, so that the parsing behaves very nicely when faced with near-CF derivations, which are a majority in practice.

Now, regarding complexity, first three "near-CF" rules yield a worst-case complexity of $O(n^5)$, wrapping adjunction on a lexical spine yields $O(n^6)$, but the sub-tree rule yields $O(n^7)$. We could change that rule into a "systematic" snb-tree extraction with arbitrary gap frontiers, in order to go down to $O(n^5)$, bnt this would generate a lot of spnrious items. Therefore we prefer a lexical check with a wrapping auxiliary tree, since their occurrence is marginal.

4 Probabilistic Thresholding

Probabilistic parsing is expressed through the deductive system as follows:

$$\begin{aligned} [_i item_j] &= P_I(_i item_j) = P(item \stackrel{*}{\Rightarrow} w_i \ldots w_j) \\ Rule &= P(rule) \end{aligned}$$

$$\frac{Rule, [_i item_j][_j item_k]}{[_i item_k] = Rule * [_i item_j] * [_j item_k]}\ d$$

Probabilities of items are *inside probabilities* i.e generative probability that an item dominates its current span of input. Now the usefulness of an item in reaching full derivations is mainly in the *outside probability* P_O of that item, defined for LTAG in (1), following (Schabes, 1992)

$$\begin{aligned} &\text{for } pos = [i, j, f_l, f_r] \\ &P\mathcal{O}([s]_{pos}) = \textstyle\sum_{U,V,T} P([\mathcal{S}] \stackrel{*}{\Rightarrow} U[s]_{pos}VT) \\ &s.t. U \stackrel{*}{\Rightarrow} W_{0,i}, V \stackrel{*}{\Rightarrow} W_{j,n}, T \stackrel{*}{\Rightarrow} W_{f_l,f_r} \end{aligned} \quad (1)$$

$$\begin{aligned} &P_{prior}([s]) = \textstyle\sum_{U,V,T,W'} P(S \stackrel{*}{\Rightarrow} U[s]VT) \\ &s.t UV[s]T \stackrel{*}{\Rightarrow} W' \end{aligned} \quad (2)$$

$$\tilde{P}([s]_{pos}) = \sum_{U_m, V_p, T_q} P(S \stackrel{*}{\Rightarrow} U_1 \ldots U_m V_1 \ldots V_p [s]_{pos} T_1 \ldots T_q \; s.t U_1..._m V_1..._p [s]_{pos} T_1 \ldots T_q \stackrel{*}{\Rightarrow} W) \quad (3)$$

For an item-path, outside probability accounts for parsing-deductions to come, i.e the connectors of the item stacks. Whereas consumed connectors are responsible for the inside probability. There is no way to compute the outside probability without the knowledge of the actual "connection" of connectors, but this decomposition gives us a very precious means to normalize inside probabilities, which put very low probabilities on large items.

item-path	$U = (U_1, ..., U_p)$
remaining stacks:	$< \Gamma_1' \| \Gamma_r' >$
consummed stacks:	$< \Gamma_l'' \| \Gamma_r'' >$
$P_I(U) \approx$	$\sqrt{\prod_i \Gamma_l' \Gamma_r' prod_i P_0'}$
$P_O(U) \approx$	$\sqrt{\prod_i \Gamma_l'' \Gamma_r''}$

(Goodman, 1998) proposes two useful approximation of the outside score of item[s], in order to correct the inside probability bias. We express them in the context of LTAG in (2) and (3). The first one simply corresponds to the prior probability of the item category. The second one is the cumulated probability of all item-paths $U = (U_1, ..., U_p)$ that include [s]. This value can be computed in several passes (Goodman, 1998; Rayner and Carter, 1996).

Computing path probabilities entails estimating the probability that sequences of items, which span the input, can build a complete derivation. This is the aim of Super-tagging, which can be viewed as a model for the first step of the chart. We generalize it to model steps n, i.e a step where edges have maximal length n. Here are some approximations which have been proposed:

$P_I(U)$	$=$	$P(U_1..U_p)$	Real
	$\approx$	$\prod_i P(U_i)$	Fully independent
	$\approx$	$\prod_i P(U_i \| U_{i-1})$	Markovian
	$\approx$	$Min_i min(P(U_i \| U_{i-1}), P_{prior}(U_i), P(U_i \| U_{i+1}))$	Fully dependent

Item sequences ressemble elementary tree sequences, as they share types, and connect through the same connectors (provided the type abstraction explained in table 2 for specialized substitutions). Hence the possible re-use, in a first approximation, of supertagging training for the generalized item-model.

Smoothing: types decompose into a very small set of connectors, with straigtforward interpretation. They can serve as a useful basis for compnting backoff probabilities. For instance by distributing the probability mass of each connection among all types that allow this connection, in the same way as (Lafferty, Sleator, and Temperley, 1992).

5 Discussion

We have presented a general framework for deductive parsing, probabilistic parsing and super-tagging. This unified approach opens a lot of perspective in the design of efficient and robust LTAG parsing. However, it remains to be fully validated.

As far as supper-tagging is concerned, supertags should perform better than linear types as their definition integrates a large amount of linguistic knowledge. Types nonetheless provide for that task a very simple, and yet relevant, smoothing scheme. As for further steps of parsing, types turn out very adequate, as they allow to express in a simple manner the essential computations involved.

References

Caroll, John and David Weir. 1997. Encoding frequency information in lexicalized grammars. In *ACL/SIGPARSE workshop on Parsing Technologies*, MIT, Cambridge.

Goodman, J. 1998. *Parsing Inside-Outside*. Ph-D, Division of Engineering and Applied Sciences, Harvard University, May.

Joshi, A. and B. Srinivas. 1994. Disambiguation of Super Parts of Speech (or Supertags): Almost Parsing. In *COLING'94*, Kyoto, Japan, August.

Lafferty, J., D. Sleator, and D. Temperley. 1992. Grammatical trigrams: A probabilistic model of link grammar. In *Symposium on Probabilistic Approaches to Natural Language*. AAAI.

Lavelli, A. and G. Satta. 1991. Bidirectional parsing of Lexicalized Tree Adjoining Grammars. In *EACL'91*.

Lopez, Patrice. 1998. Analyse guidée par connexité des TAG lexicalisées. In *TALN'98*, Paris, France.

Rayner, Manny and David Carter. 1996. Fast parsing using pruning and grammar specialization. In *ACL'96*.

Schabes, Y. 1992. Stochastic Lexicalized Tree-Adjoining Grammars. In *COLING'92*, Nantes, France, August.

Schabes, Y. and S. Shieber. 1994. An alternative conception of tree-adjoining derivation. *Computational Linguistics*, 20(1):91–124.

Shieber, Stuart M., Yves Schabes, and Fernando C. N. Pereira. 1994. Principles and implementation of deductive parsing.

van Noord, G. 1994. Head-corner parsing for TAG. *Computational Intelligence*, 10(4):525–534.

Towards a Workbench for Schema–TAGs*

Karin Harbusch and Friedbert Widmann and Jens Woch
University of Koblenz–Landau, Computer Science Department
E–mail: {harbusch|widi|woch}@uni-koblenz.de

Abstract

In the following the components of a workbench for the grammar formalism of Schema–Tree Adjoining Grammars (S–TAGs) *are outlined. This workbench can also serve as a workbench for pure TAGs because it provides a component which transforms an arbitrary TAG into an S–TAG in a non–trivial manner. Another interesting property of the workbench is that it provides a parser, which is realized as a reversible component to generate as well.*

Introduction

The formalism of augmenting *Tree Adjoining Grammars* with schemata was introduced in [Weir 87] in order to compress syntactic descriptions. For that purpose, a TAG (see, e.g., [Joshi 86]) is extended in order to provide the facility to specify a regular expression (RE). A RE is of type a.b, a+b, a^+, a^* and $a^{(0|n)}$, where a, b can uniquely refer to child nodes (via Gorn numbers) or a tree–modifying reference of the form g_1–g_2, where g_1, g_2 are Gorn numbers and g_2 denotes a subtree of g_1. This expression means that the subtree g_2 in g_1 is ignored and replaced with ϵ. Finally, a,b can be regular expressions themselves. Regular expressions are annotated at each inner node of an elementary tree. The resulting tree is called a *schematic elementary tree.* Such a tree denotes an elementary tree set just as a regular expression denotes some regular set. Thus, an individual scheme corresponds to a — possibly infinite — set of elementary trees, but itself is not the structural element to build derivation trees of.

In order to stress the power of compressing a grammar let us reconsider the coordination construction proposed in [Weir 87]. In Fig. 1, the root node NP of the substitution tree t_1 (which is element in the set of initial trees I) is annotated with a regular expression. In this regular expression, the Gorn number |n| refers to the n-th daughter of the node. For an illustration of this reference in the figure the numbers are explicitely annotated to the individual nodes. For instance, the regular

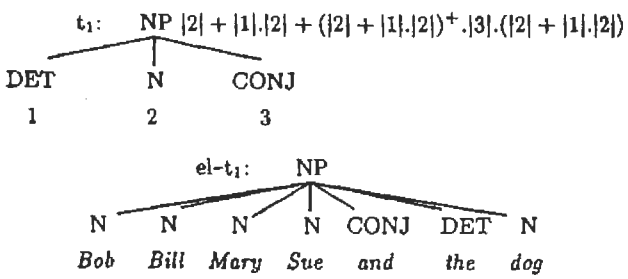


Figure 1: Coordination of NPs

expression |2| at the node NP in t_1 represents the tree with the root NP and the unique daughter N — e.g., producing *"John"*. The operation "." concatenates siblings in the same currently evaluated elementary tree. Accordingly, |1|.|2| produces an elementary tree where DET and N are the two daughters of NP (*"a man"*). The operation "+" enumerates alternative elementary trees. For instance, the regular expression |2| + |1|.|2| enumerates the two trees mentioned above. The exponents "+" and "*" produce infinite sets of elementary trees where the construction marked with such an exponent can be repeated arbitrarily often ("+" represents the infinite repetition exclusing zero occurrences and "*" inclusing zero). For instance, t_1 can produce *"Bob Bill Mary Sue and the dog"* (see tree el–t_1 in Fig. 1) but not *"and the dog"* because $(|2| + |1|.|2|)^+$ prevents the zero repetition so that at least N occurs. Furthermore a single *"and"* cannot be produced because no alternative in the regular expression at the root node starts with |3|. A finite number of repetitions can be written with the exponent $|x|^{(l|k)}$, where the component with the Gorn number x occurs at least l and up to k times.

Note, that the example is not lexicalized because Weir's dissertation proposal was earlier published than the definition of lexicalization (cf. [Schabes 90]). The coordination with Schema–TAGs works similarly with

*This work is partially funded by the DFG — German Research Foundation — under grant HA 2716/1–1.

lexicalization. Accordingly, the root node has two children (Simple_NP↓ and CONJ) and the RE is "|1| + (|1|+.|2|.|1|)". The substitution tree Simple_NP has two children (DET and N) and its root node is annotated with "|1| + |1|.|2|".

Description of the S–TAG Workbench

In the following, the components of an *S–TAG workbench (STAGWB)* are outlined. In the first subsection a facility to transform arbitrary TAG grammars (in our case the UPENN tree bench [Doran et al. 94]) into schematic trees. Then the reversible component for parsing and generation is outlined (for details s. [Woch et al. 98]).

Writing Grammar and Lexicon Rules

With respect to lexicalized TAGs [Schabes 90]) where each tree in the set of initial and auxiliary trees has at least one lexical leaf (called anchor) no lexicon component is required (cf. XTAG [Doran et al. 94]). But since the workbench should not determine the grammar formalism it is possible to specify a non–lexicalized TAG as well.

A main emphasis lies on the facility to transform an arbitrary TAG into an STAG. Obviously, an arbitrary TAG G can trivially be transformed into an S–TAG G' by annotating the concatenation of all daughters from left to right at each inner node of each elementary tree. Obviously, this transformation involves no compression. Therefore, the transformation component of the STAGWB produces an S–TAG which guarantees that each label at the root node occurs only once in the set of initial and auxiliary trees.

The component performs the following steps. Firstly, in all elementary trees all subtrees which do not contain the foot node are rewritten by substitution in order to find shared structures[1]. Since new non-terminals must be introduced to prevent the grammar from overgeneration, the adjoinable auxiliary trees are duplicated and root and foot nodes are renamed by the new non-terminals. Now, all alternatives for the same root node are collected. For each elementary tree where the root node is labelled with X (b_1, ..., b_n), a new schematic tree s_X is introduced to the S–TAG G' where its root node is labelled with X and the children result from enumerating all occurring children in all elementary trees b_1, ..., b_n without repeating the same label. In the

[1]Here, one can decide whether the structures are collapsed, although their features may differ. In the first case the disjunction of both feature descriptions is stored together with the history where they originally belonged to. Accordingly, more condensed structures are produced but the interpretation of the feature structures becomes more complicated.

```
((("NP" . "")))  ((("DetP" . "")) :substp T))
                 ((("N" . "")) :headp T)))
((("NP" . "")))  ((("N" . "")) :headp T)))
((("NP" . "r"))) ((("N" . "")) :headp T))
                 (((("S" . "")) :substp T)))
((("NP" . "")))  ((("DetP" . "")) :substp T))
                 ((("N" . "r")):constraints "NA"
                                :constraint-type :NA)
                          ((("N" . "")) :headp T))
                          ((("S" . "")) :substp T))))
((("NP" . "")))  ((("G" . "")) :headp T)))
((("NP" . "g"))) ((("NP" . "")) :substp T))
                 ((("G" . "")) :headp T)))
  ...
                                ⇓
((("NP" . "")) |1|.|2| + |2| + |2|.|3| + |1|.|4| + |5| + |6|.|5|)
                 ((("DetP" . "")) :substp T))
                 ((("N" . "")) :headp T))
                 ((("S" . "")) :substp T))
                 ((("N0" . "r")) :substp T))
                 ((("G" . "")) :headp T))
                 ((("NP" . "")) :substp T)))
((("N0" . "r")) |1|.|2|
                 :constraints "NA" :constraint-type :NA)
                 ((("N" . "")) :headp T))
                 ((("S" . "")) :substp T)))
```

Figure 2: Grammar transformation

next step the annotation of the root node of s_X is constructed by summing up all alternatives according to b_1, ..., b_n where all labels are rewritten as numerical references pointing at the respective child.

An instance of a grammar transformation is shown in Fig. 2[2]. Note, that here the first step of introducing substitutions does not have to do much, because most lexicalized TAGs already use substitution. The only new substitution node is N^0.

The resulting REs can be reformulated applying the following transformation rules:

1. $\alpha\gamma^{(i|k)}.\gamma\beta = \alpha\gamma^{(i|k+1)}\beta$,
2. $\alpha(\gamma.\delta_1)\beta + ... + \alpha(\gamma.\delta_m)\beta = \alpha\gamma.(\delta_1 + ... + \delta_m)\beta$
3. $\alpha\gamma\beta + \alpha\beta = \alpha\gamma^{(0|1)}\beta$

where $\alpha, \beta, \gamma, \delta_1, ..., \delta_m$ are arbitrary complex REs.

Note, that different compressing strategies result in different REs. For analysis grammars the rule of factoring out common prefixes is convenient, whereas the factorization according to common heads is more adequate in generation. E.g. in the example in Fig. 2 for analysis the two alternatives |1|.|2| and |1|.|4| result in |1|.(|2| + |4|). For generation the alternatives |1|.|2|+|2|+|2|.|3| result in $|1|^{(0|1)}$.|2|+|2|.|3|. Additionally, this example illustrates that an LD/LP–Schema–

[2]This transformation does not show the unification structures (c.f. footnote 1).

TAG can be advantageous especially for generation because there the alternative |2|.|3| can easily be incorporated in the compact expression.

Now, the automatically introduced substitution trees can be replaced with their original substructures and furthermore all added auxiliary trees can be eliminated again if desired. So the grammar becomes as lexicalized as it was before. Finally, in order to introduce "–" to the annotations the following process is carried out. According to the annotation of each substitution node r, substitution trees s_1 and s_2 are identified which only differ in one leaf l in s_1. For these candidates the structure must match beside the path to l. If so, the substitution of tree s_1 is explicitely realized and r is modified to refer to s_1 – <path-to-l> instead of referring to s_2.

S–TAG Parser

To be able to deal with REs and substitutions the parser extends the Earley–based TAG–parser by [Schabes 90] as follows:

Instead of computing the set of trees described by schemata (which is impossible due to its infinity) explicitely, the REs are interpreted as follows (cf. [Harbusch 94]): To indicate a certain position, $\odot$ is used to point into the current RE, i.e. $\alpha \odot \beta$ indicates, that α already has been computed. Then, two functions are introduced, namely SHIFT(ϕ), which shifts $\odot$ to the right, and NEXT(ϕ), which returns a set of nodes to be computed next. SHIFT is performed in each parsing step, in which the computation of a certain node is completed (indicated by raising the dot position to "ra"): scanning of terminals (`scanner`), the prediction of the right part[3] of auxiliary trees (`right prediction`) in which no prediction took place, and the completion of a root node of an auxiliary tree (`right completion`).

The output of NEXT is responsible for the computation of all alternatives given in the currently considered RE. Thus, each alternative g in β of NEXT($\alpha \odot \beta$) has to be taken into account for the prediction of new items. This is done in `move dot down`. Whenever an elimination $|a - b|$ occurs, it is deferred until node b is actually computed. Instead of processing b an ϵ–scan is simulated. This usually is done in `scan` obviously, but also may take place in `left prediction`, if b is non-terminal.

In order to reflect substitutions, two new operations are introduced. The formerly forbidden case of non-terminal leafs now triggers the prediction of all possible substitution trees. On the other hand, the formerly end-test-only state of being at position "ra" for non-auxiliary roots now serves for the completion of predicted substitution trees.

S–TAG Generator

As modern workbenches (cf., e.g., the workbench PAGE for Head-driven Phrase Structure Grammar [Netter, Oepen 97]) usually provide a generator, our parser is parametrised to work for generation according to the idea of bidirectional processing (cf., e.g., [Neumann 94]).

As outlined by [Shieber et al. 90] a naïve structure–driven top–down generator may not terminate (e.g. for genitive phrases in English and German). Furthermore the approach is inefficient because the input does not guide the generation process. Instead of that, possible syntactic structures are realized and their corresponding logical forms are compared to the semantic input structure.

A more natural way of guiding the generation process is to make it driven by the semantic input structure (indexing on meaning instead of indexing on string position). Generally speaking such generator predicts semantic heads. Two different procedures continue searching for a connection to sub– and the super–deriviation tree.

In the terminology of [Shieber et al. 90] the generator predicts pivots. A pivot is defined as the lowest node in the tree such that it and all higher nodes up to the root node or a higher pivot node have the same semantics. According to the definition of a pivot node the set of grammar rules consists of two subsets. The set of chain rules consists of all rules in which the semantics of some right–hand side element is identical to the semantics of the left–hand side. The right–hand side element is called the semantic head. The set of non–chain rules contains all rules which do not satisfy this condition. The traversal will work top–down from the pivot node only using non–chain rules whereas the bottom–up steps which connect the pivot node with the root node only use chain rules.

Adapting this mechanism to the generation of lexicalized TAGs means that the chain rules are completely determined by the elementary tree under consideration[4]. Adjoining and substitution represent the application of non–chain rules. In order to illustrate this kind of processing let us assume that the input structure is (frequently(see(John,friends))). Furthermore, we assume that the grammar allows to pre-

[3]Due to the possibility of arbitrary mix–ups of precedences of children by REs, the expressions "left/right to" are to be understood in a more temporal than local manner, i.e. "left of the foot node" encloses all those items that have been computed before computing the foot.

[4]Since empty semantic heads can be associated with their syntactic realization they can be processed in the same manner.

dict the trees described in Fig. 3. Since here is not the space to outline the specification lists of the individual nodes, the semantics of the trees is informally annotated at the nodes where x and y are variables to be filled during the unification at that node.

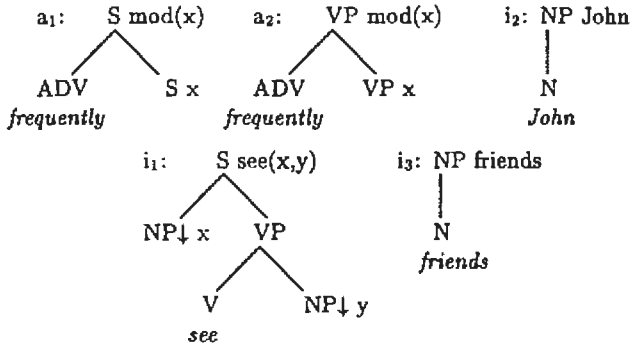


Figure 3: Predictible pivots

In a first step all predictible pivots according to the input structure can be written to the one and only item set during processing. This construction represents the unordered processing of the semantic structure. The bracketing structure of the logical form is achieved by evaluating the semantic expression associated with each elementary tree (e.g. for tree a_1 mod(x) where x is a value filled by the subtree of the foot node. The processing is successful only if a derivation tree can be constructed where all elements of the logical form occur only once[5].

Concerning the example two realizations for the input specification can be produced. The processing of the one with the sentential adverb (adjoing of a_1) is obvious whereas the adjoining of a_2 is not so clear. It also works because the variable x at the foot node is unified with the VP node of i_1 where according to the pivot definition the semantics on the spine from the root to the V node is identical. So, x contains the whole expression (see(John,friends)) and the check whether the bracketing structure is correct (i.e. the dependencies, specified in the logical form), is successful as well.

Final Remarks

All modules are implemented in JAVA [Gosling et al. 98]. Currently we run our transformation module to build a Schema–TAG equivalent to the English TAG by [Doran et al. 94]. Furthermore, we test how the average runtime varies for TAGs and Schema–TAGs. The differing size and depth of elementary trees is of special interest in incremental generation (cf. [Harbusch 94]) where the size of structures influence the time in which the processing can be finished and results can be handed over to other components.

Another topic of current considerations is how to define LD/LP–Schema–TAG which are especially interesting for generation. We assume that it suffices to rewrite the NEXT function to adapt our parser to run LD/LP–Schema–TAGs on the structural level. Our suggestion is that the separation of structural combination and linear ordering saves processing time, especially for generation.

[5]Since the bracketing structure is tested explicitely during the combination of elementary trees the accepting condition can be weaker so that the logical form equivalence problem (cf. [Shieber 93]) does not occur here.

References

[Doran et al. 94] C. Doran, D. Egedi, B.A. Hockey, B. Srinivas, M. Zaidel. XTAG System — A Wide Coverage Grammar for English. In the Proceedings of the *15th COLING*, Kyoto/Japan, 1994.

[Gosling et al. 98] J. Gosling, B. Joy, and G. Steele. *The Java Language Specification* Addison-Wesley, Reading, 2nd. ed. 1998

[Harbusch 94] K. Harbusch. Incremental sentence processing with Schema–Tree Adjoining Grammars. In the procs. of the *3rd International TAG+ Workshop*, Paris, Frankreich, September 1994, TALANA–Report 94–01, pp. 41–44.

[Joshi 86] A.K. Joshi. The convergence of mildly-contextsensitive grammar formalisms. In T. Wasow, P. Sells, eds., *The Processing of Linguistic Structures.* MIT-Press, Cambridge, MA/USA, 1986.

[Netter, Oepen 97] K. Netter, S. Oepen. PAGE – Platform for Advanced Grammar Engineering. Slides (at http://cl-www.dfki.uni-sb.de/pagegifs/slides.html), Saarbrücken/Germany, 1997.

[Neumann 94] G. Neumann. *A Uniform Computational Model for Natural Language Parsing and Generation.* PhD thesis, Saarbrücken, Germany, 1994.

[Shieber et al. 90] S.M. Shieber, F.C.N. Pereira, G. van Noord, R.C. Moore. *Semantic–Head–Driven Generation. Computational Linguistics*, 16(1): 30–42, 1990.

[Shieber 93] S.M. Shieber. *The Problem of Logical–Form Equivalence. Computational Linguistics*, 19(1): 179–190, 1993.

[Schabes 90] Y. Schabes. *Mathematical and Computational Aspects of Lexicalized Grammars.* PhD thesis, Philadelphia, PA/USA, 1990.

[Weir 87] D. Weir. Characterising Mildly Context–Sensitive Grammar Formalisms. PhD. Proposal, University of Pennsylvania, Philadelphia/USA, 1987.

[Woch et al. 98] J. Woch, F. Widmann, K. Harbusch. *A Reversible Approach to Parsing and Generation of Schema-TAGs.* Technical Report, University of Koblenz, forthcoming.

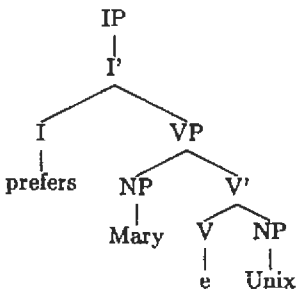
TAG and Raising in VSO Languages*

Heidi Harley and **Seth Kulick**
Institute for Research in Cognitive Science
University of Pennsylvania
Suite 400A, 3401 Walnut Street, Philadelphia, PA 19104-6228
hharley@babel.ling.upenn.edu, skulick@linc.cis.upenn.edu

The derivation of unbounded Subject-to-Subject Raising in languages like English is a problem particularly elegantly treated by Tree Adjoining Grammar. The adjoining operation inserts auxiliary trees headed by raising verbs between the subject in Spec-IP and the root verb, distancing the subject from its original local relationship with the root verb and producing a final multi-clausal structure with the subject in the final subject position in the matrix clause.

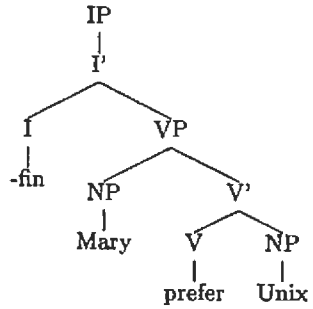
Verb-initial languages could pose a challenge to unadorned TAG in this central paradigm if it can be shown that they exhibit true raising structures. Consider the possible structures of the pseudo-English VSO finite and non-finite clauses in (1) and (2). In (1), the tensed verb appears to the left of its subject, and in (2) a structure with a non-finite verb to the right of the subject is shown. (This reflects the fact that in general, VSO clauses are only VSO in the finite case).

(1)



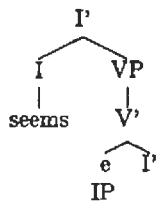
*We would like to thank Randall Hendrick, Aravind Joshi, Tony Kroch, Maggie Tallerman, and two anonymous reviewers for valuable comments. This work is supported by grant NSFSTC89-20230.

(2)



If an auxiliary *seems* tree like (3) were to adjoin to (2) above, the result would be (4) below, not a true Raising structure at all, as the subject remains in its original position in the embedded clause. In the formal system of basic TAG, it is generally true that no VSO language is predicted to exhibit a true raising structure, since the finite raising verb must appear in initial position.

(3)



(4)

The linguistic question, then, is whether it can be shown that a VSO language does exhibit a true

raising structure in which the subject is in the matrix clause. This is a non-trivial question for two reasons. First, the string will exhibit identical word order whether or not the subject is in the matrix or embedded clause, since subjects in finite clauses follow the verb. Secondly, even if it is possible to show that the subject is in the matrix clause, it must be shown that the verb in question is a true Raising verb, and not a Control verb, controlling an in situ null argument in the embedded clause. Only when both these conditions are met can we show that basic TAG is insufficient to treat VSO raising.

In Welsh, a Celtic VSO language, there are two verbs which are potential raising verbs, *digwydd* ('happen'), and *dechrau* ('begin'). We can immediately test whether or not the subject of these verbs appears in the matrix clause by using a participial form of the verb, with a finite auxiliary in initial position. If the subject is in the embedded clause, as in (4), it should make no difference whether or not the raising verb is finite or participial; it should continue to precede the embedded subject; the counterpart to *Mary has seemed to prefer Unix in the past* should be *has seemed Mary to prefer Unix in the past.* On the other hand, if the subject is in the matrix clause, the raising participle should appear to the right of the subject, since it is non- finite. We can immediately see the latter is the case:

(5) Mae Siôn yn digwydd bod yn gweld
Is John prt.happen be.inf prt.see
Mair
Mary
'John happens to be seeing Mary' (Hendrick 1988)

We must then show that *digwydd* is a raising verb, not a control verb. Following Hendrick (1988), we make this argument from the behavior of expletives. Expletives are possible as the subject of raising verbs, but not of control verbs: *There seems/*tries to be a spider on the wall.*

Welsh has an expletive subject *yna* that behaves essentially identically to English *there* , appearing in locative, existential and possessive constructions, as in (6) below.

(6) Mae yna oriad gyda John
Is there a key with John
'There is a key with John/ John has a key.'

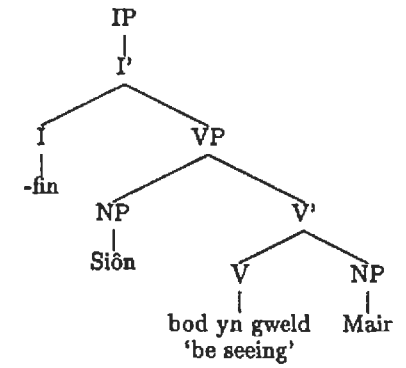
Crucially, then, this expletive may appear as the subject of *digwydd*, but not of control verbs like *mynd* ('go' in future sense). The latter also differ from the former in that they require an overt complementizer *i* to appear between the matrix and embedded clauses:

(7) a. Mae yna yn digwydd bod oriad
Is there prt.happen be.inf a key
gyda Siôn
with John
'There happens to be a key with John/ John happens to have a key.'

b. *Mae yna yn mynd i bod
Is there prt.go Comp be.inf
oriad gyda Siôn
a key with John
'There is going to be a key with John/John is going to have a key.'

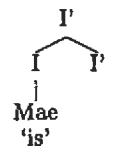
Welsh, then, is a VSO language with a true raising structure. The word order in finite clauses entails that the basic TAG adjoining mechanism will not be able to generate the structures necessary, and recourse to a multicomponent derivation must be made. Consider the non-finite tree in (2) above, repeated as (8) to represent the structure of the embedded Welsh non-finite clause in (5)[1]

(8)

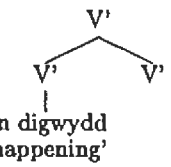


In order to get *Siôn* into subject position of the matrix clause, in a sentence like (5) above given this structure, two auxiliary trees must adjoin into the elementary tree, as shown in (9ab). One tree, headed by *Mae*, the finite copula, must substitute/adjoin in to the elementary tree in front of *Siôn*, and another, headed by the participle form of the raising verb, *yn digwydd*, must adjoin in below *Siôn*, creating the raising structure. Let us consider what such auxiliary trees must look like:

(9) (a)



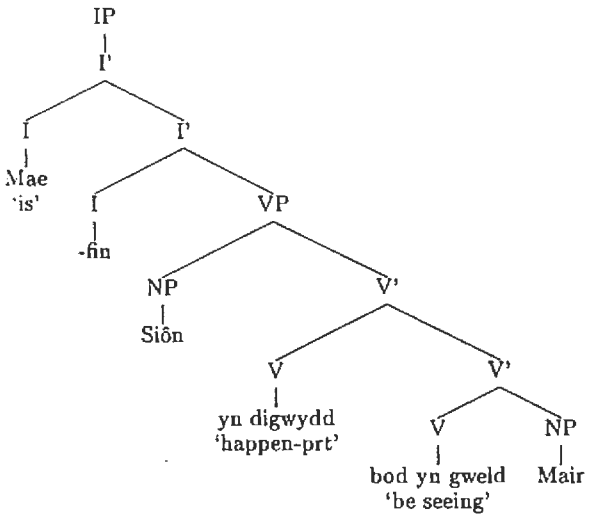
(b)



If we adjoin these trees into the elementary tree in (8), we arrive at the final structure in (10):

[1]We represent here *bod yn gweld* as a complex NP for convenience. The use of a VP-shell might be more desirable, although that issue is irrelevant for this discussion.

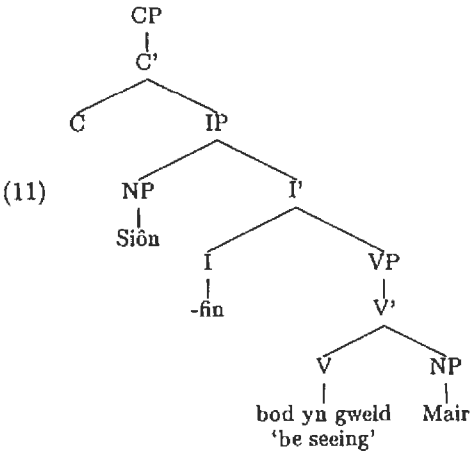
(10)



While this provides us with the correct final word order, it is linguistically unsatisfactory for two reasons. Firstly, we have destroyed the relationship between the [-fin] I head and the non-finite form of the verb *bod* by interpolating the participle *yn digwydd* (which itself needs to be related to the finite form *Mae*, now separated from it by the [-fin] head). Secondly, in a purely theory-internal problem, if Spec-VP is universally a theta-position, which is widely assumed, the subject *Siôn* is in a theta position in what is now the matrix clause. That is, "raising" has been to a theta-position, a theoretically incoherent result. Both these problems are avoided if we assume a different final clause structure for Welsh VSO sentences than that presented in the finite VSO structure in (1). The problem here is that the finite verb in (1) has raised only as far as I. This creates the dual problem above: if finite verbs are in the I head, the multicomponent auxiliary tree will always interfere with non-finite I head of the elementary tree in a raising structure, and the subject must appear in the specifier of VP, as there is no higher non-theta position available.

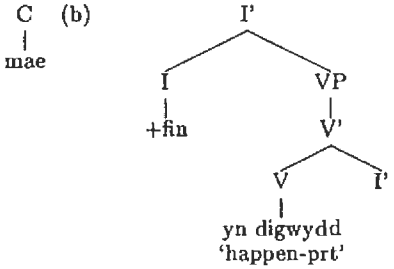
Consider, on the other hand, the possibilities which arise if finite verbs in Welsh raise as far as C. In this case the subject appears in [Spec, IP], a non-theta position, and as we shall see, no problem for the insertion of the topmost auxiliary tree will arise for the MC-adjunction necessary to derive the raising structure.

(11)

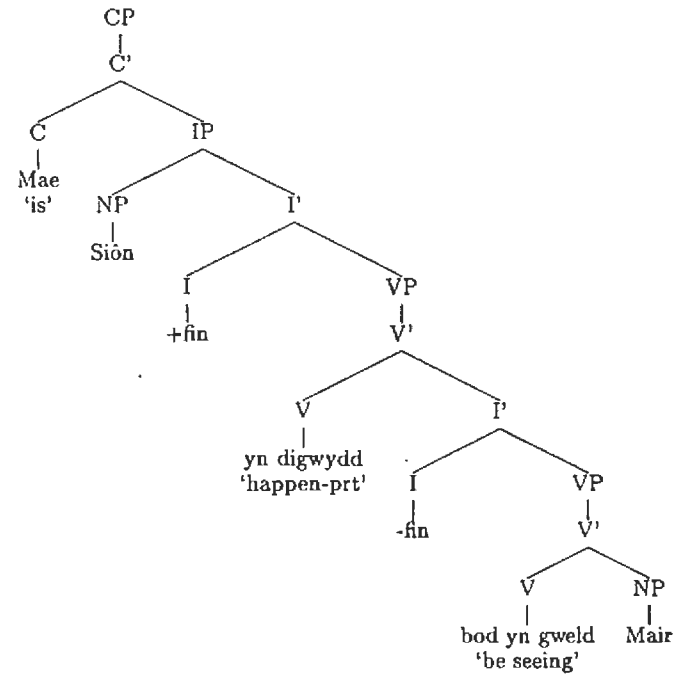


In an infinitive clause, the subject will still appear in Spec-IP, rather than Spec-VP, giving the correct SVO order for the infinitive. (Note that since in TAG there is no "movement" of the subject, it is not impossible to place the subject in [Spec, VP] in the lower clause, while ending up in [Spec, IP] in the higher clause.) Our revised elementary tree for the nonfinite clause is shown in (11), and the auxiliary trees which will adjoin into this structure are shown in (12):

(12) (a) (b)



(13)



This adjunction gives us the final structure for the raising construction, (13), which makes much more linguistic sense than the IP tree above:

The result seems to suggest that in a TAG framework, the only VSO languages which are predicted to exhibit raising structures will feature positioning the finite verb in C. [2]

The derivation we end up with is essentially identical to that proposed by Frank (1992) for an analogous problem in English: the formation of the question "Does John seem to like Mary?". This supports the view that the problem raised by that particular derivation (the requirement of a multi-component set) was not just a weird quirk, but rather just one example of the widespread need for such a derivation.

References

Frank, R. (1992) *Syntactic Locality and Tree Adjoining Grammar: Grammatical, Acquisition and Processing Perpectives*, Doctoral dissertation, University of Pennsylvania.

Hendrick, R. (1988) *Anaphora in Celtic and Universal Grammar*, Dordrecht:Kluwer Academic Publishers.

McCloskey, J. (1984a) "Case, Movement, and Raising in Modern Irish," in *West Cost Conference on Formal Linguistics*, ?

McCloskey, J. (1984b) "Raising, Subcategorization, and Selection in Modern Irish," *Natural Language and Linguistic Theory* ?, ?

McCloskey, J. (1992) "On the Scope of Verb Movement in Modern Irish," Technical Report LRC-92-10, Linguistic Research Center, Cowell College, University of California at Santa Cruz.

Stenson, N. (1981) *Studies in Irish Syntax (ars Linguistica 8)*, Gunter Narr Verlag, Tubingen.

Stowell, T. (1989) "Raising in Irish and the Projection Principle," *Natural Language and Linguistic Theory* 7, 317–359.

[2] Irish, which has been argued to not move its finite verb to C (McCloskey 1992), is an interesting case. In the "Exceptional Raising Construction" (McCloskey 1984b, Stowell 1989), an apparently-raised NP appears with a preposition (or perhaps a Case marker). The proper analysis of this construction is very unclear, and we leave it aside. Whether or not Irish has the more traditional subject raising structure is unclear (e.g., McCloskey (1984a) claims that there is, while Stenson (1981) states that there is none.) Clearly more study is required.

On Some Similarities Between D-Tree Grammars and Type-Logical Grammars

Mark Hepple
Department of Computer Science, University of Sheffield,
Regent Court, 211 Portobello Street, Sheffield S1 4DP, UK
hepple@dcs.shef.ac.uk

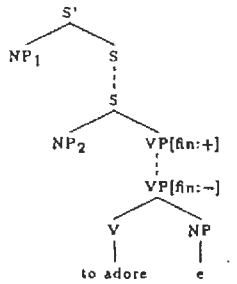
1 Introduction

This paper discusses some similarities between D-Tree Grammars and type-logical grammars that are suggested in the context of a parsing approach for the latter that involves compiling higher-order formulae to first-order formulae.[1] This comparison suggests an approach to providing a functional semantics for D-Tree derivations, which is outlined.

2 D-Tree Grammars

The D-Tree Grammar (DTG) formalism is introduced in (Rambow *et al.*, 1995). The basic derivational unit of this formalism is the d-tree, which (loosely) consists of a collection of tree fragments with domination links between nodes in different fragments (that link them into a single graph).

(1)



The above example d-tree, drawn from (Rambow *et al.*, 1995), allows topicalisation of the verb's object, as in (e.g.) *Hotdogs$_i$, he claims Mary seems to adore* t_i, where NP_1 is the fronted object, and NP_2 the verb's subject.[2] The main operation[3] for composing d-trees is *subsertion*, which, loosely, combines two d-trees to produce another, by substituting a fragment of one at a suitable node in the other, with other (dominating) fragments of the first being intercalated into domination links of the second. The approach is motivated by problems of related formalisms (such as TAG and MCTAG-DL[4]) involving linguistic coverage and the semantic interpretation of derivations.

3 Type-logical Grammar

The associative Lambek calculus (Lambek, 1958) is the most familiar representative of the 'type-logical' tradition within categorial grammar, but a range of such systems have been proposed, which differ in their resource sensitivity (and hence, implicitly, their underlying notion of 'linguistic structure'). Some of these proposals are formulated using a 'labelled deduction' methodology (Gabbay, 1996), whereby the types in a proof are associated with labels, under a specified discipline, which record proof information used in ensuring correct inferencing. Such a labelling system must be overlaid upon a 'backbone logic', commonly the implicational or multiplicative[5] fragment of linear logic. For this paper, we can ignore labellings, and instead focus on the 'core functional structure' projected by linear formulae.[6]

4 Implicational Linear Logic & First-order Compilation

In linear logic proofs, each assumption is used precisely once. Natural deduction rules of *elimination* and *introduction* for linear implication ($\multimap$) are:[7]

(2) $$\frac{A \multimap B : a \quad B : b}{A : (ab)} \multimap E \qquad \frac{\begin{array}{c}[B : v] \\ A : a\end{array}}{A \multimap B : \lambda v.a} \multimap I$$

The proof in (3) illustrates 'hypothetical reasoning', where an additional assumption, or 'hypothetical', is used that is latter discharged. The involvement of hypotheticals is driven by the presence of higher-order formulae (i.e. functors seeking an argument that bears a functional type): each corresponds to a subformula of a higher-order formula,

[1] See (Joshi *et al.*, 1997; Henderson, 1992) for other work connecting categorial formalisms (Lambek calculus and CCG, respectively) to tree-oriented formalisms.

[2] The indexation is my own, for expositional purposes.

[3] A second operation, *sister-adjunction*, used in handling modification, is discussed later in the paper.

[4] Multi-Component TAG with Domination Links (Becker *et al.*, 1991).

[5] The multiplicative fragment extends the implicational one with $\otimes$ ('tensor'), akin to the Lambek product.

[6] This means, most notably, that the representations discussed lack any encoding of linear order requirements, which would be handled within the labelling system.

[7] Eliminations and introductions correspond to steps of functional application and abstraction, respectively, as the lambda-term labelling reveals. In the $\multimap$I rule, [B] indicates a discharged or withdrawn assumption.

e.g. Z in (3) is a subformula of Xo–(Yo–Z).[8]

$$(3)\quad \dfrac{\mathrm{X}:x(\lambda z.y(wz)) \qquad \dfrac{\mathrm{Y}:y(wz)}{\mathrm{Yo–Z}:\lambda z.y(wz)}}{\mathrm{X}:x(\lambda z.y(wz))}$$

Hepple (1996) shows how deductions in implicational linear logic can be recast as deductions involving only *first-order* formulae (i.e. where any arguments sought by functors bear *atomic* types) and using only a single inference rule (a variant of o–E). The compilation reduces higher-order formulae to first-order formulae by *excising* subformulae corresponding to hypotheticals, e.g. so Xo–(Yo–Z) gives Xo–Y *plus* Z. A system of indexing is used to ensure correct use of excised subformulae, to prevent invalid reasoning, e.g. the excised Z *must* be used to derive the argument of Xo–Y. Each compiled formula has an index set with one member (e.g. $\{j\}:\mathrm{Z}$), which serves as its unique identifier. The index set of a derived formula identifies the assumptions used to derive it. The single inference rule (4) ensures correct propagation of indices (where $\uplus$ is *disjoint* union). Each argument slot of a compiled functor also has an index set, which identifies any assumptions that *must* be used in deriving its argument, as enforced by the rule condition $\alpha \subseteq \psi$.

$$\dfrac{\{i\}:\mathrm{Xo–(Y:\{}j\mathrm{\})}\ \lambda t.x(\lambda z.t) \qquad \dfrac{\{k\}:\mathrm{Yo–(W:}\emptyset\mathrm{)}\ \lambda u.yu \qquad \dfrac{\{l\}:\mathrm{Wo–(Z:}\emptyset\mathrm{)}\ \lambda v.wv \qquad \{j\}:\mathrm{Z}\ z}{\{j,l\}:\mathrm{W}:wz}}{\{j,k,l\}:\mathrm{Y}:y(wz)}}{\{i,j,k,l\}:\mathrm{X}:x(\lambda z.y(wz))}$$

In proving Xo–(Yo–Z), Yo–W, Wo–Z ⇒ X, for example, compilation yields the assumption formulae of the proof above. The leftmost (F1) and rightmost (F2) assumptions both come from Xo–(Yo–Z), and F1 requires its argument to include F2. Compilation has removed the need for an explicit introduction step in the proof, c.f. proof (3), but the effects of this step have been compiled into the semantics of the formulae. Thus, the term of F1 includes an apparently vacuous abstraction over variable z, which is the term assigned to F2. The semantics of rule (4) is handled not by simple application, but rather direct substitution for the variable of a lambda expression, employing a version of substitution which specifically does not act to avoid accidental binding. Hence, in the final step of the proof, the variable z falls within the scope of the abstraction, and so becomes bound.

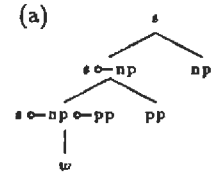
$$(4)\quad \dfrac{\phi:\mathrm{Ao–(B:}\alpha\mathrm{)}:\lambda v.a \qquad \psi:\mathrm{B}:b}{\pi:\mathrm{A}:a[b/\!\!/v]} \qquad \begin{array}{l}\pi = \phi \uplus \psi \\ \alpha \subset \psi\end{array}$$

[8]The relevant subformulae can be precisely characterised in terms of a notion *polarity*: hypotheticals correspond to maximal positive-polarity subformulae of higher-order formulae. See (Hepple, 1996) for details.

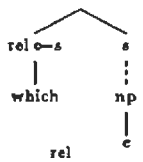
5 Relating The Two Systems

The above compilation produces results that bear more immediate similarities to the D-Tree approach than the original type-logical system. First-order formulae are easily viewed as tree fragments (in a way that higher-order formulae are not), e.g. a word w with formula so–npo–pp might be viewed as akin to (5a) below (modulo the order of daughters which is not encoded). For a higher-order formula, the inclusion requirement between its first-order derivatives is analogous to a domination link within a d-tree, e.g. a relative pronoun rel/(s/np) would yield rel o–s plus np, which we can view as akin to (5b).

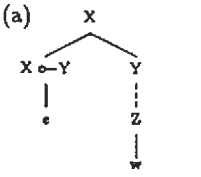
(5) (a)



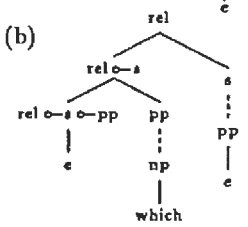
(b)



(6) (a)



(b)



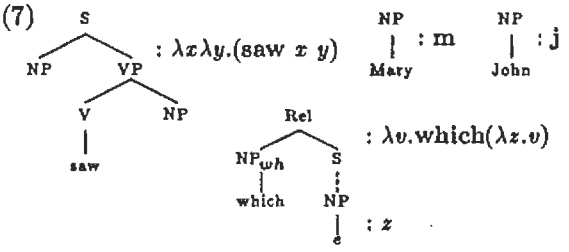
By default, it is natural to associate the string of the initial formula with its main residue under compilation, as in (5b). Following proposals in (Moortgat, 1988; 1996), some categorial systems have used connectives ↑ ('extraction') and ↓ ('infixation'), where Y↑Z is a "Y missing Z somewhere" and a type X↓(Y↑Z) infixes its string to the position of the missing Z. Thus, a word w with type X↓(Y↑Z), compiling to Xo–Y and Z, is akin to (6a). For example, the PP pied-piping relative pronoun type rel/(s↑pp)↓(pp↑np), from (Morrill, 1992), which infixes to an NP site within a PP, is akin to (6b).

6 A Functional Approach to Interpreting DTG Derivations

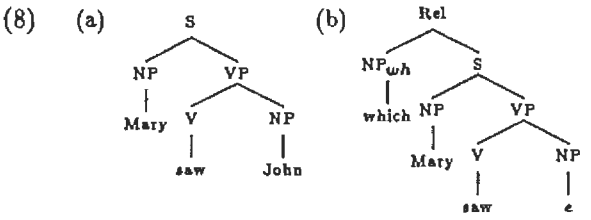
The rest of this paper explores the idea of providing a functional semantics for DTG derivations, or rather of some DTG-like formalism, in a manner akin to that of categorial grammar. The approach envisaged is one in which each tree fragment (i.e. maximal unit containing no dominance links) of an initial d-tree is associated with a lambda term. At the end of a derivation, the meaning of the resulting tree would be computed by working bottom up, applying

the meaning term of each basic tree fragment to the meanings computed for each complete subtree added in at the fragment's frontier nodes, in some fixed fashion (e.g. such as in their right-to-left order). Strictly, terms would be combined using the special substitution operation of rule (4) (allowing variable capture in the manner discussed). Suitable terms to associate with tree fragments will be arrived at by exploiting the analogy between d-trees and higher-order formulae under compilation.

(7)



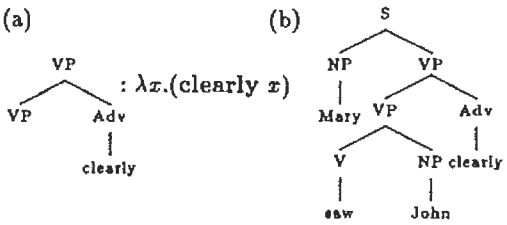
(8)



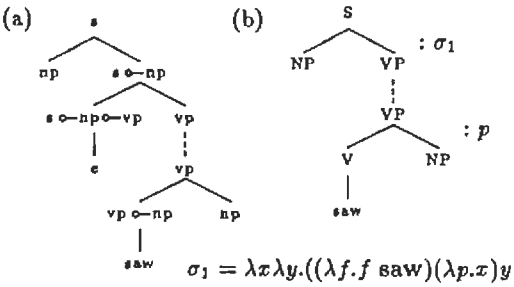
For example, consider a simple grammar consisting of the four d-trees in (7), of which only that for *which* has more than one fragment. Each tree fragment is associated with a meaning term, shown to the right of ":". The two fragments in the d-tree for *which* each have their own term, which are precisely those that would be assigned for the two compiled formulae in (5b) (assuming the meaning term for the precompilation formula rel/(s/np) to be just **which**). This grammar allows the phrase-structure (8a) for *Mary saw John*, whose interpretation is produced by 'applying' the term for *saw* to that for the NP *John* (i.e. the subtree added in at the rightmost frontier node of *saw*'s single tree fragment), and then to that of the NP *Mary*, giving (**saw j m**). The grammar allows the tree (8b) for the relative clause *which Mary saw*.[9] Here, the object position of *saw* is filled by the lower fragment of *which*'s d-tree, so that the subtree rooted at S has interpretation (**saw** z **m**). Combining this with the term of the upper fragment of *which* gives interpretation **which**(λz.**saw** z **m**).

The tree composition steps required to derive the trees in (8) would be handled in DTG by the subsertion operation. As noted earlier, DTG has a second composition operation *sister-adjunction*, used in handling modification, which adds in a modifier subtree as an additional daughter to an already existing local tree. A key motivation for this operation is so that DTG derivation trees distinguish argument vs. modifier dependencies, so as to provide an appropriate basis for interpretation. Categorial grammars typically make no such distinction in syntactic derivation, where all combinations are simply of functions and arguments. Rather, the distinction is implicit as a property of the lexical meanings of the functions that participate.[10] Accordingly, we recommend elimination of the sister-adjunction operation, with all composition being handled instead by subsertion. Thus, a VP modifying adverbial might have d-tree (9a), and give structures such as (9b).[11]

(9)



(10)



Such an analysis requires a different lexical d-tree for *saw* to that in (7), one where the VP node is 'stretched' as in (10b) to allow possible inclusion of modifiers. As a basis for arriving at suitable functional semantics for (10b), consider the following. A categorial approach might make *saw* a functor (np\s)/np with semantics **saw**. This functor could be type-raised to (np\s)↓((np\s)↑((np\s)/np)) with semantics ($\lambda f.f$ **saw**). By substituting the two embedded occurrences of (np\s) with the atom vp we get (np\s)↓(vp↑(vp/np)), which compiles to first-order formulae as in (10a), which are analogous to the desired d-tree (10b), so providing the meaning terms there assigned. Using (10b) to derive the structure (8a) involves identifying the two

[9] The treatment of *wh*-movement here exemplified is useful for expositional purposes, but clearly differs from the standard TAG/DTG approach, where a moved *wh*-item originates with a structure that includes the governor of the extraction site (typically a verb that subcategorises for the moved item). Such structures present no problem for this approach, i.e. we could simply precombine the d-trees of *which* and *saw* given in (7).

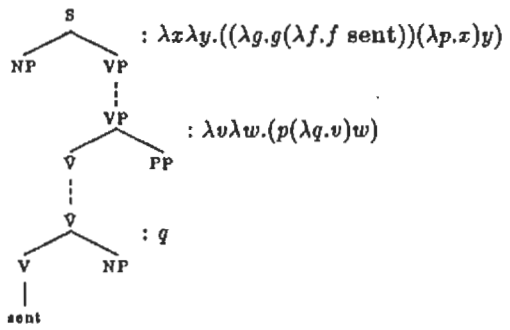
[10] This is not to say that the distinction has no observable reflex: modifiers are in general recognisable as endocentric categorial functors (i.e. having the same argument and result type).

[11] Such an analysis is more in line with the standard TAG treatment than that of DTG.

VP nodes. Such a derivation gives the interpretation $((\lambda f.f\ \textbf{saw})(\lambda p.p\ \textbf{j})\textbf{m})$ which simplifies to (**saw j m**). A derivation of (9b) gives interpretation $((\lambda f.f\ \textbf{saw})(\lambda p.\textbf{clearly}(p\ \textbf{j}))\textbf{m})$ which simplifies to (**clearly (saw j) m**).

For a ditransitive verb, we might want a structure providing more than one locus for inclusion of modifiers, such as (11). The semantics provided for this d-tree is arrived at by a similar process of reasoning to that for the previous case, except that it involves type-raising the initial categorial type of the verb *twice* (hence the subterm $(\lambda g.g(\lambda f.f\ \textbf{sent}))$ of the upper fragment's term).

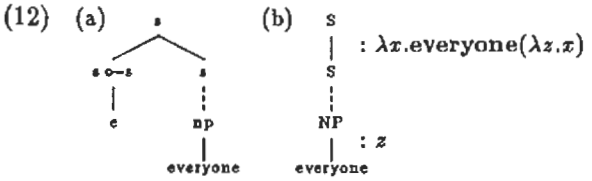
(11)



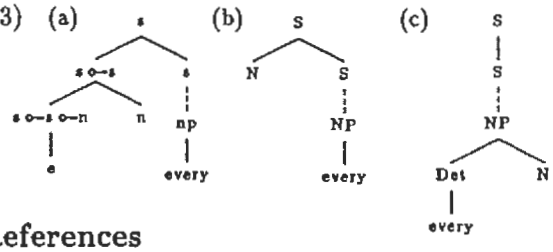
The interpretation approach outlined appears quite promising so far. We next consider a case it does not handle, which reveals something of its limitations: quantification. Following a suggestion of (Moortgat, 1996), the connectives ↑ ('extraction') and ↓ ('infixation') have been used in a categorial treatment of quantification. The lexical quantified NP *everyone*, for example, might be assigned type s↓(s↑np), so that it has scope at the level of some sentence node but its string will appear in some NP position. First-order compilation yields the results (12a). The corresponding d-tree (12b) is unusual from a phrase-structure point of view in that it's upper fragment is a purely interpretive projection, but this d-tree would serve to produce appropriate interpretations. So far so good.

A simple quantifier *every* has type s↓(s↑np)/n, to combine firstly with a noun, with the combined string of *every*+noun then infixing to a NP position. First-order compilation, however, produces the result (13a), comparable to the d-tree (13b), which is clearly an inappropriate structure. What we would hope for is a structure more like that in (13c), but although it is perfectly possible to specify an initial higher-order formula that produces first-order formulae comparable to this d-tree, the results do not provide a suitable basis for interpretation. More generally, the highly restrictive approach to semantic composition that is characteristic of the approach outlined is such that a fragment cannot have scope above its position in structure (although a d-tree having multiple fragments has access to multiple possible scopes). This means, for example, that *no* semantics for (13c) will be able to get hold of and manipulate the noun's meaning as something separate from that of the sentence predicate (c.f. s↑np), rather the former must fall within the latter.[12]

(12)



(13)



References

Becker, T., Joshi, A. & Rambow, O. 1991. 'Long distance scrambling and tree adjoining grammars. *Proc. EACL–91.*

Gabbay, D. 1996. *Labelled deductive systems. Volume 1.* Oxford University Press.

Henderson, J. 1992. 'A Structural Interpretation of CCG.' UPenn Tech. Report, MS-CIS-92-49.

Hepple, M. 1996. 'A Compilation-Chart Method for Linear Categorial Deduction.' *Proc. COLING–96.*

Joshi, A. & Kulick, S. 1997. 'Partial proof trees as building blocks for a categorial grammar.' *Linguistics and Philosophy.*

Lambek, J. 1958. 'The mathematics of sentence structure.' *American Mathematical Monthly*, 65.

Moortgat, M. 1988. *Categorial Investigations: Logical and Linguistic Aspects of the Lambek Calculus.* Foris, Dordrecht.

Moortgat, M. 1996. 'Generalized quantifiers and discontinuous constituency.' H. Bunt and A. van Horck (eds). *Discontinuous Constituency*, Mouton de Gruyter.

Morrill, G. 1992. 'Categorial Formalisation of Relativisation: Pied Piping, Islands and Extraction Sites.' Research Report LSI-92-23-R, Universitat Politécnica de Catalunya.

Rambow, O., Vijay-Shanker, K. & Weir, D. 1995. 'D-Tree Grammars.' *Proc. ACL–95.*

Shieber, S.M. & Schabes, Y. 1990. 'Synchronous tree-adjoining grammar.' *Proc. COLING–90.*

[12] See (Shieber & Schabes, 1990) for a treatment of quantification within the Synchronous TAG formalism, in which the semantics is treated as a second system of tree representations that are operated upon synchronously with syntactic trees. Their account cannot be adapted to the present approach because their operations upon syntactic and semantics representations, though *synchronous*, are not *parallel* in the way that is rigidly required in categorial semantics.

A Standard Representation Framework for TAG

Fabrice ISSAC
L.I.P.N. - Institut Galilée
Av J. B. Clément 93430 Villetaneuse
e-mail : fabrice.issac@lipn.univ-paris13.fr

Abstract

We present in this paper a markup language suitable for representing a tree adjoining grammar. Using a uniform way to represent TAG, the development of tools, e.g. parser/recognizer, editor, ..., could be done to the benefit of the entire TAG community.

Key words: SGML, linguistic tagging, data exchange.

Our work consists of proposing a framework dedicated to designing an XTAG-like standard environment. We present in this paper a markup language suitable for representing a tree adjoining grammar. The TAG formalism is used in plenty of works all around the world. However it is difficult to exchange syntactic data (*i.e.* a grammar or a piece of grammar written in the TAG formalism) as well as computer tools based on these syntactic data. Using a uniform way to represent TAG, the development of tools, e.g. parser/recognizer, editor, ..., could be done to the benefit of the entire TAG community. Note that all kinds of TAG (LTAG, MCTAG, ...) can be represented in our language. We choose SGML as descriptive language, as seen as briefly in the first section. in the second section, we provide an overview of the structure of a TAG document, followed in the third section by an example.

1 SGML

We describe our language in SGML(Goldfarb90; Herwijnen95) (Standard Generalized Markup Language). SGML is itself a metalanguage. SGML is an efficient tool to describe classes of documents because (i) it is an ISO specification[1], thus, a standard (ii) a lot of tools can be used to edit, verify or exploit SGML based documents[2].

SGML is a meta-language which allows specification through a Document Type Definition (DTD) :

- a set of markups;
- how these markups can be combined.

In our case, the class of documents is the set of TAG grammars. The most popular DTD is HTML which is used as a norm for data representation on the *Web* but there are other projects, notably the TEI project. The Text Encoding Initiative (TEI) (SMB94) is an international project to develop guidelines for the preparation and interchange of electronic texts.

The TEI proposes recommendations for feature structure markup (LS95) which can be used to represent any feature structure, including TAG. However, we think the markup set defined is not specific enough to be easily treated.

2 Structure of a TAG document

First a *good* TAG document is preceded by a prologue which indicates the TAG DTD version:
`<!DOCTYPE DTD PUBLIC "DTD TAG 0.2">`

The whole document is enclosed by the `<tag>` and `</tag>` markup. It is composed of a *header* and a *body*[3]. The *header*, enclosed by the `<tagheader>` and `</tagheader>` markup, contains information about the document itself: title, date of creation, name of the creator, origin of the data, type of data.

The *body*, enclosed by the `<ts>` (Tree Set) and `</ts>` markup, forms the usable part of the document. A tree set is a list of tree families (`<tf>`), elementary trees (`<et>`) or parsed trees (`<pt>`).

[1] ISO 8879:1986.

[2] For instance the `sgmlql` tool can extract part of document in relation to a query on the tags.

[3] As HTML, in fact it is a classical way to describe a SGML document.

<tf> (tree family) encloses a tree family. The attribute (name) indicates the name of the family. A tree family is composed of a list of trees (markup <t>).

<et> (elementary trees) encloses elementary trees. The attribute (name) indicates the name of the tree. As a family tree, elementary trees are composed of a list of trees (possibly one).

<pt> (parsed tree) encloses a parsed tree (*i.e.* a derived tree). Three markups are used to describe (i) the string recognized by the tree (<string>) (ii) the tree itself (<t>) and a set of derivation trees (DT).

A tree contains only one node (a node is indicate by <n>): the root node. The node markup can then be used recursively (a <n> can contain a <n>) to define the tree. A node contains some markups:

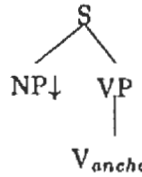
<val> (value) is the tag of the node (*i.e.* category for elementary and derived trees or tree name for a derived tree);

<fs> (feature structure) for FB-TAG.

The tree of the figure 1 indicates the relationships between markups[4].

3 An example

We give below a simple example. Let us suppose we have a tree with the features associated to the nodes :




```
NP_0.t:<num>=VP.t:<num>
NP_0:<pers>=VP.t:<pers>
S.b:<mode>=VP.t:<mode>
VP.b:<mode>=V.b:<mode>
VP.b:<num>=V.b:<num>
VP.b:<pers>=V.b:<pers>
NP_0.t:<wh>=-
S.b:<inv>=-
```

The SGML result is the following :

```
<!DOCTYPE DTD PUBLIC "DTD TAG 0.2>
<tag lang=french>
```

[4]Note that this tree is automaticaly generated with a SGML tool: dtdtree.

```
TAG
 |_(tagheader,
 |   |_(title &
 |   |   |_(#PCDATA)
 |   |
 |   |__date &
 |   |   |_(#PCDATA)
 |   |
 |   |__creator &
 |   |   |_(#PCDATA)
 |   |
 |   |__orig* &
 |   |   |_(#PCDATA)
 |   |
 |   |__type?)
 |       |_(#PCDATA)
 |
 |__ts+)
     |_(tf |
     |   |_(t+)
     |       |_(n)
     |           |_(val,
     |           |   |_(#RCDATA)
     |           |
     |           |__fs?,
     |           |   |_(f+)
     |           |       |_(l |
     |           |       |   |_(EMPTY)
     |           |       |
     |           |       |__#RCDATA)
     |           |
     |           |__n+) ...
     |
     |__et |
     |   |_(t+) ...
     |
     |__pt)+
         |_(string,
         |   |_(#PCDATA)
         |
         |__t+, ...
         |__dt?)
             |_(n) ...
```

Figure 1: DTD tree

```
<tagheader>
<title>TAG for an intransitive
structure</title>
<date>april 1998</date>
<creator>Fabrice Issac</creator>
<orig>Anne Abeillé</orig>
</tagheader> <tf name=s.np(v)>
<t name=s.np(v)>
<n id=n1>
<val>&S;</val>
<fs type=b>
<f name=mode><l id=f1></f>
<f name=inv>-</f>
</fs>
<n id=n2 type=substitute>
<val>&NP;</val>
<fs type=t>
<f name=num><l id=f2></f>
<f name=pers><l id=f3></f>
<f name=wh>-</f>
</fs>
</n>
```

```
<n id = n3>
<val>&VP;</val>
<fs type=t>
<f name=num><l id=f2></f>
<f name=pers><l id=f3></f>
<f name=mode><l id=f4></f>
</fs>
<fs type=b>
<f name=mode><l id=f5></f>
<f name=num><l id=f6></f>
<f name=pers><l id=f7></f>
</fs>
<n id=n4 type=anchor>
<val>&V;</val>
<fs type=b>
<f name=mode><l id=f5></f>
<f name=num><l id=f6></f>
<f name=pers><l id=f7></f>
</fs>
</n>
</n>
</n>
</t>
...
</tf>
</tag>
```

4 conclusion

Finally we will mention the way we can use such a description. It is obvious that an SGML document, as this one, is not supposed to be directly understandable/readable to humans. It is, in fact, used as input/output to computer tools. For instance, if developers follow these guidelines, the three steps of a parser – grammar generation, tree elimination, parsing – could be built by three different people.

The final environment will contain ;

- a graphical TAG editor, in order to create or to modify TAG grammars;
- tools for parsing (generation, disambiguation, parsing);
- miscellaneous tools (for instance a LaTeX or HTML transduction of TAG trees).

The aim of this paper is not to give a final (nor complete) version of a language providing descriptions of TAG, but rather act as a starting point. Actually, I think a data interchange norm can't be established by only one person. That's why I wish the community take a part in the development of this norm.

References

Goldfarb (Charles F.). – *The SGML handbook.* – Oxford, Clarendon Press, 1990.

Herwijnen (Eric Van). – *SGML pratique.* – Paris, International Thomson Publishing France, 1995.

Langendoen (D. Terence) et Simons (Gary F.). – A rationale for the tei recommendations for feature-structure markup. *In : Computers and the humanities*, pp. 191–209. – Kluwer Academic Publishers, 1995.

Sperberg-McQueen (C.M.) et Burnard (Lou). – *Guidelines for Electronic Text Encoding and Interchange (TEI P3).* – Text Encoding Initiative, 1994.

Appendix: The DTD

```
<!--**********************************************************
       file : tag.dtd
       author: Fabrice Issac
       date : April 1998
       Email : fabrice.issac@lipn.univ-paris13.fr
**********************************************************-->

<!ENTITY % TAG.Version
       "DTD TAG 0.2"
       -- Typical usage:
           <!DOCTYPE DTD PUBLIC "DTD TAG 0.2">
           <tag>
           ...
           </tag>
       --
       >

<!--======= Character mnemonic entities for french ====-->

<!ENTITY % Categories PUBLIC "TAG entities VERSION 1.0 FRENCH">
%Categories;

<!--======= ========================================== ====-->

<!ENTITY % TAG.Simple           "IGNORE">
<!ENTITY % TAG.Recommended      "INCLUDE">

<![%TAG.Recommended [
       <!ELEMENT TAG            - - (TAGHEADER,TS+)>
]]>

<![%TAG.Simple [
       <!ELEMENT TAG            - - (TAGHEADER?,TS+)>
]]>

<!ELEMENT TAG           - - (TAGHEADER,TS+)>
```

```
<!ELEMENT TAGHEADER     - - (TITLE & DATE & CREATOR & ORIG*
                             & TYPE?)>

<!ELEMENT TITLE         - - (#PCDATA)>
<!ELEMENT DATE          - - (#PCDATA)>
<!ELEMENT CREATOR       - - (#PCDATA)>
<!ELEMENT ORIG          - - (#PCDATA)>
<!ELEMENT TYPE          - - (#PCDATA)>

<!ELEMENT TS            - - (TF|ET|PT)+>
<!ATTLIST TS    LANG            CDATA           #IMPLIED
                ID              CDATA           #IMPLIED>

<!ELEMENT (TF|ET)       - - (T+)>
<!ATTLIST TF    NAME            CDATA           #REQUIRED
                ID              CDATA           #IMPLIED>
<!ATTLIST ET    ID              CDATA           #IMPLIED>

<!ELEMENT PT            - - (STRING,T+,DT?)>
<!ATTLIST PT    ID              CDATA           #IMPLIED>

<!ELEMENT STRING                - - (#PCDATA)>
<!ATTLIST STRING        ID      CDATA           #IMPLIED>

<!ELEMENT DT            - - (N)>

<!ELEMENT T             - - (N)>
<!ELEMENT N             - - (VAL,FS?,N*)>

<!ELEMENT VAL           - - (#RCDATA)>
<!ELEMENT FS            - - (F*)>
<!ELEMENT F             - - (L|#RCDATA)>

<!ELEMENT L             - O (EMPTY)>
<!ATTLIST L     ID              CDATA           #IMPLIED>
<!--=========================================================-->
```

Partial Proof Trees and Structural Modalities*

Aravind K. Joshi and Seth Kulick and Natasha Kurtonina
Institute for Research in Cognitive Science
University of Pennsylvania
Suite 400A, 3401 Walnut Street, Philadelphia, PA 19104-6228
{joshi,skulick,natashak}@linc.cis.upenn.edu

An important theme in current categorial research is the shift of emphasis from individual category logics to communicating families of such systems. The reason for this shift is that the individual logics are not expressive enough for realistic grammar development; the grammar writer needs access to the combined inferential capacities of family of logics. Categorial systems with structural modalities (see Moortgat 1997, Kurtonina and Moortgat 1997, Morrill 1994 for details) can incorporate not only limited relaxation of the rigid structure to provide more generative capacity, but also impose additional constraints to block undesired derivations[1]. Although they provide a powerful extension of capacities of categorial inference, their use can be linked to overgeneration in some cases. In this paper we will show how this problem can be handled if categorial systems based on partial proof trees are used as building blocks of the system. The key idea is that the use of PPTs allow us to 'localize' the management of resources, thereby freeing us from this management as the PPTs are combined.

Here we provide a very brief overview of the PPT system. See Joshi and Kulick (1997) for details. The basic idea is to associate with each lexical item one or more PPTs, obtained by unfolding the arguments of the type that would be associated with that lexical item in a simple categorial grammar, such as the Ajdukiewicz and Bar-Hillel grammar. The basic PPTs then serve as the building blocks of the grammar, and complex proof trees are obtained by 'combining' these PPTs by various inference rules, that basically allow the linking of conclusion nodes to assumption nodes, and the stretching of a node in a proof. The main motivation of this approach is to incorporate into the categorial framework the key insights from LTAG, namely the notion of an extended domain of locality and the consequent factoring of recursion from the domain of dependencies.

In CG the engine of grammatical inference is, of course, a multiplicative fragment of intuitionistic linear logic (Lambek Calculus) and logical derivability of some distinguished types from a sequence of types is crucial for determination of grammaticality of linguistic expressions. On a deductive level the logical architecture of categorial inference is reflected in the rules of a calculus (for instance, sequent calculus). In contrast to CG, the PPTs system is a tree rewriting system. However, we can make explicit the underlining logic of the system to provide a logical explanation of the resource management. In fact, two kinds of logics are involved in PPTs system. Construction of basic trees is guided by the logic of a CG, while operations of combining trees are monitored by a single rule – Cut.

We now consider the use of two kinds of structural modalities, following Moortgat (1997), Kurtonina and Moortgat (1997), Morrill (1994).

Structural Relaxation: Consider the relative clauses in (1a) and (2a):

(1) a. (the book) that John read
b. $r/(s/np), np, (np\backslash s)/np \Rightarrow r$
(2) a. (the book) that John read yesterday
b. $r/(s/np), np, (np\backslash s)/np, s\backslash s \Rightarrow r$

The two sentences correspond to the sequent derivations in (1b) and (2b). The former is a valid derivation, but the latter is not derivable. The problem is that the hypothetical np assumption is not in the required position adjacent to the verb. Here the so-called Permutation modality comes into the picture. We refine the assignment to the relative pronoun to the type $r/(s/np^{\natural})$, where the decoration with $^{\natural}$ indicates an access to *restricted* Permutation.

Structural Constraints: Interaction of the relative clause formation with coordination leads to

*We would like to thank Gerhard Jaeger, Alain Lecomte, Owen Rambow, Mark Steedman, K. Vijay-Shanker, and two anonymous reviewers for their valuable comments. This work was partially supported by NSF Grant SBR96-20230.

[1]In this paper we focus on categorial systems that use structural modalities. Another branch of categorial grammar is that represented by Combinatory Categorial Grammar (CCG) (Steedman 1996). Work is currently in progress to investigate the relationship between CCG and the partial proof tree system described here.

(3) a. (the book) that John wrote and Bob read
b. $r/(s/np), np, (np\backslash s)/np, (X\backslash X)/X, np, (np\backslash s)/np \Rightarrow r$
(4) a. (the book) that John wrote Moby Dick and Bob read
b. $r/(s/np), np, (np\backslash s)/np, np, (X\backslash X)/X, np, (np\backslash s)/np \Rightarrow r$

(5) $r/(s/np), (np, (np\backslash s)/np, (X\backslash \Box^{\downarrow} X)/X, np, (np\backslash s)/np)^{\diamond} \Rightarrow r$
(6) $np, (np\backslash s)/np, (X\backslash \Box^{\downarrow} X)/X, np, (np\backslash s)/np \Rightarrow \Box^{\downarrow}(s/np)$

(7) a. (the book) that John wrote yesterday and Bob read today
b. $r/(s/np^{\natural}), np, (np\backslash s)/np, s\backslash s, (X\backslash X)/X, np, (np\backslash s)/np, s\backslash s \Rightarrow r$

overgeneration. Sentence (3a), with the corresponding sequent (3b), is derivable with X instantiated to s/np. However, the ungrammatical (4a), corresponding to the sequent (4b), can be derived with X instantiated to s.

This problem can be fixed by refining the type assignment to 'and' to be $(X\backslash \Box^{\downarrow} X)/X$ and by closing off the coordinate structure with the dual structural modality $\diamond$. The resulting sequent corresponding to (3) is now (5), with its validity proved by (6):

The island violation (4) fails, because the hypothetical np assumption finds itself in the scope of modal operator. Thus, the idea of the approach is to *freeze* complete coordination into an island configuration. The introduction of this other type of structural modality imposes structural *constraints* rather than structural *relaxation*, as with the permutation modality.

Conflict: However, if the two types of modalities appear in the same sentence, then they require a simultaneous relaxation and constraining of the interaction between the types. Consider the derivation of (7a), with the corresponding sequent (7b).

To derive this sequent, X must be instantiated as $(s/np^{\natural})$, due to the presence of *yesterday* and *today*. And, as we just saw, the type for *and* should have the type assignment $(X\backslash \Box^{\downarrow} X)/X$, and so the type for *and* in this example becomes $((s/np^{\natural})\backslash \Box^{\downarrow}(s/np^{\natural}))/(s/np^{\natural})$. It is unfortunate that such a complex type for *and* is required simply because of the way that adverbs interact with extraction in the inference system. Using PPTs offers an interesting way to resolve the conflict, because of the way that it employs two different logics.

We cannot show the relevant PPTs here for space reasons. However, the basic idea is that, as discussed in Joshi and Kulick (1997), permutation is not needed for an adverb with a relative clause as in (2a) since the adverb is simple inserted via "stretching" a node in the object relative clause tree. Refinement of the system to account for coordination allows the derivation of (3a), while (4a) is ruled because, of course, the two conjuncts need to be of the same type, and they cannot coordinate if one is s while the other is s/np. Crucially, allowing (7a) is not a problem, since the adverbs simply come in via "stretching", and have no effect whatsoever on the type constraints for coordination. Therefore, there is no need for any modification of the basic type for coordination.

We conclude that by using PPTs, the linguistic phenomena motivating the introduction of structural modalities in categorial grammar can be handled by either eliminating them (such as for an adverb in a relative clause) or by retaining them but *localizing* them within basic PPTs (e.g., topicalization by permutation, as described in Joshi and Kulick 1997), thus avoiding the problem of overgeneration which requires constraints on modalities. This is due to the existence of of two types of logic in the PPTs, a consequence of combining trees rather than just strings, and is a very desirable consequence of localizing the management of resources in the PPT system.

References

Joshi, A. K., and S. Kulick (1997) "Partial Proof Trees as Building Blocks for a Categorial Grammar," *Linguistics and Philosophy* 20, 637–667.

Kurtonina, N., and M. Moortgat (1997) "Structural Control," in P. Blackburn and M. de Rijke, eds., *Specifying Syntactic Structures*, CSLI.

Moortgat, M. (1997) "Categorial Type Logics," in J. V. Benthem and A. T. Meulen, eds., *Handbook of Logic and Language*, North Holland.

Morrill, G. (1994) *Type Logical Grammar - Categorial Logic of Signs*, Kluwer, Dordrecht.

Steedman, M. (1996) *Surface Structure and Interpretation*, Linguistic Inquiry Monograph 30, MIT Press.

A hierarchy of local TDGs

Laura Kallmeyer
Universität Tübingen
Seminar für Sprachwissenschaft
Wilhelmstr. 113
D-72074 Tübingen, Germany
lk@sfs.nphil.uni-tuebingen.de

1 Introduction

Many recent variants of *Tree Adjoining Grammars (TAG)* allow an underspecification of the parent relation between nodes in a tree, i.e. they do not deal with fully specified trees as it is the case with TAGs. Such TAG variants are for example *Description Tree Grammars (DTG)* (Rambow, Vijay-Shanker and Weir 1995), *Unordered Vector Grammars with Dominance Links (UVG-DL)* (Rambow 1994a, 1994b), a definition of TAGs via so-called *quasi-trees* (Vijay-Shanker 1992), (Rogers and Vijay-Shanker 1994), (Rogers 1994) and *(Local) Tree Description Grammars (TDG)* (Kallmeyer 1997, 1998a). The last TAG variant, local TDG, is an extension of TAG generating tree descriptions. Local TDGs even allow an underspecification of the dominance relation between node names and thereby provide the possibility to generate underspecified representations for structural ambiguities such as quantifier scope ambiguities.

This abstract deals with formal properties of local TDGs. A hierarchy of local TDGs is established together with a pumping lemma for local TDGs of a certain rank. With this pumping lemma one can prove that the class of local TDGs of a certain rank n contains the language $L_i := \{a_1^k \cdots a_i^k \mid k \geq 0\}$ iff $i \leq 2n$.

2 Local TDGs

Local TDGs, proposed in (Kallmeyer 1997), consist of tree descriptions, so-called *elementary descriptions*, and a specific *start description*. These tree descriptions are negation and disjunction free formulas in a quantifier-free first order logic. This logic allows the description of relations between node names k_1, k_2 such as parent relation (i.e. immediate dominance) $k_1 \lhd k_2$, dominance (reflexive transitive closure of the parent relation) $k_1 \lhd^* k_2$, linear precedence $k_1 \prec k_2$ and equality $k_1 \approx k_2$. Furthermore, nodes are supposed to be labelled by terminals or by atomic feature structures. The labeling function is denoted by δ, and for a node name k, $\delta(k) \approx t$ signifies that k has a terminal label t, and $a(\delta(k)) \approx v$ signifies that k is labelled by a feature structure containing the attribute value pair (a, v).

Tree descriptions in a local TDG are of a certain form, roughly speaking they consist of fully specified (sub)tree descriptions that are connected by dominance relations.[1]

In an elementary description ψ, some of the node names are *marked* (those in the set K_ψ); this is important for the derivation of descriptions. A sample local TDG is shown in Fig. 1 (in the graphical representations, some of the node names are omitted for reasons of readability). Conjuncts such as $k_1 \lhd^* k_2$ in ϕ_S that are not entailed by the other conjuncts, are called *strong dominance*.

Starting from the start description ϕ_S, local TDGs generate tree descriptions. In each derivation step, a derived ϕ_1 and an elementary description ψ are combined to obtain a new description ϕ_2. Roughly said, ϕ_2 can be viewed as a conjunction of ϕ_1, ψ and new formulas $k \approx k'$ or $k \lhd^* k'$ where k is a name from ϕ_1 and k' a name from ψ. This derivation step must be such that

1. for a node name k_ψ in ψ, there is a new equivalence iff either k_ψ is marked or k_ψ is minimal (dominated by no other name, e.g. k_6 in ψ_1 and k_{11} in ψ_2 in Fig. 1),

2. a marked or minimal name k' in ψ that is not a leaf name (i.e. dominates other names) but does not dominate any other marked name must become equivalent to a leaf name in ϕ_1

3. the names k from ϕ_1 that are used for the new equivalences must be part of one single elemen-

[1] Some of the conditions holding for descriptions in a local TDG are left aside here. For a formal definition of local TDGs see (Kallmeyer 1998a).

tary or start description, the so-called *derivation description* of this derivation step (first locality condition),

4. for each marked name k_ψ in ψ with a parent, there must be a strong dominance $k_1 \lhd^* k_2$ in ϕ_1 such that $k_2 \approx k_\psi$ is added and the subdescription between k_ψ and the next marked or minimal name dominating k_ψ must be dominated by k_1 (second locality condition),

5. and the result ϕ_2 must be maximally underspecified.

As the first condition shows, marked names are comparable to foot nodes in an auxiliary tree in a TAG since they specify those parts of an elementary description ψ that must be connected to a derived description ϕ when adding ψ to ϕ in a derivatiou step.

The second condition describes a kind of substitution. Only leaf names in the old description can become equivalent to names that do not dominate other marked names.

Conditions 3. and 4. express the locality of the derivations. All names in the old description that are chosen for new equivalences must be part of the derivation description, and furthermore a subdescription between two minimal or marked names must be "inserted" into a strong dominance where the dominated name is part of the derivation description. These conditions can be compared to the locality restriction of the derivation in a *set-local multicomponent TAG (MC-TAG)* (Weir 1988). In fact, for each set-local MC-TAG, an equivalent local TDG can be constructed (Kallmeyer 1998a). However, local TDGs are more powerful than set-local MC-TAGs because the locality condition restricts only the derivation of descriptions but not the way a minimal structure for a derived description is obtained. This locality constitutes a crucial difference between local TDGs and DTGs since derivations in DTGs are non-local. Each subtree of a d-tree that is added in a derivation step to a derived d-tree γ can be inserted into any of the d-edges in γ.

If a marked name has no parent, then an underspecification of the dominance relation can occur in the result of a derivation step (see (Kallmeyer 1998b, Kallmeyer 1998a)). In this paper, such cases are not considered, and for the examples mentioned here, the fifth condition is of no consequence.

In Fig. 1 for example, a derivation step $\phi_S \overset{\psi_2}{\Rightarrow} \phi_1$ is possible with $\phi_1 = \phi_S \wedge \psi_2 \wedge k_1 \approx k_{11} \wedge k_2 \approx k_{17} \wedge k_4 \approx k_{23} \wedge k_3 \lhd^* k_{18}$.

A local TDG generates a set of descriptions. Each of these descriptions denotes infinitely many trees. The trees in the *tree language* of a local TDG are those trees that are "minimal" for one of the derived descriptions. A *minimal* tree of a description ϕ is a tree γ satisfying ϕ in such a way that

1. all parent relations in γ are described in ϕ, and

2. if two different node names in ϕ denote the same node in γ, then these two names neither have both a parent in ϕ nor have both a daughter in ϕ.

The first condition makes sure that everything in γ is described in ϕ, and with the second condition no parent relation in the tree is described more than once in ϕ.

For the local TDG in Fig. 1 for example, only those descriptions have a minimal tree that are derived by adding ψ_1 in the last derivation step.

The *string language* of a local TDG G is the set of all strings yielded by the trees in the tree language of G.

TDGs allow "multicomponent" derivations and a uniform complementation operation similar to subsertion in DTGs. Furthermore, they provide underspecified representations for scope ambiguities (Kallmeyer 1998b) since they allow the generation of descriptions with underspecified dominance relations.

3 Rank of a local TDG

For a given TAG, an equivalent local TDG with at most one marked name per elementary description can be easily constructed. Obviously, the extra power of local TDGs in contrast to TAGs arises from the possibility of marking more than one node name in an elementary description. In Fig. 1 for example, ψ_1 and ψ_2 both contain two marked names. The language generated by this local TDG is no TAL. This suggests the definition of a hierarchy of local TDGs depending on the maximal number of marked node names in an elemeutary description.

Two kinds of marked names can be distinguished: marked names where the part of the description dominating this name can be put somewhere "in between" on the one hand (e.g. k_{17} and k_{23} in ψ_2 in Fig. 1), and on the other hand marked node names that must be identified with a leaf name (e.g. k_3 and k_4 in ψ_2 in Fig. 2). Since there is a similarity between foot nodes of auxiliary trees in TAGs and the first kind of marked node names, these are called *adjunction-marked (a-marked)*. For similar reasons, the second

Start description:

$$\varphi_S = k_1 \lhd^* k_2 \wedge k_2 \lhd k_3 \wedge k_3 \lhd^* k_4 \wedge k_4 < k_5$$
$$\wedge\, cat(\delta(k_1)) \approx S \wedge cat(\delta(k_2)) \approx T_1$$
$$\wedge\, cat(\delta(k_3)) \approx T_2 \wedge cat(\delta(k_4)) \approx T_3 \wedge \delta(k_5) \approx \epsilon$$

Elementary descriptions:

$$\psi_1 = k_6 \lhd^* k_7 \wedge k_7 \lhd k_8 \wedge k_8 \lhd^* k_9 \wedge k_9 \lhd k_{10}$$
$$\wedge\, cat(\delta(k_6)) \approx S \wedge \cdots$$
$$\psi_2 = k_{11} \lhd^* k_{12} \wedge k_{12} \lhd k_{13} \wedge k_{12} \lhd k_{14} \wedge k_{12} \lhd k_{27}$$
$$\wedge\, k_{13} \prec k_{14} \wedge k_{14} \prec k_{27} \wedge k_{14} \lhd^* k_{15} \wedge \ldots$$
$$\ldots \wedge cat(\delta(k_{11})) \approx S \wedge cat(\delta(k_{12})) \approx S \wedge \ldots$$
$$\ldots \wedge \delta(k_{26}) \approx a_7 \wedge \delta(k_{27}) \approx a_8$$

$$K_{\psi_1} = \{k_8, k_{10}\}, K_{\psi_2} = \{k_{17}, k_{23}\}$$

Graphical representations:
(marked names with asterisk)

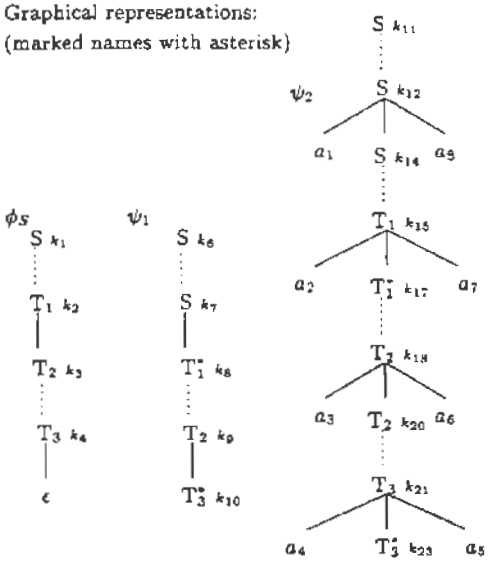


Figure 1: Local TDG for $\{a_1^n a_2^n a_3^n a_4^n a_5^n a_6^n a_7^n a_8^n \mid 0 \leq n\}$ with two a-marked names in each elementary description

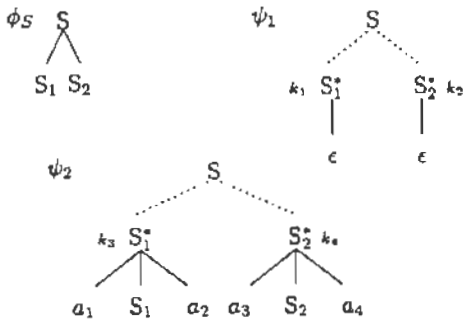


Figure 2: Local TDG for $\{a_1^n a_2^n a_3^n a_4^n \mid 0 \leq n\}$ with two s-marked names in each elementary description

kind of marked names are called *substitution-marked (s-marked)*.[2]

Roughly speaking, in a derivation step, for each s-marked name in the new elementary description, there is one substring added to the yield of the description, and for each a-marked name, two substrings are added (e.g. $a_1 a_2$ for k_3 in Fig. 2, $a_1 a_2$ and $a_7 a_8$ for k_{17} in Fig. 1 and $a_3 a_4$ and $a_5 a_6$ for k_{23} in Fig. 1). Therefore, a-marked names count twice as much as s-marked names for the rank of a local TDG: a local TDG G is of rank n iff $n = max\{i \mid$ there is an elementary ψ in G such that i is twice the number of a-marked names in ψ plus the number of s-marked names in $\psi\}$.

For a given local TDG it is always possible to find a weakly equivalent local TDG with one more s-marked name per elementary description. Therefore, the class of languages generated by local TDGs of rank i forms a subset of the class of languages generated by local TDGs of rank $i+1$ for $i \geq 0$.

As shown in (Kallmeyer 1998a), the classes of local TDLs of rank 0 and 1 are equal, they are exactly the context-free languages. The class of local TDLs of rank 2 contaius all TALs.

4 A pumping lemma

The idea of the pumping lemma for local TDGs of a certain rank n is similar to the one leading to the pumping lemma for TALs in (Vijay-Shanker 1987). As shown in (Kallmeyer 1997), the derivation process in a local TDG can be described by a context-free grammar G_{CF}. For G_{CF}, the pumping lemma for context-free languages holds. This means that in a derivation tree (of G_{CF}) from a certain tree height on, there is a subtree γ that can be iterated. For the corresponding local TDG, this signifies that an elementary ψ can be added twice such that: before adding ψ again we have the following situation for a string w yielded by the old description: $w = x_{10} v_1 \cdots x_{1m-1} v_m x_{1m}$ where $x_{1i} \in T^*$, $v_1 \cdots v_m$ is the string yielded by the subdescription derived from ψ (ordered by linear precedence). As a next derivatiou step, ψ is added again. If the grammar is of rank n, then by adding ψ, the string w can be split by inserting at most n new strings. Before the next adding of ψ (corresponding to another iteration) takes place, these substrings will be expanded to substrings $w_1, \cdots, w_n$ with $w_1 \cdots w_n = v_1 \cdots v_m$. These w_i may be split into several words (with other words in between) but the order of the letters is as

[2]These two characterizations are not exclusive, for examples of node names that are both a-marked and s-marked see (Kallmeyer 1998a).

in $v_1 \cdots v_m$. If this is repeated k times, $k \geq 1$, then one ends up with a word contaning the letters of $x_1 := x_{10} \cdots x_{1m}$ and k occurrences of all symbols of $w_1 \cdots w_n$ that are for each of these occurrences (from left to right) ordered as in $w_1 \cdots w_n$. In the last steps (after the iterations of the derivation subtree γ), the symbols of some string $x_2 \in T^*$ are added.

Therefore the pumping lemma is as follows: for each word w in the string language of a local TDG of rank n with $|w|$ greater than some constant c_G: after removing the letters of some words x_1 and x_2 from w, the resulting word has the form $w_1 \cdots w_n$. Then for each k there is a word $w^{(k)}$ in the language containing also the letters of x_1 and x_2, such that: if these letters are removed from $w^{(k)}$, the result $\tilde{w}^{(k)}$ is a word that can be obtained by taking k occurrences of $w_1 \cdots w_n$ and then, starting with ϵ, taking (in arbitrary order) always the left letter of one of these k words as the next letter in $\tilde{w}^{(k)}$. Furthermore, $\tilde{w}^{(k)}$ still contains as substrings one occurrence of each of the words $w_1, \cdots, w_n$ (in this order).

For the language $L_{2n} := \{a_1^m \cdots a_{2n}^m \mid 0 \leq m\}$ for example the lemma for rank n holds with $c_G = 2n - 1$, $x_1 = x_2 = \epsilon$: if $w = a_1^m \cdots a_{2n}^m$, then $w_i = a_{2i-1}^m a_{2i}^m$.

With the pumping lemma, it can be easily shown that for $i > 2n$, $L_i = \{a_1^m \cdots a_i^m \mid m \geq 0\}$ does not satisfy the pumping lemma for TDGs of rank n and therefore cannot be generated by a local TDG of rank n.

Consequently, for all $n \geq 1$, the string languages of TDGs of rank n form a proper subset of the string languages generated by TDGs of rank $n + 1$.

5 Conclusion

In this paper, the rank of a local TDG was defined based on the number of marked names in the elementary descriptions of the grammar. Two kinds of marked names are distinguished, namely s-marked and a-marked names. Since derivations in local TDGs can be described by a context-free grammar, the pumping lemma for context-free grammars can be applied to the derivation trees of a local TDG. This leads to the proof of a pumping lemma for local TDGs of a certain rank n. Roughly said, according to this pumping lemma, in a derivation step, for each s-marked name in the new elementary description, one substring is added, and for each a-marked name, two substrings are added. With this pumping lemma one can show that for $n \geq 1$ the languages generated by local TDGs of rank n form a proper subset of languages generated by local TDGs of rank $n + 1$.

References

Kallmeyer, L.: 1997, Local Tree Description Grammars, *Proceedings of the Fifth Meeting on Mathematics of Language, DFKI Research Report.*

Kallmeyer, L.: 1998a, *Tree Description Grammars and Underspecified Representations*, PhD thesis, Universität Tübingen. To appear in Arbeitspapiere des SFB 340.

Kallmeyer, L.: 1998b, Underspecification in Tree Description Grammars, *in* H. P. Kolb and U. Mönnich (eds), *The Mathematics of Syntactic Structures*, Mouton de Gruyter. To appear.

Rambow, O.: 1994a, *Formal and Computational Aspects of Natural Language Syntax*, PhD thesis, University of Pennsylvania.

Rambow, O.: 1994b, Multiset-Valued Linear Index Grammars: Imposing dominance constraints on derivations, *Proceedings of ACL.*

Rambow, O., Vijay-Shanker, K. and Weir, D.: 1995, D-Tree Grammars, *Proceedings of ACL.*

Rogers, J.: 1994, *Studies in the Logic of Trees with Applications to Grammar Formalisms*, PhD thesis, University of Delaware.

Rogers, J. and Vijay-Shanker, K.: 1994, Obtaining trees from their descriptions: an application to Tree-Adjoining Grammars, *Computational Intelligence* **10**(4), 401–421.

Vijay-Shanker, K.: 1987, *A Study of Tree Adjoining Grammars*, PhD thesis, University of Pennsylvania.

Vijay-Shanker, K.: 1992, Using descriptions of trees in a tree adjoining grammar, *Computational Linguistics* **18**(4), 481–517.

Weir, D. J.: 1988, *Characterizing mildly context-sensitive grammar formalisms*, PhD thesis, University of Pennsylvania.

A 'Tree Adjoining' Grammar without Adjoining
The case of scrambling in German

Gerard Kempen
Department of Psychology
Leiden University
PO Box 9555
NL-2300 RB Leiden
The Netherlands
kempen@rulfsw.fsw.leidenuniv.nl

Karin Harbusch
Computer Science Department
University of Koblenz-Landau
Rheinau 1
D-56075 Koblenz
Germany
harbusch@informatik.uni-koblenz.de

The psycholinguistically motivated grammar formalism of *Performance Grammar* (PG, [Kempen 97]) is similar to recent versions of *Tree Adjoining Grammar* (TAG; cf. [Joshi *et al.* 91]) in several important respects. It uses lexicalized initial trees; it generates derived trees synchronously linked to conceptual structures described in the same formalism (as in Synchronous TAGs [Shieber, Schabes 90]); and it factors dominance relationships and linear precedence in surface structure trees ([Joshi 87]).

PG differs from recent TAG versions in that the adjoining operation and auxiliary trees are absent. Adjunction is replaced by a combination of substitution—the only composition operation—and a special linearization component that takes care of ordering the branches of derived trees in a global manner without re-arranging the derived structures. PG has been worked out for substantial fragments of Dutch, including the well-known cross-serial dependencies in self-embedded clauses. Here we will outline how PG deals with scrambling phenomena in German without invoking adjunction. For TAG treatments of these phenomena we refer to [Becker *et al.* 91] and [Rambow 94].

PG's lexicalized initial trees, called *lexical frames*, are 3-tiered mobiles. The top layer of a frame consists of a single *phrasal* node (called the 'root'; e.g. S, NP, ADJP, PP), which is connected to one or more *functional* nodes in the second layer (e.g., SUBJect, HeaD, Direct OBJect, CoMPlement, MODifier). At most one exemplar of a functional node is allowed in the same frame, except for MOD nodes, which may occur several times (indicated by the Kleene star: MOD*). Every functional node dominates exactly one phrasal node in the third ('foot') layer, except for HD which immediately dominates a lexical (part-of-speech) node. Each lexical frame is 'anchored' to a lexical item—a 'lemma' printed below the lexical node serving as the frame's HeaD (Fig. 1).

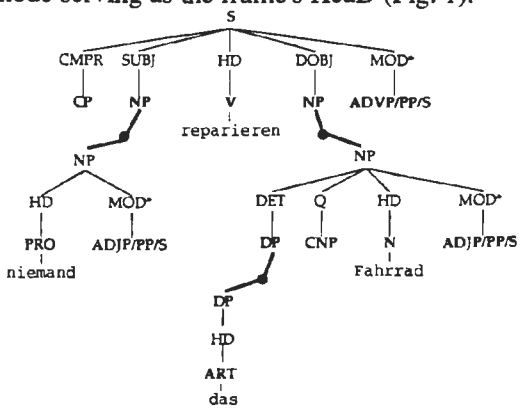


Fig. 1. Simplified examples of lexical frames. CP = Complementizer Phrase; CMPR = Complementizer; DP = Determiner Phrase. Left-to-right order of branches is arbitrary. The unifications (filled circles) correspond to German sentences such as Repariert niemand das Fahrrad? *or* Niemand repariert das Fahrrad *('Does nobody repair the bicycle?' or 'Nobody repairs the bicycle').*

Associated with nodes in the top and bottom layers are feature matrices (not discussed here), which can be *unified* with other matrices as part of the substitution process. Unification always involves one root and one foot node of two different lexical frames (see the filled circles in Fig. 1). Only non-recursive unification is used.

Left-to-right order of the branches of a lexical frame is determined by the 'linearizer' associated with a lexical frame. We assume that every lexical frame has a one-dimensional array specifying a fixed number of positions for foot nodes. For instance, verb frames (i.e., frames anchored to a verb) have an array whose positions can be occupied by a Subject NP, a Direct Object NP, the Head verb, etc. Fig. 2 shows 13 out of 14 slots where foot nodes of German verb frames can go. The positions numbered M1 through M11 belong to the Midfield (Ger. *Mittelfeld*); B1 and B2 make up the Backfield (*Nachfeld*). Not shown

is the single Forefield (*Vorfeld*) slot F1, located to the left of M1. The annotations at the arcs denote possible fillers of the slots. For example, in a main clause the Head verb is assigned the first Midfield slot (M1); in a subordinate clause it goes to the last Midfield position (M11). Subject NPs that could not enter the Forefield (e.g. in subordinate clauses) are placed in M2 if its head is a personal pronoun, in M3 otherwise. (Note that frames anchored to other parts of speech than verbs (NP, PP) have their own specialized linearization array.)

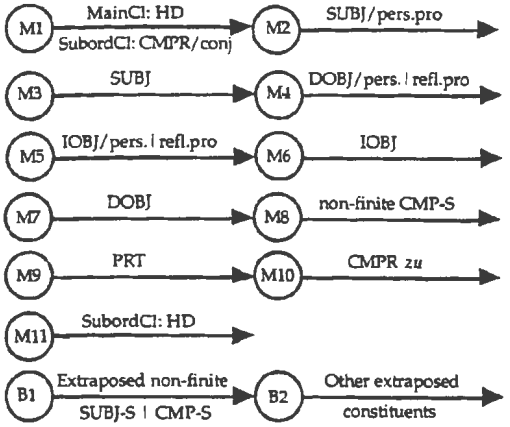


Fig. 2. Positions licensed to various types of constituents in the Midfield and Backfield of German clauses.

The fillers listed in slots M2 through M7 represent the unmarked order of verbal arguments (cf. [Uszkoreit 87]). They may be accompanied by additional constituents, in particular by modifiers and by arguments that, because of being in emphatic or contrastive focus, have been moved to the left (e.g. in *weil er ein Fahrrad den Kindern verspricht*, because-he-a-bike-the-children-promises). These companions are positioned after the 'standard' fillers (if any).

A key property of linearization in PG is that certain constituents may move out of their 'own' array and receive a position in an array located at a higher level. This is because, due to subcategorization features, a linearization array may be instantiated incompletely. For instance, if a verb takes a non-finite complement clause, then slots M1 through M3 are missing from the complement's array. If, in addition, the complement is subjected to 'clause union', slots M4 through M7 are absent as well. In such cases, verb arguments and adjuncts that need to be expressed overtly, look for a slot higher up in the hierarchy of verb frames and get hold of the first (i.e. lowest) slot that is within scope. E.g., in *daß sie den Lehrer das Fahrrad nicht reparieren sah* (that she didn't see the teacher repair the bike), *den Lehrer* and *das Fahrrad* occupy the same M7 slot, in order of increasing depth (Fig. 3).

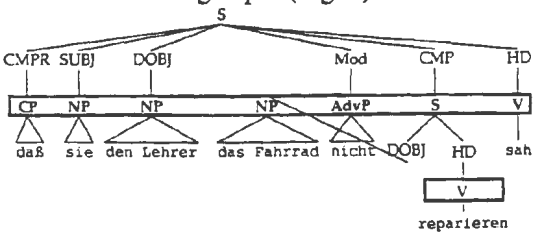


Fig. 3. The embedded DOBJ-NP has been lifted into the linearization array (rectangle) of the next higher verb frame. Due to a subcategorization feature of the lexical entry sehen *(to see), only slots M8-M10 of the complement clause have been instantiated. This causes* das Fahrrad *to land in the M7 slot of the matrix, joining* den Lehrer.

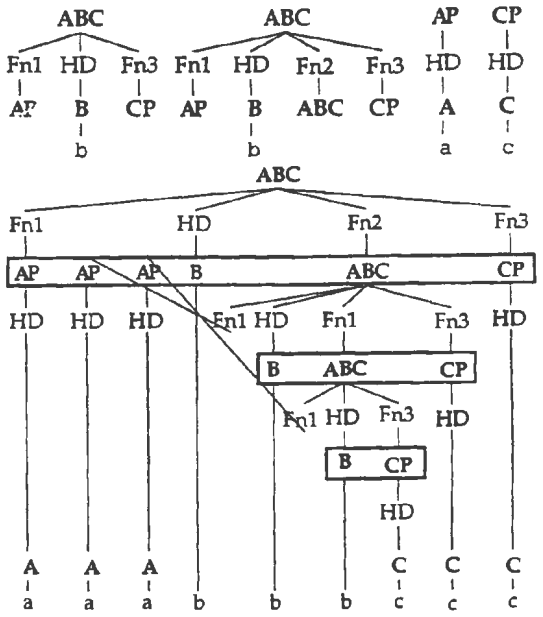


Fig. 4. Derivation of string $a^3b^3c^3$. (a) Initial lexical frames. (b) Derived tree. Notice that only the matrix linearization array is instantiated completely; the embedded ones are truncated, causing the A-phrases to be fronted.

The mechanism that controls the distribution of constituents over the slots of a linearization array, is modeled as a Finite-State Automaton (FSA). The FSA associated with a lexical frame traverses its array from left to right. At each slot, it inspects the set of constituents that are waiting for placement in the array, and inserts there any constituents meeting the place-

ment conditions on that slot (see the labels on the edges of Fig. 2).

PG is capable of generating the mildly context-sensitive language $a^n b^n c^n$. Fig. 4b illustrates a possible derivation of $a^3b^3c^3$ based on the lexical frames in Fig. 4a. The linearization array associated with ABC frames contains four slots *S1...S4* to be filled, respectively, by constituents of type AP (any number, in arbitrary order), B, ABC, and CP. Furthermore, a subcategorization feature in the ABC foot node of the recursive ABC frame causes deletion of slot S1 of the embedded ABC linearization arrays.

Certain scrambling phenomena in German are interpretable as a consequence of PG's linearization scheme. Consider sentence (1), from [Rambow 94], with two non-finite clauses embedded in one another:

[S[S *das Fahrrad zu reparieren*] *zu versuchen*]

Rambow presents acceptability ratings for 30 scrambled versions of this sentence, viz. for all permutations in which the NPs precede the verbs they belong to. (Only five constituents are permutable: two NPs and three verbs.) See Table 1 for a selection from these data.

Table 1. Acceptability ratings for some scrambled version of sentence (1), based on judgments by several native speakers of German. Data from [Rambow 94].

6	weil das Fahrrad zu reparieren niemand zu versuchen verspricht	ok
20	weil niemand das Fahrrad zu reparieren verspricht zu versuchen	?
23	weil niemand zu versuchen verspricht, das Fahrrad zu reparieren	?
25	weil niemand das Fahrrad zu versuchen verspricht zu reparieren	*?
30	weil das Fahrrad zu versuchen niemand verspricht zu reparieren	*?
10	weil das Fahrrad zu versuchen niemand zu reparieren verspricht	*
24	weil niemand zu versuchen das Fahrrad verspricht zu reparieren	*

(1) weil niemand das Fahrrad zu reparieren
because nobody the bike to repair
zu versuchen verspricht
to try promises
'because nobody promises to try to repair the bike'

(2) weil niemand verspricht das Fahrrad zu reparieren zu versuchen

(3) weil niemand das Fahrrad verspricht zu reparieren zu versuchen

The verbs *versprechen* and *versuchen* can take several types of complement in addition to the one exemplified in (1). The non-finite complement clause may be extraposed, i.e. put behind the finite verb in subordinate clauses (as in (2)). Moreover, it allows the so-called "Third Construction" where only part of the non-finite complement clause, including the infinitival verb, is extraposed.

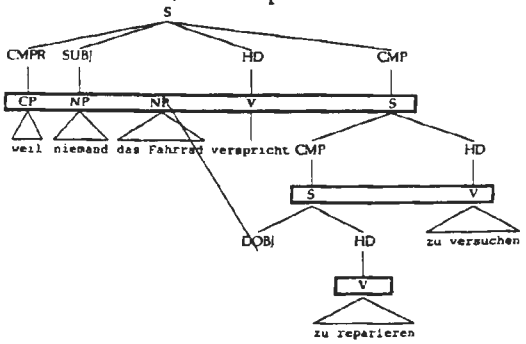


Fig 5. PG analysis of sentence (3).

In the PG treatment of these constructions (illustrated in Fig. 5), the linearization arrays play a crucial role. We assume that, in sentence (2), *reparieren*'s linearization array has been instantiated from slot M4 onward, and in sentence (3) only from slot M8 onward. Moreover, *versuchen*'s array has been truncated as well and only contains slots M8 through B2. This implies that, in (2), the direct object *das Fahrrad* could find a place in *reparieren*'s array, whereas it was moved upward into the finite clause in (3). As stated above, it is a subcategorization feature of a complement-taking verb that controls how the complement's linearization array will be instantiated.

Emphatic or contrastive focus is another factor causing a constituent to move upward. A focused constituent is assigned to early positions in a clause, e.g. M3 or M4. If that position is not available at the clause level it belongs to, it moves into the array of a higher clause.

The position of the two infinitives with respect to one another turns out to be the major source of variation in acceptability. In all fully or marginally acceptable versions ("ok" or "?"):

(A) the non-finite clauses are adjacent, or

(B) they are discontinuous, with the complement-taking infinitive (*zu versuchen*) following its complement (*zu reparieren*).

These properties are illustrated by the PG representations of Rambow's sentences (2) and

(6) in Fig. 6.

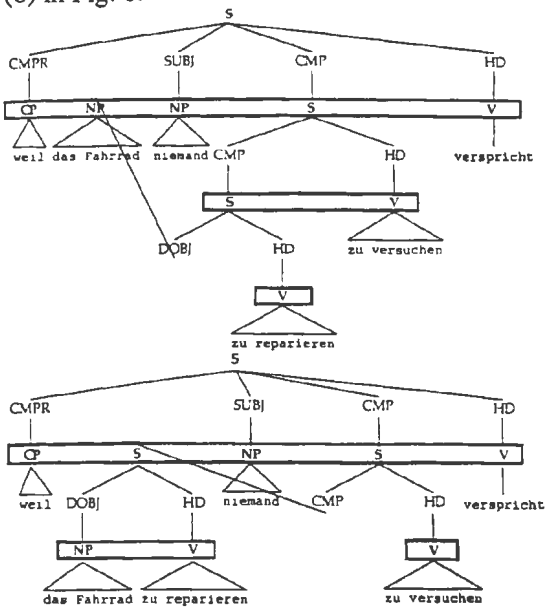


Fig. 6. PG analyses of two acceptable utterances in conformity with linearization rules. Top panel: both non-finite clauses occupy the standard position M8 in their respective arrays. NP das Fahrrad *is focused (slot M3 or M4). Bottom panel: CMP-S* versuchen *is in unmarked position M8; CMP-S* reparieren *is focused.*

On the other hand, in all unacceptable or bad versions ("*" or "*?"):

(A') the non-finite clauses are discontinuous,

(B') with the complement-taker preceding its complement.

Examples are Rambow's sentences (10) and (30), quasi-reconstructed here as Fig. 7.

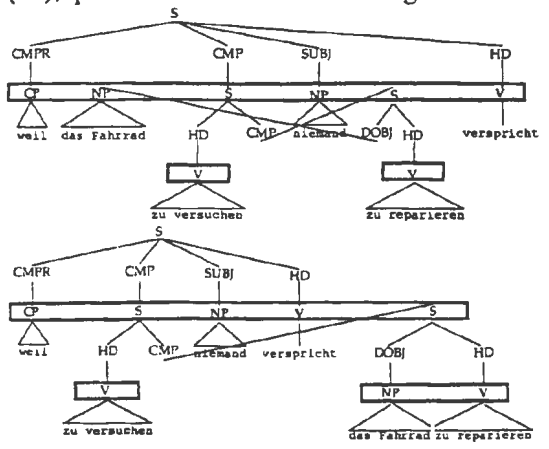


Fig. 7. Quasi-analyses of two unacceptable sentences.

The structures depicted in Fig. 7 violate PG's linearization scheme because of an illegal attempt of *zu reparieren* to move into the finite clause: this CMP-S is not moving into a focus slot and therefore will be assigned a place at its own level, i.e. in slot M8 or B1 of *versuchen*'s array. All bad or unacceptable sentences in Table 1 suffer from this problem, while those rated good or marginal all adhere to PG's linearization scheme. Version (23), whose rating is relatively good although it manifests an illegal extraposition attempt, is the only exception.

We conclude that PG is capable of accounting for a considerable portion of the variance in the acceptability judgments reported by [Rambow 94]. This suggests that the combination of 'substitution + linearization FSA' in PG could serve as an alternative to 'adjunction + substitution' in TAG.

References

[Becker *et al.* 91] Becker, T., Joshi, A.K., Rambow, O. (1991). Long Distance Scrambling and Tree Adjoining Grammars. In: *Papers presented to EACL91*, Berlin.

[Joshi 87] Joshi, A.K. (1987). The relevance of Tree Adjoining Grammar to generation. In: Kempen, G. (Ed.), *Natural language generation.* Dordrecht: Kluwer.

[Joshi *et al.* 91] Joshi, A.K., Vijay-Shanker, K., Weir, D. (1991). The convergence of mildly context-sensitive grammatical formalisms. In Sells, P., Shieber, S.M., Wasow, T. (eds.), *The Mental Representation of Grammatical Relations.* Cambridge MA: MIT Press.

[Kempen 97] Kempen, G. (1997). Grammatical performance in human sentence production and comprehension. Ms, Leiden University.

[Rambow 94] Rambow, O. (1994). *Formal and computational aspects of natural language syntax.* PhD Thesis, University of Pennsylvania.

[Shieber, Schabes 90] Shieber, S.M., Schabes, Y. (1990). Synchronous Tree-Adjoining Grammars. In: Karlgren, H. (Ed.), *COLING-90*, Helsinki.

[Uszkoreit 87] Uszkoreit, H. (1987). *Word order and constituent structure in German.* Stanford CA: CSLI.

An improved Earley parser with LTAG

Yannick de Kercadio

LIMSI-CNRS, BP 133, F-91403 Orsay cedex, FRANCE
TALANA, UFR de Linguistique, Université Paris 7
kercadio@talana.linguist.jussieu.fr or kercadio@limsi.fr

1 Introduction

This paper presents an adaptation of the Earley algorithm (EARLEY, 1968) for parsing with lexicalized tree-adjoining grammars (LTAGs).

This algorithm constructs the derivation tree following a top-down strategy and verifies the valid prefix property. Many earlier algorithm do not have both of this properties (SCHABES, 1994). The Earley-like algorithm described in (SCHABES and JOSHI, 1988) verifies the valid prefix property, but the algorithm presented here is thought to be easier to improve using some properties of LTAGs.

2 Representation of a LTAG with a set of rules

A LTAG is a context-free grammar (CFG) on trees, the elementary operations of which are the adjunction and the substitution. The Earley algorithm can be used for parsing with any CFG insofar as the elementary operation is the concatenation. Hence, the Earley algorithm cannot simply be used for LTAGs, but the meaning of an edge in the derivation tree needs to be specified in terms of words strings and concatenations.

Substitution and terminal nodes can be handled using ordinary context-free rules. Such a rule represents a node in the derivation tree and captures the linear word order of the derived string.

An adjunction can be seen as two correlated substitutions: the derived string of the part of the adjoined tree on the left of the foot node is inserted in some location while the other part of the string is inserted in some other location farther in the string. The string located between the two substitution points is the derived string of the subtree under the adjoined node. The correlation between these two substitutions is that either none or both of them should occur. thus a synchronization must be transmitted up to the second location in order to preserve this constraint.

The locations of these pairs of places follows a stack order: there is an equal number of "first places" and "second places" between two matching places. Therefore, a unique symbol (# hereafter) can be used to represent any "second place", while a βX notation can be used to represent a "first place" for an adjunction of a tree with root X.

The figure 1 shows a few rules representing some elementary trees. A star denotes a foot node in an auxiliary tree. The drawn links implements the correlation information between the two substitution points representing an adjunction. Because of the stack structure of this information, the links need not to be explicitly stored. Also note that these trees are flat (no VP). See (ABEILLÉ, 1991). This is not mandatory and the trees usually used for English can be encoded the same way.

As each node in the derivation tree represents an elementary tree, and as every elementary

I'd like to thank A. Abeillé, M.-H. Candito, F. Issac and P. Paroubek for their valuable help and advices

Rule for the transitive verb *to love* ($\alpha n0Vn1$), without adjunctions:

$$\alpha S \rightarrow \alpha N \quad love \quad \alpha N$$

Rule for the transitive verb *to love* ($\alpha n0Vn1$), with possible adjunctions on S and on V:

$$\alpha S \rightarrow \overbrace{\beta S \quad \alpha N \quad \overbrace{\beta V \quad love \quad \#}^{} \quad \alpha N \quad \#}^{}$$

Rule for the determiner *the* ($\beta DetN$), with a possible adjunction on the root N:

$$\beta N \rightarrow \overbrace{\beta N \quad the \quad * \quad \#}^{}$$

Figure 1: Examples of rules

tree can be represented by a rule which capture the linear word order of the derived string, this is a way to capture the linear word order in the derivation tree. The usual derivation tree (as defined in (VIJAY-SHANKER, 1987)) can be obtained by linking the subtree of every "first place" to the left of the subtree of the matching "second place" and by storing the resulting structure under the "second place".

3 Earley-like parsing driven by the derivation tree

In this section, we show how the stacked relationships between the "first places" and "second places" can be represented in a structure which is suitable for the Earley algorithm.

Following Earley, a partial parsing can be represented by an item, which consists in a rule, a position in the rule (all the symbols located on its left have been recognized), and two lists of pairs of references to items. The first list keeps track of the requesters of the rule, that is to say the items which are waiting for the rule to be recognized in order to be shifted. The second element of each pair is used as a relay storage during the recognition of the second part of an auxiliary tree. The second list implements the previously mentioned stack of "first places". The first element of each pair it contains is the data part of the stack item. It is a reference to an item waiting on a foot symbol. The second element in each pair is used to implement the stack. It is a reference to an item waiting for an adjunction.

A number of primitive operations will be applied on this data structure. They are summed up in the table 2. When a primitive is applied on a given set, the second column indicates how many actions are to be taken. The rule and mark columns indicate which item is to be introduced. If no item with this rule and this position mark is present in the set, it is introduced with the indicated lists for the requesters list and the stack list. Otherwise, the indicated lists are merged with the ones of the existing item in the set. This merging step ensures that the spatial complexity has a polynomial upper bound.

The algorithm consists in working on each set in turn, following the word order. The initial set is initialized using **init**. Then an evolution stage applies a **predict** or **reduce** primitive on every newly introduced item, the type of which is chosen from the symbol in the rule which is right after the mark. For instance, if it is an αX (a substitution is expected), then **predict** α**(item, X)** is used. If there is no such symbol, them a **reduce** primitive is used, depending on the type (α or β) of the left part of the rule. This process is then run on each set in turn, replacing **inits** with a **shift** on every item expecting (i.e. with the mark right on the left of) the word associated with the current set.

The sentence is accepted if there is an item in the last set with a rule deriving the axiom (S), with the mark at the end of the rule, with an empty requesters list. It should be noted that this algorithm does not give an analysis of the sentence. An additional structure is required in each item to keep the analysis information.

primitive	applied for each	rule	mark	req	stack
init()	rule r with root αS	r	0	{}	{}
shift(item)	*once*	item.rule	item.mark + 1	item.req	item.stack
predict α(item, X)	rule r with root αX	r	0	{(item, -)}	{}
predict β(item, X)	rule r with root βX	r	0	{(item, -)}	{}
and	*once*	item.rule	item.mark + 1	item.req	{(-, item)}
predict $*$(item)	(x, y) in item.req	x.rule	x.mark + 1	x.req	{(item, x)}
reduce #(item)	(x, y) in item.stack, where x is not -	x.rule	x.mark + 1	{(item, y)}	x.stack
	(-, y) in item.stack	item.rule	item.mark + 1	item.req	y.stack
reduce α(item)	(x, y) in item.req	x.rule	x.mark + 1	x.req	x.stack
reduce β(item)	(x, y) in item.req	x.rule	x.mark + 1	x.req	y.stack

Figure 2: Primitives of the algorithm

However, every edge in the derivation tree is detected through the fact that a **reduce** primitive is run. This additional structure should cope with the ambiguities and permit a polynomial representation of ambiguities from other level of analysis (features unification, semantic analysis and so on). This is a quite general matter: the number of solutions to the problem of parsing being (potentially) exponential, a simple list of analyses would require an exponential time to be output. The usual assumption that the number of analyses is "small" is not acceptable in the context of parsing oral utterances (because of potential auto-repairing constructs). Therefore, the representation of the outputs should grow polynomially (and not exponentially) with the number of ambiguities.

4 Benefits in using this strategy

The top-down strategy of this algorithm has a trivial, but very useful property: this algorithm do not require the utterance to be cut into sentences in order to parse it. Instead, one can perform an **init** primitive in every set where a rule with the axiom as its left part and an empty requesters list is found. It has the effect of concurrently trying to parse a new sentence from this point. This property is very important when parsing oral utterances: there is no practical other way to find out where sentences begin and end.

Moreover, the combination of both the top-down strategy and the valid prefix property enables valuable performance improvements. Many of the LTAGs properties (SRINIVAS, 1997) can be used to avoid the introduction of unrelevant elementary trees, thus allowing the use of a richer grammar.

The data structures construct a derivation tree. Therefore, a rough semantic analysis can be performed to check whether some newly discovered potential edge in the derivation tree makes sense or not. If not, it can be invalidated as soon as it is discovered.

When features are used, they can be checked following only the derivation tree (the derived tree is not needed). This is due to the fact that the nodes in the derivation tree are more than simple atoms: they are the rules that have been used for parsing. Like with semantic analysis, the features unification can be done on partial analysis, after every reduction. However, it is not clear whether this would result in an improvement or not: the cost of the unification might overcome the benefits of invalidating some partial analysis as soon as possible.

Due to the lexicalization, terminals (words) are put in the trees during lexicon access. When a rule is invocated in a set S, it always contains at least one terminal (lexicalization). All the symbols on the left of the first terminal have to be recognized before the set where this terminal is to be found. This is a way to filter the candidate rules for recognizing these symbols.

Former parsers already used the span of trees to eliminate trees that are too large to parse the sentence (XTAG, for instance), but this algorithm permits considering the span properties locally, at every prediction stage.

Last but not least, the data structures used for this algorithm can be enriched in successive analysis stages. That is to say, when no analysis is found, it is possible to enrich the sets with new rules. This property is useful to construct a fault tolerant parser, accepting unknown words, using weighted syntactic rules (the weights indicating whether a given rule is linguistically perfect or somewhat deviant), and accounting for auto-repairing sequences in an oral utterance.

5 Prospects

Using these properties enables the design of an efficient oral-specific robust parser using a grammar of the written language (ABEILLÉ, 1991). We plan to incorporate a syntactic LTAG-based component in a working real-time speech understanding system (GAUVAIN et al., 1997,) to improve its recognition performances.

References

ABEILLÉ, A. 1991. *Une grammaire lexicalisée d'Arbres Adjoints pour le français.* Ph.D. thesis, Université Paris 7.

ABEILLÉ, A., K. BISHOP, S. COTE, and Y. SCHABES. 1990. A lexicalized tree adjoining grammar for english. Technical report, Department of Computer and Information Science, University of Pennsylvania, Philadelphia.

EARLEY, J. C. 1968. *An efficient context-free parsing algorithm.* Ph.D. thesis, Carnegie-Mellon University, Pittsburgh.

GAUVAIN, J.-L., S. BENNACEF, L. DEVILLERS, L. LAMEL, and S. ROSSET. 1997. Spoken language component of the mask kiosk. In K. Varghese and S. Pfleger, editors, *Human Comfort and Security of Information Systems.* Springer-Verlag, pages 93–103.

SCHABES, Y. 1994. Left-to-right parsing of lexicalized tree-adjoining grammars. *Computational Linguistics*, 10(4):506–524.

SCHABES, Y. and A. K. JOSHI. 1988. An earley-type parsing algorithm of tree adjoining languages. In *26th Meeting of the Association for Computational Linguistics*, Buffalo.

SRINIVAS, B. 1997. *Complexity of lexical descriptions and its relevance to partial parsing.* Ph.D. thesis, Department of Computer and Information Science, University of Pennsylvania, Philadelphia.

VIJAY-SHANKER, K. 1987. *A study of tree adjoining grammars.* Ph.D. thesis, Department of Computer and Information Science, University of Pennsylvania, Philadelphia.

VIJAY-SHANKER, K. and A. K. JOSHI. 1985. Some computational properties of tree adjoining grammars. In *23rd Meeting of the Association for Computational Linguistics*, pages 82–93, Chicago.

Clitic Climbing in Romance: "Restructuring", Causatives, and Object-Control Verbs *

Seth Kulick
Institute for Research in Cognitive Science
University of Pennsylvania
Suite 400A, 3401 Walnut Street, Philadelphia, PA 19104-6228
skulick@linc.cis.upenn.edu

1 Introduction

"Restructuring" in Romance refers to constructions which appear to violate standard locality constraints, thereby presenting a challenge for syntactic theory. One of the most well-studied cases of restructuring[1] is that of clitic climbing. This is illustrated in the Italian example (1a), in which the clitic *lo*, the apparent object of *leggere*, appears on the higher verb *vuole*. As shown in (1b), it can even move past more than one verb.

(1) a. Mario lo vuole leggere
'Mario wants to read it'
b. Mario lo vuole poter leggere
Mario it wants to be able to read
'Mario wants to be able to read it'

Such clitic climbing is possible only with certain verbs, such as *voler* and *poter* in (1), which I will refer to as the "trigger" verbs, following Aissen and Perlmutter (1983). Bleam (1994) argued in detail that clitic climbing causes problems for TAG, and that set-local multi-component TAG is required. In previous work (Kulick 1997), I have proposed that due to the limited nature of the trigger verbs (aspectuals, motion verbs, modals) they can be treated as "adjunct predicates" that adjoin into a TAG tree, as if they were raising verbs, taking advantage of their semantic "weakness". An advantage of this approach is that the apparent unboundedness of clitic climbing, as in (1b), can be handled in TAG by repeated adjoinings of these trigger verbs. There are also several aspects of "restructuring" other than clitic-climbing (e.g., long reflexive passive, long tough-movement, Italian auxiliary change, etc.) which I cannot comment on here. The case of the long reflexive passive is discussed in Kulick (1997)[2].

However, this "adjunct" predicate" approach to clitic climbing in TAG is clearly insufficient for two other major cases of clitic climbing: the Romance causatives, and object-control verbs in Spanish such as *permitir* (Strozer 1977, Moore 1991). In this work I extend the analysis to handle these two cases. The relation of these cases to the "restructuring" trigger verbs has long been a matter of debate, and I argue that it is desirable that TAG enforces a sharp distinction between them. Still, an analysis must be given in TAG for these cases, and I propose a tree-local multi-component TAG analysis for both cases. This raises again the issue of the unboundedness of clitic climbing with these verbs.

2 Causatives

The Romance causative, as illustrated by the French example (2), of course has a number of unusual features which have been the focus of much research[3]. As illustrated in (2), the word order and Case marking of the causee in the lower clause is strikingly different than the usual. Of particular interest here is that when the lower object is cliticized, as in (3),

*I would like to thank Tonia Bleam, Robin Clark, Robert Frank, Heidi Harley, Aravind Joshi, Alexandra Kinyon, Tony Kroch, Miriam Meyerhoff, Paolo Monachesi, Beatrice Santorini, two anonymous reviewers, and the members of the Xtag project for comments on various aspects of this work. I would also like to thank Filippo Beghelli, Claudia Brovetto, Alexandra Kinyon, Marisel, Zoe Lacroix, Paola Merlo, and Carmen Rio-Rey for native speaker judgements. This work is supported by grant NSFSTC89-20230.

[1] I am using "restructuring" as a descriptive term only, and not to refer to the particular analysis proposed in Rizzi (1982).

[2] Cinque has proposed that clitic climbing and other transparency effects can he handled by treating the trigger verbs as being "directly inserted in the extended projection of a lexical verb", according to the abstract for a talk. This depends on the trigger verbs being limited to modal, aspectual, and motion verbs, which for Cinque correspond to functional heads, and so can be so inserted into the extended projection. Clearly, this proposal seems to have much in common with that in Kulick (1997). However, I have not seen Cinque's full analysis, and so I cannot currently comment further on the connection.

[3] I am putting aside here the *faire-par* causative

it appears on *a fait* rather than with the verb that it is semantically associated with, *manger*.

(2) Jean a fait manger ce gateau à Pierre
Jean has made eat the cake to Pierre
'Jean made Pierre eat the cake

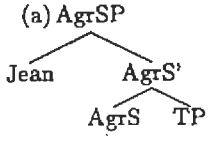
(3) Jean l'a fait manger à Pierre
Jean it has made eat to Pierre
'Jean made Pierre it'

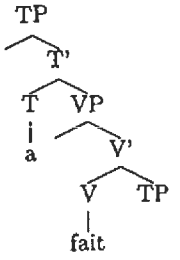
Clearly, the approach taken for the "adjunct predicates" is insufficient here. Adjoining *fait* or *a fait* into a tree which has both *Jean* and *Pierre* is absurd, since the latter tree would be a radical violation of the most basic principles of what constitutes an elementary tree.

There have been two basic approaches taken in TAG to handling the problem of non-local movement in the French causative. (Abeillé 1991, Abeillé 1993) treated *faire* as a co-anchor of an elementary tree headed by the infinitival verb. Then the clitic movement is local to an elementary tree, and there is no problem. Santorini and Heycock (1988) argue, however, that the French causative must be considered bi-clausal, and therefore two separate TAG trees, since the complement object is not able to passivize (unlike the case with the Italian causative). However, they did not discuss how to handle the clitic movement.

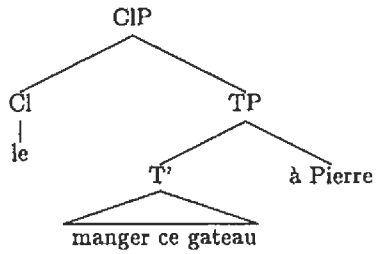
There are arguments for both approaches, but in this work I follow Santorini and Heycock (1988), and adopt a bi-clausal analysis. I extend Santorini and Heycock (1988)'s analysis to handle the clitic movement by using a tree-local multi-component TAG, which allows a tree set for *Jean* and *a fait* to wrap around *le* in *le manger à Pierre*. This depends on the clitic moving to the top of the *manger* tree[4]

One way to work this out is to use the tree set in (4ab) for the matrix clause, and the tree in (5) for the embedded clause. The derivation proceeds by (4b) adjoining at the TP node, while (4a) adjoins at the root of (5) to produce the tree (6) (this requires that the AgrSP node be treated as a TP node for purposes of adjoining).

(4) (a) 

(b) 

[4]In work-in-progress, I propose using this same approach to handle long-distance-scrambling in German (Rambow 1994), thus hopefully unifying the machinery needed for these two cases of non-local movement.

(5) 

There are obviously some issues here concerning the Case marking and word-order which require further discussion. For now, I am assuming that the dative Case on *á Pierre* is assigned by the causative verb, and that if the lower verb was intransitive, it would get accusative Case[5].

However, just as with the restructuring Case, it becomes a crucial question as to how unbounded such clitic movement is. For the causatives, this relates to the issue of how recursive causative forma-

[5]I am also assuming that the causee is a structural subject, as opposed to being generated as an indirect object. As a reviewer notes, Abeillé et al. (1996) argue that the causee is a true indirect object or direct object, depending on the transitivity of the lower verb. They note that when the lower verb takes a dative argument, it is possible for the accusative causee to appear between the lower verb and its dative argument:

(1) Maire fera parler Jean à Paul
Marie will make Jean speak to Paul

Thus, the arguments of the causative and embedded verbs follow the unmarked ordering of clausal arguments in French. This is not expected given the type of analysis as in (6). However, these facts are not new, and were discussed in Santorini and Heycock (1988), in which they suggested, following Burzio (1986) that there are "late reordering rules" to fix up the order. I follow Santorini and Heycock (1988) in this regard, although such rules of course are somewhat undesirable.

Most of the arguments in Abeillé et al. (1996) point out that the causative construction acts differently from a control construction, in terms of how the arguments of the two verbs can be reordered. While this is correct, I don't see how it's an argument against a structure as in (6) (again, assuming the existence of reordering rules), which is clearly not a control structure.

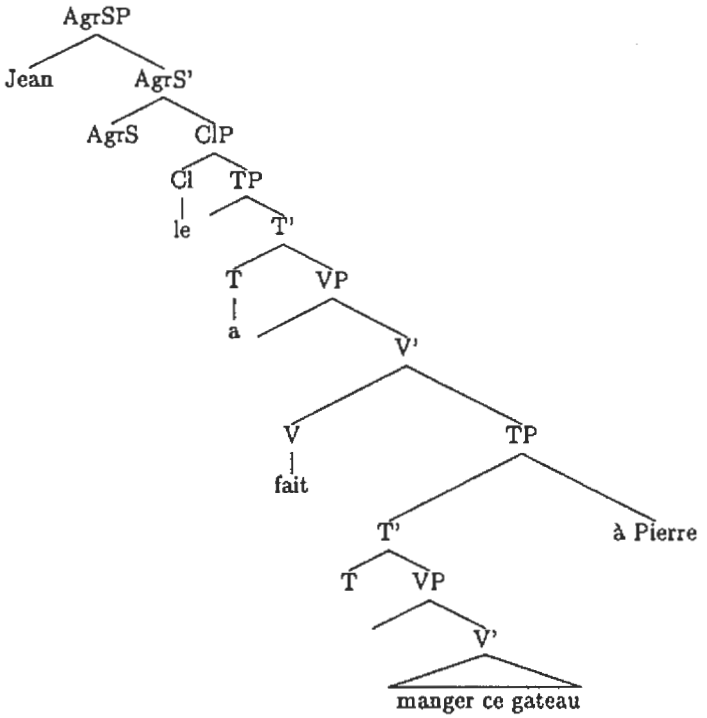
They also note that since quantitative *en* can be extracted out of an accusative causee, as in (2), this shows that it must be a structural object. However, since such extraction can also take place out what might be analyzed as a small-clause subject (3), it's not clear to me how strong this argument is.

(2) Il en fera courir trois
He will make three of them run

(3) a. Paul entend 3 femmes chanter
b. Paul en entend 3 chanter

Their strongest argument, I think, concerns the ability of "tough movement" to take place across the causative in French. Clearly, for the approach taken here, this deserves further study.

(6)



tion is. There has been very little discussion of this issue in the literature[6], and the data is somewhat murky, but it seems to be the case that sentences with lower unaccusative verbs are acceptable[7]. For example, (7b) has an additional causative verb on top of the causative construction in (7a). In such a case, it is possible to place a clitic for the causee (*lui* for *à son fils*) and for the object (*le*, for *le pont*), on *a fait*, as shown in (7c).

(7) a. Son fils a fait sauter le pont
His son made blow up the bridge
His son made the bridge blow up

b. Elle a fait faire sauter le pont
She made make blow up the bridge
à son fils
to her son
She had her son make the bridge blow up

c. Elle le lui a fait faire sauter
'She made him make it blow up' or
'She had it blown up by him'

d. Elle me l'a fait faire sauter
She me it made make blow up
'She made me make it blow up' or
'She had it blown up by me'

I discuss the consequences for TAG of the possibility of sentences such as (7c), which appear to require the use of set-local MCTAG. However, there is a "trick" that can be done to allow a tree-local derivation for (7c), although space prohibits here any explanation of what I'm talking about. This approach, however, will not work for the case in which the clitics are in a different order, as in (7d), and I discuss the consequences of that.

3 Spanish object-control verbs

An example of clitic climbing with *permitir* is shown in (8ab), in which (b) shows that *la* can optionally move from *arreglar* to *permitió*[8].

(8) a. Juan le permitió arreglarla a
Juan him permitted to repair it
Pedro
Pedro
Juan permitted Pedro to repair it

b. Juan se *la* permitió arreglar a
Juan him it permitted to repair
Pedro
Pedro
Juan permitted Pedro to repair it

Similar issues arise here as with the causatives. Again, the "adjunct predicate" analysis is inadequate, and a tree-local TAG analysis seems appropriate. Following the approach of Bleam (1994) and

[6] Kayne (1975) is an exception.

[7] For sentences with lower intransitive verbs, I have gotten mixed reactions from native speakers.

[8] *le* in (8a) is a clitic double of *a Pedro*, and a morphological rule changes *le la* to *se la* in (8b).

others (e.g., Moore 1991), I adopt a "reduced complement" analysis.

The question of unboundedness is quite interesting, since it seems to be the case that clitic climbing over these verbs is much more constrained than with the "adjunct predicate" trigger verbs, and speakers are very reluctant to accept even a highly simplified sentence such as (9b). This is true even for speakers who can accept clitic climbing over two or even three "adjunct predicates" without any hesitation.

(9) a. Juan ordenó permitir comprarla
Juan ordered to permit to buy it
Juan ordered someone to permit someone to buy it
b. * Juan la ordenó permitir comprar

Since tree-local TAG can clearly handle such cases as (8b), it might be appropriate to say that the increased difficulty of clitic climbing in cases such as (9) is a reflection of the need to move to set-local TAG. However, the force of this argument is weakened if the same approach for clitic climbing out of two embedded clauses with the causatives (as in (7c)) can be applied in this case.

More interesting is the question of why the object-control verbs that allow clitic climbing in Spanish are limited to those that take dative, not accusative, controllers. I offer the tentative suggestion that the complements of accusative controllers such as *forzar* are not "defective" enough, since they take a preposition which takes a sentential complement, as in (10): (example from Bordelois (1988))

(10) * Se lo forzó a hablar
her-DAT him-ACC he forced to speak
'He forced him to speak to her

For this argument to go though, of course, the *a* in (10) must be fundamentally from the *a* that follows some of the "adjunct predicate" trigger verbs which do allow clitic climbing. It also depends on a correlation between the accusative controller taking prepositional complements, while the dative controllers do not. I am currently unsure whether this correlation holds fully, and of course it leaves open the question of why such a correlation might exist.

However, it is very interesting to note that the same facts concerning which object-control verbs allow long movement appear to hold for long-distance scrambling in German. Bayer and Kornfilt (1989) suggest that this is because all German verbs with accusative controllers can take "prepositional adverbs", while those with dative controllers do not. Thus, if the Romance data cooperates, it appears promising that there can be a unified explanation for the similar behavior of the object-control verbs in Spanish and German.

References

Abeillé, A. (1991) *Une grammaire lexicalisée d'arbres adjoints pour le français*, Doctoral dissertation, University Paris 7.

Abeillé, A. (1993) *Les Nouvelles Syntaxes: grammaires d'unification et analyse du français*, Armand Colin, Paris.

Abeillé, A., D. Godard, P. Miller, and I. A. Sag (1996) "French Bounded Dependencies."

Aissen, and Perlmutter (1983) "Clause Reduction in Spanish," in *Studies in Relational Grammar*, University of Chicago Press.

Bayer, J., and J. Kornfilt (1989) "Restructuring Effects in German," in *Parametric Variation in Germanic and Romance: Proceedings from a DYNA Workshop.* Center for Cognitive Science, University of Edinburgh.

Bleam, T. (1994) "Clitic Climbing and The Power of Tree Adjoining Grammar," in *Symposium on Tree Adjoining Grammar.* To Appear.

Bordelois, I. (1988) "Causatives: From Lexicon to Syntax," *Natural Language and Linguistic Theory* 6, 57–93.

Burzio (1986) *Italian Syntax*, D. Reidel.

Kayne, R. S. (1975) *French Syntax*, The MIT Press.

Kulick, S. (1997) "Generalized Transformations and Restructuring in Romance," in *Proceedings of the Eastern States Conference on Linguistics.*

Moore, J. (1991) *Reduced Constructions in Spanish*, Doctoral dissertation, University of Santa Cruz.

Rambow, O. (1994) *Formal and computational aspects of natural language syntax*, Doctoral dissertation, University of Pennsylvania.

Rizzi (1982) *Issues in Italian Syntax*, Foris Publications.

Santorini, B., and C. Heycock (1988) "Remarks on Causatives and Passive," Technical Report MS-CIS-88-33, University of Pennsylvania.

Strozer, J. R. (1977) *Clitics in Spanish*, Doctoral dissertation, UCLA.

Wh-dependencies in Romanian and TAG

Manuela Leahu
TALANA , Université Paris 2,place Jussieu , case 7003,75251 Paris cedex 05
manuel@pirae.linguist.jussieu.fr

Introduction

The aim of this paper is to analyse the wh - movement for Romanian in TAG formalism. Romanian shares free extractibility from tensed clauses with its Romance sister languages and it has borrowed multiple wh-fronting from the slavic languages. These features of Romanian are quoted by Kroch (1989) from Comorovsky (1986) , where he justifies the analysis of extractions in TAG. This formalism allows a correlation between the absence of wh-islands and the possibility of multiple wh - movement. But the facts of the Romanian language are more complex .We consider here several phenomena like simple questions, unbounded dependencies, wh-islands, multiple wh – movement . Because of order between the free wh-words for the multiple wh-movement , a complete analysis is not possible with TAG .

TAG derivation trees do not provide a good representation of the dependencies between the words of the sentence, i.e., of the predicate - argument and modification structure.

Also, the derivation structures of MCTAG (Joshi,1987) cannot be given a linguistically meaningful interpretation (Section 3). We show here that an analysis is possible with DTG formalism (Vijay-Shanker, D. Weir, O. Rambow,1995) that resolve these problems with the use of a single operation -that we call subsertion -for handling all complementation.[1]

Simple Questions

(1) Pe cine$_i$ vede Ion e_i ?
Who$_i$ sees Ion e_i ?

This sentence in the TAG formalism is represented as a transitive tree with object extraction and the initial place of the extraction is marked by a trace . A characteristic feature for questions is the inversion of the subject.

Unbounded dependencies

The following sentences illustrate some examples of unbounded dependencies :

(2) Ce $_i$ regreti ca a citit Maria e_i ?
What $_i$do you regret that Mary has read e_i ?

The wh -pronoun in the initial tree is in the same verb with which it is construed and its interpretation as the object of the verb " to read " is thus guaranteed . Following standard conventions we represent the relationship between the fronted constituent and the position in which phrases with its grammatical role normally appear by coindexing the fronted wh with an empty category. The relationship between an indexed empty category and the categorially identical, c- commanding node with which it is in coindexed , we call " linking ". The adjunction of the auxiliary tree in the initial tree produces the final tree in which the wh-word is now initial in the matrix sentence.

Strikingly, there is no bound on the depth of the embedding :

(3)Pe cine $_i$ crezi ca Paul a zis ca Ion a placut e_i?
Who $_i$do you think that Paul said that Ion liked e_i?

In (3), the wh-word is an argument of the most deeply embedded verb " like " , thus causing the non-projectivity . A TAG can capture the long-distance dependency naturally, since the recursive adjunction operation allows an unbounded number of clauses to intervenue between directly dependent lexemes. We first substitute all nominal arguments into their respective verbal trees , and then adjoin the intermediate *say* –clause into the most deeply embedded *like*-clause at the S node immediately dominated by the root. This has the effect of separating the wh-word from its verb. even though they originated in the same structure . We than subsequently adjoin the matrix *think* - clause into the intermediate *say*-clause .

[1] We are grateful to Anne Abeille Sylvain Kahane and Owen Rambow.

Wh- islands

Islands phenomena can be found in Romanian for relative clauses and adjuncts.

(4)* Pe cine $_j$ cunosti femeia $_i$ care $_i$ a intalnit e $_j$?

Who $_j$ do you know the woman $_i$ which $_i$e $_i$ met e $_j$?

(5) * Pe cine $_j$ ai plecat inainte sa examineze e_j Ionescu ?

Who $_j$ did you leave before that examine e $_j$ Ionescu ?

These violate for locality reasons : there is no way to localize the wh-element and its co-indexed base position in the same tree set (MCTAGs) which can then be adjoined into a single elementary tree.

But in the case of interrogative clauses are not islands for extraction:

(6) Pe cine $_i$ crezi ca Paul detesta e_i ?

Who $_i$ do you think that Paul detests e $_i$?

(7) Pentru care clauza $_i$ vrei sa afli cine $_j$e $_j$ nu a decis inca ce $_k$ va vota e_k e_i? (Comorovsky 1986)

For which paragraph $_i$ do you want to learn who$_j$e$_j$ has not decided yet what$_k$ he will vote e_k e_i ?

For this Kroch suggests an interesting account that reduces the constraint on movement out of an island to a local well-formedness condition on elementary trees.

Multiple Wh - Movement

In Romanian language multiple wh movement are rare but grammatical.Wh pronouns are strictly ordered:

(8a) Cine cui$_j$ promite o masina e_j?

Who to whom $_j$ promises a car e $_j$?

(8b) *Cui $_j$ cine promite o masina e_j ?

To whom $_j$ who promises a car e $_j$?

(9a) Cui $_i$ ce$_j$ zice e $_i$ ca vezi e_j?

To whom $_i$what $_j$ he says e_i you see e $_j$?

(9b) *Ce $_i$ cui $_j$ zice e $_j$ca vezi e $_i$?

(10a) Cui $_i$ pe cine$_j$ Paul zice e $_i$ca vezi e_j?

To whom $_i$ whom$_j$ Paul says e_i you see e_j ?

(10b) * Pe cine$_i$ cui$_j$ Paul zice e_j ca vezi e $_j$?

(11a) Cine $_i$ pe cine $_j$ a zis e $_i$ ca a vazut e_j?

Who$_i$ whom$_j$ said e_i he has seen e $_j$?

(11b)*Pe cine$_i$ cine $_j$ a zis e_j ca a vazut e $_i$?

(12a) Cine$_i$ ce$_j$ ziceai ca e $_i$ isi inchipuie ca ai descoperit e $_j$?

Who $_i$what $_j$you were saying that e_i to himself imagines that you have discovered e_j?

(12b) *Ce$_i$ cine$_j$ ziceai ca e $_j$si inchipuie ca ai descoperit e $_i$?

(13) Cine $_i$ cui $_j$ ce$_k$ ziceai ca e_i i_j- a promis e $_j$ e $_{jk}$?

Who$_i$ to whom $_j$what$_k$ you were saying that e_i to him $_j$has promised $e_j e_k$?

Examples 8(b)-12(b) are not correct because they don't respect the ordering on the wh-pronouns , which is the following:

cine(who)<cui (to whom)<pe cine(whom)<ce(what)

Nom<Dat<Acc

The ordering constraint is kept even if the Wh extractions are not dependent on the same verb(9-12(a)).

When a non pronominal NP is also extracted . several word orders are possible :

(14a) Ce masina $_i$ cui $_j$ Paul promite e_i sa repare e_i?

What car$_i$ to whom $_j$ Paul promises e $_i$to repair e $_i$?

(14b)Cui $_i$ce masina $_j$ Paul promite e_i sa repare e_j?

To derive the sentence in (14a) ,for example. we adjoin the tree " to whom Paul promises " into the elementary tree " which car to repair " and this example can be analysed in TAG formalism.(Figure1)

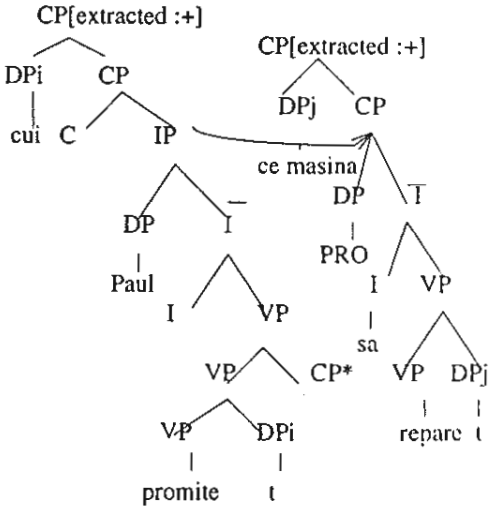


Figure 1 : trees for (14a)

In examples 9-12(a) and 14(b) because of the ordering constraint, the TAG formalism is not able to analyse these cases , given the predicate - argument coocurrence constraint on elemen-

tary trees.
The problem in describing this phenomena with TAG arises from the fact observed by Vijay-Shanker 1992 , that adjoining is an overly restricted way of combining structures.
In Multi-Component TAG (MCTAG) (Joshi, 1987), trees are grouped into sets which must be adjoined together (multicomponent adjunction). The elementary tree is split up into parts, which are grouped together into sets. All trees from one set must be adjoined at the same time, at different nodes into the single tree representing the embedded clause. However, MCTAG lack expressive power since, while syntactic relations are invariably subject to c- command or dominance constraints , there is no way to state that two trees from a set must be in a dominance relation in the derived tree.(Figure 2)

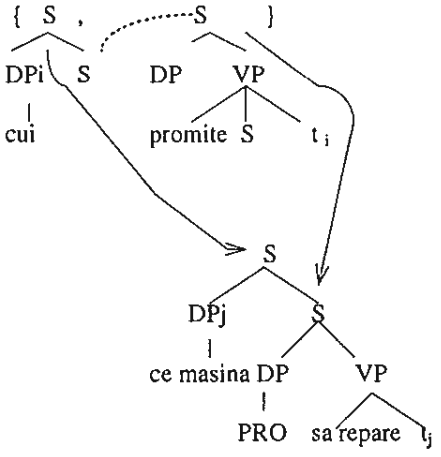


Figure :MC-trees for (14b)

DTG is designed to overcome this limitation . Subsertion can be viewed as a generalization of adjunction in which components of the clausal complement (the subserted structure) can be interspersed within the structure that is the site of the subsertion . DTG provide a mechanism involving the use of domination links(d-edges) that ensure that parts of the subserted structure that are not substituted dominate those parts that are . Furthermore,there is a need to constrain the way in which the non - substituted components can be interspersed.
The derivation proceeds as follows: we first subsert the embedded clause tree into the matrix clause tree. After we subsert the wh-pronoun of the first clause and the wh-pronoun of the second clause (The extraction of the first clause precedes the extraction of the second clause)(Figure 3)

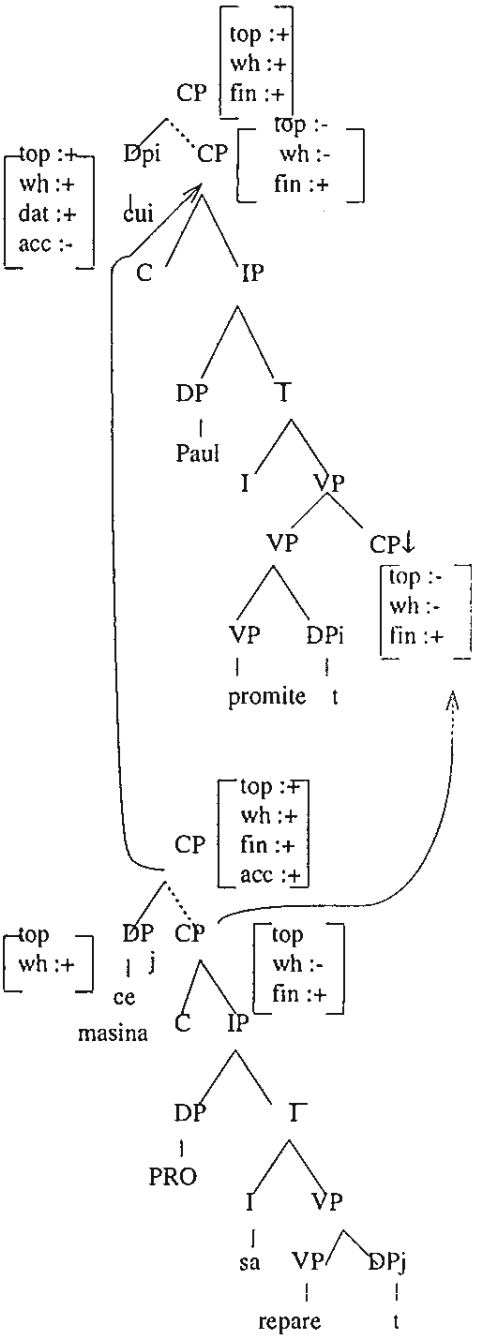


Figure 3 :quasi-trees for (14b)

In DTG formalism,the ordering constraint on the extractions is marked by the feature " topic ".The final tree is the desired , semantically motivated , dependency structure :

the embedded clause depends on the matrix clause), with respect to the ordering constraint on the wh-pronouns.

Conclusion

DTG are designed to share some of the advantages like other formalisms in the TAG family, while overcoming some of their limitations . The most distinctive feature of DTG is that there is complete uniformity in the way that the subsertion relate lexical items.
Furthermore , DTG can provide a uniform analysis for wh-movement in Romanian, despite the fact that the wh-elements in Romanian can appear in sentence –initial position and in sentence –second position.

References :

A. Abeillé ,1991 : Une grammaire lexicalisée d'arbres adjoints pour le français. Ph.D.Thesis.Univ. Paris 7.

A. Abeillé, forthcoming : Extraction out of NP and clitic-noun dependencies in French . TAG+3, Paris.

I. Comorovsky ,1986 : Multiple WH Movement in Romanian , Linguistic Inquiry.

C. Dobrovie ,1994 :The syntax of Romanian. Mouton de Gruyter .

R. Frank ,1992 : Syntactic Locality and TAG : Grammatical , Acquisition and Processing Perspectives.Ph.D.thesis.Univ.of Pennsylvania

A .Joshi, O. Rambow,1992 : A formal Look at Dependency Grammars and Phrase –Structure Grammars ,with Special Consideration of Word-Order Phenomena. In Leo Wanner(ed.) *Recent trends in Meaning-Text Theory*

A. Kroch 1989 :Asymmetries in Long Distance Extraction in a TAG. *In Alternative conceptions of phrase structure.* M.Baltin and A. Kroch(eds), Univ.of Chicago Press, Chicago

A. Kroch and A. Joshi 1986 :Analysing extraposition in a TAG. In G.Huckand A. Ojeda (eds) , *Syntax and Semantics : Discontinous Constituents*

O. Rambow, K.Vijay-Shanker, D. Weir, 1995 :D-Tree Grammars, ACL 1995

XTAG for English 1995, Institute for Research in Cognitive Science, Univ.of Pennsylvania

Which rules for the robust parsing of spoken utterances with Lexicalized Tree Adjoining Grammars ?

Patrice Lopez
LORIA & University of Nancy 1
lopez@loria.fr

David Roussel
Thomson CSF/LCR
roussel@thomson-lcr.fr

Abstract

In the context of spoken dialogue systems, we investigated a bottom-up robust parsing for LTAG (Lexicalized Tree Adjoining Grammars) that interleaves a syntactic and a semantic structure. When the regular syntactic composition rules fail, the syntactic islands and the corresponding partial semantic structures are combined thanks to additional local rules. We supply some descriptive limits of the grammar with these rules which depend on the immediate syntactic context of the islands. In this paper, we focus on their application to few spoken phenomena.

Introduction

Robust parsing is needed to cope with spontaneous uses of language. In particular, it is needed to deal with out-of-grammar utterances occurring in spoken man-machine interfaces. Because of the restricted application domain of such interfaces, it is expected that a robust architecture can interpret an unexpected utterance. This is illustrated with examples in French like :

(1) *Je voudrais un euh un billet pour Paris*
I would like a hum a ticket for Paris.

(2) *Départ à vers 20h.*
Depart at well at about 8 p.m.

(3) *Départ à huit enfin vingt heures.*
Depart at 8 I mean 8 p.m.

(4) *Je voudrais le premier qui part.*
I would like the first (one) which leaves.

(5) *Je voudrais un billet maintenant pour Paris.*
I would like a ticket now for Paris.

Those utterances represent a typical variety of spoken phenomenon namely a repetition (with hesitation) in (1), a self repair in (2), a correction in (3), a noun ellipsis in (4) and the insertion of an adverbe within a noun phrase (5). Parsing failures are respectively due to the impossible mapping of the parasite determiner into the derived tree (1), to the presence of a self repair (2) and (3), to a non canonical constituent (4) and finally to the prepositional attachment across the adverb barrier (5).

In the LTAG framework, we propose to represent the syntactic (partial) trees as connected routes (section 1.2). Adjunction, substitution but also additional local operation are applied to connected routes to make up the descriptive limits of the TAG formalism. Iu section 2, we expose a small set of rules which handle those routes -instead of the trees- and force operations between the trees. Assuming that local disruptions can be resolved by semautic mechanisms, some robust analyses receive a semantic counterpart in a synchronous TAG framework (section 3). Overgeneration remains a major challenge that we discuss in section 4. We will begin briefly explain the Connection driven parsing principles.

1 Connection driven parsing for lexicalized TAG

1.1 Connected routes

We define a *connected route* as a list of internal and root nodes crossed successively according to a left to right tree transversal (Schabes, 1994) until reaching a substitution or a foot node (included barriers) or an anchor (excluded barrier). Each elementary or derived tree can be represented as a list of connected routes. As the list of connected routes is ordered from left to right, we define the function *next* which gives from a given connected route the next connected route.

In (Lopez, 1998b) we explain how to lead a bottom-up bidirectional parsing focused on connected routes instead of focused on nodes as for other algorithms for TAG. Two data structures are used : the table of connected routes which gathers all the connected routes and a chart of parsing states which stores the sequences of well recognized anchors and their left and right connected routes.

1.2 Island representation with connected route

When no connected parse can span the whole sentence, the result of the parsing consists in representations of islands and its both right and left connected routes. An interesting point of this representation

(a) **Rule for hesitations :**

$$\frac{(i,j,\Gamma_G,\Gamma_D,idf) \qquad (j,k,\Gamma'_G,\Gamma'_D,idf') \qquad (k,l,\Gamma''_G,\Gamma''_D,idf'')}{(i,k,\Gamma_G,\Gamma_D,idf) \qquad (k,l,\Gamma''_G,\Gamma''_D,idf'')} \quad (\Gamma'_G = \Gamma_D = (root, H))$$

(b) **Rule for head ellipsis on the left :**

$$\frac{(i,j,\Gamma_G,\Gamma_D,idf) \qquad (j,k,\Gamma'_G,\Gamma'_D,idf')}{(i,k,\Gamma_G,\Gamma'_D,idf'')} \quad \begin{array}{l}(\exists(foot,X) \in \Gamma_D \wedge \\ (\exists(subs,X) \in \Gamma'_G \quad \vee \quad \exists(foot,X) \in \Gamma'_G))\end{array}$$

(c) **Rule for argument ellipsis on the right :**

$$\frac{(i,j,\Gamma_G,\Gamma_D,idf)}{(i,j,\Gamma_G,\Gamma'_D,idf')} \quad (\exists(subs,X) \in \Gamma_D \quad \wedge \quad \Gamma'_D = next(\Gamma_D))$$

(d) **Rule 1 for self repair :**

$$\frac{(i,j,\Gamma_G,\Gamma_D,idf_p) \qquad (j,k,\Gamma'_G,\Gamma'_D,idf_q)}{(i,k,\Gamma_G,\Gamma'_D,idf_r)} \quad \begin{array}{l}(\exists(v,w,\Gamma''_G,\Gamma''_D,idf) \in \Delta, idf \Rightarrow^* idf_p \quad \wedge \\ (\exists(root|internal,X) \in \Gamma''_D \quad \wedge \quad \exists(foot,X) \in \Gamma'_G) \vee \\ (\exists(subs,X) \in \Gamma''_D \quad \wedge \quad \exists(root,X) \in \Gamma'_G))\end{array}$$

(e) **Rule 2 for self repair :**

$$\frac{(i,j,\Gamma_G,\Gamma_D,idf_p) \qquad (j,k,\Gamma'_G,\Gamma'_D,idf_q) \qquad (k,l,\Gamma''_G,\Gamma''_D,idf_r)}{(i,l,\Gamma_G,\Gamma''_D,idf_s)} \quad \begin{array}{l}((\exists(foot,Y) \in \Gamma'_D \quad \wedge \\ (\exists(root|internal,X) \in \Gamma_D \quad \wedge \\ \exists(foot,X) \in \Gamma''_G) \quad \vee \\ (\exists(subs,X) \in \Gamma_D \quad \wedge \\ \exists(root,X) \in \Gamma''_G))\end{array}$$

Figure 1: Example of repairing rules for connection driven parsing

is that these connected routes correspond to the left and right context of the well recognized islands. A parsing state e is defined as the following 5-tuple :

state : (left index, right index, left connected route, right connected route, idf)

The two indices are the bounds of the input string covered by the island (anchors or the consecutive anchors) corresponding to the parsing state. During the initialization, we build a state for each anchor present in the input string. As each elementary and derived tree is identified, the anchor or the connected anchors belong to the tree *idf*. Those representation allows efficient partial parsing. This is the starting point of our robust strategy.

2 Robust Parsing with rules

2.1 Connected routes as flexible categories

A classical bidirectional TAG parsing (Lavelli and Satta, 1991) (van Noord, 1994) can not directly combine incomplete islands but it is possible to adapt the parser behaviour to the remaining syntactic material. Adaptations can be easily simulated by considering a connected route as a flexible category. The midly context sensitive power of LTAGs and CCGs has already suggested that elementary trees can be considered as flexible structured categories (Doran and Srinivas, 1994). According to the linguistic context, local rules can proceed to local adaptation of the routes. Then, the parser can try again to expand islands in both directions.

2.2 Inference rules system

The new derivation processes can be viewed as inference rules (Shieber et al., 1995) which use the parsing states described in section 1. The inference rules (Schabes, 1994) have the following meaning, if $(item_i)_i$ are present in the chart Δ and if the conditions are verified then add $(item_j)_j$ in Δ :

$$\frac{(item_i)_i}{(item_j)_j} \quad (conditions)$$

We note $\Rightarrow^*$ the reflexive transitive closure of the derivation relation between two elementary or derived trees : if $idf \Rightarrow^* idf'$ then the tree identified with idf' can be obtained from idf after applying to it a set of derivations.

The full system (including adjunction and substitution) increases the worst case complexity to $O(n^8)$ and deals with the following phenomena among others.

2.3 Ellipsis

The TAG formalism presents difficulties to describe these very common spoken productions. For instance, the parsing of utterance (4) does not succeed to find any complete derivation if *premier* does not exist in the lexicon as a noun or without the use of a sophisticated non lexicalized structure.

Two rules and their two symetrical configurations try to detect and recover respectively an empty head (b) and an empty argument (c). For instance, rule (b) attempts to make available an adjunction on a node marked for substitution if adjacent and categorial constraints are respected. When the rule (b) applies during the parsing of the example (4), the N1 node of the structure *N0-vouloir-N1* becomes candidate for an adjunction of the nominal auxiliary trees associated with sequences *le premier* and *qui part*.

2.4 Self repairs

The (Cori et al., 1997) definition of self repairs stipulates that the right side of the interrupted structure (the partial derived tree on the left of the interruption point) and the reparandum (the adjacent syntactic island) must match. Instead of modifing the parsing algorithm as (Cori et al., 1997) do, we consider a connected route matching condition. Rule (d) deals with self repair where the repaired structure has been connected on the target node. Rule (e) applies when the repaired structure has not been connected. In example (2), rule (d) detects the structural matching between the two prepositions à and *vers*. Then the rule reintroduces the target node on which the prepositional phrase *vers Paris* must be adjoined. The corresponding semantic tree of the à preposition is deleted.

Rule (e) remains relevant even if islands are separated by an hesitation (1) or a modification marker (3). Indeed the rule for hesitation (a) absords adjacent elementary trees whose head is a *H* node. Such a tree may correspond to different kind of hesitation forms. Rule (a) deletes an hesitation which can play the role of a barrier and a trace is kept in the chart.

3 Robust parsing with a Synchronous Semantic Tree Grammar

In combinatorial Grammars and lexicalized synchronous Semantic Tree Grammars (Shieber and Schabes, 1990) (Kallmeyer, 1997), predicate argument relations are directly encoded in the lexicon. This provide a syntax/semantic correspondence and additional well-formed criterion to validate an analysis (Abeillé, 1992). Robust parsing can take advantage of this property to only combine the syntactic islands in respect to the combination that the corresponding semantic fragments accept. In the case of robust parsing of an elliptic construction, the mechanism which allows such syntactic and semantic control consists in lambda abstractions.

For instance, the parsing of sentence (4) gives rise in the semantic tree shown Fig 2. Rules (b) and (c) combine islands without considering the empty argument. To control that the missing argument is present at the leftmost side of the partial derivation

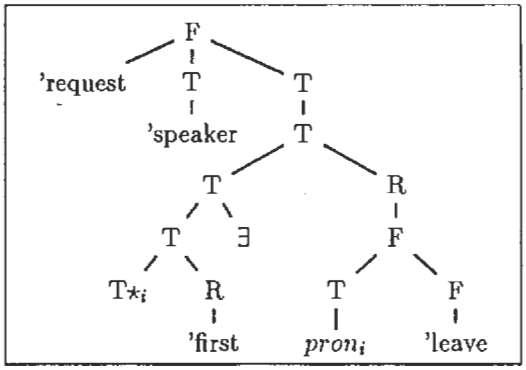


Figure 2: Logical form associated to the robust analyse of sentence (5) by synchronous adjunctions and substitutions

(3) or in the discourse structure (4), the semantic tree (see Fig 2) is translated in a denotational semantic by the introduction of lambda operators. We obtain the following formula for example (3) :

$$\lambda x : term\ (request(speaker(x), \lambda y : term\ \exists y(first(y) \quad \wedge \quad leave(y))))$$

To check if a well- formed formula is obtained, one can test the application of lambda abstraction on the missing predicates and curryfication on independent variables. The resulting semantic tree is correct for the previous example but if we consider a sequence like (6) :

(6) *Je voudrais le.*
I would like the.

the following formula obtained is not correct because the predicate *w* needs to be instanciated :

$$\lambda x : term\ (request(speaker(x), \lambda w : form\ \lambda y : term\ \exists y(w(y)))$$

4 Discussion

4.1 Rules vs specific lexical descriptions

Another way to deal with a sentence like (4) is to adopt a specific elementary tree for the adjective *premier* as explained in (Lopez, 1998a). In that case, the ellipsis resolution is not triggered directly by the parsing failure and a sentence like (7) is rejected.

(7) *Je voudrais le qui part.*
I would like the which leaves.

The same approach could be applied to the description of word order variation. In a Tree Grammar, word order must be determined by dependency relations. While substitution often corresponds to an ordered relation between argument in a syntactic structure, this is not the case for adjoined constituents, especially for adverbs. For instance, the

parsing of utterance (5) needs to consider the adverb *maintenant* as an unusual nominal modifier. The compositionality principle restricts the combination of this syntactic unit to trigger a synchronous combination on the same semantic node that the sentencial adverb does. It is expressed in synchronous TAG by a semantic tree which is synchronously combined at a different node than the syntactic tree.

In this paper, we argue for a rule based approach because we suppose that ambiguous analyses are taken into account at a upper level in a given application domain. By this way, we have to consider more analyses but we avoid inherent restrictions of the "augmented representation".

Indeed, the latter is limited because the semantic derivation can not always be built synchronously with the syntactic derivation. That is the case with the following sentence (8) :

(8) *Un train maintenant pour Paris doit-il partir?*
Does a train now for Paris have to leave?

Moreover, a sentence like (9) triggers redundant analysis because the both elementary trees for the adverb *maintenant* (sentencial and nominal modifier) are valid concurents.

(9) *Je voudrais un train pour Paris maintenant.*
I would like a train for Paris now.

4.2 Constraints vs preferential mechanisms

A previous experiment (Roussel and Halber, 1997) has shown that a robust parsing strategy based on a lexicalized grammar and a set of additional rules can improve the performances of a spoken dialogue system. However, in this experiment, a lot of spurious concurent hypothesis were still hard to eliminate whereas the lexicalized tree grammar was enriched with specific semantic constraints. This result adresses the need of a scoring method to cross-check more knowledge sources. In this framework, the use of semantic control could be use independently among other criteria (hesitation cues, conditions on speech acts, dialogue history, focus, ...) (Roussel and Modave, 1998).

Conclusion

We have shown that connected routes and categorial abstractions gives robustness capacities in a lexicalized tree grammar framework. Many questions are always investigated as the scoring method. A complementary perspective is to extend the rules to more complex discourse representations (Webber and Joshi, 1998).

References

Anne Abeillé. 1992. Synchronous TAGs and French Pronominal Clitics. In *COLING*, Nantes, France.

Marcel Cori, Michel de Fornel, and Jean-Marie Marandin. 1997. Parsing Repairs. In Ruslan Mitkov and Nicolas Nicolov, editors, *Recent advances in natural language processing*. John Benjamins.

Christine Doran and Bangalore Srinivas. 1994. Bootstrapping A Wide-Coverage CCG from FB-LTAG. In *3rd International Workshop on Tree Adjoining Languages (TAG+3)*, Paris, France.

Laura Kallmeyer. 1997. A Syntax-Semantic Interface with Synchronous Tree Description Grammars. In *Formal Grammar Workshop : ESSLLI*, pages 112–124, Aix-en-Provence, France.

Alberto Lavelli and Giorgio Satta. 1991. Bidirectional parsing of lexicalized tree adjoining grammars. In *Fifth Conference of the European Chapter of the Association for Computational Linguistics*, Berlin, Germany.

Patrice Lopez. 1998a. A LTAG grammar for parsing incomplete and oral utterances. In *European Conference on Artificial Intelligence (ECAI)*, Brighton, UK.

Patrice Lopez. 1998b. Analyse guidée par la connexité de TAG lexicalisées. In *Conférence sur le Traitement Automatique du Langage Naturel (TALN)*, Paris, France.

David Roussel and Ariane Halber. 1997. Filtering errors and repairing Linguistic Anomalies for Spoken Dialogue Systems. In *Workshop on Interactive Spoken Dialog Systems : ACL/EACL*, pages 74–81, Madrid.

David Roussel and Francois Modave. 1998. A multicriteria scoring method to parse recognition hypotheses. In *the International Workshop on Speech and Computer (SPECOM)*, St.-Petersburg, Russia.

Yves Schabes. 1994. Left to Right Parsing of Lexicalized Tree Adjoining Grammars. *Computational Intelligence*, 10:506–524.

Stuart Shieber and Yves Schabes. 1990. Synchronous Tree Adjoining Grammars. In *COLING*, volume 3, pages 253–260, Helsinki.

Stuart Shieber, Yves Schabes, and Fernando Pereira. 1995. Principles and Implementation of Deductive Parsing. *Journal of Logic Programming*, 24:3–36.

Gertjan van Noord. 1994. Head Corner Parsing for TAG. *Computational Intelligence*, 10:525–534.

Bonnie Lynn Webber and Aravind K. Joshi. 1998. Anchoring a Lexicalized Tree-Adjoining Grammar for Discourse. In *COLING*, COLING-ACL'98 Workshop on Discourse Relations and Discourse Markers.

"Category Families" for Categorial Grammars

Mary McGee Wood
Department of Computer Science
University of Manchester
Manchester M13 9PL U.K.
mary@cs.man.ac.uk

Abstract

Categorial Grammars (CGs; Wood 1993), grounded in algebra (Lambek 1958) and mathematical logic (Ajdukiewicz 1935), have rightly pushed to the limit the use of logically and algebraically justifiable rules for the combination and alternation of types in describing natural language. However, when TAG trees are mapped to CG categories, tree-families - linguistically well-motivated objects - can only be mapped to arbitrary category sets.

To capture predictable category alternations, such as noun / adjective alternations, or valency alternations for verbs, this paper proposes extending a CG with non-algebra-preserving rules, comparable to the "lexical redundancy rules" of early lexicalist theory. The theoretical argument is backed by an analysis of the degree of compaction which could be achieved by applying such rules to the CG "Large Lexicon" developed at IRCS, UPenn. The reduction which could be achieved both in the number of lexical entries and, more significantly, in the number of categories needed is considerable.

Redundancy rules in theory

CGs have always included both binary rules (such as function application and function composition) and unary (type-shifting) rules, and indeed the interactions between these two rule types have been involved in many debates within CG. The unary rules have been restricted to those which preserved algebraic identity: type-raising NP to S/(S\NP), for example, does not in itself affect the descriptive power of the grammar. However, it is notorious that words can be highly ambiguous as to category, even in a phrase structure grammar with categories of a fairly coarse grain size (such as "verb"), but far more so in a CG. One of the central advances of the lexicalist movement in linguistic description (eg Bresnan (ed) 1982, Gazdar et al 1985) was the recognition and formalization of patterns in the lexicon such as active / passive alternation. Indeed it is ironic that the most extreme of lexicalist grammars has not adopted such lexical rules.

CGs could and, I believe, should have such type alternation rules. For example:

Nominals:
a lexical noun can also serve as a noun phrase, or a noun modifier or noun phrase modifier
$N \Longrightarrow \{NP, N/N, NP/NP\}$

Passives:
a lexical verb will also have a passive form taking one fewer nominal complement
$(S\backslash NP)/NP \Longrightarrow S\backslash NP$
$((S\backslash NP)/NP)/NP \Longrightarrow (S\backslash NP)/NP$
etc.

Gerundives:
a verb (function into S) will also have a gerundive form (function into NP)
$(S\backslash NP)/NP \Longrightarrow (NP\backslash NP)/NP$
$((S\backslash NP)/NP)/NP \Longrightarrow ((NP\backslash NP)/NP)/NP$
etc.

The exact semantics of the rewrite arrow is not at issue here. It is perhaps best taken as a well-formedness constraint or licensing statement along the lines of GPSG meta-rules: "if that is legal, so is this". Nor are we concerned with implementation details such as whether the rules cause expansion at run-time or compile-time. The claim is that these alternations are facts of natural language, and a linguistic theory must have rules to describe them, as indeed most linguistic theories do.

Redundancy rules in practice

The UPenn Combinatory CG "Large Lexicon" (Doran and Srinivas, forthcoming) was created by automatic translation from the large TAG lexicon developed by the TAG Group at the UPenn Institute for Research in Cognitive Science (XTAG Group 1995). TAG trees were mapped to CG categories, and the result modified by hand, principally by Christy Doran, B. Srinivas, and Mark Steedman. Some debugging remains to be done, so these figures are approximate:

36,950 entries
17,960 words
11 POS values
86 CG categories
120 CG category "families"
effectively about 110,000 entries (word / category pairs)

Category families are sets of categories which typically and predictably are assigned together to a word, causing the expansion from 37,000 word entries to 110,000 word / category pairs. In the original TAG lexicon, words are assigned *tree families*, which are linguistically well-motivated objects (Xia et al, in preparation). In the translation from TAG trees to CG categories, the motivation is lost, and we are left with seemingly arbitrary category sets. It is these which can be both motivated and compressed using redundancy rules.

Here are some example entries from the lexicon. (The index numbers serve to distinguish atoms within each complex category, and have no other significance. I give the corresponding TAG trees for the first entry only.)

Verbs: each verb stem has one or two block entries, with some redundancy in passive and gerundive categories:

```
INDEX:  crease/1
ENTRY:  crease
POS:    V
CAT:    S_0\NP_0
        NP_0\(NP_1/N_0)
        NP_0\NP_1

;;; Intransitives
GnxOV    NP_0\NP_1        #INTRANSger
InxOV    S_0\NP_0         #INTRANS
WOnxOV   S_0\NP_0         #INTRANS
nxOV     S_0\NP_0         #INTRANS
NOnxOV   S_0\NP_0         #INTRANS
DnxOV    NP_0\(NP_1/N_0)  #INTRANSger
                           #LagrpassNP_0
```

```
INDEX:  crease/2
ENTRY:  crease
POS:    V
CAT:    (S_0\NP_0)/NP_1
        (S_0\NP_0)/PP_0
        (NP_0\NP_1)/NP_2
        NP_0/NP_1
        N_0/N_1
FS:     #TRANS+
```

Nouns: each noun stem has four block entries, containing 12 categories (singular / plural x head noun / modifier, plus predicatives) which could be reduced to one:

```
INDEX:  Afghan/1
ENTRY:  Afghan
POS:    N
CAT:    (S_0\NP_0)\(NP_1/N_0)
        (S_0\S_1)\(NP_0/N_0)
FS:     #N_refl- #N_wh-

INDEX:  Afghan/2
ENTRY:  Afghan
POS:    N
CAT:    NP_0
        N_0
        N_0/N_1
        NP_0/NP_1
FS:     #N_refl- #N_wh-

INDEX:  Afghans/1
ENTRY:  Afghans
POS:    N
CAT:    (S_0\NP_0)\(NP_1/N_0)
        (S_0\S_1)\(NP_0/N_0)
FS:     #N_refl- #N_wh-

INDEX:  Afghans/2
ENTRY:  Afghans
POS:    N
CAT:    NP_0
        N_0
        N_0/N_1
        NP_0/NP_1
FS:     #N_refl- #N_wh-
```

Adjectives: each adjective has two block entries, containing four categories (singular / plural modifier, plus predicatives) which could be reduced to one:

```
INDEX:  Canadian/1
ENTRY:  Canadian
POS:    A
CAT:    NP_0/NP_1
        N_0/N_1
FS:     #A_WH-

INDEX:  Canadian/2
ENTRY:  Canadian
POS:    A
```

```
CAT:    S_0\NP_0
  (NP_0\NP_1)\((S_0\NP_2)/(S_1\NP_3))
FS:     #A_WH-
```

Since the exact figures for this sort of simple numerical compression are entirely dependent on incidental details of the composition of the original lexicon, it is more significant to look at the size of the set of categories used in the lexicon.

It is well known that CG categories are more detailed, and therefore more numerous, than the traditional categories of phrase structure grammars ("verb" becomes the set S\NP, (S\NP)/NP, ((S\NP)/NP)/NP, ..., etc.). It is less commonly observed that a single CG category can correspond to more than one PSG category, where different parts of speech have the same syntactic behaviour. For example,

S_0\NP_0

Intransitive active
The scuffling and miaowing **abated.**
Transitive bare passive
The food **was accepted.**
Predicative adjective
That proposal is **absurd.**
Predicative nominal
Pepper is **a tabby cat.**
Predicative pp
The president is **abroad.**

I refer to these as the *senses* of a category, and to a category with more than one sense as *ambiguous*. A *primary sense* is basic or irreducible, like the first sense (intransitive active) above. A *secondary sense* is a derived usage which could be predicted or derived by rule from some other category. Thus S_0\NP_0 (transitive bare passive) is derived from (S_0\NP_)/NP (transitive active) by a passive rule which systematically reduces the number of argument NPs to a verb by one. The three predicative senses are derived from basic adjectival, nominal, and prepositional categories by rules which are less neat schematically, but do make the appropriate predictions.

(Bear in mind that only the structural syntactic category itself is being considered here. Since TAG trees include part-of-speech information, "similar" looking trees are distinguished by the part-of-speech that anchors them. In CG categories, since part-of-speech information is not explicitly encoded, it appears that there are redundancies. However, as we saw above, lexical entries in the CCG Large Lexicon contain a POS field, so during lexical access, given a part-of-speech, there will not be any confusion of this nature.) Further, structurally identical categories will often be distinguished at a finer grain-size by having different features. The detailed form of any redundancy rules will have to include these.)

Although the proposed redundancy rules do give a worthwhile reduction in the number of categories needed, the number of senses which can be omitted, and the number of ambiguous categories, are more dramatically reduced.

The present CCG Large Lexicon category set includes:

86 categories, with
113 senses

of these, there are:

19 ambiguous categories, with
46 senses

By using redundancy rules to predict gerunds, passives, predicatives, and secondary nominal uses, we reduce this to

86 $\longrightarrow$ 65 categories, with
113 $\longrightarrow$ 73 senses

including:

19 $\longrightarrow$ 6 ambiguous categories, with
46 $\longrightarrow$ 14 senses.

The 40 senses eliminated (over one-third of the total) are made up of

12 gerunds
13 passives
13 predicatives
2 nominals

The 20 categories eliminated entirely include, for example:

((NP_0\NP_1)/NP_2)/NP_3

Gerund of ditransitive
John **giving** *the cats an unusually large breakfast kept them happy for a few hours.*

S_0/NP_0

Predicative
Pepper is a tabby cat - **What** *is Pepper?*

The thirteen ambiguous categories which become unambiguous include the example of S_0\NP_0 given above, which keeps only its primary sense of intransitive verb, losing four sec-

ondary senses, one passive and three predicative. When one considers that at present the first 15 words in the lexicon with this category are: *abate, abdicate, aberrant, abhorrent, abide, abject, able, abnormal, abominable, aboriginal, abort, abortive, above, abrasive, abroad* one advantage of the simplification is ovbvious. Similarly:

`NP_0\NP_1`
abate, abdicate, abide, abort, above, abroad, abscond, abstain, abut, accede, accelerate, accept, acclimatize, accord, accrue

Gerund of intransitive
the noise **abating**
Independent preposition
the stars **above**, *an Englishman* **abroad**

keeps only its prepositional sense and loses the gerundive.

The remaining ambiguities are entirely reasonable: for example,

`(S_0\NP_0)/(S_1\NP_1)`

Adverbs
Pepper **already** *was demanding breakfast.*
Auxiliary verbs
She **had** *prodded John's face several times.*

`(S_0\NP_0)\(S_1\NP_1)`

Adverbs
Pepper was demanding breakfast **already**.
Negation on auxiliaries
John did **not** *want to get up that early.*
Exhaustive PPs
He moved her **away**.

Redundancy rules will not only compress the explicitly given category set, but expand the set implicitly available. Crossing seven of the basic verb categories (intransitive, intrans + particle, intrans + adjective, transitive, trans + PP comp, trans + VP comp, trans + V comp) with five of the derived forms (active, bare passive, by-passive, gerund, gerund + determiner) should give 29 categories (as intransitives have no passive forms). Of these, only 18 are actually given in the current lexicon, presumably due to accidental gaps in the corpus data from which its parent TAG lexicon was originally derived.

Conclusion

This proposal will not be popular with the logical purists in the CG community. In language engineering terms, it will be necessary to control the appplicability of redundancy rules and to explore their effect on parsing. What I offer here is some quantified evidence, derived from a realistically large large lexicon intended for serious linguistic description, for the nature and scope of the benefits that a categorial grammar could gain from a systematic formalization of predictable lexical relations through lexical redundancy rules or category families.

Acknowledgements

This work rides on the shoulders of the people who developed the CCG Large Lexicon: I am deeply grateful to Christy Doran and B. Srinivas, in particular, for their generosity and support. Thanks also to David Brée, Jong C. Park, and Mark Steedman. Much of the credit is theirs; any errors or idiocies are mine.

References

Ajdukiewicz, K. 1935. "Die syntaktische Konnexität". *Studia Philosophica* 1:1-27; translated as "Syntactic connexion" in McCall (ed) *Polish Logic.* Oxford.

Bresnan, J. ed. 1982. *The Mental Representation of Grammatical Relations.* MIT Press, Cambridge, Mass.

Doran, Christy and B. Srinivas. Forthcoming. "Developing a Wide-Coverage CCG System" To appear in a CSLI Volume of the *Proceedings of TAG+3*, ed. Anne Abeille and Owen Rambow.

Gazdar, G., E. Klein, G. Pullum and I. Sag. 1985. *Generalized Phrase Structure Grammar.* Blackwell, Oxford.

Lambek, J. 1958. "The Mathematics of Sentence Structure". *American Mathematical Monthly* 65:154-70; reprinted in Buszkowski, Marciszewski and van Benthem (eds) *Categorial Grammar.* John Benjamins, Amsterdam.

Wood, M.M. 1993. *Categorial Grammars.* Routledge, London.

Xia, Fei, Martha Palmer, K. Vijay-Shanker, Joseph Rosenzweig. In preparation. "Consistent Grammar Development Using Partial-Tree Descriptions for Lexicalized Tree-Adjoining Grammar".

The XTAG Group. 1995. "A Lexicalized Tree Adjoining Grammar for English". Technical Report IRCS 95-03, University of Pennsylvania. Updated version available at `http://www.cis.upenn.edu/ xtag/tr/tech-report.h`

Packing of Feature Structures for Optimizing the HPSG-style Grammar translated from TAG

Yusuke Miyao†, Kentaro Torisawa†, Yuka Tateisi†, Jun'ichi Tsujii†‡
†Department of Information Science, Graduate School of Science, University of Tokyo*
Hongo 7-3-1, Bunkyo-ku, Tokyo 113-0033, Japan
‡CCL, UMIST, Manchester, U. K.
E-mail: {yusuke, torisawa, yucca, tsujii}@is.s.u-tokyo.ac.jp

1 Introduction

This paper describes a method for packing feature structures, which is used for reducing the number of constituents generated during parsing, and for improving the parsing speed. The method was developed for optimizing a parsing system for *XHPSG* (Tateisi et al., 1998) translated from XTAG (The XTAG Research Group, 1995). The XHPSG system is a wide-coverage parsing system for English based on HPSG framework (Pollard and Sag, 1994). This system is also intended to be used for processing large amounts of texts, for the purposes such as information extraction. Current parsing speed of our system is not sufficient enough to achieve this goal.

Our method improves the parsing speed by solving the problem which the XHPSG and the XTAG system have. That is, many lexical entries are assigned to a word, and many constituents are produced during parsing. The experimental results show that our method leads to a significant speed-up. The results also suggest the possibility of optimizing the XTAG system by introducing packing of feature structures and packing of tree structures, although these operations are not currently so apparent.

2 The XHPSG System

This section describes the current status of the XHPSG system and the efficiency problem in the system. Both of the grammar and the parser in the XHPSG system are implemented with feature structure description language, *LiLFeS* (Makino et al., 1998). The grammar consists of lexical entries for about 317,000 words, and 10 schemata, which follows schemata of the HPSG framework in (Pollard and Sag, 1994) with slight modifications. The parser is a simple CKY-based parser.

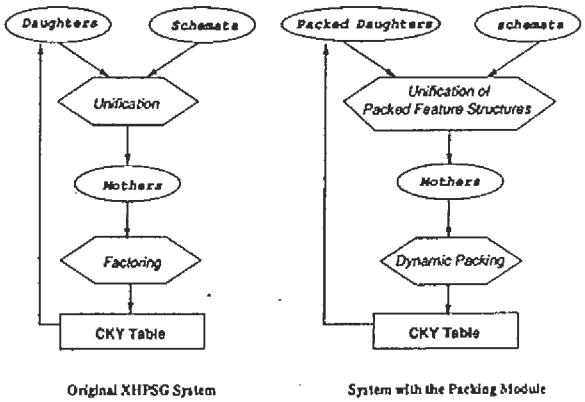


Figure 1: Data flow in the parsers for the XHPSG system.

Currently, the parsing speed of this system is not satisfactory, and we need further improvement of the parsing speed. One of the major reasons of inefficiency is that the XHPSG system assigns many lexical entries to a single word. For example, a noun is assigned 11 lexical entries, a verb is assigned 20–30 lexical entries, and some words are even assigned more than 100 entries.

This characteristic is inherited from the XTAG grammar. The XTAG grammar assigns many elementary trees to a single word, and there is a one-to-one correspondence between a lexical entry in XHPSG and an elementary tree in the XTAG grammar. The XTAG system applies a POS tagger before parsing in order to overcome this inefficiency by reducing the number of lexical entries assigned to a word. However, this method sacrifices the soundness of the

This work is partially founded by Japan Society for the Promotion of Science (JSPS-RFTF96P00502).

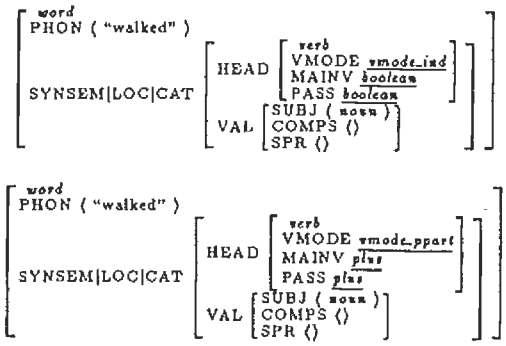


Figure 2: Two of the lexical entries for an English verb "walked". Underlined values are different. Most of the features are omitted for simplicity.

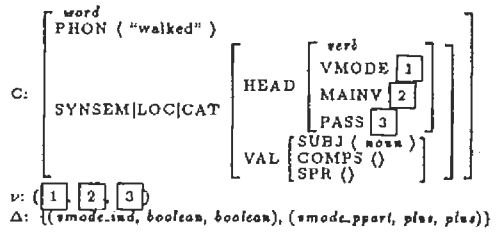


Figure 3: A *packed feature structure* for the lexical entries shown in Figure 2.

parsing process. In the case that the tagger fails to assign the correct POS to a word, correct syntactic structures may not be created even when the grammar potentially covers such structures.

To solve the same problem, we propose a new method described in the next section. The method can gain a similar effect, but does not sacrifice the soundness of parsing.

3 Packing of Feature Structures

The left hand side of Figure 1 illustrates the data flow of the original parser of the XHPSG system. There are two major operations, *unification* and *factoring*. When we apply a schema to daughters, a unification operation is performed, and a mother is created. A set of mothers are reduced to a smaller set of feature structures by factoring operation[1], and these constituents are put into CKY table.

```
create_PFS(F)
  C := ⊥, ν := (), δ := ()
  for each f ∈ features(F)
    if f ∈ DisjFeatures then
      ν := (follow(C, f))@ν
      δ := (follow(F, f))@δ
    else
      F' :=follow(F, f)
      (C', ν', {δ'}) =create_PFS(F')
      C := C ⊔ [f C']
      ν := ν@ν', δ := δ@δ'
    end_if
  end_for
  return (C, ν, {δ})
```

Figure 4: Algorithm for creating new packed feature structure from a feature structure. '@' denotes the concatenation operation of sequences.

The right hand side of Figure 1 illustrates the parser with the packing module. The unification and the factoring operation in the original parser was replaced by *unification of packed feature structures* and *dynamic packing*. These operations are more efficient than the corresponding one, because multiple applications of schemata are reduced to one unification of packed feature structures, and multiple operations of factoring are reduced to one dynamic packing. In addition, *dynamic packing* reduces the constituents further than the factoring operation.

With a simple example, now we see how feature structures are packed into one. Figure 2 shows two of the lexical entries that the XHPSG system assigns to an English verb "*walked*". These lexical entries correspond to distinct elementary trees of XTAG. They are different in only a few features, while each feature structure has over 100 features. That is, most of them have equivalent values, so that it is redundant to have each of them as two independent feature structures.

For these feature structures, a packed feature structure is described as in Figure 3. C specifies the common part of the original two feature

[1] A factoring operation in a CKY parser for CFG reduces the number of constituents by identifying constituents described by the identical non-terminals. The operation plays a crucial role for avoiding an exponential explosion of the time complexity of the parsing of CFG. In the case of HPSG, the similar effect can be accomplished by the factoring operation, which identifies the constituents with equivalent feature structures in this case. We have observed that parsing time with syntactic grammars can be reduced significantly, though this operation does not lead to a reduction of computational time complexity to polynomial.

```
pack_feature_structures(PFS)
  PFS' := φ
  for each P = (C, ν, Δ) ∈ PFS
    if P' = (C', ν', Δ') ∈ PFS' such that
      C' is equivalent to C and,
      for each i(0 ≤ i ≤ k)
      paths(C, n_i) = paths(C', n'_i)
      where ν = (n_0, ···, n_k) and ν' = (n'_0, ···, n'_k)
    then
      Δ'' := Δ ∪ Δ'
      PFS' := (PFS'\{P'}) ∪ {(C, ν, Δ'')}
    else
      PFS' := PFS' ∪ {P}
    end_if
  end_for
  return PFS'
```

Figure 5: Algorithm for packing a set of packed feature structures.

```
unify_packed_feature_structures(P_1, P_2)
  P_1 = (C_1, ν_1, Δ_1)
  P_2 = (C_2, ν_2, Δ_2)
  Δ := φ
  if success C := C_1 ⊔ C_2 then
    ν := ν_1@ν_2
    for each δ_1 ∈ Δ_1 and δ_2 ∈ Δ_2
      δ := copy((ν_1 ⊔ δ_1)@(ν_2 ⊔ δ_2))..U_1
      Δ := Δ ∪ {δ}
      Cancel the side-effect of ⊔
      occuring during computation of U_1.
    end_for
  end_if
  return (C, ν, Δ)
```

Figure 6: Algorithm for unifying two packed feature structures.

structures. ν expresses the nodes[2] in the feature structure, to which disjunctive structures are incorporated. The nodes are expressed as tags for structure sharings such as $\boxed{1}$. Δ expresses a set of different values, that come to the position specified by the nodes in ν. Hence, the original feature structures are obtained by unifying one of the elements of Δ to the nodes in ν. A packed feature structure holds exactly the equivalent information of the original feature structures with a smaller data size.

4 Algorithms

This section describes three algorithms, (1)conversion of a feature structure to a packed feature structure, (2)packing of packed feature structures and (3)unification of packed feature structures.

The last two algorithm requires packed feature structures as their inputs and the first algorithm is used for convert non-disjunctive feature structures to such inputs to the two algorithms. Figure 4 shows the first algorithm for converting a feature structure to a new packed feature structure. We assume that a packed feature structure is given as a triple (C, ν, Δ) as described in Section 3. The input to this algorithm is a (non-disjunctive) feature structure and a set of features, to which the disjunction is introduced. In the figure, F is a feature structure and $DisjFeatures$ is a set of features. The function follow(F, f) returns the node in F reached by the feature f from a root of F. What the algorithm does is to split F into two parts, the first part is C and the other part is a set of nodes and a set of substructures represented by ν and Δ respectively.

Figure 5 shows the algorithm for packing already packed feature structures. In the figure, $\mathcal{PFS}$ denotes a given set of packed feature structures, and $\mathcal{PFS}'$ denotes a newly created set of packed feature structures. The function $paths(F, n)$ returns a set of all the paths to the node n in F. The algorithm for packing lexical entries is straightforwardly obtained from this algorithm and the previous algorithm.

Figure 6 shows the algorithm for unifying two packed feature structures. The overall algorithm is similar to the one in (Kasper, 1987), although data structures for disjunctive feature structures are different. Intuitively, we first unify common parts (C_1 and C_2), and next check consistency of each combination of disjuncts in Δ_1 and Δ_2. The operator $\sqcup$ denotes the unification of non-disjunctive feature structures[3]. The unification is regarded as an destructive procedure in the figure. It has a side effect to the input feature structures. For instance, suppose that feature structures stored in the variables F and F' have the nodes stored in the variable n and n' as their substructure and that for some path π $follow(F, \pi) = n$, $follow(F', \pi) = n'$ and $n \neq n'$. After performing the unification $F \sqcup F'$, the values of F, F', n and n' are automatically updated and, as a result of the update, $F = F'$ and $n = n'$ hold. In the algorithm in the figure, this type of side-

[2] Though feature structures are expressed in a conventional matrix-like notation, they can be seen as directed graph with a root whose nodes and arcs are labeled. Features are labels for arcs and the labels for nodes are called types.

[3] Unification of tuples is a tuple of the results of the unification of corresponding elements of the tuples.

Features incorporated from XTAG
PRD, CASE, PRON, REFL, VMODE, MAINV, EXTRACT, TRANS, PASS, PERF, PROG, ASSIGN_CASE, INV
Other features
HEADPHON, MARKING, CONT, TRF

Table 1: Specified features for the experiments.

	Parsing time in avg. (sec.)	
	Test set A	Test set B
Old System(O)	2.31	14.45
New System(N)	1.29	5.88
Improvement Ratio(O/N)	1.79	2.46

The experiments are performed on Alpha Station 500 (500MHz CPU, 256MB Memory), and the times are measured in User Time.

Table 2: Results of the experiments.

effects is assumed to occur for the values stored in the variables such as $C_1, C_2, \nu_1, \nu_2, \delta_1$ and δ_2. The mechanisms for the side-effect and its canceling are similar to the execution mechansims of Prolog, including backtracking. They are also implemented in LiLFeS. The *copy* is a procedure to create a *distinct* feature structure equivalent to the input feature structure and the newly created feature structure is free from the side-effect of the unification against the original input feature structure.

5 Experiments

This section shows the experimental results of the current implementation of our packing method. Experiments are performed by specifying features originated in XTAG and a few other features as in Table 1.

The packing module is implemented with LiLFeS, and is incorporated into the XHPSG system. We compared the parsing times of (1)*Test set A* (337 sentences, 8.37 words/sentence)[4] and (2)*Test set B* (16 sentences, 11.88 words/sentence)[5], between the (1)*New System* (with the packing module) and the (2)*Old System* (without the packing module). The parsers of both systems are simple CKY-based parsers. As Table 2 shows, the parsing speed improves by 1.79 times in Test set A, and 2.46 times in Test set B, which consists of sentences costing much time to parse. In Test set A, the number of lexical entries is reduced by 35.3%, and that of constituents in the CKY table by 46.7% on average.

[4]Test set A is bundled in the XTAG system for checking the grammar.

[5]Test set B is a subset of Test set A. The subset consists of 16 sentences, each of which costs more than 10 seconds to parse.

6 Conclusion and Future Work

We proposed a method for packing feature structures by introducing disjunctions into feature structures. This method reduces the number of lexical entries in HPSG grammars and constituents created during parsing. As a result, we achieved 1.74 times improvement in parsing time for the test corpus bundled in the XTAG system. We expect to gain the similar effect with the XTAG system by applying our packing method, though it is currently not so apparent.

For realizing a practical parsing system, we are currently integrating our packing method with other two optimization techniques: (1)implementation with a native compiler version of LiLFeS (Makino et al., 1998), and (2)compilation of HPSG to CFG (Torisawa and Tsujii, 1996). As a result of the latter optimization, current XHPSG system can parse sentences in the ATIS corpus in 1.12 seconds on average without any POS taggers. Further speed-up is expected by integrating our method to this system.

References

Robert T. Kasper. 1987. A unification method for disjunctive feature descriptions. In *Proc. 26th ACL*, pages 235–242.

Takaki Makino, Minoru Yoshida, Kentaro Torisawa, and Jun'ichi Tsujii. 1998. LiLFeS — towards a practical HPSG parser. In *Proc. of COLING-ACL '98*. To appear.

C. Pollard and I. A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press.

Yuka Tateisi, Kentaro Torisawa, Yusuke Miyao, and Jun'ichi Tsujii. 1998. Translating the XTAG English grammar to HPSG. In *Proc. TAG+ Workshop*.

The XTAG Research Group. 1995. A lexicalized tree adjoining grammar for English. Technical report, IRCS Report 95-03, University of Pennsylvania, March.

Kentaro Torisawa and Jun'ichi Tsujii. 1996. Computing phrasal-signs in HPSG prior to parsing. In *Proc. 16th COLING*, pages 949–955.

TAGS M-CONSTRUCTED

Uwe Mönnich
Seminar für Sprachwissenschaft
Tübingen University
Wilhelmstrasse 113
D-72074 Tübingen
Uwe.Moennich@uni-tuebingen.de

Abstract

This paper puts TAGs into an algebraic perspective. The operation of tree adjunction is shown to be a special case of function substitution within a derived theory. The underlying process of theory derivation is illustrated with the concrete example of free continuous tree algebras.

1 Introduction

The aim of this paper is to relate two notions. The first one is that of tree adjunction. The operation of tree adjunction serves to seperate dependency and recursion within a mild extension of the context-free grammar formalism. The second notion is that of a polyadic procedure. It generalizes the operation of making several identical copies of a string and was introduced in formal language theory by Fischer (1968).

The two notions are related in the following way. The operation of tree adjunction builds a new tree t from two input trees t_1 and t_2 by replacing a subtree of t_1 displaying a root label identical to t_2's root label with the tree t_2 and appending the replaced subtree of t_1 to an especially marked leaf node of t_2. The name of a polyadic procedure in a tree can similary be replaced by a tree with dummy symbols at some of its leaves into which the arguments of the replaced procedure are to be inserted.

It has long been realized that the introduction of higher order auxiliairy symbols into a grammar formalism is an iterable process that leads to an algebraic refinement of the Chomsky hierarchy. The most general characterization of this iterable process is due to the ADJ group and presented by them within the category theoretic framework of finitary algebraic theories (Bloom et al. 1983). Based on their presentation, we propose an abstract formulation of tree-adjoining grammars in which its rule systems correspond to morphisms of an algebraic theory that is constructed from the algebraic theory of context-free grammars along the lines indicated by the ADJ group.

The notion of an algebraic refinement of the Chomsky hierarchy was first formulated by Wand (1975). He shows that solving regular equations in function spaces over languages leads to a hierarchy of language families beginning with the regular languages, the contex-free languages and the indexed language. His conjecture that these language families are but the first steps in an infinite hierarchy was later confirmed by Damm (1982).

The original motivation for our interest is an algebraic formulation of tree adjoining grammars has come from a long term project on denotational semantics for grammar formalisms. Algebraic semantics seems to provide a uniform framework for such an attempt. In the present connection the algebraic perspective not only adds another characterization of the tree adjoining languages to the already long list of equivalences with restricted production systems, but it also makes available the whole gamut of techniques that have been developed in the tradition of algebraic language theory (Maibaum 1978, Mehlhorn 1979, Schimpf and Gallier 1985).

In the interest of a more concrete presentation we restrict ourselves to the special case of tree algebras. The basic notions from universal algebra which we need in the sequel are introduced in the next section. For reasons of space we have refrained from supplying the details of the general M-functor.

2 Basic Definitions

Let S be a set of sorts. A *many-sorted signature* Σ is an indexed family $(\Sigma_{w,s}|w \in S^*, s \in S)$ of disjoint sets. A symbol in $\Sigma_{w,s}$ is called an *operator* of *type* $\langle w, s\rangle$, *arity* w, *sort* s and *rank* $\ell(w)$, where $\ell(w)$ denotes the length of w. In the case of a single-sorted signature we write $\Sigma_{s^n,s}$ as Σ_n. The set of n-ary

trees over such a single-sorted signature Σ is built up from a finite set $X_n = \{x_1, \ldots, x_n\}$ of *variables* using the operators in the expected way: If $\sigma \in \Sigma_n$ and $t_1, \ldots, t_n$ are n-ary trees, then $\sigma(t_1, \ldots, t_n)$ is an n-ary tree.

The operator symbols induce operations on an algebra of the appropiate structure. A Σ-*algebra* A consists of an S-indexed family of sets $A = \langle A_s \rangle_{s \in S}$ and for each operator $\sigma \in \Sigma_{w,s}$, a function $\sigma : A^w \to A^s$ where $A^w = A_1^w \times \cdots \times A_n^w$ and $w = w_1 \cdots w_n$. The set of n-ary trees $T(\Sigma, X_n)$ can be made into a Σ-algebra by specifiying the operations as follows. For every $\sigma \in \Sigma_n$ and every $t_1, \ldots, t_n \in T(\Sigma, X_n)$ we identify $\sigma_{T(\Sigma,X_n)}(t_1, \ldots, t_n)$ with $\sigma(t_1, \ldots, t_n)$.

3 Lawvere Theories

Our main notion is that of an *algebraic (Lawvere) theory*. Given a set of sorts S, an algebraic theory, as an algebra, is an $S^* \times S^*$-sorted algebra T, whose carriers $\langle T(u,v) | u, v \in S^* \rangle$ consist of the morphisms of the theory and whose operations are of the following types:

- projection: $x_i^u \in T(u, u_i)\ (u = u_1 \ldots u_n \in S^*)$
- composition: $\cdot_{u,v,w} \in T(u,v) \times T(v,w) \to T(u,w)\ (u,v,w \in S^*)$
- target tupling: $(\ , \ldots, \ )_{u,v} \in T(u,v_1) \times \cdots \times T(u,v_n) \to T(u,v)\ (u, v = v_1 \ldots v_n \in S^*)$

The projections and the operations of target tupling are required to satisfy the obvious identities for products and the composition operations are required to be associative:

- $x_i^v \cdot (\alpha_1, \ldots, \alpha_n)_{u,v} = \alpha_i$ for all $\alpha_i \in T(u, v_i)$
- $(x_1^v \cdot \beta, \ldots, x_n^v \cdot \beta)_{u,v} = \beta$ for all $\beta \in T(u,v)$, where $v = v_1 \cdots v_n$
- $(\gamma \cdot \beta) \cdot \alpha = \gamma \cdot (\beta \cdot \alpha)$ for all $\alpha \in T(u,v), \beta \in T(v,w), \gamma \in T(w,z)$
- $\alpha \cdot (x_1^u, \ldots, x_1^u)_{u,u} = \gamma$ for all $\alpha \in T(u,v)$, where $u = u_1 \cdots u_n$

By rearranging the ingredients of the prededing definition algebraic theories can be looked upon as categories. Under this conceptualization an algebraic theory T has as objects $|T|$ the set of sort-strings S^*, the elements of the carrier sets become morphisms in the category theoretic sense and the following tuples of the projection morphisms $(x_1^u, \ldots, x_1^u)_{u,u}$ function as identities. The axioms for the composition operation ensure that it behaves as is required by the basic category theoretic postulates for the operation of the same name and the axioms for target tupling ensure its status as a category theoretic product.

With S being a singleton, the powerset $\wp(T(\Sigma))$ of n-ary trees constitutes the central example of interest for formal language theory. The carriers $\langle \wp T(n,m) | n, m \in w \rangle$ consist of sets of m-tuples of n-ary trees $\{(t_1, \ldots, t_m)\}$. The operation of composition is defined as substitution for the projection constants and target tupling is just tupling.

The M-construction can be characterized as a functorial generalization of the device of signature extension. For lack of space we abstain from giving the general definition and restrict ourselves to outlining the relevant features for the case of free continuous theories. Suppose that Σ is an one-sorted signature. Elements of $S^* \times S^*$ can then be identified with elements of $\omega \times \omega$. Given a finite set of *function variables* F, we obtain the extended signature $\Sigma + F$, where $(\Sigma + F)_n = \Sigma_n \cup \{f | f \in F \ \&\ \text{arity}(f) = n\}$. Based on this signature we are able to define the notion of a finite tree t of *recursion-sort* n and *recursion-arity* w, $w \in \omega^*$. This says that nodes in t dominating w_i daughters may be labeled with $f \in F$ of arity w_i and that its projection labels come from $X_n = \{x_1, \ldots, x_n\}$. Given Σ and F, we can now define the M-constructed *continuous, one-sorted recursion theory* $M(\wp(T(\Sigma)))$ as follows. For $v \in \omega^n$, $w \in \omega^*$, $M(\wp(T(\Sigma)))(w,v)$ is the powerset of all n-tuples of trees $t = (t_1, \ldots, t_n)$, where t_i is of recursion sort v_i and of recursion arity w. Tupling is again tupling, the function variables play the role of "higher-order" projections, but composition is specified as substitution for function-variables which label internal tree nodes, rather than as substitution for projection labels at the leaves of trees. For $u \in \omega^n$, $v \in \omega^p$ and $w \in \omega^*$, let T' be a set of p-tuples of trees $t' = (t'_1, \ldots, t'_p)$ of recursion arity w and of recursion sort v and let T be a set of n-tuples of trees $t = (t_1, \ldots, t_n)$ of recursion arity v and of recursion sort u , then their composition $T \cdot T' = \{t''\} = \{(t''_1, \ldots, t''_n)\} = \{(t_1 \cdot t', \ldots, t_n \cdot t')\}$ is defined recursively as follows:

- $t''_i = \{x_j^{v_i}\}$ for $t_i = x_j^{v_i}$
- $t''_i = \{\sigma(\tau_1 \cdot t', \ldots, \tau_q \cdot t')\}$ for $t_i = \sigma(\tau_1, \ldots, \tau_q)(\sigma \in \Sigma_q)$
- $t''_i = \{t'_j(\tau_1 \cdot t', \ldots, t_r \cdot t')\}$ for $t_i = f_j(\tau_1, \ldots, \tau_r)(f_j \in F_r)$

4 Context-Free and Tree Adjoining Languages

Consider the example of a single-sorted signature of monadic algebras:

$$\Sigma_0 = \{\epsilon\} \qquad \Sigma_1 = \{a | a \in V\}$$

Due to the fact that Σ is a *monadic* signature trees in $T(\Sigma, X)$ may not contain more than a single variable. Observe that this corresponds exactly to the rule format of regular (string) languages, where the righthand sides of production rules are either strings in the terminal alphabet or concatenations of such a string with a single non-terminal. The regular language V^*, e.g., is the solution of the set of equations $\{x = a(x) | a\}\{a \in V\}$ in the space $\wp(T(\Sigma))$. It should be pointed out that V^* and the set of all variable-free trees in the monadic signature Σ, introduced a moment ago, are, strictly speaking, not the same sets. They are nevertheless related by an obvious one-to-one correspondence.

Once the signature Σ is extended with one nullary and one monadic variable, the following example shows that we obtain the context-free langnage $L = \{a^n b^n\}$ as solution in the same space $\wp(T(\Sigma))$, where $\Sigma_1 = \{a, b\}$:

$$G = \langle \Sigma, F, S, E \rangle$$

$$F_0 = \{S\} \qquad F_1 = \{F\}$$

$$E = \left\{ \begin{array}{lll} S & = & \{F(\epsilon), \epsilon\} \\ F(x) & = & \{a(F(b(x))), a(b(x))\} \end{array} \right\}$$

$$L(E, S) = \{\overbrace{a(a \ldots}^{n} \overbrace{(b(b \ldots}^{n} (\epsilon) \ldots)\}$$

The pair of equations E in the preceding example is represented by a morphism

$$E = (E_0, E_1) : 0 \cdot 1 \to 0 \cdot 1$$

in the recursion theory $M(P(T(\sigma)))$ and the language $L = \{a^n b^n\}$ is the first component of the least fixpoint that solves the equational system E.

Observe again that the preceding equational system looks suspiciously similar to the usual production system for the "same" langnage in a concatenative signature Σ':

$$G' = \langle \Sigma', F, S, P \rangle$$

$$\Sigma_0 = \{\epsilon, a, b\} \qquad \Sigma_2 = \{\frown\} \qquad F_0 = \{S\}$$

$$P = \{S \to \epsilon | \frown(a, \frown(S, b))\}$$

$$L(G', S) = \{\frown(a, \frown(\ldots, \frown(\epsilon, b) \ldots b) \ldots)\}$$

where n occurrences of a precede the same number of occurrences of b for $n \geq 0$.

The following result expresses the fact that the situation above characterizes already the whole class of context-free languages: Every context-free language can be represented as the solution of a morphism in an algebraic theory that is M-constructed on the basis of a monadic tree theory.

There is actually a mechanical procedure that allows one to convert an arbitrary context-free grammar $G = \langle V, N, S, P \rangle$ in Chomsky Normal Form into a weakly equivalent equational system $E = \langle \Sigma_V, F, E \rangle$ that has a solution in the space of monadic trees (Maibaum 1974). The procedure consists in first forming the monadic signature Σ_V corresponding to the terminal vocabulary V of G:

$$(\Sigma_V)_0 = \{\epsilon\} \qquad (\Sigma_V)_1 = \{V\}$$

The new function variables F are similarly in a one-to-one correspondence with the nonterminals of G:

$$F_0 = \{S\} \qquad F_1 = \{A | A \in N\}$$

The equational system E is then obtained through the following replacements:

$S \to AB$	$\Rightarrow$	$S = \{A(B(\epsilon))\}$	
$S \to a$	$\Rightarrow$	$S = \{a(\epsilon)\}$	
$S \to \epsilon$	$\Rightarrow$	$S = \{\epsilon\}$	
$A \to BC$	$\Rightarrow$	$A(x) = \{B(C(x))\}$	for $A \neq S$
$A \to a$	$\Rightarrow$	$A = \{a(x)\}$	for $A \neq S$

$L(G, S)$ equals the least solution of E at its S-component.

Recall that context-free languages are also captured by the notion of a frontier or yield of a regular tree set. The obvious question that presents itself in this connection is which language family is reached by the addition of monadic function variables to an arbitrary signature.

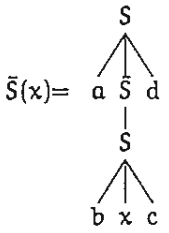
In the way of motivating the answer to this question it is useful to consider once more the example of a simple morphism $E' : 0 \cdot 1 \to 0 \cdot 1$ in an M-constructed recursion theory that is based on a signature Σ of arity 3:

$$\Sigma = \Sigma_0 \cup \Sigma_3 \text{ where } \Sigma_0 = \{a, b, c, d\} \text{ and } \Sigma_3 = \{S\}$$

$$F = F_0 \cup F_1 \text{ where } F_0 = \{S'\} \text{ and } F_1 = \{\bar{S}\}$$

$$E = \{S' = \{\bar{S}(\epsilon)\}, \bar{S}(x) = \{S(a, \bar{S}(S(b, x, c)), d), x)\}\}$$

In tree form the last equation has the following shape:



This system specifies the string language $\{a^n b^n c^n d^n\}$. Apart from minor notational modifications the grammar in the last example corresponds to a well-known tree adjoining grammar. Note that apart from the start symbol the only other nonterminal is of arity one. As was the case in connection with the context-free string languages, the preceding example is a particular instance of the general situation. The tree adjoining languages correspond to languages that are M-constructed from arbitrary signatures through the addition of monadic function variables.

As in the case of context-free grammars there exists a mechanical procedure that allows one to produce for any given tree adjoining grammar G a weakly equivalent equational system E that specifies the "same" set of trees. Strict identity is not guaranteed for grammars that contain nonterminals with variable arities. To remain within the algebraic setup, nonterminals that label nodes which branch out into different numbers of daughters, have to be assigned to different components of the indexed set Σ. Otherwise the procedure that resulted in the system of the example is completely general. Terminals and nonterminals alike are collected into the new signature Σ. All nonterminals that are free for an adjunction become duplicated by a monadic member of the set of function variables F. Adjunction constraints have to be taken over with one modification: When sa is the empty set the nonterminal has no duplicate in F.

5 Conclusion

The M-construction in its general form is conceived for Lawvere theories regarded as categories. The main prerequisites a category of such theories has to satisfy in order for it to be M-able is the existence of a free theory and of coproducts. Both conditions are fulfilled by the powerset of n-ary trees.

In compliance with the spirit of algebraic semantics I have considered tree adjoining languages as solutions of morphisms in a derived theory. Under the perspective of an operational semantics an analoguous characterization can be obtained by considering tree adjoining grammars as a restricted form of context-free tree grammars (Engelfriet and Schmidt 1977). This has been the topic of a previous publication where it is shown that not only any tree adjoining language is presentable as a monadic context-free tree language, but that the opposite implication holds as well (Mönnich 1997). The proof in that paper for this opposite direction of the implication is easily adapted to the framework of denotational semantics. As was adumbrated in the introductory section, the particular conception of denotational semantics that is being developed within the algebraic tradition promises to provide the right level of abstraction from where to investigate the connections between different types of grammatical formalisms.

References

Bloom, S.L., J.W. Thatcher, E.G. Wagner, and J.B. Wright. 1983. Recursion and iteration in continuous theories: The M-construction. *J. of Computer and System Sciences*, 27(2):148–164.

Damm, Werner. 1982. The IO- and OI-hierarchies. *Theoretical Computer Science*, 20:95–207.

Engelfriet, Joost and E.M. Schmidt. 1977. IO and OI, part I. *J. Comput. System Sci.*, 15:328–353.

Fischer, Michael J. 1968. Grammars with macro-like productions. In *Proceedings of the 9th Annual Symposium on Switching and Automata Theory*, pages 131–142. IEEE.

Joshi, A.K. and Y. Schabes. 1997. Tree-adjoining grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages Vol. 3: Beyond Words*, volume 3. Springer, pages 69–124.

Maibaum, T.S.E. 1974. A generalized approach to formal languages. *J. Comput. System Sci.*, 8:409–439.

Maibaum, T.S.E. 1978. Pumping lemmas for term languages. *Journal of Computer and System Science*, 17:319–330.

Mehlhorn, Kurt. 1979. Parsing macro grammars top down. *Information and Control*, 40:123–143.

Mönnich, U. 1997. Adjunction as substitution. In G-J. M. Kruijff, G.V. Morrill, and R.T. Oehrle, editors, *Formal Grammar 1997: Proceedings of the Conference*. Aix-en-Provence, pages 169–178.

Schimpf, Karl and Jean Gallier. 1985. Tree pushdown automata. *Journal of Computer and System Sciences*, 30:25–40.

Wagner, E.G. 1994. Algebraic semantics. In S. Abramsky, D.M. Gabbay, and T.S.E. Maibaum, editors, *Handbook of Logic in Computer Science Vol. 3: Semantic Structures*. OUP, pages 323–393.

Wand, Michell. 1975. An algebraic formulation of the chomsky hierarchy. In *Category Theory Applied to Computation and Control*, number 25 in Lecture Notes in Computer Science, pages 209–213.

Description Theory, LTAGs and Underspecified Semantics*

Reinhard Muskens
Department of Linguistics
Tilburg University
P.O. Box 90153
5000 LE Tilburg, The Netherlands
r.a.muskens@kub.nl

Emiel Krahmer
IPO
Eindhoven University of Technology
P.O. Box 513
5600 MB Eindhoven, The Netherlands
krahmer@ipo.tue.nl

An attractive way to model the relation between an underspecified syntactic representation and its completions is to let the underspecified representation correspond to a *logical description* and the completions to the *models* of that description. This approach, which underlies the *Description Theory* of (Marcus et al. 1983) has been integrated in (Vijay-Shanker 1992) with a pure unification approach to *Lexicalized Tree-Adjoining Grammars* (Joshi et al. 1975, Schabes 1990). We generalize Description Theory by integrating semantic information, that is, we propose to tackle both syntactic *and* semantic underspecification using descriptions.[1] Our focus will be on underspecification of *scope*. We use a generalized version of LTAG, to which we shall refer as LFTAG. Although trees in LFTAG have surface strings at their leaves and are in fact very close to ordinary surface trees, there is also a strong connection with the *Logical Forms* (LFs) of (May 1977). We associate logical interpretations with these LFs using a technique of *internalising the logical binding mechanism* (Muskens 1996). The net result is that we obtain a Description Theory-like grammar in which the descriptions underspecify semantics. Since everything is framed in classical logic it is easily possible to *reason* with these descriptions.

1 Syntactic Composition

Descriptions in our theory model three kinds of information. First, there are *input descriptions*, which vary per sentence. For example, for sentence (1) we have (2) as an input description. It says that there are two lexical nodes,[2] labeled *John* and *walks* respectively; that the first of these precedes the second; and that these two lexical nodes are all that were encountered. Secondly, there is a *lexicon* which includes semantic information. The entries for *John* and *walks* are given in (3) and (4).

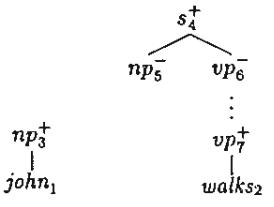
(1) John walks.

(2) $\exists n_1 n_2(lex(n_1) \wedge lex(n_2) \wedge n_1 \prec n_2 \wedge lab(n_1, john) \wedge lab(n_2, walks) \wedge \forall n(lex(n) \rightarrow (n = n_1 \vee n = n_2)))$

(3) $\forall n_1(lab(n_1, john) \rightarrow \exists n_3(lab(n_3, np) \wedge n_3 \lhd n_1 \wedge \alpha^+(n_3) = n_1 \wedge \sigma(n_3) = John \wedge \forall n(\alpha^+(n) = n_1 \rightarrow (n = n_3 \vee n = n_1)) \wedge \forall n(\alpha^-(n) = n_1 \rightarrow n = n_1)))$

(4) $\forall n_2(lab(n_2, walks) \rightarrow \exists n_4 n_5 n_6 n_7(lab(n_4, s) \wedge lab(n_5, np) \wedge lab(n_6, vp) \wedge lab(n_7, vp) \wedge n_4 \lhd n_5 \wedge n_4 \lhd n_6 \wedge n_6 \lhd^* n_7 \wedge n_7 \lhd n_2 \wedge n_5 \prec n_6 \wedge \alpha^+(n_4) = \alpha^+(n_7) = n_2 \wedge \forall n(\alpha^+(n) = n_2 \rightarrow (n = n_4 \vee n = n_7 \vee n = n_2)) \wedge \alpha^-(n_5) = \alpha^-(n_6) = n_2 \wedge \forall n(\alpha^-(n) = n_2 \rightarrow (n = n_5 \vee n = n_6 \vee n = n_2)) \wedge \sigma(n_4) = \sigma(n_6)(\sigma(n_5)) \wedge \sigma(n_7) = \lambda v.walk\ v))$

The function symbol α^+ used in these descriptions *positively anchors* nodes to lexical nodes, α^- *negatively anchors* nodes and σ gives a node its semantic value. Since descriptions are unwieldy we partially abbreviate them with the help of pictures:



Here uninterrupted lines represent immediate dominance ($\lhd$) and dotted lines represent dominance ($\lhd^*$), as usual. Additionally we mark positive and

*We wish to thank Kurt Eberle, Barbara Partee, Stanley Peters and all other participants of the Bad Teinach Workshop on Models of Underspecification and the Representation of Meaning (May 1998) for their comments and criticisms on an earlier version of this paper.

[1] The approach to underspecified semantics taken in (Muskens 1995) was very much inspired by Description Theory and the work of Vijay-Shanker in (Vijay-Shanker 1992) but did not offer an actual integration with Tree-Adjoining Grammars. In this paper we endeavour to set this right.

[2] With *lexical nodes* we mean those leaves in a tree which carry a lexeme.

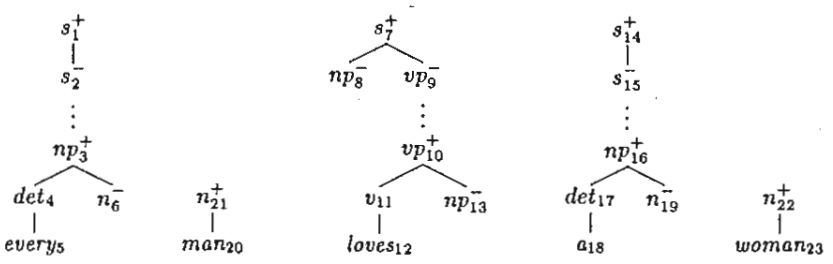


Figure 1: Elementary descriptions for *every man loves a woman*

negative anchoring in the following way. If a description contains the information that a certain nonlexical node is positively (negatively) anchored, the term referring to that node gets a plus (minus) sign. But pluses and minuses cancel and terms that would get a $\pm$ by the previous rule will be left unmarked. Terms marked with a plus (minus) sign are to be compared with the bottom (top) parts of Vijay-Shanker's 'quasi-nodes' in (Vijay-Shankar 1992). There is also an obvious close connection with positive (negative) occurrences of types in complex types in Categorial Grammar.

To the third and final kind of descriptions belong axioms which say that $\lhd$, $\lhd^*$ and $\prec$ behave like immediate dominance, dominance and precedence in trees ($\mathcal{A}1$ - $\mathcal{A}10$, see also e.g., Cornell 1994, Backofen et al. 1995:9)[3] combined with other general information, such as the statements that labeling is functional ($\mathcal{A}11$), and that different label names denote different labels ($\mathcal{A}12$). $\mathcal{A}13$ and $\mathcal{A}14$ say that all nodes must be positively anchored to lexical nodes and that all lexical nodes are positively anchored to themselves. The axioms for negative anchoring ($\mathcal{A}15$ and $\mathcal{A}16$) are similar, but allow the root r to be negatively anchored to itself.

$\mathcal{A}1$ $\forall k\,[r \lhd^+ k \vee r = k]$

$\mathcal{A}2$ $\forall k \neg k \lhd^+ k$

$\mathcal{A}3$ $\forall k_1 k_2 k_3\,[[k_1 \lhd^+ k_2 \wedge k_2 \lhd^+ k_3] \rightarrow k_1 \lhd^+ k_3]$

$\mathcal{A}4$ $\forall k \neg k \prec k$

$\mathcal{A}5$ $\forall k_1 k_2 k_3\,[[k_1 \prec k_2 \wedge k_2 \prec k_3] \rightarrow k_1 \prec k_3]$

$\mathcal{A}6$ $\forall k_1 k_2\,[k_1 \prec k_2 \vee k_2 \prec k_1 \vee k_1 \lhd^+ k_2 \vee k_2 \lhd^+ k_1 \vee k_1 = k_2]$

$\mathcal{A}7$ $\forall k_1 k_2 k_3\,[[k_1 \lhd^+ k_2 \wedge k_1 \prec k_3] \rightarrow k_2 \prec k_3]$

$\mathcal{A}8$ $\forall k_1 k_2 k_3\,[[k_1 \lhd^+ k_2 \wedge k_3 \prec k_1] \rightarrow k_3 \prec k_2]$

[3]Note that $\mathcal{A}9$ and $\mathcal{A}10$ in themselves do not suffice to exclude that some nodes are connected by a dominance relation without there being a (finite) path of immediate dominances between them. In fact the nature of our input descriptions and the form of our lexicon exclude this.

$\mathcal{A}9$ $\forall k_1 k_2\,[k_1 \lhd k_2 \rightarrow k_1 \lhd^+ k_2]$

$\mathcal{A}10$ $\forall k_1 k_2 k_3 \neg[k_1 \lhd k_3 \wedge k_1 \lhd^+ k_2 \wedge k_2 \lhd^+ k_3]$

$\mathcal{A}11$ $\forall k \forall \ell_1 \ell_2\,[[lab(k, \ell_1) \wedge lab(k, \ell_2)] \rightarrow \ell_1 = \ell_2]$

$\mathcal{A}12$ $l_1 \neq l_2$, if l_1 and l_2 are distinct label names

$\mathcal{A}13$ $\forall k\, lex(\alpha^+(k))$

$\mathcal{A}14$ $\forall k\,[lex(k) \rightarrow \alpha^+(k) = k]$

$\mathcal{A}15$ $\forall k\,[k = r \vee lex(\alpha^-(k))]$

$\mathcal{A}16$ $\forall k\,[[lex(k) \vee k = r] \rightarrow \alpha^-(k) = k]$

Together with this extra information (2), (3) and (4) conspire to determine a single model. Only n_1 and n_2 are lexical nodes. All nodes must be positively anchored to a lexical node. The set of nodes positively anchored to n_1 is $\{n_1, n_3\}$ and the set positively anchored to n_2 is $\{n_2, n_4, n_7\}$. So the remaining n_5 and n_6 must corefer with one of the constants mentioned, the only possibility being that $n_5 = n_3$ and that $n_6 = n_7$. The reader will note that in the resulting model $\sigma(n_4) =$ *walk John*. The general procedure for finding out which models satisfy a given description is to identify positively marked terms with negatively marked ones in a one-to-one fashion. The term r, denoting the root, counts as negatively marked.

In the given example only one tree was described, but this is indeed an exceptional situation. It is far more common that a multiplicity of trees satisfy a given description. This kind of underspecification enabled (Marcus et al. 1983) to define a parser which does not only work in a strict left-right fashion but is also incremental in the sense that at no point during a parse information need be destroyed. A necessary condition for this form of underspecification is that there are *structures* which can be described. In the context of semantic scope differences it therefore is natural to turn to (May 1977)'s Logical Forms, as these are the kind of models required. In fact we use a variant of May's trees which is very close to ordinary surface structure: although we will allow NPs to be raised, the syntactic material of such NPs will in fact remain *in situ*. But while the only syntac-

tic effect of raising will be the creation of an extra S node and Logical Forms will have their corresponding surface structures as subtrees, the 'movement' has an important effect on semantic interpretation. Consider example (5).

(5) Every man loves a woman.

We have depicted its five lexical items in fig. 1. With two exceptions they pretty much conform to expectation. The exceptions are that each determiner comes with a pair of S nodes dominating its NP. The basic idea here is that the long-distance phenomenon of quantifying-in is treated within the domain of extended locality of a determiner. In each case the semantics of the higher S will be composed out of the semantics of the lower S and the semantics of the NP, the semantic composition rule being quantifying-in.[4] The two Ss are to be compared to the two Ss at the adjunction site of a raised NP in May's theory. There is also an obvious connection with the (single) S where 'NP-retrieval' occurs in Cooper's theory of Quantifier Storage (see Cooper 1983).

It is easily seen that in any model of the descriptions in fig. 1 (+ the input description for (5) + our axioms) certain identities must hold: $n_6 = n_{21}$, $n_{19} = n_{22}$, $n_9 = n_{10}$, $n_8 = n_3$, and $n_{13} = n_{16}$ are derivable. But there is a choice between two further possibilities, as it can be the case that $n_2 = n_{14}$ and $n_{15} = n_7$, or, alternatively, that $n_{15} = n_1$ and $n_2 = n_7$. These two possibilities will correspond to the two different readings of the sentence.

2 Internalising Binding

How can we assign a semantics to the lexical descriptions in fig. 1? We must e.g. be able to express the semantics of n_1 in terms of the semantics of n_2, *whatever the latter turns out to be*, i.e. we must be able to express the result of quantification into an arbitrary context. In mathematical English we can say that, for any φ, the value of $\forall x\varphi$ is the set of assignments a such that for all b differing from a at most in x, b is an element of the value of φ. We need to be able to say something similar in our logical language, i.e. we must be able to talk about things that function like variables and constants, things that function like assignments, etc. The first will be called *registers*, the second *states*. Two primitive types are added to the logic: π and s, for registers and states respectively. We shall have *variable registers*, which stand proxy for variables and *constant registers* for constants. However, since registers are simply objects in our models, both variable registers and constant registers can be denoted with variables as well as with constants. Here are some axioms:

$\mathcal{A}17$ $\forall i_s \forall v_\pi \forall x_e [VAR(v) \rightarrow \exists j_s [i[v]j \wedge V(v)(j) = x]]$

$\mathcal{A}18$ $\forall k\, VAR(u(k))$

$\mathcal{A}19$ $\forall k_1 k_2 [u(k_1) = u(k_2) \rightarrow k_1 = k_2]$

$\mathcal{A}20$ $\forall i.V(John_\pi)(i) = \text{john}_e$,
$\forall i.V(Mary)(i) = \text{mary}_e, \ldots$

Here VAR is a predicate which singles out variable registers, V assigns a value to each register v in each state j, and $i[\delta]j$ is an abbreviation of $\forall w[w \neq \delta \rightarrow V(w)(i) = V(w)(j)]$. $\mathcal{A}17$ forces states to behave like assignments in an essential way. The function u assigns variable registers to nodes ($\mathcal{A}18$). Each node is assigned a fresh register ($\mathcal{A}19$). Constant registers have a fixed value ($\mathcal{A}20$). For more information on a strongly related set of axioms see (Muskens 1996).

These axioms essentially allow our logical language to speak about binding and we can now use this expressivity to embed predicate logic into (the first-order part of) type theory, with the side-effect that binding can take place on the level of registers. Write

$$
\begin{array}{rll}
R\delta_1 \ldots \delta_n & \text{for} & \lambda i.R(V(\delta_1)(i), \ldots, V(\delta_n)(i)), \\
\textbf{not } \varphi & \text{for} & \lambda i \neg\varphi(i), \\
\varphi \,\&\, \psi & \text{for} & \lambda i[\varphi(i) \wedge \psi(i)], \\
\varphi \Rightarrow \psi & \text{for} & \lambda i[\varphi(i) \rightarrow \psi(i)], \\
\textbf{some } \delta\ \varphi & \text{for} & \lambda i \exists j[i[\delta]j \wedge \varphi(j)], \\
\textbf{all } \delta\ \varphi & \text{for} & \lambda i \forall j[i[\delta]j \rightarrow \varphi(j)].
\end{array}
$$

We have essentially mimicked the Tarski truth conditions for predicate logic in our object language and in fact it can be proved that, under certain conditions,[5] we can reason with terms generated in this way as if they were the predicate logical formulas they stand proxy for (see Muskens 1998).

It should be stressed that the technique discussed here can be used to embed any logic with a decent interpretation into classical logic. For example, (Muskens 1996) shows that we can use the same mechanism to embed Discourse Representation Theory (Kamp & Reyle 1993) into classical logic. In a fuller version of this paper we shall also present a version of LFTAG based on Discourse Representations.

[4] In this paper only quantification into S is considered, but in a fuller version we shall generalise this to quantification into arbitrary phrasal categories.

[5] The relevant condition is that in each term φ we are using in this way, and each pair $u(n), u(n')$ occurring in φ, with n and n' syntactically different, we must be justified to assume $n \neq n'$. In the application discussed below this condition is met automatically.

$$\sigma(r) = \textbf{all}\ u_{n_5}[man\ u_{n_5} \Rightarrow \textbf{some}\ u_{n_{18}}[woman\ u_{n_{18}}\ \&\ u_{n_5}\ loves\ u_{n_{18}}]] \vee$$
$$\sigma(r) = \textbf{some}\ u_{n_{18}}[woman\ u_{n_{18}}\ \&\ \textbf{all}\ u_{n_5}[man\ u_{n_5} \Rightarrow\ u_{n_5}\ loves\ u_{n_{18}}]]$$

Figure 2: A Derivable Disjunction

3 Semantic Composition

We can now integrate semantic equations with the lexical items occurring in fig. 1.

$$\begin{aligned}
\sigma(n_3) &= u_{n_5} \\
\sigma(n_1) &= \textbf{all}\ u_{n_5}[\sigma(n_6)(u_{n_5}) \Rightarrow \sigma(n_2)] \\
\sigma(n_{10}) &= \lambda v.v\ loves\ \sigma(n_{13}) \\
\sigma(n_7) &= \sigma(n_9)(\sigma(n_8)) \\
\sigma(n_{16}) &= u_{n_{18}} \\
\sigma(n_{14}) &= \textbf{some}\ u_{n_{18}}[\sigma(n_{19})(u_{n_{18}})\ \&\ \sigma(n_{15})] \\
\sigma(n_{21}) &= \lambda v.man\ v \\
\sigma(n_{22}) &= \lambda v.woman\ v
\end{aligned}$$

The first two equations derive from the lexical item for *every*, the third and fourth from *loves*, the fifth and sixth from *a*, and the last two from the common nouns. Note that in the translation of *every*, n_3 only gets a referent as its translation (namely $u(n_5)$, which for readability we write as u_{n_5}), while the real action is taking place upstairs. A similar remark holds for the other determiner.

As we have seen earlier, in any model of the relevant descriptions $n_6 = n_{21}$, $n_{19} = n_{22}$, $n_9 = n_{10}$, $n_8 = n_3$, and $n_{13} = n_{16}$ hold. From this it follows that

$$\begin{aligned}
\sigma(n_7) &= u_{n_5}\ loves\ u_{n_{18}} \\
\sigma(n_1) &= \textbf{all}\ u_{n_5}[man\ u_{n_5} \Rightarrow \sigma(n_2)] \\
\sigma(n_{14}) &= \textbf{some}\ u_{n_{18}}[woman\ u_{n_{18}}\ \&\ \sigma(n_{15})]
\end{aligned}$$

The relevant constraints further imply that either $n_2 = n_{14}$ and $n_{15} = n_7$, or, alternatively, that $n_{15} = n_1$ and $n_2 = n_7$. For the moment let us assume the second possibility. Since u_{n_5} *loves* $u_{n_{18}}$ is a *closed term* (u is a function constant and n_5 and n_{18} are constants that witness existential quantifiers in the input description of (5)), the assumption that $n_2 = n_7$ allows us to conclude that

$$\sigma(n_1) = \textbf{all}\ u_{n_5}[man\ u_{n_5} \Rightarrow u_{n_5}\ loves\ u_{n_{18}}]$$

Note that this is the point where we have made essential use of our internalisation of binding: had we used ordinary variables instead of our register-denoting terms, the substitution would not have been possible.

Continuing our reasoning, we see that nnder the given assumption the root node r ($=n_{14}$ in this case) will be assigned the $\exists\forall$ reading of the sentence. Without assumptions the disjunction in fig. 2 is derivable.

We conclude that the leading idea behind Marcus' Description Theory allows us to underspecify semantic information much in the same way as syntactic information is underspecified in this theory. The price is that we must accept that different semantic readings correspond to different structures, as the method only allows underspecification of the latter.

References

Backofen, R., J. Rogers and K. Vijay-Shanker. 1995. A First-Order Axiomatization of the Theory of Finite Trees. *Journal of Logic, Language and Information* 4:5-39.

Cooper, R. 1983. *Quantification and Syntactic Theory.* Reidel. Dordrecht.

Cornell, T. 1994. On Determining the Consistency of Partial Descriptions of Trees. *Proceedings of ACL 94.* 163-170.

Joshi, A., L. Levy and M. Takahashi. 1975. Tree Adjunct Grammars. *Journal of the Computer and System Sciences* **10**: 136-163.

Kamp H. and U. Reyle. 1993. *From Discourse to Logic.* Kluwer, Dordrecht.

Marcus, M., D. Hindle and M. Fleck. 1983. D-theory: Talking about Talking about Trees. *Proceedings of the 21st ACL.* 129–136.

May, R. 1977. *The Grammar of Quantification*, PhD thesis, MIT, Cambridge.

Muskens, R. 1995. Order-Independence and Underspecification. In J. Groenendijk, editor, *Ellipsis, Underspecification, Events and More in Dynamic Semantics.* DYANA Deliverable R.2.2.C, 1995.

Muskens, R. 1996. Combining Montague Semantics and Discourse Representation. *Linguistics and Philosophy* **19**: 143–186.

Muskens, R. 1998. Underspecified Semantics. Manuscript. Tilburg University.

Schabes, Y. 1990. *Mathematical and Computational Aspects of Lexicalized Grammars*, Ph.D. thesis, University of Pennsylvania.

Vijay-Shanker, K. 1992. Using Descriptions of Trees in a Tree Adjoining Grammar. *Computational Linguistics* **18**:481–518.

Prefix Probabilities for Linear Indexed Grammars

Mark-Jan Nederhof
DFKI
Saarbrucken, Germany
nederhof@dfki.de

Anoop Sarkar
Dept. of Computer and Information Science
University of Pennsylvania
anoop@linc.cis.upenn.edu

Giorgio Satta
Dip. di Elettronica e Informatica
Università di Padova
satta@dei.unipd.it

Abstract

We show how prefix probabilities can be computed for stochastic linear indexed grammars (SLIGs). Our results apply as well to stochastic tree-adjoining grammars (STAGs), due to their equivalence to SLIGs.

1 Introduction

The problem of computing prefix probabilities for stochastic context-free languages is defined as follows. Given a word sequence $a_1 \cdots a_n$ over some alphabet Σ, which we call the input prefix, we must compute quantity $\sum_{w \in \Sigma^*} \Pr(a_1 \cdots a_n w)$. This problem has been discussed in [1, 4] with the main motivation of applications in speech recognition, where we are given some word sequence $a_1 \cdots a_{n-1}$, and must hypothesize the next word a_n.

The main idea leading to the solution of this problem is that all parts of context-free derivations that are potentially of unbounded size are captured into a set of equations that can be solved "off-line", i.e., before a specific prefix is considered. This is possible because the involved derivations do not depend on the given prefix. Once these equations have been solved, the results are stored. When computing the prefix probability for an actual input string, all possible derivations are then considered and a probability is computed, but for certain parts of these derivations the results that were computed off-line are used, in such a way that the computation is guaranteed to terminate.

Cases of derivations of potentially unbounded size might arise because of so called *unit rules*, i.e., rules of the form $A \rightarrow B$. Such rules potentially cause the grammar to be cyclic, which means that $A \rightarrow^* A$ might hold for some nonterminal A. This allows certain strings to have derivations of unbounded size. However, also a rule of e.g. the form $A \rightarrow Ba$ may effectively behave like a unit rule if a contributes to the unknown suffix following the actual input that is considered as prefix.

For stochastic tree-adjoining grammars (STAGs) similar problems arise. STAGs that are well-behaved and allow a bounded number of derivations for each complete sentence may require an unbounded number of derivations to be considered, once the input is regarded as a prefix followed by a suffix of unbounded length. The key idea to solving this problem is again to break up derivations into parts that are of potentially unbounded size and are

independent on actual input, and parts that are always of bounded length and do depend on input symbols. The probabilities of the former subderivations can be computed off-line, and the results are combined with subderivations of the latter kind during computation of the prefix probability for a given string.

The distinction between the two kinds of subderivations requires a certain notational system that is difficult to define for tree-adjoining grammars. We will therefore concentrate on stochastic linear indexed grammars instead, relying on their equivalence to STAGs [3]. The solution proposed in the present paper is an alternative to a different approach by the same authors in [2]. In that publication, a set of equations is transformed in order to distinguish off-line and on-line computations.

2 Computation of prefix probabilities

We refer the reader to [2] for the definition of LIG. In what follows, we use $\alpha, \beta, \ldots$ to denote strings of nonterminals associated with empty stacks of indices, $x, y, v, w, z, \ldots$ to denote strings of terminal symbols, and a to denote a terminal symbol. Without loss of generality we require that rules are of the form $A[\eta \circ\circ] \rightarrow \alpha\ B[\eta' \circ\circ]\ \beta$ with $|\eta\eta'| = 1$, or of the form $A[\,] \rightarrow z$, where $|z| \leq 1$.

As usual, $\rightarrow$ is extended to a binary relation between sentential forms, and its transitive and reflexive closure is denoted by $\rightarrow^*$. When we write $A[\sigma] \rightarrow^* \alpha\ B[\tau]\ \beta$, the indicated occurrence of $B[\tau]$ is the symbol that inherits the stack content of $A[\sigma]$ in the derivation, which we will call the *distinguished descendant* of $A[\sigma]$. We extend this notation to $A[\sigma] \rightarrow^* \alpha\ a\ \beta$, when a is generated in one step from the distinguished descendant of $A[\sigma]$ in a previous sentential form.

We first introduce a subrelation of $\rightarrow^*$ defined by $A[\sigma] \Rightarrow^* \varepsilon$ if $A[\sigma] \rightarrow^* \varepsilon$, and $A[\sigma] \Rightarrow^* B[\tau]$ if $A[\sigma] \rightarrow^* B[\tau]$ and this derivation does not end on a subderivation of the form $C[\tau] \rightarrow^+ B[\tau]$, for any C, where no elements that belong to τ are popped and pushed again. When we write $A[\sigma] \Rightarrow^* X$, then X is of the form $B[\,]$ or ε.

Based on this, two further subrelations of relation $\rightarrow^*$, written $\rightarrow^*_{ver}$ and $\rightarrow^*_{hor}$, are defined below by means of deduction steps. The distinction between $\rightarrow^*_{ver}$ and $\rightarrow^*_{hor}$ is made in order to record how derivations were built up from subderivations. In the case of $\rightarrow^*_{hor}$, the derivation was constructed from two subderivations $A[\,] \rightarrow^* v\ B[\,]\ w$ and $B[\,] \rightarrow^* x\ C[\,]\ y$. In all other cases, we use $\rightarrow^*_{ver}$. This distinction is needed to avoid spurious ambiguity in applications of the deduction steps: the result from combining $A[\,] \rightarrow^* v\ B[\,]\ w$ and $B[\,] \rightarrow^* x\ C[\,]\ y$, viz. $A[\,] \rightarrow^*_{hor} v\ x\ C[\,]\ y\ w$, is not allowed to combine with a third subderivation $C[\,] \rightarrow^* z\ D[\,]\ q$. Note that the desired derivation $A[\,] \rightarrow^*_{hor} v\ x\ z\ D[\,]\ q\ y\ w$ can be derived by combining $B[\,] \rightarrow^* x\ C[\,]\ y$ and $C[\,] \rightarrow^* z\ D[\,]\ q$, and then $A[\,] \rightarrow^* v\ B[\,]\ w$ with the result of this.

$$\frac{}{\varepsilon \rightarrow^*_{ver} \varepsilon} \qquad (1) \qquad \frac{A[\,] \rightarrow^*_{ver} v \quad \alpha \rightarrow^*_{ver} w \quad \alpha \neq \varepsilon}{A[\,]\ \alpha \rightarrow^*_{ver} v\ w} \qquad (2)$$

$$\frac{A[\,] \rightarrow^* a}{A[\,] \rightarrow^*_{ver} a} \qquad (3)$$

$$\frac{\begin{array}{ll} A[\,] \rightarrow^* B[\sigma] & \alpha \rightarrow^*_{ver} v_\alpha \\ B[\circ\circ] \rightarrow \alpha\ C[p \circ\circ]\ \beta & \beta \rightarrow^*_{ver} v_\beta \\ C[\,] \rightarrow^* D[\,] & \gamma \rightarrow^*_{ver} v_\gamma \\ D[p \circ\circ] \rightarrow \gamma\ E[\circ\circ]\ \delta & \delta \rightarrow^*_{ver} v_\delta \\ E[\sigma] \Rightarrow^* X & v_\alpha v_\beta v_\gamma v_\delta \neq \varepsilon \end{array}}{A[\,] \rightarrow^*_{ver} v_\alpha\ v_\gamma\ X\ v_\delta\ v_\beta} \qquad (4)$$

$$\frac{\begin{array}{ll} A[\,] \rightarrow^* B[\sigma] & lab \in \{ver, hor\} \\ B[\,\circ\circ\,] \rightarrow \alpha\ C[p\circ\circ\,]\ \beta & \alpha \rightarrow^*_{ver} v_\alpha \\ C[\,] \rightarrow^*_{lab} v\ D[\,]\ w & \beta \rightarrow^*_{ver} v_\beta \\ D[\,] \rightarrow^* E[\,] & \gamma \rightarrow^*_{ver} v_\gamma \\ E[p\circ\circ\,] \rightarrow \gamma\ F[\,\circ\circ\,]\ \delta & \delta \rightarrow^*_{ver} v_\delta \\ F[\sigma] \Rightarrow^* X & v_\alpha v_\beta v_\gamma v_\delta \neq \varepsilon \end{array}}{A[\,] \rightarrow^*_{ver} v_\alpha\ v\ v_\gamma\ X\ v_\delta\ w\ v_\beta} \quad (5)$$

$$\frac{\begin{array}{l} A[\,] \rightarrow^*_{ver} v\ B[\,]\ w \\ B[\,] \rightarrow^*_{lab} x\ C[\,]\ y \quad lab \in \{ver, hor\} \end{array}}{A[\,] \rightarrow^*_{hor} v\ x\ C[\,]\ y\ w} \quad (6)$$

$$\frac{\begin{array}{l} A[\,] \rightarrow^*_{ver} v\ B[\,]\ w \\ B[\,] \rightarrow^*_{ver} x \end{array}}{A[\,] \rightarrow^*_{ver} v\ x\ w} \quad (7)$$

$$\frac{\begin{array}{l} A[\,] \Rightarrow^* B[\sigma] \\ B[\,] \rightarrow^*_{hor} v\ C[\,]\ w \\ C[\sigma] \Rightarrow^* X \quad \sigma \neq \varepsilon \end{array}}{A[\,] \rightarrow^*_{ver} v\ X\ w} \quad (8)$$

We now discuss how LIG derivations are uniquely partitioned into subderivations by the above steps. We will explain later how the above steps can be used in the computation of prefix probabilities. We call *spine* any path in the parse tree that leads from a node that is not a distinguished child of its father (or that does not have a father, in the case of the root), down to a leaf following distinguished children. This means that for an instance of a rule $A[\eta\circ\circ\,] \rightarrow \alpha\ B[\eta'\circ\circ\,]\ \beta$ in the parse tree, the nodes corresponding to symbols in α and β are each the first node of a distinct spine. Also, the spine belonging to the node which corresponds to $A[\eta\circ\circ\,]$ leads down along the node corresponding to $B[\eta'\circ\circ\,]$. At both ends of a spine, the stack of indices associated with the nonterminals is empty. In between, the height of the stack may alternately grow and shrink. This is shown in Figure 1. The horizontal axis represents nodes along the spine, and the vertical axis represents the height of the stack.

At some instances of rules, non-empty input is found at some child of a node on the spine that does itself not belong to the spine. We always investigate such rules in pairs: if one rule pushes p on the stack, we locate the unique rule that pops that p; only one of the two rules needs to be associated with non-empty input. Three instances of such pairs of rules are indicated in the figure.

In Figure 1, the parts of the spine labelled by a and b are accounted for by step (4). From these two parts, the part labelled c is obtained through step (6). This step combines paths

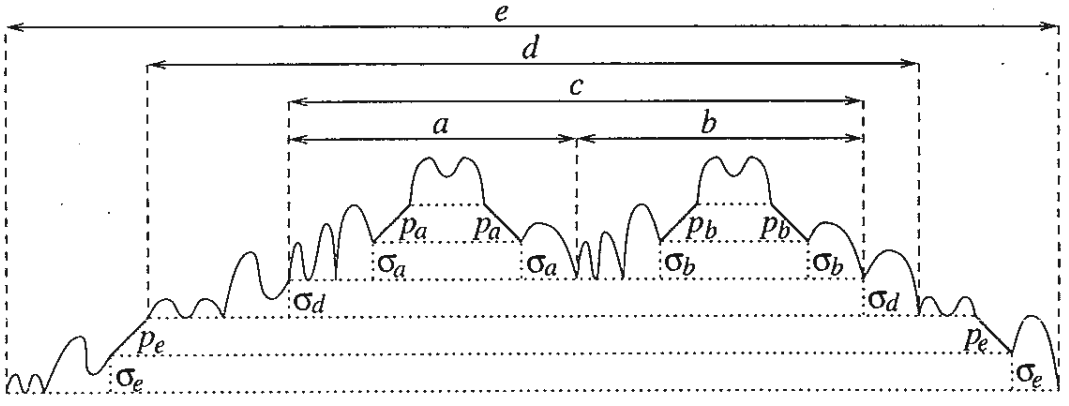


Figure 1: Development of the stack along a spine, and partitioning according to deduction steps.

in a "horizontal" way, hence the label *hor* in the consequent. The path is then extended to the path d in a vertical way by applying step (8). Again vertically, step (5) extends the path to path e by identifying one more pair of rules where non-empty input is found.

Each stack development along a spine, exemplified by Figure 1, can be partitioned in exactly one way according to the deduction steps. The proof of this fact is rather involved and is not reported in this long abstract.

We can now discuss how to compute prefix probabilities using steps (1) to (8). We can compute the inside probability of a given string w by applying the deduction steps in reverse for the relation $S[\,] \rightarrow^*_{ver} w$. This gives rise to a unique partitioning into subderivations for each possible derivation of w in the grammar. We multiply the probabilities attached to the rules that are used in the derivations, and we add probabilities where more than one derivation exists due to ambiguity.

We see that statements of the form $C[\,] \rightarrow^* D[\,]$ in e.g. step (4) and $A[\,] \rightarrow^* a$ in step (3) cannot themselves be derived by the deduction steps. It is assumed the probabilities of such derivations are computed off-line, which is possible since they do not depend on actual input. Also, the joint probability of the pair of derivations $A[\,] \rightarrow^* B[\sigma]$ and $E[\sigma] \Rightarrow^* X$ in step (4) can be precomputed for a given combination of A, B, E, and X, even though there may be an infinite number of stacks σ. These off-line computations can be carried out by solving systems of equations that express recursive relations among probabilities of derivations. Again, due to space limitations these systems will not be introduced in this long abstract.

It is easy to see that the backward application of the deduction steps must necessarily terminate. This is independent of whether a LIG allows infinite ambiguity.

If prefix probabilities are to be computed instead of inside probabilities, the deduction steps need to be slightly altered. For example, the condition $v_\alpha v_\beta v_\gamma v_\delta \neq \varepsilon$ in step (4) needs to be reformulated to the effect that at least one symbol from $v_\alpha v_\beta v_\gamma v_\delta$ should belong to the input, i.e. the prefix. Further, probabilities of derivations of the form $A[\,] \rightarrow^* B[\,] w$ should be computed off-line, where w belongs to the unknown suffix. (Cf. unit rules and rules of the form $A \rightarrow Ba$ in the case of context-free grammars.)

It is easy to see that the deduction steps are consistent, in the sense that $\alpha \rightarrow^*_{ver} \beta$ or $\alpha \rightarrow^*_{hor} \beta$ implies $\alpha \rightarrow^* \beta$. That the deduction steps are also complete, i.e., that $A[\,] \rightarrow^*_{ver} w$ can be derived if $A[\,] \rightarrow^* w$, is more difficult to show and cannot be explained here due to length restrictions. The proof relies on the already mentioned uniqueness of the proposed partitioning of spines, on which steps (1) to (8) are based.

References

[1] F. Jelinek and J.D. Lafferty. Computation of the probability of initial substring generation by stochastic context-free grammars. *Computational Linguistics*, 17(3):315–323, 1991.

[2] M.-J. Nederhof, A. Sarkar, and G. Satta. Prefix probabilities from stochastic tree adjoining grammars. In *36th Annual Meeting of the ACL, Proceedings of the Conference*, Montreal, Canada, August 1998. To appear.

[3] Y. Schabes. Stochastic lexicalized tree-adjoining grammars. In *Proc. of the fifteenth International Conference on Computational Linguistics*, volume 2, pages 426–432, Nantes, August 1992.

[4] A. Stolcke. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics*, 21(2):167–201, 1995.

Automatic Extraction of Stochastic Lexicalized Tree Grammars from Treebanks

Günter Neumann
DFKI
66123 Saarbrücken, Germany
neumann@dfki.de

Abstract

We present a method for the extraction of stochastic lexicalized tree grammars (S-LTG) of different complexities from existing treebanks, which allows us to analyze the relationship of a grammar automatically induced from a treebank wrt. its size, its complexity, and its predictive power on unseen data.

Processing of different S-LTG is performed by a stochastic version of the two-step Early-based parsing strategy introduced in (Schabes and Joshi, 1991).

1 Introduction

In this paper we present a method for the extraction of stochastic lexicalized tree grammars (S-LTG) of different complexities from existing treebanks, which allows us to analyze the relationship of a grammar automatically induced from a treebank wrt. its size, its complexity, and its predictive power on unseen data. The use of S-LTGs is motivated for two reasons. First, it is assumed that S-LTG better capture distributional and hierarchical information than stochastic CFG (cf. (Schabes, 1992; Schabes and Waters, 1996)), and second, they allow the factorization of recursion of different kinds, viz. extraction of left, right, and wrapping auxiliary trees and possible combinations. Existing treebanks are used because they allow a corpus-based analysis of grammars of realistic size. Processing of different S-LTG is performed by a stochastic version of the two-phase Early-based parsing strategy introduced in (Schabes and Joshi, 1991).

This abstract describes work in progress. So far, we have concentrated on the automatic extraction of S-LTGs of different kinds (actually S-LTSG, S-LTIG, and S-LTAG). This phase is completed and we will report on first experiments using the Penn-Treebank (Marcus et al., 1993) and Negra, a treebank for German (Skut et al., 1997). A first version of the two-phase parser is implemented, and we have started first tests concerning its performance.

2 Grammar extraction

Given a treebank, grammar extraction is the process of decomposing each parse tree into smaller units called subtrees. In our approach, the underlying decomposition operation

1. should yield lexically anchored subtrees, and
2. should be guided by linguistic principles.

The motivation behind (1) is the observation that in practice stochastic CFG perform worse than non-hierarchical approaches, and that lexicalized tree grammars may be able to capture both distributional and hierarchical information (Schabes and Waters, 1996). Concerning (2) we want to take advantage of the linguistic principles explicitly or implicitly used to define a treebank. This is motivated by the hypothesis that it will better support the development of on-line or incremental learning strategies (the cutting criteria are less dependent from the quantity and quality of the existing treebank than purely statistically based approaches, see also sec. 5) and that it renders possible a comparison of an induced grammar with a linguistically based competence grammar. Both aspects (but especially the latter one) are of importance because it is possible to apply the same learning strategy also to a treebank computed by some competence grammar, and to investigate methods for combining treebanks and competence grammars (see sec. 6).

However, in this paper we will focus on the use of existing treebanks using the Penn-Treebank (Marcus et al., 1993) and Negra, a treebank for German (Skut et al., 1997). First, it is assumed that the

treebank comes with a notion of lexical and phrasal head, i.e., with a kind of *head principle* (see also (Charniak, 1997)). In the Negra treebank, head elements are explicitly tagged. For the Penn treebank, the head relation has been determined manually. In case it is not possible to uniquely identify one head element there exists a parameter called DIRECTION which specifies whether the left or right candidate should be selected. Note that by means of this parameter we can also specify whether the resulting grammar should prefer a left or right branching.

Using the head information, each tree from the treebank is decomposed from the top downwards into a set of subtrees, such that each non-terminal non-headed subtree is cut off, and the cutting point is marked for substitution. The same process is then recursively applied to each extracted subtree. Due to the assumed head notion each extracted tree will automatically be lexically anchored (and the path from the lexical anchor to the root can be seen as a head-chain). Furthermore, every terminal element which is a sister of a node of the head-chain will also remain in the extracted tree. Thus, the yield of the extracted tree might contain several terminal substrings, which gives interesting patterns of word or POS sequences. For each extracted tree a frequency counter is used to compute the probability $p(t)$ of a tree t, after the whole treebank has been processed, such that $\sum_{t:root(t)=\alpha} p(t) = 1$, where α denotes the root label of a tree t.

After a tree has been decomposed completely we obtain a set of lexicalized elementary trees where each nonterminal of the yield is marked for substitution. In a next step the set of elementary trees is divided into a set of initial and auxiliary trees. The set of auxiliary trees is further subdivided into a set of left, right, and wrapping auxiliary trees following (Schabes and Waters, 1995) (using special foot note labels, like :lfoot, :rfoot, and :wfoot). Note that the identification of possible auxiliary trees is strongly corpus-driven. Using special foot note labels allows us to trigger carefully the corresponding inference rules. For example, it might be possible to treat the :wfoot label as the substitution label, which means that we consider the extracted grammar as a S-LTIG, or only highly frequent wrapping auxiliary trees will be considered. It is also possible to treat every foot label as the substitution label, which means that the extracted grammar only allows substitution.

3 Two-phase parsing of S-LTG

The resulting S-LTG will be processed by a two-phase stochastic parser along the line of (Schabes and Joshi, 1991). In a first step the input string is used for retrieving the relevant subset of elementary trees. Note that the yield of an elementary tree might consist of a sequence of lexical elements. Thus in order to support efficient access, the deepest left-most chain of lexical elements is used as index to an elementary tree. Each such index is stored in a decision tree. The first step is then realized by means of a recursive tree traversal which identifies all (longest) matching substrings of the input string (see also sec. 4). Parsing of lexically triggered trees is performed in the second step using an Earley-based strategy. In order to ease implementation of different strategies, the different parsing operations are expressed as inference rules and controlled by a chart-based agenda strategy along the line of (Shieber et al., 1995). So far, we have implemented a version for running S-LTIG which is based on (Schabes and Waters, 1995). The inference rules can be triggered through boolean parameters, which allows flexible hiding of auxiliary trees of different kinds.

4 First experiments

We will briefly report on first results of our method using the Negra treebank (4270 sentences) and the section 02, 03, 04 from the Penn treebank (the first 4270 sentences). In both cases we extracted three different versions of S-LTG (note that no normalization of the treebanks has been performed): (a) lexical anchors are words, (b) lexical anchors are part-of-speech, and (c) all terminal elements are substituted by the constant :term, which means that lexical information is ignored. For each grammar we report the number of elementary trees, left, right, and wrapping auxiliary trees. The following table summarizes the results:

Negra	**words**	**pos**	**:term**
elem. trees:	26553	10384	6515
leftaux trees	184	60	40
rightaux trees	54	35	25
wrapping trees	39	36	29
Penn	**words**	**pos**	**:term**
elem. tree:	31944	11979	8132
leftaux trees	701	403	293
rightaux trees	649	246	153
wrapping trees	386	306	249

In a second experiment we evaluated the performance of the implemented S-LTIG parser using the extracted Penn treebank with words as lexical anchors. We applied all sentences on the extracted grammar and computed the following average valnes for the first phase: sentence length: 27.54, number

of matching snbstrings: 15.93, number of elementary trees: 492.77, number of different root labels: 33.16. The average run-time for each sentence (measnred on a Sun Ultra 2 (200 mhz): 0.0231 sec. In a next step we tested the run-time behaviour of the whole parser on the same input, however ignoring every parse which took longer than 30 sec. (about 20 %). The average run-time for each sentence (exhaustive mode): 6.18 sec. This is promising, since the parser is still not optimized.

We also tried first blind tests, but it turned ont that the current considered size of the treebanks is too small to get reliable results on unseen data (randomly selecting 10 % of a treebank for testing; 90 % for training). The reason is that if we consider only words as anchors then we rarely get a complete parse result (around 10 %). If we consider only POS then the number of elementary trees retrieved through the first phase increases causing the current parser prototype to be slow (due to the restricted annotation schema).[1] A better strategy seems to be the use of words only for lexical anchors and POS for all other terminal nodes, or to use only closed-class words as lexical anchors (assuming a head principle based on functional categories). In that case it would also be possible to adapt the strategies described in (Srinivas, 1997) wrt. supertagging in order to reduce the set of retrieved trees before the second phase is called.

5 Related work

Here we will discuss alternative approaches for converting treebanks into lexicalized tree grammars, namely the Data-oriented Parsing (DOP) framework (Bod, 1995) and approaches based on applying Explanation-based Learning (EBL) to NL parsing (e.g., (Samuelsson, 1994; Srinivas, 1997)).

The general strategy of our approach is similar to DOP with the notable distinction that in our framework all trees must be lexically anchored and that in addition to substitution, we also consider adjunction and restricted versions of it. In the EBL approach to NL parsing the core idea is to use a competence grammar and a training corpus to construct a treebank. The treebank is then used to obtain a specialized grammar which can be processed much faster than the original one at the price of a small loss in coverage. Samuelsson (1994) presents a method in which tree decomposition is completely automatized using the information-theoretical concept of entropy, after the whole treebank has been indexed in an and-or tree. This implies that a new grammar has to be computed if the treebank changes (i.e., reduced incrementallity) and that the generality of the induced subtrees depends much more on the size and variation of the treebank than ours. On the other side, this approach seems to be more sensitive to the distribution of sequences of lexical anchors than our approach, so that we will explore its integration.

In (Srinivas, 1997) the application of EBL to parsing of LTAG is presented. The core idea is to generalize the derivation trees generated by an LTAG and to allow for a finite state transducer representation of the set of generealized parses. The POS sequence of a training instance is used as the index to a generalized parse. Generalization wrt. recursion is achieved by introducing the Kleene star into the yield of an auxiliary tree that was part of the training example, which allows generalization about the length of the training sentences. This approach is an important candidate for improvements of our two-phase parser once we have acquired an S-LTAG.

6 Future steps

The work described here is certainly in its early phase. The next future steps (partly already started) will be: (1) measuring the coverage of an extracted S-LTG, (2) incremental grammar induction, (3) combination of a competence grammar and a treebank. I already applied the same learning strategy on derivation trees obtained from a large HPSG-based English grammar in order to speed up parsing of HPSG (extending the work described in (Neumann, 1994)). Now I am exploring methods for merging such an "HPSG-based" S-LTG with one extracted from a treebank. The same will also be explored wrt. a competence-based LTAG, like the one which comes with the XTAG system (Doran et al., 1994).

7 Acknowledgment

The research underlying this paper was supported by a research grant from the German Bundesministerium für Bildung, Wissenschaft, Forschung und Technologie (BMBF) to the DFKI project PARADIME, FKZ ITW 9704. I would like to thank Tilman Becker for many fruitful discussions.

[1] Applying the same test as described above on POS, the average number of elementary trees retrieved is 2292.86, i.e., the number seems to increase by a factor of 5.

References

R. Bod. 1995. *Enriching Linguistics with Statistics: Performance Models of Natural Language.* Ph.D.

thesis, University of Amsterdam. ILLC Dissertation Series 1995-14.

E. Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *AAAI-97*, Providence, Rhode Island.

C. Doran, D. Egedi, B. Hockey, B. Srinivas, and M. Zeidel. 1994. Xtag system - a wide coverage grammar for english. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING)*, Kyoto, Japan.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19:313–330.

G. Neumann. 1994. Application of explanation based learning for efficient processing of constraint-based grammars. In *Proceedings of the Tenth IEEE Conference on Artifical Intelligence for Applications*, pages 208–215, San Antonio, Texas, March.

C. Samuelsson. 1994. Grammar specialization through entropy thresholds. In *Proceedings of the 32nd Annual Meeting of the Association forComputational Linguistics*, pages 188–195.

Y. Schabes and A. K. Joshi. 1991. Parsing with lexicalized tree adjoining grammar. In M. Tomita, editor, *Current Issues in Parsing Technology*, pages 25–48. Kluwer, Boston.

Y. Schabes and R. 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.

Y. Schabes and R. Waters. 1996. Stochastic lexicalized tree-insertion grammar. In H. Bunt and M. Tomita, editors, *Recent Advances in Parsing Technology*, pages 281–294. Kluwer Academic Press, London.

Y. Schabes. 1992. Stochastic lexicalized tree-adjoining grammars. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING)*, pages 426–432, Nantes.

S. Shieber, Y. Schabes, and F. Pereira. 1995. Principles and implementation of deductive parsing. *Journal of Logic and Computation*, 24:3–36.

W. Skut, B. Krenn, T. Brants, and H. Uszkoreit. 1997. An annotation scheme for free worder order languages. In *5th International Conference of Applied Natural Language*, pages 88–94, Washington, USA, March.

B. Srinivas. 1997. *Complexity of Lexical Restrictions and Its Relevance to Partial Parsing.* Ph.D. thesis, University of Pennsylvania. IRCS Report 97–10.

Memoisation in Sentence Generation with Lexicalised Grammars

Nicolas Nicolov
Dept of Artificial Intelligence
Univ. of Edinburgh
Edinburgh EH1 1HN, UK
nicolas@dai.ed.ac.uk

Abstract

This paper discusses a sentence generation system PROTECTOR which uses: (i) a non-hierarchical semantic representation which allows for flexible lexical choice and uniform treatment of different languages, (ii) a lexicalised D-Tree Grammar which is very similar to Tree-Adjoining Grammar in spirit, and (iii) dynamic programming techniques to avoid doing redundant computations. We review the motivation for choosing such an organisation of the generation system and give an example of the generation of a sentence which involves a lexical gap. The generation of the example sentence requires a non-deterministic mode of computation (the lexical gap forcing backtracking). We show how dynamic programming techniques can be used to save re-generating structures using a top-down generation algorithm.

Keywords: natural language generation, non-hierarchical semantics, lexicalised d-tree grammars, dynamic programming.

1 Introduction

Natural language generation is the process of generating text from a set of abstract communicative goals. It attempts to model the human language production mechanisms in man-machine communication. As part of the overall generation process computer systems will need to consider how the communicative goals can be mapped onto conceptual representations and these in turn into sentences in a natural language. The latter process is known as sentence generation and this paper discusses a system for doing this task (realising sentences from meaning representations).

2 Conceptual input

Early work on sentence generation assumed input of the form: $\texttt{pred}(\texttt{arg}_1, \ldots \texttt{arg}_n)$ and the generation process was reduced to mapping $\texttt{pred} \rightarrow$ *verb*, $\texttt{arg}_1 \rightarrow$ first *complement*, etc. This approach, of course, makes the "semantic structures" be nothing more than disguised syntactic representations and reduces the sentence generation problem to finding out the ordering of the constituents. The tree-like semantic assumption does not allow for handling head switching examples (Nicolov, 1993), incorporation of modifiers in the syntactic head (*French blond* and *blond French girl* cannot be generated from `french(blond(girl))`) and cases like: *She smiled a welcome to the guests./ She welcomed the guests with a smile.*

Such phenomena can be addressed more elegantly using non-hierarchical semantic representations. In PROTECTOR conceptual graphs are used (Sowa, 1992). The same generation mechanisms can be used with underspecified discourse representation structures.

3 D-Tree Grammars

D-Tree Grammar (Rambow et al., 1995) is a grammar formalism which arises from work on Tree-Adjoining Grammars (TAG) (Joshi, 1987).[1] In the context of generation, TAGs have been used in a number of systems MUMBLE (McDonald and Pustejovsky, 1985), SPOKESMAN (Meteer, 1990), WIP (Wahlster et al., 1991), synchronous TAGs (Shieber and Schabes, 1991) the system reported by McCoy (McCoy et al., 1992), the first version of PROTECTOR (Nicolov et al., 1995), and SPUD (Stone and Doran, 1997). TAGs have been given a prominent place in the VERBMOBIL project — they have been chosen to be the framework for the generation module (Caspari and Schmid, 1994; Harbusch et al., 1994; Becker et al., 1998). In the area of grammar development TAG has been the basis of one of the largest grammars developed for English (Doran et al., 1994).

[1] DTG and TAG are very similar, yet they are not equivalent (Weir pc).

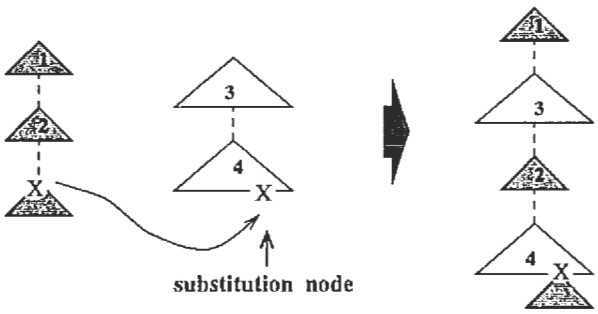


Figure 1: Subsertion

DTGs uses two operations to combine elementary structures — subsertion (Figure 1) and sister adjunction (Figure 2). The elementary structures are d-trees (descriptions of trees) which in addition to immediate dominance relation allow for stating dominance relationships between nodes in the d-tree.

Unlike TAGs, DTGs provide a uniform treatment of complementation and modification at the syntactic level. DTGs are seen as attractive for generation because a close match between semantic and syntactic operations leads to simplifications in the overall generation architecture. DTGs try to overcome the problems associated with TAGs while remaining faithful to what is seen as the key advantages of TAGs (Joshi, 1987):

1. the extended domain of locality over which syntactic dependencies are stated; and
2. function argument structure is captured within a single initial construction in the grammar.

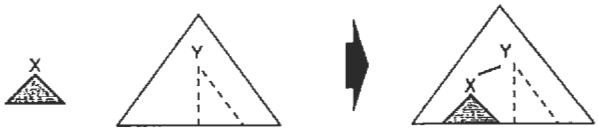


Figure 2: Sister adjunction

We use a lexicalised (every elementary structure contains a terminal node (anchor) which 'justifies' the construction), feature-based (nonterminals are feature structures) DTG.

4 Generation strategy

PROTECTOR uses declarative specification of the relation between semantics and syntax encoded as mapping rules. The mapping rules are elementary d-trees (i.e., tree descriptions) annotated with applicability semantics a match with which will licence the applicability of the mapping rule. In addition if the d-tree has nonterminal leaf nodes relevant parts of the applicability semantics are related to these nodes so that we know how the semantics is decomposed. PROTECTOR employs a top-down (recursive descent) strategy for generating the complements once an initial top-level mapping rule has been chosen (this stage is called generation of skeletal structure). PROTECTOR keeps track how much of the input semantics it has consumed. Then in a consequent stage the remaining semantics is consumed which involves the use of modification and sister-adjunction.

5 Example

In this section we discusss the generation of a sentence which involves a lexical gap:

> **Alexander attacked the town 'full-scalely'.*
> *Alexander launched a full-scale attack on the town.*

The input semantics and the search space are are shown in Figure 3 (see next page). At the onset of generation there are at least two top-level mapping rules that can be chosen (*attack* and *launch an attack*) and the default one (*attack*) leads to a dead end. The reason is the lack of a mapping rule (not only in the linguistic knowledge base of the generator but worse of all in the English language) that would allow us to express the concept FULL-SCALE as a structure that we can intergrate to the existing skeletal syntactic structure (*Alexander attacked the town*). Such is the nature of lexical gaps and this forces backtracking. The generator would need to reconsider its previous decisions, it would have to undo (forget) about all the structures it had built all the way up to the point when it chose the wrong mapping rule. This was the first choice that was made so practically every computation is lost. All the work that went into building the subject and object NPs has to be duplicated. Choosing the alternative (*launch an attack*) mapping rule

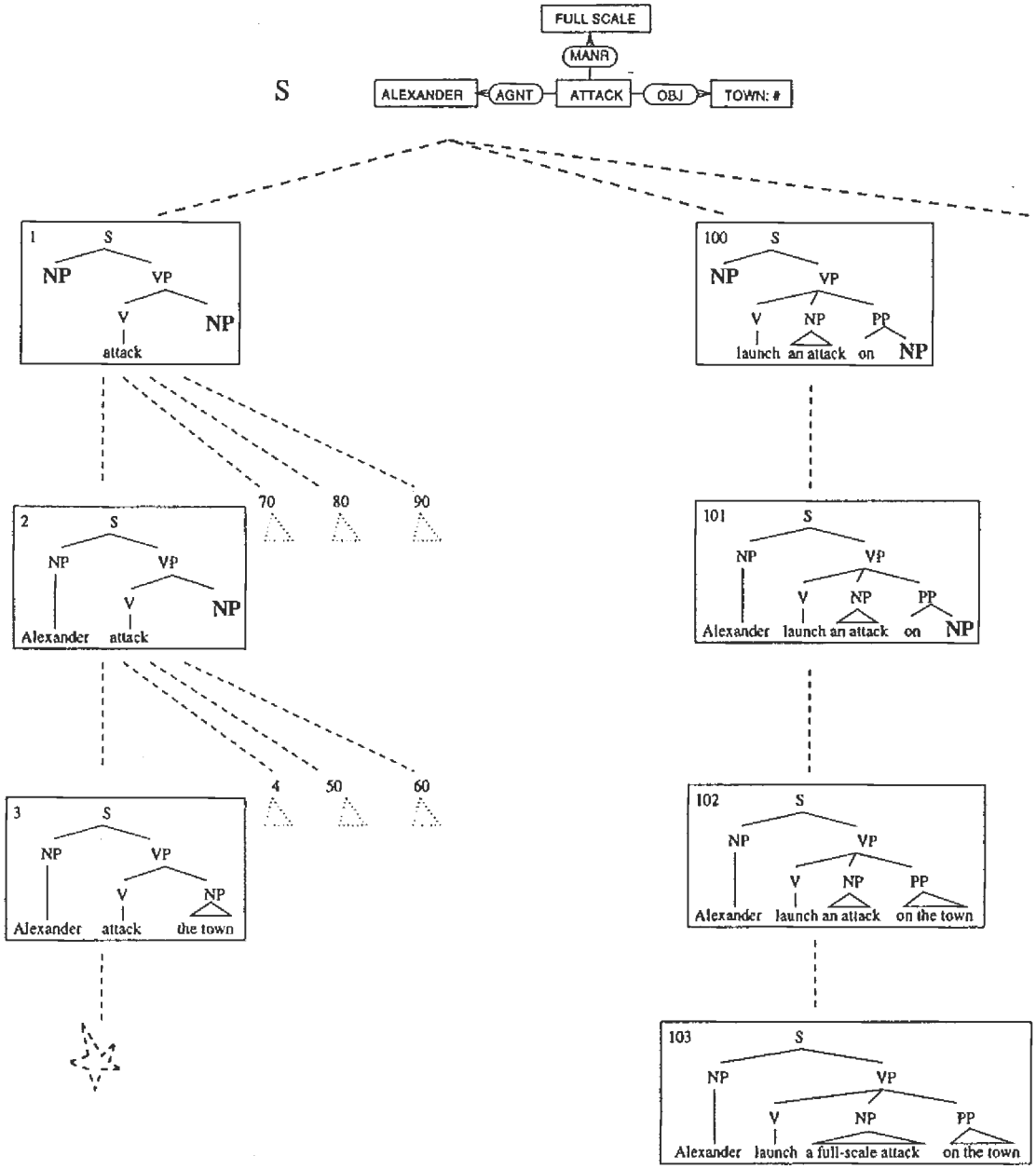


Figure 3: The search space for the example

and generating its required complements will result in re-computation of the subject and object NPs. These NPs can be arbitrarily large and in order to avoid doing redundant computations we store the results of previous generation goals and reuse them if needed again. Such dynamic programming techniques have been exploited heavily in parsing and PROTECTOR's declarative mapping rules and flexibility of incorporating alternative generation strategies allows us to take advangates of that work. This approach is gaining popularity in generation (Shieber, 1988; Haruno et al., 1993; Pianesi, 1993; Gerdemann and Hinrichs, 1995; Kay, 1996; Nicolov et al., 1997). The other approaches to chart generation are based on CFGs and in a bottom-up strategy one has to make sure that in moving from an $\bar{N}$ to NP all modifiers have been ex-

pressed. This causes serious overhead in backtracking. Our use of DTGs and flexible way of adding modifiers using precedence constraints between semantic classes of modifiers does not suffer from this problem.

PROTECTOR does not assume that lexical choice is performed prior to surface realisation. It chunks the input semantics appropriately on the basis of the mapping rules.

6 Conclusions

We have described a sentence generator which takes non-hierarchical input, uses mapping rules to relate parts of the semantics to elementary d-trees, combines the syntactic structures in a manner that closely mirrors the semantic decomposition and employs dynamic programming to avoid re-generation of structures on backtracking which cannot always be predicted in advance as is the case for lexical gaps. Our architecture allows for easy encoding of alternative generation strategies (e.g., bottom-up, best-first, etc.) which other systems have not considered and in fact find rather difficult to do. Thus, PROTECTOR can be seen as a test bed for experimenting and evaluating alternatives methods for generation.

References

Tilman Becker, Wolfgang Finkler, Anne Kilger, and Peter Poller. 1998. An efficient kernel for multilingual generation in speech-to-speech dialogue translation. In *Proceedings of COLING-ACL'98)*, Montreal, Canada.

Rudolf Caspari and Ludwig Schmid. 1994. Parsing und generierung in trug. Technical Report Verbmobil-Report 40, Siemens AG, December.

Christine Doran, Dania Egedi, Beth Ann Hockey, Bangalore Srinivas, and Martin Zaidel. 1994. XTAG — A Wide Coverage Grammar for English. In *Proceedings of the 15th Int. Conference on Computational Linguistics (COLING-94)*, pages 922–928, Kyoto, Japan, 5-9 August.

Dale Gerdemann and Erhard Hinrichs. 1995. Some open problems in head-driven generation. In Jerry Morgan, Georgia Green, and Jennifer Cole, editors, *Linguistics & Computation*, pages 65–197. CSLI Publications.

K. Harbusch, G. Kikui, and A. Kilger. 1994. Default handling in incremental generation. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING-94)*.

Masahiko Haruno, Yasuharu Den, Yuji Matsumoto, and Makoto Nagao. 1993. Bidirectional chart generation of natural language texts. In *Proceedings of the 11th National Conference on Artificial Intelligence (AAAI'93)*, pages 350–356, Menlo Park, CA. American Association for Artificial Intelligence, The MIT Press.

Aravind Joshi. 1987. The Relevance of Tree Adjoining Grammar to Generation. In Gerard Kempen, editor, *Natural Language Generation*, pages 233–252. Kluwer Academic, Dordrecht, The Netherlands.

Martin Kay. 1996. Chart generation. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL'96)*, pages 200–204, Santa Cruz, California.

Kathleen F. McCoy, K. Vijay-Shanker, and Gijoo Yang. 1992. A functional approach to generation with tag. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics(ACL'92)*, pages 48–55, University of Delaware, U.S. Association of Computational Linguistics.

David McDonald and James Pustejovsky. 1985. TAGs as a grammatical formalism for generation. In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, pages 94–103.

Marie Wenzel Meteer. 1990. *The "Generation Gap": The Problem of Expressibility in Text Planning*. Ph.D. thesis, Computer and Information Science Department, University of Massachusetts, February. COINS Technical Report 90-04.

Nicolas Nicolov, Chris Mellish, and Graeme Ritchie. 1995. Sentence Generation from Conceptual Graphs. In Gerard Ellis, Robert Levinson, William Rich, and John Sowa, editors, *LNAI 954, Conceptual Structures: Applications, Implementation and Theory*, pages 74–88. Springer, Berlin, 14–18 August. Proceedings of the 3rd Int. Conf. on Conceptual Structures (ICCS'95), Santa Cruz, CA, USA.

Nicolas Nicolov, Chris Mellish, and Graeme Ritchie. 1997. Approximate Chart Generation from Non-Hierarchical Representations. In Ruslan Mitkov & Nicolas Nicolov, editor, *Recent Advances in Natural Language Processing: Selected papers from RANLP'95*, Current Issues in Linguistic Theory (CILT), 136, pages 273–294. John Benjamins, Amsterdam & Philadelphia.

Nicolas Nicolov. 1993. *Head Selection in NLG*. DAI discussion paper 140, Dept. of Artificial Itelligence, Univ. of Edinburgh.

Fabio Pianesi. 1993. Head-driven bottom-up generation and Government and Binding: a unified perspective. In Helmut Horacek and Michael Zock, editors, *New Concepts in Natural Language Generation*, chapter 3, pages 187–214. Pinter, London.

Owen Rambow, K. Vijay-Shanker, and David Weir. 1995. D-Tree Grammars. In *Proceedings of the 33rd Meeting of the Association for Computational Linguistics (ACL'95)*, pages 151–158.

Stuart Shieber and Yves Schabes. 1991. Generation and synchronous tree-adjoing grammars. *Computational Intelligence*, 7(4):220–228. Special issue on Natural Language Generation.

Stuart M. Shieber. 1988. A uniform architecture for parsing and generation. In *COLING-88*, Budapest.

John F. Sowa. 1992. Conceptual graphs summary. In Timothy E. Nagle, Janice A. Nagle, Laurie L. Gerholz, and Peter W. Eklund, editors, *Conceptual Structures: Current Research and Practice*, Ellis Horwood Series in Workshops, pages 3–51. Ellis Horwood Limited, London, England.

Matthew Stone and Christine Doran. 1997. Sentence Planning as Description Using Tree Adjoining Grammar. In *Proceedings of the 35th Annual Meeting of the Association of Computational Linguistics(ACL'97)*, pages 198–205, Madrid, Spain.

Wolfgang Wahlster, Elisabeth André Son Bandyopadhyay, Winfried Graf, and Thomas Rist. 1991. WIP: the coordinated generation of multimodal presentations from a common representation. RR 91-08, DFKI.

Constructive models of extraction parameters[1]

Dick Oehrle
Linguistics & Cognitive Science
University of Arizona

Recursion as the basis of long-distance dependencies

An important and central insight of Tree Adjoining Grammar is its factorization of local dependencies—handled through local INITIAL TREES—and recursion—handled through AUXILIARY TREES and successive applications of the ADJUNCTION operation. Many different frameworks of grammatical description have converged on a conceptually similar distinction. In the transformational tradition, the idea of long-distance movement—movement across an 'essential variable'—has been abandoned in favor of sequences of short-distance hops (or checks). In feature-based phrase structure grammars such as GPSG and HPSG, the analog of recursive movement is the transitive closure of local consistency conditions on local trees containing the SLASH feature.

At first glance, then, this convergence in a variety of theoretical approaches suggests that recursion in some form is the essential engine in the characterization of natural language long-distance dependencies. And this assumption might lead us to the following thesis concerning the relation between recursion and extraction.

> *Thesis: if* $\Gamma[\alpha]$ *is a well-formed expression of category A containing a gap* α *of category B and* $\Delta[\beta]$ *is a well-formed expression of category B containing a gap* β *of category C, then the result of replacing the gap* α *in* $\Gamma[\alpha]$ *with* $\Delta[\beta]$*, which we write* $\Gamma[\Delta[\beta]]$ *is a well-formed expression of category A containing a gap* β *of category C.*

As an example of a case which might be adduced in support of this thesis, consider the unbounded nature of extraction from noun phrases, as discussed by Kroch [6]. The well-formedness of Which painting did you see? indicates that did you see is a well-formed expression containing a gap of type *np*, and the well-formedness of Which painting did you see a photograph of? and Which painting did you see a copy of? suggests (in a way consistent with the thesis) that a photograph of and a copy of are well-formed *np*'s containing *np* gaps. Accordingly, the thesis, if correct, requires that Which painting did you see a copy of a photograph of? also be well-formed, as indeed it is. Yet this simple and elegant thesis concerning recursion encounters well-known difficulties, which

[1]This paper is the product of joint work with Michael Moortgat, with whom a more comprehensive treatment of these questions is under preparation. This work has been supported by the National Science Foundation under Grant No. SBR-9510706, which we gratefully acknowledge.

have been construed as supporting additional theoretical devices such as filters and other forms of surface constraints. The goal of this paper is to show in the most direct possible way that in one well-known case, it is possible to formulate recursive principles in a way that obviates the need for additional theoretical mechanisms and, at the same time, offers a simple formal characterization of a proposed typological distinction of long-standing interest.

A typological parameter

As Perlmutter [15, 16] first observed, extraction from the *np*-position following a complementizer is possible in some languages, but not in others. Thus, we have:

(1a)	**French**	Marie se demandait qui Jean a dit que Martin a vu? (after [16]) 'Marie wondered who Jean said that Martin saw?'
(1b)		*Marie se demandait qui Jean a dit que a vu Martin? (after [16]) 'who did he say saw Martin?'
(2a)	**English**	Marie wondered who Jean said that Martin saw?
(2b)		*Mary wondered who Jean said that saw Martin?
(3a)		Mary wondered who Jean said Martin saw?
(3b)		Mary wondered who Jean said saw Martin?
(4a)	**Nederlands**	Wie zei Marie dat die appel opgegeten heeft? (after [8]) who said Marie that this apple eaten has 'who did Marie say ate this apple?'
(4b)		Wie zei Marie dat Martin gezien heeft? (after [8]) 'who did you say Martin saw'
	OR	'who did Marie say saw Martin'

There are two basic strategies to deal with these issues. The first is to propose general grammatical rules (selectively chosen by each language) which generate exactly the grammatical examples and fail to generate the ungrammatical examples; the second is to propose general grammatical principles which generate all the good examples and couple these principles wih constraints (selectively chosen by each language) which weed out particular cases. We call the first strategy 'constructive' and the second 'co-constructive'. There have been many co-constructive proposals to account for the above phenomena: we mention here only [15, 16, 1, 2]. In the sections to follow, we develop simple and appealingly symmetrical constructive accounts of these constrasting systems of extraction.

The framework

We work in the framework of multi-modal grammatical logic [10, 11, 4, 9, 14, 12], a framework we describe here only in enough depth to support the goals of this paper. From this perspective, the problem of grammatical composition, within and across such different dimensions of linguistic structure, is regarded as an inference problem: the component pieces of a complex linguistic

structure are taken to be the premisses of a deductive problem, and its global structure to be a conclusion deducible from these premisses in a system of grammatical inference. Thus, grammaticality is identified with validity within this system. Moreover, the formal system characterizing validity offers a natural model, in the style of denotational semantics for programming languages [17], of the cognitive computation that must be assumed to provide the basis for real-time understanding of running speech.[2] Thus the logical methods described here are not introduced in a blind search for formal rigor; on the contrary, they are introduced because they provide an armentarium of subtle and suitable tools and methods that allow us to probe the properties of grammatical reasoning.

In such a system, if A is deducible from a structured set of premisses Γ, we write $\Gamma \Rightarrow A$. It is reasonable to suppose that the deducibility relation is reflexive and transitive: that is, for every formula A, we have $A \Rightarrow A$; and for every triple of formulas A, B, C, if $A \Rightarrow B$ and $B \Rightarrow C$, then $A \Rightarrow C$.

A *uni-modal* deductive system contains a single way (or *mode*) of putting resources premisses together. To reason about this mode, we introduce a product operator—a form of conjunction—together with its residuals (or adjoints)—forms of implication. For example, given a binary mode of composition, we have a product $\bullet$ and two directionally-sensitive implications written, as in the categorial tradition, / and \. Every product and its adjoints are connected by the basic adjointness laws. In the binary case, as here, these take the form:

$$A \Rightarrow C/B \quad \textit{iff} \quad A \bullet B \Rightarrow C \quad \textit{iff} \quad B \Rightarrow A\backslash C$$

As a simple illustration of the consequences of the adjointness laws, take A to be C/B; by reflexivity, we have $C/B \Rightarrow C/B$; using the first adjointness law (left to right), we have $(C/B) \bullet B \Rightarrow C$. This is called the *co-unit* of the adjunction and is also known variously as Modus Ponens (in the logical literature) or (functional) application (in the categorial literature).

There are a number of different presentations of this system of pure binary residuation logic: Gentzen style, natural deduction, Hilbert-style, proof nets. These can be easily shown to be equivalent with regard to provability and we identify them all with the non-associative Lambek calculus **NL** [7].

Keeping the logical rules expressed by the adjointness laws invariant, we may obtain other logical systems by adding *structural rules* [3, 5], such as the following:

RAssoc	$(A \bullet B) \bullet C \Rightarrow A \bullet (B \bullet C)$
LAssoc	$A \bullet (B \bullet C) \Rightarrow (A \bullet B) \bullet C$
Perm	$A \bullet B \Rightarrow B \bullet A$
Contr	$A \Rightarrow A \bullet A$
RWeak	$A \bullet B \Rightarrow A$
LWeak	$A \bullet B \Rightarrow B$

[2] Analogously, we may think of models of the unfolding processes of speech comprehension at the psychological and neurological levels as approximations, at different levels of scale, of the operational semantics of this process.

The presence or absence of these rules defines a family of unimodal logics of conjunction and implication, some of whose members (with characteristic arrows) are:

logic	*structural rules*	*arrows*
NL	none	$(A/B) \bullet B \Rightarrow A$, $B \Rightarrow (A/B)\backslash B$
L	*RAssoc, LAssoc*	$A/B \Rightarrow (A/C)/(B/C)$, $A\backslash(B/C) \Rightarrow (A\backslash B)/C$
LP	*RAssoc,LAssoc,Perm*	$A/B \Rightarrow B\backslash A$, $(A/B)/C \Rightarrow (A/B)/C)$

When a particular formula is provable in a particular logical system, we indicate this using Frege's symbol $\vdash$. Thus,

$$\mathbf{NL} \vdash s/(np\backslash s) \bullet (s/(np\backslash s))\backslash s) \Rightarrow s$$

$$\mathbf{L} \vdash vp/np \Rightarrow (vp/pp)/(np/pp)$$

If a formula is not provable in a particular system, we draw a slash through the turnstile, as in

$$\mathbf{NL} \not\vdash vp/np \Rightarrow (vp/pp)/(np/pp)$$

From this general perspective, then, binary unimodal deductive systems are definable simply by specifying, once and for all, what structural rules the single mode of composition enjoys.

Although the applicability of these systems to the analysis of natural language properties has been the subject of intense scrutiny, it is clear that natural languages differ from unimodal deductive systems in an essential way. Namely, they exhibit a much more subtle control of inference than the all or nothing choice of structural rules allows. For example, individual languages often exhibit varying sensitivity to order. Japanese and Korean, for example, are strict about the position of the tensed verb in a clause but not strict about the position of the arguments preceding the verb. This suggests a richer deductive system, one based on multiple modes of combination.[3] Each mode has a fixed arity, an associated product operator of that arity and an implication for each argument position, satisfying the adjointness laws. Each mode is associated with a set of structural rules. However, something new arises as well: structural postulates involving more than one mode.

As an illustration which will be important in the sequel, consider a system with a single binary mode, associated with the binary product $\bullet$ and adjoints / and \, and a single unary mode, associated with a unary operator $\Diamond$ and a single adjoint $\Box^{\downarrow}$. The adjointness laws for the unary operator take the form:

$$\Diamond A \Rightarrow B \quad \textit{iff} \quad A \Rightarrow \Box^{\downarrow} B$$

[3]In fact, the presence of more than one mode of combination is implicit in linguistic practice: phonologists and morphologists have recognized different kinds of boundaries between elements; $\overline{X}$-bar theory recognizes different modes of combination ('spec-head' relation, for example) at different levels.

Just as we derived the co-unit above by starting with the sequent $C/B \Rightarrow C/B$, if take A above to be $\Box^{\downarrow}B$, then the right hand side holds by reflexivity and the left hand side gives us a unary counterpart to Modus Ponens:[4]

$$\Diamond\Box^{\downarrow}A \Rightarrow A$$

In other words, if the unary operator $\Diamond$ has an adjoint, then the composition of $\Diamond\Box^{\downarrow}$ has an interesting property: it can play a role in part of a deduction and then disappear. This property is the first of two crucial properties of multi-modal type logic we will need below. The second, a small set of structural rules involving the interaction of $\Diamond$ and $\bullet$, will be developed below, after we prepare the ground by developing some very small fragments which will support the illustration of the extraction parameters of interest here.

Fragments without extraction

We now develop the simplest possible fragments of French, English, and Dutch without extraction which can be directly extended to support the extraction constructions of interest. The many points of grammatical interest that these fragments touch on that are not directly relevant to the problem at hand will be systematically ignored. The logical framework is simply the pure residuation logic NL: $\bullet$, /, and \ connected by the adjointness laws; no added structural rules. From this point of view, all that remains to be added is a set of atomic formulas (categories), common to all the fragments, and a set of lexical assumptions associating basic expressions with formulas.

[4]One may connect this straightforwardly with the binary case discussed earlier by regarding the product $A \bullet B$ as the result of applying the unary operator $A \bullet -$ to B. This unary operator may be regarded as a modality $\Diamond_A$, whose corresponding adjoint $\Box^{\downarrow}_A$ is the unary operator $A\backslash -$ which yields $A\backslash B$ when applied to B. Applying the unary adjointness law in this case, we have

$$\Diamond_A B \Rightarrow C \quad \textit{iff} \quad B \Rightarrow \Box^{\downarrow}_A C$$

But this is just another way of writing

$$A \bullet B \Rightarrow C \quad \textit{iff} \quad B \Rightarrow A\backslash C$$

Similarly, we can write $A \bullet B$ as the unary operator $\Diamond_B$ applied to A, and regard C/B as $\Box^{\downarrow}_B C$. Applying the unary adjointness law here gives

$$A \bullet B \Rightarrow C \quad \textit{iff} \quad A \Rightarrow C/B.$$

atom	*vernacular category*
s	sentence
is	inverted sentence
fs	verb-final clause
np	noun phrase (including proper names)
$partp$	participle phrase
c	*that*-clause, *que*-clause, *dat*-clause

The full set of formulae (categories) is obtained as usual by closing the set of atoms under the binary type constructors $\bullet$, $/$, and $\backslash$.

The lexical declarations we need are given in the table below:[5]

language	*category*	*lexical inhabitants*
fr*nch	np	Marie, Jean, Martin
	$(np\backslash s)/partp$	a
	$partp/np$	vu
	$partp/c$	dit
	c/s	que
***ngl*sh**	np	Marie, Jean, Martin
	$(np\backslash s)/np$	saw
	c/s	that
	$(np\backslash s)/c$	said
	$((np\backslash s)/(np\backslash s))/np$	said
d*tch	np	Marie, Martin, die appel
	$np\backslash partp$	opgegeten, gezien
	$np\backslash(partp\backslash fs)$	heeft
	c/fs	dat
	$(is/c)/np$	zei

When word w inhabits category t, we write $w \Rightarrow t$.

For any logical system λ, a lexical type assignment ω is extended to *binarily bracketed sequences* of words in the standard way: thus, if I is an appropriate index set and $\prod_{i\in I} w_i$ is a binarily bracketed sequence of words and τ is a formula, if there are categories $\{\tau_i\}_{i\in I}$ such that $\omega \vdash w_i \Rightarrow \tau_i$ and

$$\lambda \vdash \prod_{i\in I} \tau_i \Rightarrow \tau$$

To show both dependencies, we may indicate that such a situation holds by

$$\lambda, \omega \vdash \prod_{i\in I} w_i \Rightarrow \tau$$

[5]The presence of asterisks is to emphasize the fragmentary character of these simple grammatical systems.

For example, we have

$$\mathbf{NL}, \mathbf{fr*nch} \vdash \mathsf{Jean} \bullet (\mathsf{a} \bullet (\mathsf{vu} \bullet (\mathsf{Martin}))) \Rightarrow s$$

because

$$\begin{array}{l}
\mathbf{fr*nch} \vdash \mathit{Jean} \Rightarrow np \\
\mathbf{fr*nch} \vdash \mathsf{a} \Rightarrow (np\backslash s)/partp \\
\mathbf{fr*nch} \vdash \mathsf{vu} \Rightarrow partp/np \\
\mathbf{fr*nch} \vdash \mathsf{Martin} \Rightarrow np \\
\text{AND} \\
\mathbf{NL} \vdash (np \bullet ((np\backslash s)/partp \bullet (partp/np \bullet np))) \Rightarrow s
\end{array}$$

The first four lines come directly from our lexical assumptions; the final line can be straightforwardly demonstrated as displayed in the proof tree below, where inference steps are marked with t, a, or r, according to whether they depend on transitivity, adjointness, or reflexivity, respectively.[6]

$$\dfrac{\dfrac{\dfrac{}{partp/np \bullet np \Rightarrow partp}\,a,r \qquad \dfrac{}{(np\backslash s)/partp \bullet partp \Rightarrow np\backslash s}\,a,r}{(np\backslash s)/partp \bullet (partp/np \bullet np) \Rightarrow np\backslash s}\,t \qquad \dfrac{}{np \bullet np\backslash s \Rightarrow s}\,a,r}{(np \bullet ((np\backslash s)/partp \bullet (partp/np \bullet np))) \Rightarrow s}\,t$$

Similarly, as the reader is invited to show, we have:

$$\begin{array}{ll}
\mathbf{NL}, \mathbf{fr*nch} & \vdash \mathsf{Marie} \bullet (\mathsf{a} \bullet (\mathsf{dit} \bullet (\mathsf{que} \bullet (\mathsf{Jean} \bullet (\mathsf{a} \bullet (\mathsf{vu} \bullet (\mathsf{Martin})))))))) \Rightarrow s \\
\mathbf{NL}, \mathbf{*ngl*sh} & \vdash \mathsf{Jean} \bullet (\mathsf{said} \bullet (\mathsf{that} \bullet (\mathsf{Martin} \bullet (\mathsf{saw} \bullet \mathsf{Marie})))) \Rightarrow s \\
\mathbf{NL}, \mathbf{d*tch} & \vdash \mathsf{zei} \bullet (\mathsf{Marie} \bullet (\mathsf{dat} \bullet (\mathsf{Martin} \bullet ((\mathsf{die\ appel} \bullet \mathsf{gegeten}) \bullet \mathsf{heeft})))) \Rightarrow is
\end{array}$$

These fragments are of course extremely simple. This is obvious at the lexical level, since each fragment contains fewer than 10 words and speakers of natural languages are estimated to know

[6] Actually, we let t stand for a generalization of transitivity which is easily shown to be valid in the presence of the adjointness laws. We illustrate with a simple special case. Suppose $A \Rightarrow B$ and $C \bullet B \Rightarrow D$. By adjointness,

$$C \bullet B \Rightarrow D \quad \textit{iff} \quad B \Rightarrow C\backslash D$$

By our second premise, the lefthand side holds; thus, the righthand side holds; by our first premise and transitivity, we have $A \Rightarrow C\backslash D$; taking this as the righthand side of the adjointness law, the lefthand side gives us $C \bullet A \Rightarrow D$. Thus, we have proved the derived rule of inference (with premisses represented on top of the line and conclusion below):

$$\dfrac{A \Rightarrow B \quad C \bullet B \Rightarrow D}{C \bullet A \Rightarrow D}$$

By an easy inductive argument, this simple result can be generalized to show that we can generalize transitivity to substitution inside a product of arbitrary depth.

on the order of tens of thousands of words. This can be remedied in part by enriching the lexicon. But enriching the lexicon is not in and of itself a sufficient remedy.

In the next section, we will examine the well-known inadequacies of **NL** as a logic of extraction and show how simple extensions of it can accommodate the properties of interest here of languages like French, English, and Dutch.

Extraction: preliminaries

An embedded question, such as qui a vu Martin in a French sentence such as Jean s'est demandé qui a vu Martin or who saw Martin in an English sentence like Jean wondered who saw Martin, consists of two basic parts: the question word *who* and the *body*—the clausal remnant saw Martin. Although the system **NL** is too weak to deal adequately with French or English embedded questions, its type system can handle this particular case and shows the way toward a system that handles a much broader range of cases.

We begin with the following fact, which follows directly from the lexical properties of the words in question by the adjointness laws:

$$\mathbf{NL}, *\mathbf{ngl}*\mathbf{sh} \vdash \mathsf{saw} \bullet \mathsf{Martin} \Rightarrow np\backslash s$$

Now, writing cq for the type of an embedded question, adjointness allows us to solve for the unknown type x in the sequent

$$(x \bullet (np\backslash s)) \Rightarrow cq \quad \textit{iff} \quad x \Rightarrow cq/(np\backslash s)$$

Thus, adding cq to our stock of atoms and extending our lexical assignment by the declaration $\mathsf{who} \Rightarrow cq/(np\backslash s)$, we can prove:

$$\mathsf{Jean} \bullet (\mathsf{wondered} \bullet (\mathsf{who} \bullet (\mathsf{saw} \bullet \mathsf{Martin}))) \Rightarrow s$$

This analysis is lexically extendable to embedded questions with complementizer whether, by the addition of the lexical type declaration

$$\mathsf{whether} \Rightarrow cq/s$$

But further generalizations within the system **NL** are only possible if completely unacceptable forms of lexical polymorphism are allowed. For example, to treat the embedded question who Martin saw from this perspective, we would need to be able to assign a type to Martin saw, which requires a new type $np\backslash(s/np)$ for saw, relative to which we can show Martin • saw $\Rightarrow s/np$. But we also need a new type for who, $cq/(s/np)$, in order to be able to derive who Martin saw as a cq. Switching the basic inference system from **NL** to **L** by adding the two Associativity rules allows one to combine all the cases in which the gap is rightmost into a single category (since it is possible to show that in the presence of Associativity that all clausal remnants with a single, final np gap belong to the

type s/np), but distinct types are still needed for intial and final gaps and non-peripheral cases still remain.

Before proceeding further, it is worthwhile to take stock of the situation. We seek a system of inference with the following properties:

1. there is a type ξ such that we may take who, for example, to be of type $cq/(\xi\backslash s)$ and we may show using hypothetical reasoning, that the body is provably of type $\xi\backslash s$;
2. to show that the body is of type $\xi\backslash s$, we must be able to show

$$\xi \bullet [\text{body}] \Rightarrow s$$

   This step requires communication between the hypothetical premise ξ and the position of the gap inside the body of the embedded question;
3. communication between the hypothetical premise ξ and the position of the gap must be statable by logical principles; and
4. the additional logical principles allowing communication between the hypothetical premise ξ and the position of the gap must not lead to overgeneration (as occurs if we extend our logical system from **NL** to **LP** by adding both the Associativity Rules and Permutation; while this would allow communication between the hypothetical premise ξ and any possible position in the body, it would also completely destroy the possibility of distinguishing expressions by the order of their components (just as the associativity rules destroy the possibility of distinguish expressions by the grouping of sub-expressions)).

All these desiderata can be simultaneously satisfied in a simple multi-modal system of grammatical inference.

Extraction: a multi-modal approach

Extend **NL** by the addition of a unary mode associated with the unary type constructor $\Diamond_{wh}$ and its adjoint $\Box^{\downarrow}_{wh}$, to form the system we shall refer to as $\mathbf{NL}_{\Diamond_{wh}}$. Recall that by the adjointness laws, we have

$$\Diamond_{wh}\Box^{\downarrow}_{wh}A \Rightarrow A$$

Now, if we assume that the *single* type assignment in our fragment for who is

$$\text{who} \Rightarrow cq/(\Diamond_{wh}\Box^{\downarrow}_{wh}np\backslash s),$$

then we can treat who saw Martin as an embedded question, since we have

$$\mathbf{NL}_{\Diamond_{wh}} \vdash (cq/(\Diamond_{wh}\Box^{\downarrow}_{wh}np\backslash s) \bullet np\backslash s) \Rightarrow cq.$$

It is worth seeing how the proof of this theorem unfolds, in order to appreciate the deductive role played by the modalities.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Diamond_{wh}\Box^{\downarrow}_{wh}np \Rightarrow np}\ unary\ a!,r \quad \dfrac{}{np \bullet (np\backslash s) \Rightarrow s}\ a,r}{(\Diamond_{wh}\Box^{\downarrow}_{wh}np \bullet (np\backslash s)) \Rightarrow s}\ t}{np\backslash s \Rightarrow \Diamond_{wh}\Box^{\downarrow}_{wh}np\backslash s}\ a \quad \dfrac{}{(cq/((\Diamond_{wh}\Box^{\downarrow}_{wh}np)\backslash s) \bullet (\Diamond_{wh}\Box^{\downarrow}_{wh}np)\backslash s) \Rightarrow cq}\ a,r}{(cq/((\Diamond_{wh}\Box^{\downarrow}_{wh}np)\backslash s) \bullet np\backslash s) \Rightarrow cq}\ t$$

Thus, for the special case in which the body of the embedded question is of type $np\backslash s$, we now have two types for who which satisfy all our desiderata (some vacuously), namely the **NL**-type $cq/(np\backslash s)$ and the $\mathbf{NL}_{\Diamond_{wh}}$-type $cq/(\Diamond_{wh}\Box^{\downarrow}_{wh}np\backslash s)$. We have already seen that the first of these is difficult to extend uniformly to a larger range of relevant cases, for at least two reasons:

- atomic categories like np are not part of the logical vocabulary, so our logical system cannot formulate general laws in terms of particular atoms;
- on the other side of the coin, formulating filler-gap communication in terms of particular atoms would miss the point, since similar communication rules hold with respect to other atomic categories (such as ap and pp).

In fact, in standard generative syntax, these problems were recognized very early, and movement rules were formulated not with regard to particular categories, but with regard to a particular feature (or set of features), such as $[+wh]$. But in contrast to the inert feature $[+wh]$, which has no intrinsic logical behavior, the type constructor $\Diamond_{wh}$ is a logical operator, with an adjoint $\Box^{\downarrow}_{wh}$. But over and above the behavior of the operator $\Diamond_{wh}$ with its adjoint $\Box^{\downarrow}_{wh}$ (which plays a role in the proof displayed above), as a product operator, $\Diamond_{wh}$ can also appear in interaction rules, connecting it with other operators.

We have already seen how the type $cq/(\Diamond\Box^{\downarrow}np\backslash s)$ accounts for French, English, and Dutch sentences such as:

French Jean se demandait qui a vu Martin.
Jean refl asked:impf who has seen Martin
'Jean was wondering who saw Martin.'

English Jean was wondering who saw Martin.

Dutch Jan vroeg zich af wie slaapte
Jan asked refl who slept
'Jan wondered who slept.'

The next simplest step of communication between filler and gap involves sentences such as:

French Jean se demandait qui Martin connait.
Jean refl asked:impf who Martin knows
'Jean was wondering who Martin knows.'

English Jean was wondering who Martin saw.

Dutch Jan vroeg zich af wie Martin plaagte
Jan asked refl who Martin teased
'Jan wondered who Martin teased.'

In French and English, these sentences will be derivable if we add the following interaction postulate:

$$\vec{K}\ 2r \quad \Diamond_{wh}A \bullet (B \bullet C) \Rightarrow B \bullet (C \bullet \Diamond_{wh}A).$$

In Dutch, the required interaction postulate is:

$$\vec{K}\ 2l \quad \Diamond_{wh}A \bullet (B \bullet C) \Rightarrow B \bullet (\Diamond_{wh}A \bullet C).$$

These postulates are pleasantly symmetric. To see that they do what we say they do, look at the proofs below:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{(np\backslash s)/np \bullet np \Rightarrow np\backslash s}\,a,r \quad \dfrac{}{np \bullet np\backslash s \Rightarrow s}\,a,r}{(np \bullet ((np\backslash s)/np \bullet np)) \Rightarrow s}\,t}{(np \bullet ((np\backslash s)/np \bullet \Diamond\Box^{\downarrow}np)) \Rightarrow s}\,a}{(\Diamond\Box^{\downarrow}np \bullet (np \bullet (np\backslash s)/np)) \Rightarrow s}\,\vec{K}\ 2r!}{(np \bullet (np\backslash s)/np) \Rightarrow (\Diamond\Box^{\downarrow}np)\backslash s}\,a \quad \dfrac{}{(cq/((\Diamond\Box^{\downarrow}np)\backslash s) \bullet ((\Diamond\Box^{\downarrow}np)\backslash s)) \Rightarrow cq}\,a,r}{cq/((\Diamond\Box^{\downarrow}np)\backslash s) \bullet (np \bullet (np\backslash s)/np) \Rightarrow cq}\,t$$

$$\dfrac{cq/((\Diamond\Box^{\downarrow}np)\backslash s) \bullet (np \bullet (np\backslash s)/np) \Rightarrow cq}{\mathsf{who} \bullet (\mathsf{Martin} \bullet \mathsf{saw}) \Rightarrow cq}\,lex$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{(np \bullet np\backslash s) \Rightarrow s}\,a,r \quad \dfrac{\dfrac{}{\Diamond\Box^{\downarrow}np \Rightarrow np}\,a \quad \dfrac{}{(np \bullet np\backslash(np\backslash s)) \Rightarrow np\backslash s}\,a,r}{(\Diamond\Box^{\downarrow}np \bullet np\backslash(np\backslash s)) \Rightarrow np\backslash s}\,t}{(np \bullet (\Diamond\Box^{\downarrow}np \bullet np\backslash(np\backslash s))) \Rightarrow s}\,t}{(\Diamond\Box^{\downarrow}np \bullet (np \bullet np\backslash(np\backslash s))) \Rightarrow s}\,\overrightarrow{K}\,2l!}{np \bullet np\backslash(np\backslash s) \Rightarrow (\Diamond\Box^{\downarrow}np)\backslash s}\,a \quad \dfrac{}{cq/((\Diamond\Box^{\downarrow}np)\backslash s) \bullet ((\Diamond\Box^{\downarrow}np)\backslash s) \Rightarrow}}{cq/((\Diamond\Box^{\downarrow}np)\backslash s) \bullet (np \bullet np\backslash(np\backslash s)) \Rightarrow cq}$$

$$\dfrac{}{\mathsf{wie} \bullet (\mathsf{Martin} \bullet \mathsf{plaagte}) \Rightarrow cq}\,lex$$

The postulate $\overrightarrow{K}\ 2l$ recursively allows a modally decorated type to adjoin to the left of any right branch. For example, starting with

$$\Diamond\Box^{\downarrow}A \bullet (B \bullet (C1 \bullet C2))$$

the modally-decorated subformula can move in one step to the left of the product $(C1 \bullet C2)$ and subsequently in a second step to the left of $C2$, as illustrated below:

$$\Diamond\Box^{\downarrow}A \bullet (B \bullet (C1 \bullet C2)) \Rightarrow B \bullet (\Diamond\Box^{\downarrow}A \bullet (C1 \bullet C2)) \Rightarrow B \bullet (C1 \bullet (\Diamond\Box^{\downarrow}A \bullet C2))$$

This correctly allows for

Jan vroeg zich af wat (Marie (Piet zou geven)
Jan ask refl particle what Marie Piet would give
'Jan wondered what Marie would give Pete'

While $\overrightarrow{K}\ 2l$ allows the modally-decorated type to look recursively down the left branch of a right branch, it is also possible in Dutch to find the gap down the left branch of a left branch:[7]

Jan vroeg zich af ((op wie)(Marie (gestellt was)))
Jan asked refl particle prep whom Marie like
'Jan wondered who Marie liked'

In this example, the extracted phrase must communicate with the position to the left of gestellt. This is accomplished by adding to the Dutch postulate package the interaction postulate $\overrightarrow{K}\ 1l$, formulated below:

$$\overrightarrow{K}\ 1l \quad \Diamond A \bullet (B \bullet C) \Rightarrow (\Diamond A \bullet B) \bullet C$$

[7]The example involves pied-piping with the preposition op; this fact is orthogonal to our interests here, so is not pursued here. For treatments of pied-piping, see Morrill [13].

Unlike $\vec{K}\ 2l$, the postulate $K2r$ is not recursive, since its output can never be matched to its input. Still, in English, the output of $K2r$ must be able to communicate with more deeply embedded positions, as in

Jean wondered (who (Maxima (tried (to (telephone)))))
Jean wondered (who (Maxima (persuaded (to (telephone Kim)))))

These examples are obtainable with the mirror images of the postulates for Dutch:

$$\begin{array}{ll} \overleftarrow{K}\ 1r & ((A \bullet B) \bullet \Diamond C) \Rightarrow ((A \bullet \Diamond C) \bullet B) \\ \overleftarrow{K}\ 2r & ((A \bullet B) \bullet \Diamond C) \Rightarrow (A \bullet (B \bullet \Diamond C)) \end{array}$$

We assume that these postulates hold for French as well as English. On this view then, the differences between French and English, on the one hand, and Dutch, on the other, reside in the choice between two sets of interaction postulates, displayed in Figures 1 and 2.

$$\begin{array}{ll} \vec{K}\ 2r & \Diamond_{wh} A \bullet (B \bullet C) \Rightarrow B \bullet (C \bullet \Diamond_{wh} A) \\ \overleftarrow{K}\ 1r & ((A \bullet B) \bullet \Diamond C) \Rightarrow ((A \bullet \Diamond C) \bullet B) \\ \overleftarrow{K}\ 2r & ((A \bullet B) \bullet \Diamond C) \Rightarrow (A \bullet (B \bullet \Diamond C)) \end{array}$$

Figure 1: postulates for French and English

$$\begin{array}{ll} \vec{K}\ 2l & \Diamond_{wh} A \bullet (B \bullet C) \Rightarrow B \bullet (\Diamond_{wh} A \bullet C) \\ \vec{K}\ 1l & \Diamond A \bullet (B \bullet C) \Rightarrow (\Diamond A \bullet B) \bullet C \end{array}$$

Figure 2: postulates for Dutch

The Dutch postulates allow an extracted phrase to occur directly following a complementizer. For example, consider the sentence Wie zei Marie dat die appel opgegeten heeft? Figure 3 displays the bracketing we assume and the succession of structures involved in a proof.[8]

On the other hand, the postulates proposed here for English and French do not allow extraction sites to follow a complementizer. More precisely, although it is possible for a modally-decorated expression to communicate with the position following a complementizer, this requires the expression to be on the right branch of a binary structure whose left branch is the complementizer, and this position makes it impossible for the expression to combine with the predicate.

[8]Full details of the proof depend on an analysis of extraposition, which we need not pursue here.

$$\dfrac{\dfrac{\dfrac{((\text{zei Marie})\ (\text{dat}\ (\Diamond\Box^{\downarrow}np\ ((\text{die appel})\ (\text{opgegeten heeft}))))}{((\text{zei Marie})(\Diamond\Box^{\downarrow}np\ (\text{dat}\ ((\text{die appel})\ (\text{opgegeten heeft})))))}\ \overrightarrow{K}\ 2l}{\Diamond\Box^{\downarrow}np((\text{zei Marie})(\text{dat}\ ((\text{die appel})\ (\text{opgegeten heeft}))))}\ \overrightarrow{K}\ 2l}$$

$$\dfrac{\dfrac{\dfrac{\text{FAIL}}{(\text{Marie}\ (\text{said}\ ((\text{that}\ \Diamond\Box^{\downarrow}np)(\text{saw Martin}))))}\ \overleftarrow{K}\ 1r}{(\text{Marie}\ ((\text{said}\ (\text{that}\ (\text{saw Martin})))\ \Diamond\Box^{\downarrow}np))}\ \overleftarrow{K}\ 2r}{\Diamond\Box^{\downarrow}np(\text{Marie}\ (\text{said}\ (\text{that}\ (\text{saw Martin}))))}\ \overrightarrow{K}\ 2r}$$

Discussion

The principles of distributivity on which the above account of extraction systems depends on are non-deterministic and dynamic. These properties distinguish this approach from alternatives in the literature and offer new perspectives on natural language extraction systems. The fuller report on this research in preparation will contain a comparison with current theoretical alternatives mentioned in the introduction.

References

[1] J.W. Bresnan. 1972. *Theory of Complementation in English Syntax.* Ph.D. dissertation. MIT.

[2] N. Chomsky & H. Lasnik. 1977. Filters and control. *Linguistic Inquiry* 8.425-504.

[3] G. Gentzen. 1934-5. Untersuchungen über das logische Schliessen, *Mathematische Zeitschrift* 39, pp. 176-210, 405-431. (English translation in M. E. Szabo, ed., *The Collected Papers of Gerhard Gentzen*, North-Holland, Amsterdam).

[4] P. Hendriks. 1995. *Comparatives and Categorial Grammar.* Ph.D. dissertation. Rijksuniversiteit Groningen.

[5] S. C. Kleene. 1952. *Introduction to Metamathematics.* D. Van Nostrand, New York.

[6] A.S. Kroch. 1989. Asymmetries in Long-Distance Extraction. In M.R. Baltin & A.S. Kroch, eds., *Alternative Conceptions of Phrase Structure*, 66-98. Chicago and London: University of Chicago Press.

[7] J. Lambek. 1961. On the calculus of syntactic types. In R. O. Jakobson, ed., *Structure of Language in its Mathematical Aspects. Proceedings of the 12th Symposium in Applied Mathematics.* American Mathematical Society, Providence.

[8] J. Maling & A. Zaenen. 19xx. The non-universality of a surface filter. In J. Maling & A. Zaenen, eds., *Syntax and Semantics xx: Icelandic Syntax.* xxx-xxx. Orlando: Academic Press.

[9] M. Moortgat. 1996. Categorial type logics. In J. van Benthem and A. ter Meulen, eds., *Handbook of Logic and Language.* Elsevier, Amsterdam.

[10] M. Moortgat and R. T. Oehrle. 1993. *Categorial Grammar: Logical Parameters and Linguistic Variation.* Lecture notes. European Summer School in Logic, Language, and Information. Faculdade de Letras, Universidade de Lisboa, Portugal.

[11] M. Moortgat and R. T. Oehrle. 1993. Adjacency, dependency, order. P. Dekker & M. Stokhof, eds. *Proceedings of the Ninth Amsterdam Colloquium.* 447-467. Institute for Logic, Language, and Computation, Universiteit van Amsterdam.

[12] G. Morrill. 1994. *Type Logical Grammar: Categorial Logic of Signs.* Kluwer, Dordrecht.

[13] G. Morrill. 1995. Discontinuity in Categorial Grammar. *Linguistics & Philosophy* 18.175-219.

[14] R.T. Oehrle. 1997. Substructural logic and linguistic inference. Under review.

[15] D.M. Perlmutter. 1968. *Deep and Surface Constraints in Syntex.* Ph.D. dissertation. MIT.

[16] D.M. Perlmutter. 1971. *Deep and Surface Constraints in Syntax.* New York: Holt, Rinehart & Winston.

[17] J. Stoy. 1981. *Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory.* Cambridge, Mass. & London: MIT Press.

Two-step TAG Parsing Revisited

Peter Poller, Tilman Becker
DFKI GmbH, Stuhlsatzenhausweg 3, D-66123 Saarbrücken, Germany
{poller,becker}@dfki.de

Introduction

Based on the work in (Poller, 1994) and a minor assumption about a normal form for TAGs, we present a highly simplified version of the two-step parsing approach for TAGs which allows for a much easier analysis of run-time and space complexity. It also snggests how restrictions on the grammars might result in improvements in run-time complexity.

The main advantage of a two-step parsing system shows in practical applications like Verbmobil (Bub et al., 1997) where the parser must look at multiple hypotheses supplied by a speech recognizer (encoded in a *word hypotheses lattice*) and filter out illicit hypotheses as early as possible. The first (context-free) step of our parser filters out some illicit hypotheses fast ($O(n^3)$); the constructed parsing matrix is then reused for the second step, the complete ($O(n^6)$) TAG parse.

Simplifying Root and Foot Nodes

The normal form that we assume in the following is only a very minor modification and allows for a trivial retrieval of parses from the results of the normal form-based parser.

We call a TAG *clean* if the root node of every elementary tree and the foot node of every auxiliary tree is labeled with the null-adjoining constraint. Obviously, every TAG can be transformed into a clean TAG by simply adding to every elementary tree an additional node, immediately dominating the root node, with the same label as the root node and the null-adjoining constraint and also adding an additional node, immediately dominated by the foot node, with the the same label as the foot node and the null-adjoining constraint (see figure 1). While this transformation adds new nodes to the derived trees, no adjunctions can take place at these additional nodes and they can easily be eliminated again from a derived tree, resulting in the derived tree of the original grammar. Thus, every TAG can be transformed into an *"almost"* strongly equivalent clean TAG.

In a clean TAG, no adjunction can take place at the root or foot node. This allows us to drop numerous special data structures and steps from the algorithm in (Poller, 1994), resulting in a much cleaner presentation. We also omit the treatment of linear precedence rules, which can easily be added.

A Simplified Two-Step TAG parser

An initial offline step is the extraction of the *context-free kernel* from the TAG G, a context-free grammar G_K which overgenerates, i.e., $L(G) \subset L(G_K)$.

The first step of the parser is a standard parse with the Earley-algorithm (Earley, 1970). The second step is the repeated *elimination* of adjoined trees from the parser's matrix. Thus a TAG derivation is constructed *inside-out*[1].

First, we describe the additional data structure which is added to the items of the Earley parser. An item is a tuple $(i, j, S \rightarrow \alpha \bullet \beta)$, representing a derivation of $a_{i+1} \ldots a_j$ from α. In addition, every non-terminal node in the item carries a list with *node numbers*, taken from the TAG grammar G, uniquely identifying a node in an elementary tree of G which contributed the rule $S \rightarrow \alpha\beta$. Furthermore, every node number in an item can store a list of pointers, called *foot node pointers* (see below). Figure 2 shows two elementary trees and an example item with node numbers.

[1] Or rather bottom-up in terms of the derivation tree of the TAG.

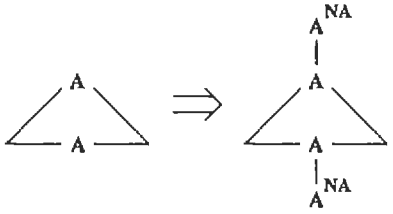


Figure 1: Transforming an auxiliary tree into a clean tree.

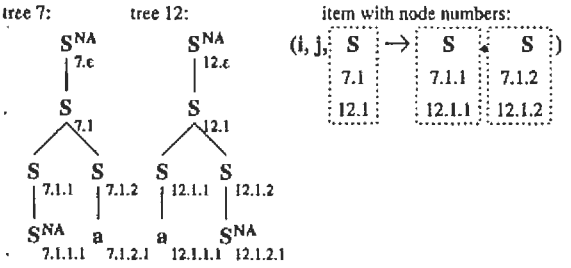


Figure 2: Two example trees and an item from the context-free kernel with node numbers.

Our stepwise approach to TAG-parsing is open to different degrees of precision for the context-free parsing step depending on how much information about the elementary TAG-trees is integrated into the context-free parsing step. We expect that the following alternatives have different influences on the realtime behaviour of a practical system mainly depending on the grammar's characteristics (size, ambiguitiy, ...).

(1) Solely the node labels are used to generate the context-free kernel. In this case the node numbers attached to the terminals and nonterminals have no influence at all on the Earley operators. In this paper we describe this alternative.

(2) An other possibility is to integrate the node addresses from the elementary TAG trees into the rules of the context-free kernel[2]. This requires extensions of the Earley operators because they are now controlled by the characteristics of a specific node of a TAG tree instead of just a label. In particular, the prediction of a nonterminal node now only produces items for context-free rules that are valid according to the TAG grammar and also don't violate any adjunction constraint of the predicted node. This allows for the integration of the TAG constraint check into the context-free parsing step. Similarly, the completor also works only with valid derivation steps according to the TAG grammar. On the other hand, we cannot share node number alternatives in one item anymore. But this increases the overall number of items only by a constant factor.

While the first alternative filters out only invalid context-free derivation steps with respect to node labels, the second one is a stronger filter because it only produces items which represent locally valid derivation steps with respect to the TAG grammar but reduced to the context-free domain of locality. Furthermore it requires one item for each occurence of a context-free rule in different TAG trees. This is a trade off between the number of items to be produced in the first parsing step and the precision of its filtering effect.

It is interesting to note that it is also possible to derive the node number specific items of the second alternative from the parsing matrix of the first one. If the node number check is organized top-down starting with successful context-free derivations (similar to the initialization of the TAG parsing step below) we get a 3-step parser functioning as a cascade of filters.

Independent of these alternatives there is a special parsing strategy for lexicalized TAGs (Schabes et al., 1988). As each terminal is associated with a set of elementary trees we can immediately restrict the relevant TAG trees for the parser to those that are associated with the terminals of the input string. This strategy can still be applied since the rules of the context-free kernel can be computed in advance for each elementary tree separately. Once the relevant elementary trees are determined for an input string, the context-free kernel is simply the union of the associated context-free rules.

For all variants of the context-free parsing step, the second step (the actual TAG parsing step) remains basically the same.

Within the second step an initialization procedure filters out irrelevant items by a top-down traversal starting from roots of successful context-free derivations through the parsing matrix. This sets the ground for an iterated elimination of complete, adjoined trees. This initialization is not strictly necessary (and takes $O(n^3)$ time), but it provides an important speed-up because now only valid context-

[2]Thanks to the anonymous reviewer who suggested this procedure.

free derivations are considered. Invalid context-free derivation steps are filtered out which might become relevant in practical systems with large grammars.

Initially, all leaf nodes (including foot nodes) are marked, i.e., the corresponding node numbers in all items, are labeled *ok*, then these initial *ok's* are propagated "bottom-up" along the context-free derivation steps if they took place inside the same elementary tree which can easily be checked by comparing the unique node numbers. This *ok*-propagation also propagates relevant information about foot node positions.

While recovering elementary trees in the parsing matrix, we need to keep track of possible foot node positions. Each node number in an item is associated with a set of corresponding *foot node pointers*. A foot node pointer points to a particular node number in some item. Thus, when an *ok* is eventually propagated to a node number that represents the root node of an elementary tree, all possible positions of its foot node have been collected in the *foot node pointer* list. Note that there can be $O(n^2)$ foot node pointers for each node number, since there are $O(n^2)$ items.

The relevant computational steps during the iteration are: *elimination, upwards propagation,* and *horizontal propagation. Elimination* of an adjoined tree in the Earley matrix is realized by propagating all *ok's* from immediately "below" all possible foot nodes to all immediate supertrees of the root node[3]. *Upwards propagation* is the propagation of an *ok* from a complete[4] item to its ancestor. *Horizontal propagation* is the propagation of an *ok* to an item where the dot has moved one position to the right.

In the following, all complexity statements are based on the limited number of items that are produced by the Earley algorithm, in particular the number of items in a so called itemlist[5]. Each itemlist I_k contains at most $O(k)$ items so that the number of all items produced by the Earley algorithm is bound by: $\sum_{k=0}^{n} O(k) = O(n^2)$. Another important point for our complexity statements is that each individual item is stored exactly once by the Earley algorithm (even though it might be derived by more than one operation), which means that there are no two identical items.

We can now present a sketch of the algorithm:

```
for j from 0 to n
  for i from j downto 0
    foreach item (i, j, A → α • β)
```

There are only three cases for a node number N of A labeled *ok* of an item $(i, j, A \rightarrow \alpha \bullet \beta)$:

1. $\beta = \epsilon$:
   - 1.1 N is the root of an auxiliary tree: perform an *elimination* of all encodings of this tree,
     This can be done in $O(n^4)$ time.
   - 1.2 N is an inner node of an elementary tree: perform an *upwards propagation.*
     This can be done in $O(n^3)$ time.
2. $\beta \neq \epsilon$:
   perform a *horizontal propagation.*
   This can require $O(n^3)$ time.

```
end foreach; end for j; end for i;
```

The most expensive step is *elimination* (step 1.1). For each root node of an adjoined tree to be eliminated there can be $O(n^2)$ foot node pointers because there are at most $O(n^2)$ items to which they can point to. They result in $O(n^2)$ positions from which this tree can be eliminated, i.e., *ok's* at these positions and their foot node pointer lists must be propagated. Collecting all these foot node pointers lists (of size $O(n^2)$ each) from each of the $O(n^2)$ positions results in $O(n^4)$ time complexity (see figure 3). It is important to note that this computational step cannot produce more than $O(n^2)$ new foot node pointers at the current root node although their computation costs $O(n^4)$. Therefore the fact that each node has at most $O(n^2)$ foot node pointers is an invariant of the iterative elimination.

The complexity of step 1.2 (*upwards propagation*) is also based on the limited number of foot node pointers. Since the *ok* of a node is propagated to its "context-free" ancestors and all possible ancestors are contained in the same itemlist, the complexity is limited by the $O(n^2)$ foot node pointers and the $O(n)$ items ("*supertree*" in figure 4) to which they have to be propagated to, which results in $O(n^3)$ time complexity.

Finally, step 2 (*horizontal propagation*) can also be done in $O(n^3)$ time. Again, $O(n^2)$ foot node pointers of an *ok* have to be propagated to all items where the dot has moved one position

[3]Implementations of the concepts "below" and "supertree" are already provided by the Earley parser.

[4]A complete item has the dot at the rightmost position.

[5]An itemlist I_k is defined as the set of all items $(i, j, S \rightarrow \alpha \bullet \beta)$ where $k = j$.

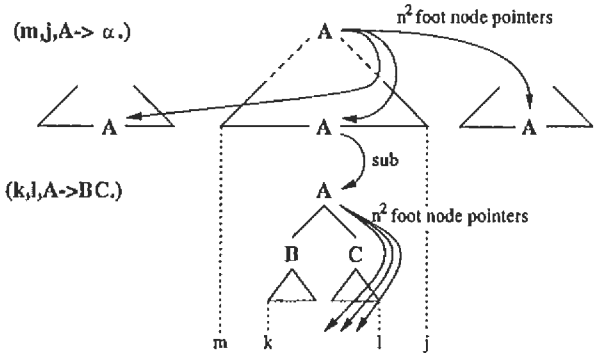


Figure 3: Complex elimination of a completely recognized auxiliary tree.

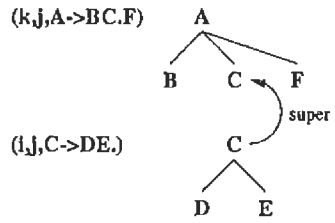


Figure 4: Simple propagation of *ok* to the ancestor node, i.e., the corresponding items.

to the right. Since there are no identical items there can be at most $O(n)$ items (one item from each itemlist; see figure 5)[6].

$$(i, j, S \rightarrow \alpha \bullet X\beta) \overset{next}{\rightarrow} (i, j+k, S \rightarrow \alpha X \bullet \beta)$$

Figure 5: Horizontal propagation of *ok* to the next item.

The overall time complexity of the parsing algorithm is $O(n^6)$ since there are $O(n^2)$ items for which *elimination* ($O(n^4)$) must be performed.

Obviously, the two-step parsing algorithm does not have the correct prefix property (Nederhof, 1997) as it requires the entire sentence to be analyzed by the Earley parser before the second (TAG parsing) step begins. However, the Earley step itself has the correct prefix property wrt. the context-free kernel and also the discussion in (Poller, 1994) of a completely incremental setup also applies to the simplified two-step TAG parsing algorithm presented here.

[6]The concepts "next" and "previous" explicitly represent links between items coming from dot movements by the Earley parser.

Current Work

Although our parser does not have the correct-prefix property it can run incrementally as described in (Poller, 1994) namely by running the TAG parsing step in parallel to the construction of the context-free parsing matrix. Although this may require additional computational steps on unsuccessful context-free derivation steps, the effects on the realtime behavior of a practical system again depend on grammar characteristics. So it would be very helpful to find some kind of "grammar classification" with respect to their "parser suitability" in practical implementations answering the question "Which TAG parser is best suitable for my current task?".

The analysis of the *elimination* step shows clearly that the time complexity of our TAG parser stems from the number of possible foot node positions. We are currently investigating whether certain restrictions on TAG grammars can lower this number. E.g., this is obviously the case for unambiguous grammars.

References

Thomas Bub, Wolfgang Wahlster, and Alex Waibel. 1997. Verbmobil: The combination of deep and shallow processing for spontaneous speech translation. In *Proceedings of ICASSP-97*, pages 71–74, Munich.

J. Earley. 1970. An efficient context–free parsing algorithm. *Communications of the ACM*, 13(2):94–102.

M. J. Nederhof. 1997. Solving the correct-prefix property for TAGs. In T. Becker and H.-U. Krieger, editors, *Proceedings of the Fifth Meeting on Mathematics of Language (MOL5)*, number D-97-02, pages 124–130, Saarbrücken, August. DFKI GmbH.

Peter Poller. 1994. Incremental parsing with LD/TLP-TAGs. *Computational Intelligence*, 10(4):549–562, November.

Yves Schabes, Anne Abeillé, and Aravind K. Joshi. 1988. Parsing strategies with 'lexicalized' grammars: Application to Tree Adjoining Grammars. pages 578–583, Budapest, August.

Wh-islands in TAG and Related Formalisms

Owen Rambow* and K. Vijay-Shanker†
* CoGenTex, Inc. (owen@cogentex.com)
† University of Delaware (vijay@udel.edu)

1 Introduction: TAG and *wh*-Movement

The analysis of *wh*-movement given within TAG is a very convincing argument for the use of a constrained tree-rewriting formalism in syntax, since *wh*-movement does not require any special mechanism in TAG. *wh*-movement can be localized to elementary trees, and island effects are obtained naturally. This situation contrasts with approaches based on string-rewriting formalisms such as CFG, which require extensions (mathematical or at any rate definitional) to the basic mathematical formalism (resulting in theories such as GPSG, HPSG, LFG, or transformational grammar).

However, the question arises how other *tree-rewriting* formalisms such as D-Tree Grammar (Rambow et al., 1995) can handle *wh*-movement. Specifically, the question arises whether an equally elegant solution to the problem of *wh*-movement can be found. In this paper, we propose to study exactly which what features of the formal (mathematical) definition of TAG contribute to the correct analysis of *wh*-movement (in English). We will mainly concentrate on TAG, but occasionally mention tree-local MC-TAG.

The paper is structured as follows. In Section 2, we present the relevant elements of the definition of TAG. We then proceed to discuss specific island types and how these can be expressed in TAG: relative clause and other adjunct islands in Section 3, sentential subject islands in Section 4, and *wh*-islands in Section 5.

2 Elements of the Definition of TAG

In this paper, we will distinguish the following elements of the definition of TAG. (For a full mathematical definition, see (Vijay-Shanker, 1987).)

- The **extended domain of locality** (EDL). In TAG, the elementary structures are trees (rather than strings), so we can state extensive linguistically motivated restrictions on the shape of the elementary trees of a grammar. In fact, any such linguistic restriction on the shape of elementary structures exploits EDL.
- The **geometry of adjunction** (GA). By this term, we mean the specific, mathematical definition of the adjunction operation in TAG and, especially, the shape of the resulting derived tree. Specifically, an auxiliary tree β has a designated footnode; when β is adjoined in a tree α at node ν, it is inserted in its entirety into α. In the process, β remains intact, but α is divided in two subtrees at node ν, with β now attached at ν and the subtree formerly rooted in ν now attached to the footnode of β.
- The **factoring of recursion** (FR). By definition, in an auxiliary tree β, the footnode and the root node must have the same label, A. Furthermore, β can only be adjoined at a node labeled A. We observe that this aspect of the definition of TAG is not essential in the sense that the restrictions could be lifted without affecting the remainder of the definition, in particular the geometry of adjunction. The crucial part for the geometry of adjunction is the presence of a footnode; its label does not *a priori* matter.

We observer that tree-local MC-TAG has the same notion of EDL as TAG, and it has it own

notion of GA. FR is limited to those cases in which adjunction of on eof the component trees takes place.

By definition, any other tree-rewriting system[1] will also have EDL, while GA and FR are specific to TAG. Thus, we are in particular interested in the extent to which GA and FR are used in deriving island constraints, since such use would not necessarily carry over to other tree-rewriting systems.

In the following, we will be making an important assumption. Because of the EDL of the elementary structures of TAG, it is possible to *lexicalize* TAG in a straightforward manner (Schabes, 1990), meaning that each elementary tree in a grammar is associated with exactly one lexical item. Furthermore, we can require that each tree corresponding to a lexical item has positions (substitution nodes or a footnode) corresponding to each syntactic argument of that lexical item, and that the derivation thus reflects the syntactic relation between the lexical items involved (Rambow and Joshi, 1996) (the "lexical derivation constraint"). In this paper, we will only be interested in lexicalized grammars and in derivations that conform to the lexical derivation constraint.

3 Relative Clause Islands and Other Adjunct Islands

Sentence-initial extraction from certain adjuncts such as relative clauses modifying non-fronted object NPs or VP sentential adjuncts is ruled out simply by GA (in conjunction with the lexical derivation constraint). It is simply impossible to adjoin (or substitute) a tree into a (non-fronted) object, or adjoin a tree at a VP node (in a tree which has a subject NP to the left of the VP node), and obtain a derived tree in which some part of the adjoined tree is now in sentence-initial position.

In contrast, it is quite possible to adjoin a relative clause to a subject or adjoin an S-adjunct to a clausal tree (i.e., and adjunct phrase rooted in S), and obtain a *wh*-extraction to sentence-initial position. A sample auxiliary tree that would result in illicit extraction is shown in Figure 1. This tree can be ruled out in several different ways resorting to linguistic arguments. For example, one could exclude the tree by saying that extraction beyond the root node of an adjunct is impossible since the root node is not part of the projection of the lexeme anchoring the adjunct, or one could say that the tree in Figure 1 is illicit because of independently formulated constraints on node labels. In any case, one would be exploiting the EDL to express linguistically motivated constraints on the shape of elementary structures in the grammar. But, crucially, these constraints would carry over to the case of the relative clause modifying an object NP, and to the case of the VP-adjunct: it is not plausible that the linguistic constraints would be formulated in such a way that they only apply to subject relative clauses (or S adjuncts), but not to object relative clauses (or VP adjuncts). Thus, these cases are redundantly rules out by GA.

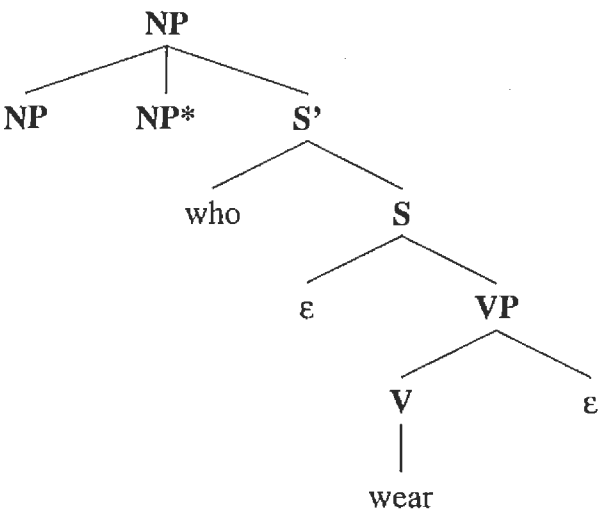


Figure 1: Relative clause with *wh*-moved element

Furthermore, there is a point that is easily overlooked. While object relative clauses with sentence-initial fronting are ruled out by GA, we *also* need to rule out non-initial fronting:

(1) *I saw what$_i$the man who was wearing t_i

While these kinds of sentences may be pathologically bad, they still need to be ruled out in a TAG grammar

[1]We include in this category systems which operate on tree-like structures.

We close by observing that if we are using tree-local MC-TAG, an argument very similar to the one above can be made to demonstrate that any predictive power obtained from the geometry of tree-local multicomponent adjunction is redundant with respect to independently required linguistic restrictions on the shape of the elementary tree sets. We omit the details.

4 Sentential Subjects

It would be possible to derive extraction from sentential subjects in the same manner that we derive extraction from sentential objects, namely by adjoining a matrix clause of the type shown in Figure 2 into the subordinate clause. In order to exclude such a derivation, we must say that the subject position, even when labeled S, cannot be a footnode. Thus, simply saying that we have factoring of recursion does not limit the extraction patterns: we must, in addition, make a linguistically motivated choice among possible footnodes. Designating a footnode is equivalent to allowing extraction from that position.

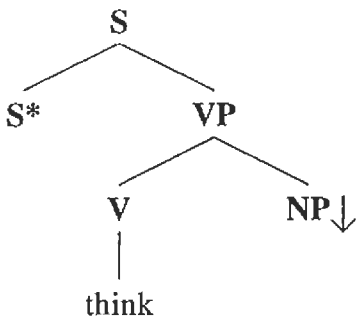


Figure 2: Matrix clause with sentential subject

However, the designation of the footnode is not sufficient. This is because of a well-known asymmetry in extraction from picture-NPs: while extraction from certain object NPs is possible, extraction from subject NPs never is.

(2) a. What$_i$ did you buy a picture of t_i?

b. * What$_i$ did a picture of t_i fall on your head?

Thus, if we use tree-local multicomponent MC-TAG to derive picture-NP extraction by substituting the main NP and substituting or adjoining the extracted *wh*-element, we must still specifically rule out extraction from subject position in some manner.[2] Furthermore, the same problem arises when we want to distinguish between verbs that allow picture-NP extraction and those that do not (as readily). Therefore, we will need some formal device (say, a feature EXTRACT on frontier nodes which regulates multicomponent derivations across them) for blocking extraction from certain positions *in addition to the choice of footnodes.* (This will also exclude extraction from sentential subjects if these are analyzed as projecting to NP.) The use of the device will need to be linguistically motivated. Some sort of equivalent device with similar linguistic motivation for its use can be used in tree rewriting systems which do not have FR or GA.

5 *Wh*-Islands

In English, we can exclude some wh-islands by restricting the shape of elementary trees.

(3) *What$_i$ do you know whom$_j$ Mary gave t_j t_i?

(3) is excluded because the elementary tree for give, which would need t have two *wh*-moved elements, is already excluded (we never have multiple *wh*-movement in English elementary trees). This analysis exploits the EDL and transfers to other tree-rewriting formalisms.

But that does not cover all cases of *wh*-islands.

(4) *What$_i$ do you know whom$_j$ Mary told t_j that she had bought t_i?

In (4), there is only one *wh*-extraction per elementary verbal tree. These cases can be excluded in several ways, but they all use FR. We

[2]Kroch (1989) suggests instead that the traces of picture-NP extractions are found in the elementary structures of the main verb. They are not licensed in the verbal tree because not bound; an index is adjoined through multi-component adjunction (along with the *wh*-element), which provides the binding. However, unlike traces in object position, traces in subject position are never licensed to begin with. This analysis exploits the EDL and could be expressed in other tree-rewriting formalisms as well.

take (Frank, 1992) as the most advanced example. There, trees in which *wh*-extraction from below takes place are footed in C′ and (hence by FR) are rooted in C′, while those without *wh*-extraction from below are both footed and rooted in CP. This ensures that if there is a *wh* element below (and assuming *wh* elements are always in SPEC(CP)), then the tree below must project to CP, and then the foot node must be CP, and hence the root node as well. Therefore, there is no room for a further *wh* element up front that would come from below. Note that if there is a single *wh*-movement at any depth of embedding, then because of the recursion part of FR, all trees above it must be CP-footed-and-rooted as well.

Frank's analysis makes use of several linguistic constraints on elementary structures (exploiting EDL), among which:

1. In an elementary tree, a C′ may never dominate a CP.
2. An elementary tree may not have two CP nodes one immediately dominating the other (the "anti-CP-recursion stipulation").
3. Each tree can only contain a single lexical item and its projection and (crucially) *no part of a different lexical item's projection.* Otherwise, we could have *(did) (john) wonder whether* in one tree which is rooted and footed in C′. Such a tree would allow sentences such as **Who did John wonder whether Sue saw?.*

Given these linguistic constraints as well as FR, it is impossible to obtain a node labeled CP immediately dominating a *wh*-element on the path separating a "moved" *wh*-element from the rest of its tree.

In tree rewriting systems that do not have FR, it will be necessary to derive the path constraint in some other manner. In DTG, it is possible to include path constraints explicitly in the elementary structures. In such an approach, the linguistic restrictions can be relaxed; it is not necessary to assume the anti-CP-recursion constraint, for example, and it would even be possible to allow an inversion of CP and C′.

6 Conclusion

In conclusion, we have seen that for relative clause islands and clausal adjunct islands, and for sentential subject islands, the TAG analysis exploits EDL but not GA or FR. These analyses would therefore carry over to other tree-rewriting systems. In the case of *wh*-islands, FR is exploited in conjunction with several linguistic EDL-type constraints in order to limited the occurrence of certain nodes on the path of *wh*-"movement". While this can not be replicated exactly in a system without FR, any other device to restrict the path has the same effect.

References

Frank, Robert (1992). *Syntactic Locality and Tree Adjoining Grammar: Grammatical, Acquisition and Processing Perspectives.* PhD thesis, Department of Computer and Information Science, University of Pennsylvania.

Kroch, Anthony (1989). Asymmetries in long distance extraction in a Tree Adjoining Grammar. In Baltin, Mark and Kroch, Anthony, editors, *Alternative Conceptions of Phrase Structure*, pages 66–98. University of Chicago Press.

Rambow, Owen and Joshi, Aravind (1996). A formal look at dependency grammars and phrase-structure grammars, with special consideration of word-order phenomena. In Wanner, Leo, editor, *Current Issues in Meaning-Text Theory.* Pinter, London.

Rambow, Owen; Vijay-Shanker, K.; and Weir, David (1995). D-Tree Grammars. In *33rd Meeting of the Association for Computational Linguistics (ACL'95)*, pages 151–158. ACL.

Schabes, Yves (1990). *Mathematical and Computational Aspects of Lexicalized Grammars.* PhD thesis, Department of Computer and Information Science, University of Pennsylvania.

Vijay-Shanker, K. (1987). *A study of Tree Adjoining Grammars.* PhD thesis, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA.

On Defining TALs with Logical Constraints

James Rogers
School of Computer Science
Univ. of Central Florida
jrogers@cs.ucf.edu

In Rogers (1997b) we introduced a new class of models, *three-dimensional tree manifolds* (3-TM), that can serve as both the derived and derivation structures for TAGs in the same way that trees serve as both derived and derivation structures for CFGs. These tree-manifolds are higher-dimensional analogs of trees; in a 3-TM the children of a node form an ordinary (two-dimensional) tree just as in ordinary trees the children of a node form a string. From this point of view the elementary structures of a TAG can be interpreted as labeled local 3-TMs—a root node and its set of children (a pyramidal structure)—analogous to the interpretation of the rewrite rules of a CFG as local trees. Adjunction in TAGs and substitution in CFGs both reduce to a form of concatenation, of local trees in CFGs, of local 3-TMs in TAGs. In Figure 1, for example, the local 3-TMs corresponding to the elementary trees α_1 and β_1 are concatenated to form the 3-TM corresponding to the result of adjoining β_1 into α_1. The two-dimensional yield of this structure is the corresponding derived tree and its one-dimensional yield is the derived string.

This analogy can be extended downward to encompass the regular languages and upward generating the *control language hierarchy* of Vijay-Shanker et al. (1987), Weir (1988), Weir (1992). And it turns out to be quite deep. The ordinary finite-state automata (over strings—the one-dimensional level) accepting the regular languages become, at the two-dimensional level, the tree-automata accepting the recognizable sets of trees. The corresponding automata over 3-TM turn out to accept exactly the sets of tree manifolds that are generated by TAGs (with adjoining constraints) modulo a relaxation of the usual requirement that the root and foot of an auxiliary tree be labeled identically to each other and to the node at which it adjoins. (We refer to these sets as the recognizable sets of three-dimensional tree manifolds.) Moreover, essentially all of the familiar automata-theoretic proofs of properties of regular languages lift directly to automata over tree-manifolds of arbitrary dimension—the dimensionality of the structures is simply a parameter of the proof and plays no essential role.

In Rogers (1998) we exploit this regularity to obtain results analogous to Büchi's characterization of the regular languages in terms of definability in wS1S (the weak monadic second-order theory of the natural numbers with successor) (Büchi, 1960) and Doner's (1970) and Thatcher and Wright's (1968) characterizations of the recognizable sets (of trees) in terms of definability in wSnS (the weak monadic second-order theory of n successor functions—the complete n-branching tree). The recognizable sets of 3-TM are exactly the finite 3-TM definable in the weak monadic second-order theory of the complete n-branching three-dimensional tree manifold, which we refer to as wSnT3. This raises the prospect of defining TALs through the medium of collections of logical constraints expressed in the signature of wSnT3 rather than with explicit TAGs. In this paper, we introduce this approach and begin to explore some of its ramifications in the context of TAGs for natural languages.

Rather than work in wSnT3 directly, we work with an equivalent class of structures that is linguistically more natural. A *Labeled Headed Finite 3-TM* is a structure:

$$(T, \lhd_3, \lhd_2, \lhd_1, \bar{\lhd}_3, \bar{\lhd}_2, \bar{\lhd}_1, \lhd_3^+, \lhd_2^+, \lhd_1^+, \mathrm{H}_1, \mathrm{P}_\sigma)_{\sigma\in\Sigma},$$

where T is a rooted, connected, finite subset of the complete n-branching 3-TM (for some n); $\lhd_i$ is immediate domination, $\bar{\lhd}_i$ is local proper domination (among siblings) and $\lhd_i^+$ is global proper domination (inherited), all in the i^{th} dimension;[1] H_1 is the set of *Heads* (exactly one in each string of children—these are underlined in the figures) and P_σ are the labels (each picking out the set of nodes labeled σ, not necessarily mutually exclusive).

We begin by looking at a simple example: assignment of case in XTAG main verb (α_1) and auxiliary verb (β_1) trees. We interpret node names as first-order variables and tree names as monadic second-order variables with, e.g., $\alpha_1(x)$ satisfied iff x is

[1] Domination, in its familiar form in trees, is domination in the second dimension here. Domination in the first dimension is usually known as linear precedence. We will refer to domination in the third dimension as *above*.

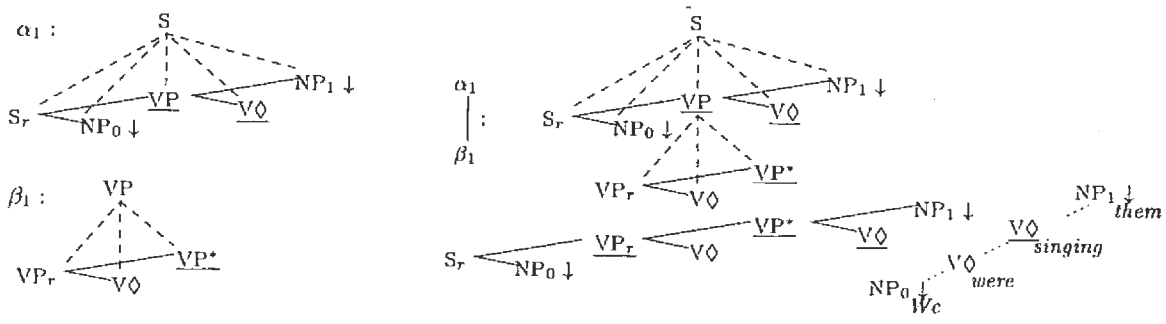


Figure 1: Tree Manifolds

the (3^{rd} -dimensional) root of the local 3-TM corresponding to α_1:

$$
\begin{array}{l}
\alpha_1(s) \leftrightarrow \\
(\exists s_r, np_0, vp, v, np_1)[\\
\quad s \lhd_3 s_r \wedge s \lhd_3 np_0 \wedge s \lhd_3 vp \wedge s \lhd_3 v \wedge s \lhd_3 np_1 \wedge \\
\quad \mathrm{Min}_2(s_r) \wedge \mathrm{Max}_2(np_0) \wedge \mathrm{Max}_2(v) \wedge \mathrm{Max}_2(np_1) \wedge \\
\quad s_r \lhd_2 np_0 \wedge s_r \lhd_2 vp \wedge \mathrm{H}_1(vp) \wedge \\
\quad \mathrm{Min}_1(np_0) \wedge np_0 \lhd_1 vp \wedge \mathrm{Max}_1(vp) \wedge \\
\quad vp \lhd_2 v \wedge vp \lhd_2 np_1 \wedge \mathrm{H}_1(v) \wedge \\
\quad \mathrm{Min}_1(v) \wedge v \lhd_1 np_1 \wedge \mathrm{Max}_1(np_1) \wedge \\
\quad \mathrm{Initial}(s) \wedge \mathrm{Anchor}(v) \wedge \mathrm{Subst}(np_0) \wedge \mathrm{Subst}(np_1) \\
]
\end{array}
$$

Here Min_i and Max_i pick out minimal (root) and maximal (leaf) nodes wrt the i^{th} dimension—these are defined predicates:

$$\mathrm{Min}_i(x) \equiv \neg(\exists y)[y \lhd_i x].$$

Initial(x) is true at the root of each local 3-TM encoding an initial tree, Anchor(x) is true at each anchor node (we will ignore insertion of the lexical items), and Subst(x) is true at each node marked for substitution—these are labels, in Σ. We require all Subst nodes to have children in the 3^{rd} -dimension and require the set of Initial nodes to be exactly the Subst nodes plus the root of the entire 3-TM:

$$(\forall x)[\mathrm{Subst}(x) \rightarrow (\exists y)[x \lhd_3 y]]$$
$$(\forall x)[\mathrm{Initial}(x) \leftrightarrow (\mathrm{Subst}(x) \vee \mathrm{Min}_3(x))]$$

Figure 2 shows the distribution of features responsible for case assignment in the XTAG grammar. Following the approach of Rogers (1997a) we interpret the paths occurring in the feature structures decorating the trees as monadic predicates: Σ includes each sequence of features that is a prefix of a path occurring in a feature-structure derivable in the grammar.[2] We will refer to this set of sequences as Feat. Each node is multiply labeled: the feature-structure associated with it is the union of the paths labeling it. In order to capture the distinction between top and bottom feature-structures we will prefix their paths with 't' and 'b', respectively. We can then add to the definition of α_1:

$$\langle \mathrm{t} : \mathrm{case} : \mathrm{acc} \rangle(np_1) \wedge \langle \mathrm{b} : \mathrm{assign\text{-}case} : \mathrm{nom} \rangle(v).$$

This encoding of feature-structures gives us a straightforward definition of predicates for path equations as well. For any sequences $w, v \in$ Feat:

$$\langle w = v \rangle(x, y) \equiv \bigwedge_{\substack{w:u \in \mathrm{Feat} \\ \text{or } v:u \in \mathrm{Feat}}} [\langle w : u \rangle(x) \leftrightarrow \langle v : u \rangle(y)].$$

With this we can add the re-entrancy tags:

$$
\begin{array}{l}
\langle \mathrm{b} : \mathrm{assign\text{-}case} = \mathrm{t} : \mathrm{assign\text{-}case} \rangle(vp, v) \wedge \\
\langle \mathrm{b} : \mathrm{assign\text{-}case} = \mathrm{t} : \mathrm{assign\text{-}case} \rangle(s_r, vp) \wedge \\
\langle \mathrm{b} : \mathrm{assign\text{-}case} = \mathrm{t} : \mathrm{case} \rangle(s_r, np_0) \wedge \\
\langle \mathrm{t} = \mathrm{t} \rangle(s, s_r).
\end{array}
$$

The labeling of the elementary trees can then be interpreted as a collection of constraints on local 3-TM, with the set of structures licensed by the grammar being the set of 3-TM in which every node satisfies one of these collections of constraints. Note that for a 3-TM in which the β_1 3-TM expands the VP node in an α_1 3-TM to be licensed, the VP node must satisfy both the constraints of the α_1 3-TM and the constraints on the root of the β_1 3-TM. Thus the top feature-structure of the VP is unified with the top feature-structure of VP_r and the bottom feature-structure with the bottom feature-structure of the foot VP by simple transitivity of equality. There is no need for additional path equations and no extra-logical mechanisms of any sort; licensing is simply a matter of ordinary model-theoretic satisfaction. To get the (default) unification of top and bottom feature structures of nodes that are not expanded by adjunction we add a single universal principle:

$$(\forall x)[\mathrm{Max}_3(x) \rightarrow (t = b)(x, x)].$$

[2] As is typical in FTAG, we assume finite feature-structures.

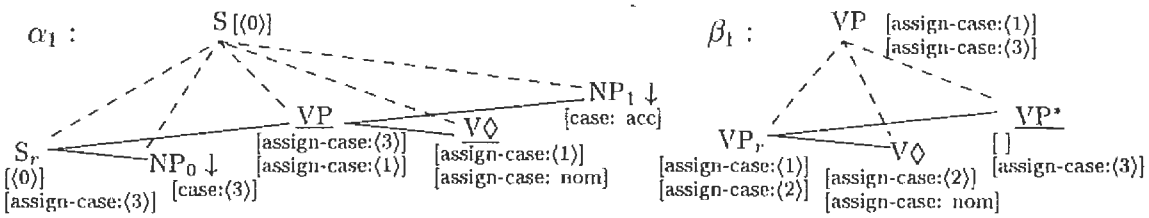


Figure 2: Case assignment in XTAG.

Taken literally, this approach yields little more than a fully declarative restatement of the original grammar. But, in fact, a large proportion of the features decorating elementary trees are there only to facilitate the transport of features through the tree: there is no obvious linguistic motivation for positing that "assign-case" is a feature of VPs or of S. In the language of wSnT3 there is no need for these intermediate "functional" features or even any need to distinguish top and bottom feature structures—we can state directly that the value of the *case* feature of the subject NP, for instance, must agree with the value of the *assign-case* feature of the verb. Of course, what is interesting about this relationship is the effect of adjoined auxiliaries. The TAG analysis includes an *assign-case* feature for the intermediate VP in order to allow auxiliary verbs adjoined at the VP to intercept this relationship by interposing between the VP's top and bottom feature structures. In wSnT3 we obtain the same result from the way in which we identify the relevant verb. For instance, if we take it to be the last adjoined verb[3]—the one most deeply embedded in the third dimension—we can add to the definition of α_1:

$$(\exists x, y)[vp \lhd_3^* x \wedge \mathrm{Max}_3(x) \wedge x \bar{\lhd}_2 y \wedge \langle\text{assign-case}\rangle(y) \wedge \langle\text{assign-case} = \text{case}\rangle(np_0, y)].$$

In somewhat more linguistically natural terms[4] we might say that a verbal head governs, for the purposes of case assignment, all arguments in its local tree manifold (i.e., the minimal associated structure). Furthermore a verbal head in an auxiliary tree governs all nodes in the structure it adjoins into, as well as all nodes governed by them—effectively each case assigner governs every child of each node properly above it up to the first Initial node:

$$\begin{aligned}&\mathrm{Governs}(x, y) \equiv \\ &\quad \langle\text{assign-case}\rangle(x) \wedge \\ &\quad (\exists z)[z \lhd_3^+ x \wedge z \lhd_3 y \wedge \\ &\qquad (\forall z')[(z \lhd_3^+ z' \wedge z' \lhd_3^+ x) \rightarrow \neg\mathrm{Initial}(z')]].\end{aligned}$$

[3]This is correct only if the foot nodes have null-adjoining constraints, as is usual.

[4]This is not meant to be a proposal of an analysis of assignment of case in XTAG, only to be an example of the style of analyses that can be supported by this approach.

Then v assigns case to np_0 iff it governs it and is not, itself, governed by some other case assigner:

$$(\forall x, y)[(\mathrm{Governs}(x, y) \wedge \neg(\exists z)[\mathrm{Governs}(z, x)]) \rightarrow \langle\text{assign-case} = \text{case}\rangle(x, y)].$$

Alternatively, we could adopt existing accounts based on the more familiar relationships in the two-dimensional projections of the 3-TMs such as traditional GB accounts or Rizzi's (1990) *Relativized Minimality*. All of these are definable in wSnT3 and all, therefore, correspond to some TAG account of case assignment to subjects. The central question, perhaps, is which comes closest to the intuitions informing the existing grammar.

This factoring of a TAG grammar into component linguistic principles is not a new idea. Vijay-Shanker and Schabes's (1992) hierarchical encoding of TAG lexicons using partial descriptions of trees becomes, from this perspective, a matter of classifying the lexicon on the basis of shared properties—every verbal anchor is associated with a subject and the associated structure (see Figure 3):

$$\begin{aligned}&(\forall v)[(\mathrm{Anchor}(v) \wedge \mathrm{Verb}(v)) \rightarrow \\ &\quad (\exists s_r, np_0, vp)[s_r \lhd_2 np_0 \wedge s_r \lhd_2 vp \wedge vp \lhd_2 v \wedge \\ &\qquad \langle\text{case}\rangle(np_0) \wedge \langle\text{assign-case} : \text{nom}\rangle(v) \wedge \cdots]],\end{aligned}$$

transitive verbs, in addition, are associated with an object:

$$\begin{aligned}&(\forall v)[(\mathrm{Anchor}(v) \wedge \mathrm{Verb}(v) \wedge \mathrm{Transitive}(v)) \rightarrow \\ &\qquad (\exists np_1)[v \lhd_1 np_1 \wedge \langle\text{case:acc}\rangle(np_1) \wedge \cdots]],\end{aligned}$$

and so on. Note that, since concatenation of 3-TMs does not disturb relationships internal to them, there is no non-monotonicity here (or,rather, the apparent non-monotonicity is an artifact of the yield operation)—there is no need to distinguish top and bottom quasi-nodes, no need for partial trees.

A more obvious connection can be made to Frank's (1992) exploration of universal grammatical principles as interactions of the TAG mechanism with linguistically motivated constraints on the elementary structures. From the current perspective, these constraints are just properties of the local 3-TMs occurring in well-formed grammatical structures. Here, again, the constraints are not disturbed

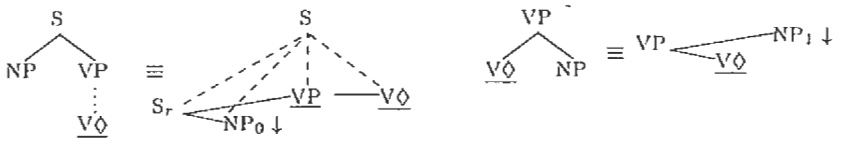


Figure 3: Shared structure as shared properties

by the process of building 3-TMs from these local structures—these are properties not just of the elementary structures but of every local 3-TM in all well-formed structures. More interestingly, not all of these constraints are simple properties of the elementary trees, some depend on the derivations. The Specifier Licensing Condition (SLC), for instance, in its basic form, can only be satisfied once an adjunction has taken place. As it turns out, the mechanism employed in capturing this as a condition on the elementary trees is to encode it as a requirement that certain features of the sort we have been calling "functional" are instantiated.[5] Again in this context, in abstracting away from such implementation details, wSnT3 offers a more direct expression of the constraint.

The key feature of this approach is that it isolates the linguistic theory being expressed from the mechanical details of the grammar formalism expressing it—in this respect there is a strong parallel to Mosier's category theoretic approach to HPSG (Mosier, 1997)—*without* losing the restrictions that the formalism imposes. Thus, while the linguistic principles can usually be stated directly, the fact that they must be expressible within the signature of wSnT3 limits them to principles which can be enforced by TAGs. In fact the characterizations of the recognizable sets of 3-TM by definability in wSnT3 and of TAG tree and string languages as the yields of recognizable sets of 3-TM are constructive and when these constructions are carried out many "functional" features of the sort that the logical approach eschews are instantiated in the resulting TAG. This raises the possibility of using the logical definitions not just as an abstract means of discussing the linguistic theory, bnt also as a sort of higher-level language which can be compiled into TAGs of the familiar sort.[6]

[5] Perhaps coincidentally, these attribute case-assignment to IPs and Is in close parallel to the XTAG example we started with.

[6] There are some formidable obstacles to realizing this idea, not the least of which is the fact that the process of compiling wSnT3 formulae into 3-TM automata has, at least potentially, non-elementary complexity. Nonetheless, prior experience at the one- and two-dimensional levels suggests that the process may be feasible even for relatively substantial theories and here we have the knowledge that reasonably compact grammars for similar theories exist (as witnessed by the XTAG grammar). Thus, in some sense, the potential intractability of the theoretical approach may be thought of as a compiler technology issue.

References

J. R. Büchi. 1960. Weak second-order arithmetic and finite automata. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 6.

John Doner. 1970. Tree acceptors and some of their applications. *J. Computer and System Sciences*, 4.

Robert Evan Frank. 1992. *Syntactic Locality and Tree Adjoining Grammar: Grammatical, Acquisition and Processing Perspectives.* Ph.D. thesis, University of Pennsylvania.

M. Andrew Mosier. 1997. Is HPSG featureless or unprincipled. *Linguistics and Philosophy*, 20. (MOL4).

Luigi Rizzi. 1990. *Relativized Minimality.* MIT Press.

James Rogers. 1997a. "Grammarless" phrase structure grammar. *Linguistics and Philosophy*, 20. (MOL4).

James Rogers. 1997b. A unified notion of derived and derivation structures in TAG. In *Proceedings of the Fifth Meeting on Mathematics of Language*, Dagstuhl.

James Rogers. 1998. A descriptive characterization of tree-adjoining languages. In *To Appear: COLING-ACL'98.* Project Note.

J. W. Thatcher and J. B. Wright. 1968. Generalized finite automata theory with an application to a decision problem of second-order logic. *Mathematical Systems Theory*, 2.

K. Vijay-Shanker and Yves Schabes. 1992. Structure sharing in lexicalized tree-adjoining grammars. In *Proceedings COLING'92.*

K. Vijay-Shanker, David J. Weir, and Aravind K. Joshi. 1987. On the progression from context-free to the tree adjoining languages. In A. Manaster-Ramer, editor, *Mathematics of Language.* John Benjamins.

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

David J. Weir. 1992. A geometric hierarchy beyond context-free languages. *Theoretical Computer Science*, 104:235–261.

Exploiting Semantic Dependencies in Parsing

William Schuler
Computer and Information Science Dept.
University of Pennsylvania
Philadelphia, PA 19103
schuler@linc.cis.upenn.edu

Abstract

In this paper we describe a semantic dependency model for estimating probabilities in a stochastic TAG parser (Resnik, 1992) (Schabes, 1992), and we compare it with the syntactic dependency model inherent in a TAG derivation using the flat treatment of modifiers described in (Schabes and Shieber, 1994).

1 Introduction

The use of syntactic dependencies to estimate parser probabilities is not uncommon (Eisner, 1996) (Collins, 1997) (Charniak, 1997). Typically, a maximum probability parse is estimated from bigram statistics of lexical items that participate in head-modifier or head-complement dependencies with other lexical items. These dependencies can be characterized as (head, label, modifier) triples and ⟨ head, label, complement ⟩ triples – or as labeled directed arcs in a graph – which have the property that each lexical item may participate as a modifier or a complement in no more than one dependency. Using a TAG derivation tree (Joshi, 1987) with a flat treatment of modifiers (Schabes and Shieber, 1994), it is possible to capture the long distance dependencies of wh-extractions and relative clauses as adjacent arcs in a dependency structure, making them available for probability estimates within the parser as well. In this case, the head-complement dependencies for a sentence correspond to a set $\mathcal{S}$ of substitution triples $\langle \gamma, \eta, \alpha \rangle$ (where tree α substitutes into tree γ at note address η), and the head-modifier dependencies correspond to a set $\mathcal{A}$ of adjunction triples $\langle \gamma, \eta, \beta \rangle$ (where tree β adjoins into tree γ at node address η), in a probabilistic TAG (Resnik, 1992).[1]

Although the TAG-based syntactic dependency model has the necessary domain of locality (in terms of adjacent arcs on the derivation tree) to accurately guide a statistical parser, it is still susceptible to sparse data effects, in part because it does not generalize attachment statistics across syntactic transformations. An adjective used as a declarative predicate, for example, could not draw on attachment statistics for the same adjective used as a modifier, or as a predicate in a relative clause, and vice versa, because each transformation uses a different syntactic dependency structure. The triples in the syntactic dependency sets $\mathcal{S}$ and $\mathcal{A}$ for the sentences, "The damaged handle is attached to the drawer," and "The handle attached to the drawer is damaged," are represented as arcs in Figure 1.

In order to group these attachment statistics into denser pools of data, we need to abstract a common semantic structure from the various syntactic structures, effectively adopting a common argument frame for each transformation. This means that each auxiliary tree must have an argument position corresponding to the subject substitution site in its predicative transformation if it is a modifier auxiliary, or corresponding to the wh-object substitution site in its object-extraction transformation if it is a predicative auxiliary.[2] For convention, we place

[1] Although Resnik uses a direct function $S(\gamma, \eta, \alpha)$ to the $[0 - 1]$ interval where we use a probability of set membership $\mathcal{P}(\langle \gamma, \eta, \alpha \rangle \in \mathcal{S})$. Also note that this correspondence between head-complement dependencies and substitution dependencies is not strictly true in the case of predicative auxiliaries (Schabes and Shieber, 1994), which are handled by adjunction in TAG.

[2] See (Schabes and Shieber, 1994) for a description of the distinction between modifier and predicative auxiliaries.

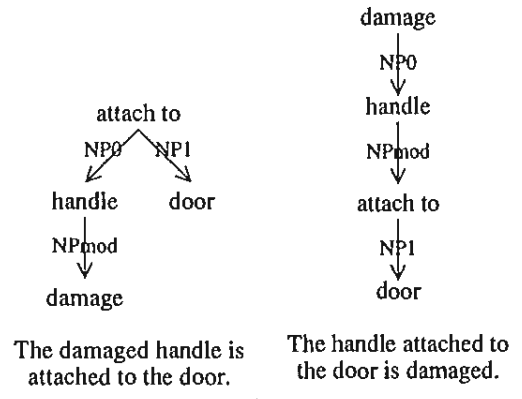


Figure 1: Syntactic dependencies in TAG

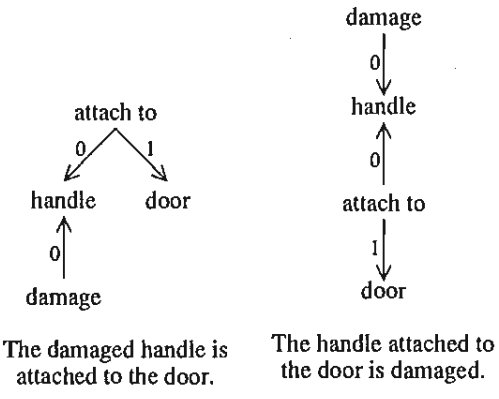


Figure 2: Semantic dependencies

this extra argument position at the foot node of the auxiliary tree, so the auxiliary takes the tree it adjoins into as an argument. This means that our semantic dependency model effectively reverses the direction of dependencies involved in adjunction from the syntactic model.The triples in the semantic dependency set $\mathcal{D}$ for the sentences, "The damaged handle is attached to the drawer," and "The handle attached to the drawer is damaged," are represented as arcs in Figure 2.

Formally, we augment the syntactic dependency sets $\mathcal{S}$ and $\mathcal{A}$ with a semantic dependency set $\mathcal{D}$ of $\langle$ predicate, label, argument $\rangle$ triples defined as follows:

- For every substitution (head-complement) dependency $\langle \gamma, \eta, \alpha \rangle$ in $\mathcal{S}$ add a predicate-argument dependency $\langle anchor(\gamma), argnum(\gamma, \eta), anchor(\alpha) \rangle$ to $\mathcal{D}$; and
- For every adjunction (head-modifier) dependency $\langle \gamma, \eta, \beta \rangle$ in $\mathcal{A}$ add a predicate-argument dependency $\langle anchor(\beta), argnum(\beta, foot(\beta)), anchor(\gamma) \rangle$ to $\mathcal{D}$;

where $anchor(\alpha)$ returns the lexical anchor of tree α, and $argnum(\alpha, \eta)$ returns the semantic argument position corresponding to node η in tree α. In this way we can combine argument attachment distributions for initial tree transformations and auxiliary tree transformations into a common attachment distribution for the underlying predicate.

2 Parsing

Parsing proceeds in three passes of $O(n^6)$ complexity. First, the chart is filled in from the bottom up, as described in (Schabes et al., 1988), and the input is recognized or rejected. The parser then constructs a shared forest (Vijay-Shanker and Weir, 1993) top-down from the elements in the chart, ignoring those items on bottom-up dead ends. Finally, the parser proceeds with the more expensive operations of feature unification and probability estimation on the reduced set of nodes in the shared forest. The chart consists of a set of items that each specify a node address η in an elementary tree α, a top ($\top$) or bottom ($\bot$) marker denoting the phase of operation on the node, and four indices i,j,k, and l, composing the extent of the node's coverage in the sentence: $\langle \alpha, \eta, \top, i, j, k, l \rangle$. The shared forest consists of an and/or graph, with 'or' arcs from each non-dead-end chart item to instantiations of the parser productions that could have produced it, and 'and' arcs from each instantiation of a parser production to the chart items it would have required.

In order to select a most-preferred parse for an ambiguous input, a highest-probability item is selected from the top node in the shared forest, and a parse is read off below it by traversing the subordinate items with the most probable dependencies. The probability of each shared forest item is computed as the maximum of the probabilities of its 'or'-adjacent parser productions. The probability of each instantiation of a parser production is computed as the proba-

bility of the relevant dependency for that production multiplied by the probabilities of the chart items that production required. Finally, the probability of each parse must be multiplied by the probability of each elementary tree given a lexical item in the input.

The probability model is adapted from (Resnik, 1992), which assigns a probability to any arc $\langle \alpha, \eta, \beta \rangle$ (where tree β is attached to tree α at node address η) being in the set of substitutions $\mathcal{S}$ or adjunctions $\mathcal{A}$ in a derivation. The root of the derivation tree is represented as $\langle MAIN, 0, \alpha \rangle$ in $\mathcal{S}$, and null adjunctions (which terminate the adjunction of modifiers at a node) are represented as $\langle \alpha, \eta, \epsilon \rangle$ in $\mathcal{A}$. Finally, the probability of a tree α is represented as the probability of the double $\langle anchor(\alpha), tree(\alpha) \rangle$ being in the set $\mathcal{T}$ of elementary trees used in a parse.

Probabilities for the dependencies in a parser production are estimated from observed frequencies that a child predicate c (the base-form anchor of a tree) occurs in argument position a of a parent predicate p (the base-form anchor of another tree), within some training set $\mathcal{D}$ of dependency structures: $\hat{F}(\langle a, p, c \rangle \in \mathcal{D})$. The top-level dependency is represented in $\mathcal{D}$ as $\langle MAIN, 0, c \rangle$, and null adjunctions are represented as $\langle NULL, 0, c \rangle$.[3] Note that we use the same dependencies as Resnik (the syntactic dependency sets $\mathcal{S}$ and $\mathcal{A}$) in describing the probability model, and use the semantic dependencies ($\mathcal{D}$) only in the estimation of those probabilities.

Probabilities are estimated as follows:

- For any topmost item in a derivation tree:
  $\langle \alpha, 0, \top, 0, -, -, n \rangle$
  the initial probability would be:
  $\mathcal{P}(\langle MAIN, 0, \alpha \rangle \in \mathcal{S} \mid \langle MAIN, 0, \_ \rangle \in \mathcal{S})$
  which we estimate as:
  $\frac{\hat{F}(\langle MAIN, 0, anchor(\alpha) \rangle \in \mathcal{D})}{\hat{F}(\langle MAIN, 0, \_ \rangle \in \mathcal{D})}$
- For any chart production for the substitution of initial tree α into γ at node address η, where i and j are indices, and η is a substitution site in γ with the same label as the root of α:
  $\frac{\langle \alpha, 0, \top, i, -, -, j \rangle}{\langle \gamma, \eta, \top, i, -, -, j \rangle}$
  the probability would be:
  $\mathcal{P}(\langle \gamma, \eta, \alpha \rangle \in \mathcal{S} \mid \langle \gamma, \eta, \_ \rangle \in \mathcal{S})$
  which we estimate as:
  $\frac{\hat{F}(\langle anchor(\gamma), argnum(\gamma, \eta), anchor(\alpha) \rangle \in \mathcal{D})}{\hat{F}(\langle anchor(\gamma), argnum(\gamma, \eta), \_ \rangle \in \mathcal{D})}$
- For any chart production for adjunction of auxiliary tree β into γ at node address η, where i, j, i', j', p and q are indices, and η is an adjunction site in γ with the same label as the root of β:
  $\frac{\langle \gamma, \eta, \bot, i', p, q, j' \rangle \quad \langle \beta, 0, \top, i, i', j', j \rangle}{\langle \gamma, \eta, \bot, i, p, q, j \rangle}$
  the probability would be:
  $\mathcal{P}(\langle \gamma, \eta, \beta \rangle \in \mathcal{A} \mid \langle \gamma, \eta, \_ \rangle \in \mathcal{A})$
  which we estimate as:
  $\frac{\hat{F}(\langle anchor(\beta), argnum(\beta, foot(\beta)), anchor(\gamma) \rangle \in \mathcal{D})}{\hat{F}(\langle \_, \_, anchor(\gamma) \rangle \in \mathcal{D})}$
- For any chart production for closing adjunction at a node address η in tree γ:
  $\frac{\langle \gamma, \eta, \top, i, j, k, l \rangle}{\langle \gamma, \eta, \bot, i, j, k, l \rangle}$
  the probability would be:
  $\mathcal{P}(\langle \gamma, \eta, \epsilon \rangle \in \mathcal{A} \mid \langle \gamma, \eta, \_ \rangle \in \mathcal{A})$
  which we estimate as:
  $\frac{\hat{F}(\langle NULL, 0, anchor(\gamma) \rangle \in \mathcal{D})}{\hat{F}(\langle \_, \_, anchor(\gamma) \rangle \in \mathcal{D})}$
- For any other chart production, the probability would be 1.
- Finally, the probability that each elementary tree α is in the set of trees $\mathcal{T}$ used in the parse, given a lexical item is:
  $\mathcal{P}(\langle anchor(\alpha), tree(\alpha) \rangle \in \mathcal{T} \mid \langle anchor(\alpha), \_ \rangle \in \mathcal{T})$
  which we estimate as:
  $\frac{\hat{F}(\langle anchor(\alpha), tree(\alpha) \rangle \in \mathcal{T})}{\hat{F}(\langle anchor(\alpha), \_ \rangle \in \mathcal{T})}$

[3] Although since the null-adjunction probability only conditions on the parent tree, it will be a constant in every case, and can be ignored in estimating the maximum probability.

3 Practical Issues

The extended goal of this project was to provide a natural language interface for "Jack" (Badler et al., 1993), a human-like agent that answers questions and carries out instructions in a virtual 3-D environment. The system's restricted domain makes unknown words and unknown syntactic structures unlikely, and the goal of trauslating inputs into a formal language for the agent avoids the danger of modifier scoping

ambiguity (which our model does not evaluate), since the scoping of modifier adjuncts can usually be ignored in transfer. It is for this reason that we concentrate our attention on parsing attachment ambiguity at the expense of other problems which might seem more relevant in free text applications.

We consider our approach orthogonal to statistical smoothing techniques such as (Charniak, 1997) for addressing the sparse data problem, and for this reason do not discuss them.

References

Badler, N., Phillips, C., and Webber, B. (1993). *Simulating humans: Computer graphics animation and control.* Oxford University Press, New York, NY.

Charniak, E. (1997). Statistical parsing with a context-free grammar and word statistics. In *Fourteenth National Conference on Artificial Intelligence*, Providence, Rhode Island.

Collins, M. (1997). Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL '97).*

Eisner, J. (1996). Three new probabilistic models for dependency grammar. In *Proceedings of the Sixteenth International Conference on Computational Linguistics (COLING '96).*

Joshi, A. K. (1987). An introduction to tree adjoining grammars. In Manaster-Ramer, A., editor, *Mathematics of Language.* John Benjamins, Amsterdam.

Resnik, P. (1992). Probabilistic tree-adjoining grammar as a framework for statistical natural language processing. In *Proceedings of the Fourteenth International Conference on Computational Linguistics (COLING '92)*, Nantes, France.

Schabes, Y. (1992). Stochastic lexicalized tree-adjoining grammars. In *Proceedings of the Fourteenth International Conference on Computational Linguistics (COLING '92)*, Nantes, France.

Schabes, Y., Abeillé, A., and Joshi, A. K. (1988). Parsing strategies with lexicalized grammars: Applications to tree adjoining grammars. In *Proceedings of the 12th International Conference on Computational Linguistics (COLING '88)*, Budapest, Hungary.

Schabes, Y. and Shieber, S. M. (1994). An alternative conception of tree-adjoining derivation. *Computational Linguistics*, 20(1):91–124.

Vijay-Shanker, K. aud Weir, D. (1993). The use of shared forests in tree adjoining grammar parsing. In *Proceedings of EACL '93*, pages 384–393.

Comparison of XTAG and LEXSYS Grammars

Martine Smets
Cognitive and Computing Sciences
University of Sussex, Brighton BN1 9HQ, UK
martines@cogs.susx.ac.uk

1 Introduction

This paper presents work that forms part of the ongoing LEXSYS project on wide-coverage parsing,[1] and more precisely, some differences between our D-Tree grammar and XTAG 1995.

2 Grammar Formalism

We use the Lexicalised D-Tree Grammar (LDTG) formalism (Rambow et al. 95), which is based on the Lexicalized Tree Adjoining Grammar (LTAG) formalism. In LDTG, there are two types of edges between nodes: *d-edges*, represented with a broken line, and *p-edges*, represented by a solid line. Trees are combined by two substitution-like operations, both of which involve combining two descriptions, by equating exactly one node from each description. One of the operations is always used to add complements, and involves equating a frontier node (in the d-tree that is getting the complement) with the root of some component (in the d-tree that is providing the complement), such that the two nodes being equated are compatible . An example of substitution is shown in Figure 1.

The d-tree for *to adore* is composed with the d-tree for *seems* by equating the two nodes labelled VP[fin: –]. The top component of the *to adore* tree can then be fitted into the resulting d-tree by equating the root of the *seems* tree with the lower S of the *to adore* tree.

A second operation is used to add modifiers, but we are not going to discuss it in this paper.

[1]This work is supported by UK EPSRC project GR/K97400 'Analysis of Naturally-occurring English Text with Stochastic Lexicalized Grammars' (http://www.cogs.susx.ac.uk/lab/nlp/dtg/).

3 Differences between XTAG and LEXSYS Grammars

3.1 Trees Are Syntactic Representations

A first difference between our DTG and TAG is that we do not claim that elementary trees express in all cases the predicate-argument structure of their anchor; instead, they represent the syntactic requirements of their anchor. To illustrate, because raising verbs subcategorize for a syntactic subject, they anchor a standard verb tree with a subject, and not a tree rooted in VP without a subject, as in TAG. On the other hand, there are trees rooted in VP which represent VP complements and can be anchored by any verb. In those trees, there is no subject (because VP complements do not have syntactic subjects), and a semantic argument of the verb is thus missing.

This choice allows us to adopt other linguistic analyses than the ones supported by XTAG, as will be shown in the next sections.

3.1.1 Complementation and Long Distance Dependency

A main difference between the two grammars is that there are VP complements in our grammar, when there are only S complements in XTAG (except for auxiliaries and raising verbs). To the sentence in (1), our grammar gives the analysis in (1a), while XTAG gives the analysis in (1b).

```
(1a) [S He wants [VP to [VP come]]]
(1b) [S He wants
        [S [NP PRO] [VP to [VP come]]]]
```

In (1a), the complement of *want* is a VP; in (1b), it is an S, and the subject of the sentence is *PRO* (an empty pronominal). The analysis in (1a) is the one proposed in lexical theories

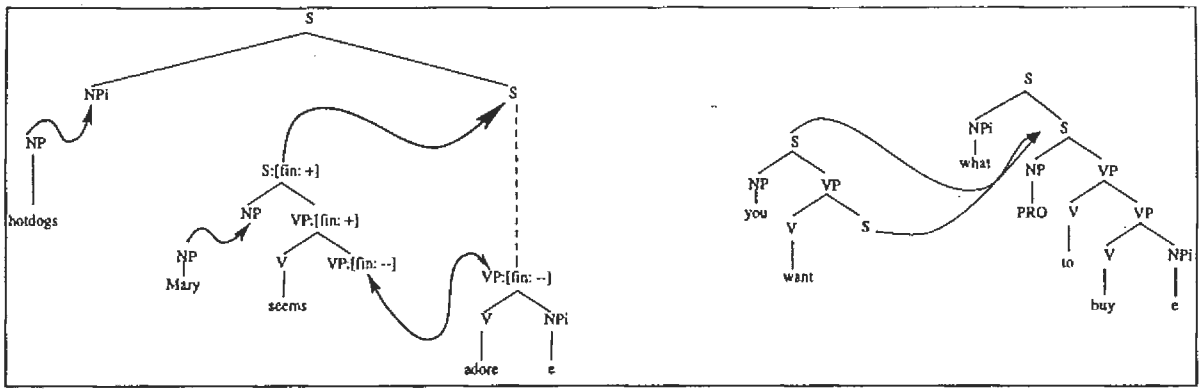


Figure 1: Example of unbounded dependency in DTG (left) and in TAG (right)

such as Head-Driven Phrase Structure Grammar (HPSG, Pollard and Sag 1994) and Lexical Functional Grammar (LFG, Bresnan 1982) while the analysis in (1b) is the analysis of Government and Binding (GB, Haegeman 1991).

Arguments given in the XTAG report for the representation in (1b) include a uniform treatment for indicative, infinitive and gerund embedded clauses (XTAG report 1995, 1998). This implies that both infinitive and gerunds are analyzed as having an empty subject, which is questionable, because there is no evidence for the existence of *PRO*[2] ; it is even more questionable for gerunds, which have the same distribution as NPs (this is true even for verbal gerunds), and can hardly be characterized as clauses (Malouf 1997 *inter alia*).

An important reason for XTAG to adopt the analysis in (1b) is that it seems to be the only type of analysis possible in that formalism (except if *equi* verbs like *want* anchor the elementary tree for raising verbs). This comes from the fact that unbounded dependencies which extend across more than one clause boundary are achieved through the use of auxiliary trees in XTAG: to derive the sentence in (2), an initial tree for *buy* is combined with an auxiliary tree for *want* (Figure 1).

(2) What do you want to buy?

[2] *PRO*, besides being unmotivated, creates theory-internal problems: XTAG has to define two different infinitive auxiliaries *to*, one which assigns the case *no case* (when the subject is *PRO*) and the other one which does not assign any case (when the complementizer *for* assigns accusative case to the subject). This distinction between two *to* is of course ad-hoc.

The auxiliary tree for *want* is grafted onto the lower S of the *buy* tree, and the recursivity of the process creates unbounded dependency. And because in auxiliary trees the root node and the foot node must be of the same category, verbs such as *want* cannot take a VP complement (assuming *want* anchors an S-tree).

In our grammar, on the other hand, there is no such restriction, and verbs can take S complements as well as VP complements. This decision to introduce VP complements in the grammar has a number of consequences (some of which are related to what was discussed in section 1):

- auxiliaries and raising verbs anchor the same tree family as other verbs which take VP complements;
- passive trees are rooted in VP;
- because trees for auxiliaries and raising verbs are rooted in S as any other verb tree, there are no predicative trees;
- the grammar has at least twice as many verb trees as XTAG 95 (each tree rooted in S has a counterpart rooted in VP), and in fact, more than that as we use multiple instances of the same tree to represent disjunctive feature values.

Each of these points will be addressed in the next sections.

3.2 Verbs of *Considering*

Another type of construction for which we assume the existence of a VP complement is the

Subject-to-Object Raising (SOR) structure illustrated in (3).

(3) We believe [Kim] [to be very smart]

In that analysis[3], which we adopt, raising verbs such as *believe* have two complements, an NP and a VP. In the XTAG analysis, SOR verbs have only one complement, a clause.

We assign the same kind of analysis to another type of verbs, referred to as *verbs of considering* in Pollard and Sag (1994): *consider* in (4) and *regard* in (5) have two complements, an NP and respectively an AP and a PP.

(4) I consider Jack quite intelligent
(5) We regard him as a nuisance

This analysis has been debated since the early seventies, and supported by a number of researchers, Pollard and Sag (1994) among others.

XTAG, on the other hand, adopt the GB analysis, which considers that verbs of *considering* and the like have only one complement, a small clause. Small clauses are *Ss* headed by an empty verb, and anchored by the complement of that verb (NP, PP or AP). This account is not without problems. First, it has to postulate an unmotivated empty verb position: there is no evidence that such a position should exist. Its purpose is to allow adjunction of raising and auxiliary verbs, but this is a purely technical device which is not supported by linguistic evidence.

A more important problem is the fact that verbs which take small clause complements must be able to constrain the small clause predicate: *consider* allows PPs, NPs and APs (6) while *prefer* allows PPs only (7).

(6)
We consider Kim a good teacher
We consider Kim quite good
We consider Kim out of his mind

(7)
*We prefer Kim a good teacher
*We prefer Kim quite good
We prefer Kim out of here

[3]The SOR analysis has been advocated with compelling arguments by Bresnan (1982), Postal and Pullum (1988) and Pollard and Sag (1994) *inter alia*.

Verbs who subcategorize for clausal complements cannot specify the subcategorization requirements of the verb in the complement clause; for example, there is no example of a verb like *say* which would stipulate what kind of complement the verb in its clausal complement should have. Accordingly, in the XTAG account, the clausal complement is not expanded, whether it is a standard clause or a small clause. But the data in (6) and (7) show that verbs of *considering* and the like do select the type of phrases which follow the NP; the solution adopted in XTAG is to use the feature *mode* (whose values are usually *indicative, imperative, subjunctive*, etc.) and to add to the range of features *nom* and *prep* (for NP and AP, and PP respectively). The verb *consider* selects an S which has a feature mode with value *nom/prep*, while *prefer* selects a small clause with *prep* as value for the feature mode. Of course, the decision to add these values to the range of values of the feature mode is ad-hoc, as they have nothing to do with verb mode, and are only a technical device to match the subcategorization requirements of the verb of *considering* with the actual category of the complement in the embedded small clause. Our solution, on the other hand, is straightforward: if the verb *consider* constrains the type of phrase that follows the NP it is because this phrase is also one of its complements.

Our choice of analysis, besides being straightforward and motivated by the data, also allows for a more uniform account of passive: the passive of verbs of *considering* and the like is handled by the same lexical rules as for other transitive verbs.

3.3 Auxiliaries and Raising Verbs

In XTAG, raising verbs and auxiliary verbs anchor the same auxiliary tree rooted in VP. In our grammar, on the other hand, those verbs anchor trees rooted in S, and belong to different families.

There have been debates in the literature about the status of auxiliary verbs, and several authors have argued that auxiliaries and modals should be considered as main verbs (Pullum and Wilson (1977), Gazdar et al. (1982)). Arguments include the fact that some auxiliaries behave also like main verbs (*be* and *have*, *ought*, *is* in *is to*), and the existence of semi-auxiliaries

(*need, used, dare* and *have to*[4]) which behave like main verbs in certain environments and like auxiliaries in other environments. So, the distinction between auxiliaries and main verbs is not clear-cut, and either the tree family for auxiliary verbs will include verbs which do not always behave like auxiliaries, or verbs classified as main verbs will share characteristics with auxiliary verbs. In both cases, the obvious solution is to abandon the distinction between main verbs and auxiliaries in terms of drastically different types of tree, and adopt instead a unified representation for both kinds of verbs.

A second issue is the fact that in the tree for auxiliaries and raising verbs, the complement of the anchor is a VP. This implies that all subject raising verbs subcategorize for VP, which is clearly not the case (*become* subcategorizes for AP or NP, *turn out* for AP, NP or VP). Thus, in order to get the right distribution of subcategorization, constraints on the complement of the raising verbs have to be expressed through percolation of the *mode* feature, which use has already been shown to be ad-hoc in similar instances.

3.3.1 Predicative Trees

There are no predicative trees in our grammar: this is a consequence of our decision to adopt a tree rooted in S for both raising verbs and auxiliaries. Also, we want a uniform treatment of predicative complements, and this would not be the case if we adopted different trees for predicative complements of verbs of *considering* and predicative complements of other types of verbs. So, predicative complements just substitute in the tree of their governing verb, like other types of complements.

A main criticism of our approach will be that the basic trees do not express all semantic relations: a predicative complement places semantic restrictions on the subject, and this cannot be captured in the basic trees, because predicative complements are substituted in the tree for the auxiliary/raising verb; similarly, for the VP complement trees, which do not have a subject[5]; finally, in the case of passive, the passive participle anchors a VP tree too, and the subject is not expressed either in the elementary tree.

We agree with this, but we do not claim that we can express every type of relation between constituents in basic trees; instead, we believe that it is impossible to capture all relevant information, syntactic and semantic, in the basic trees. We therefore adopt a modular representation, with the basic trees expressing mainly syntactic information, and the derivation tree most of the semantic information. We hope that this division of labour will allow us to express motivated syntactic analyses in the grammar, without having to compromise in order to also express at the same level semantic relations.

3.4 Conclusion

This paper has presented some differences between XTAG and the grammar we are developing in the LEXSYS project. It has shown that the DTG formalism gives us the possibility to adopt linguistic analyses which have proven to be more motivated than the GB ones (which can also be expressed with the same formalism).

The fact that we will have much more trees than TAGs might seem like a drawback to our approach. But Evans and Weir (1998) are exploring ways to allow a compact representation of the grammar for parsing purposes.

References

Bresnan, Joan. 1982. Control and complementation. In J. Bresnan, eds., *The Mental Representation of Grammatical Relations*.

Evans, Roger and David Weir. 1998. A structure-sharing parser for lexicalized grammars. In *COLING/ACL'98*.

Haegeman, Liliane. 1991. *Introduction to Government and Binding Theory*. Blackwell Publishers, Oxford, UK.

Malouf, Rob. 1996. A constructional approach to English verbal gerunds. In *Proceedings of the Twenty-second Meeting of the Berkeley Linguistics Society*, 1996, pages 255-266.

Pollard, C., and I. Sag. 1994. *Head-Driven Phrase Structure Grammar*. Chicago University Press, Chicago, IL.

[4]Actually, *have to* behaves like a main verb in all environments, but has a meaning very similar to *must*. This shows that which verbs are auxiliaries cannot be predicted from semantic information alone, as was noted by Pullum and Wilson (1977).

[5]I do not see any advantage of having *PRO* instead of no subject, though.

Pullum, Geoffrey and Paul Postal. 1988. Expletive noun phrases in subcategorized positions. In *Linguistic Inquiry*, 19, pages 635-670.

Pullum, Geoffrey and Deirdre Wilson. 1977. Autonomous syntax and the analysis of auxiliaires. In *Language*, v.53,4, pages 741-788.

Rambow, Owen, K. Vijay-Shanker and David Weir. 1995. Parsing D-Tree Grammars. In *Proceedings of the 33rd Meeting of the Association for Computational Linguistics (ACL '95)*, pages 151-158.

XTAG research group. 1995. A lexicalized Tree Adjoining Grammar of English, Technical Report IRCS 95-03, University of Pennsylvania.

A Compact Encoding of a DTG Grammar

Martine Smets
Cognitive and Computing Sciences
University of Sussex
Brighton, BN1 9QN, UK
martines@cogs.sussex.ac.uk

Roger Evans
Information Technology Research Institute
University of Brighton
Brighton, BN2 4GJ, UK
Roger.Evans@itri.brighton.ac.uk

Abstract

This paper describes the use of a compact encoding scheme to represent the trees of the wide-coverage DTG grammar currently being developed in the LEXSYS project (Caroll et al 1998). The encoding scheme is derived from the scheme for LTAG grammars described in Evans, Gazdar and Weir (1995), but the LEXSYS grammar is the first attempt to apply these ideas on a larger scale. In this paper we report on the approach taken and discuss some technical improvements to the encoding scheme that we have introduced to overcome problems of scaling.

1 Compact encoding of LTAG/ DTG trees

Evans, Gazdar and Weir (1995) describe a compact representation of LTAG trees using the default inheritance language DATR (Evans and Gazdar 1996). This representation uses two techniques to make tree grammars more compact. First, inheritance between trees allows them to share common structure: for example a transitive verb tree can inherit the structure of an intransitive verb, adding a direct object argument, and a ditransitive can inherit all this structure from the transitive, adding a further indirect argment. Second, the grammar includes rules which derive new trees from old: tree relations such as passive, dative movement and topicalisation are encoded as rules, allowing the full grammar to be encoded as a set of base trees plus a set of such generative rules.

The representation of rules is *internal*, in the sense that they are expressed as part of the tree definitions themselves, rather than externally as a set of rules in a separate representation system. For example, passive is represented as a constraint between an 'input' tree with a direct object and an 'output' tree without one. This relation is part of the definition of any transitive verb and can be 'invoked' by setting the input to be the base tree for the verb and reading off the outpu as the resultin surface tree (see Evans, Gazdar and Weir 1995 for full details).

This approach has a number of advantages. Rule definitions can directly access test or modify any part of the tree under consideration, and they can themselves use the inheritance mechanisms to share structure between rules. In addition rules can be positioned in the main tree hierarchy so that they are only visible to trees they can sensibly be applied to. For example, the passive rule definition can be located as part of the transitive verb tree definition, and so will be inheritied by ditransitives (which also passivise) but not by intransitives, sentential complement verbs etc., which cannot. Thus by internalising rules, one can simply express generalisations about their scope.

However, the specific proposal in Evans, Gazdar and Weir (1995) also had some less desirable features. In order to apply a rule, it was necessary to 'plumb together' inputs and outputs using inheritance statements in a new DATR node. In addition, rule definitions included specific references to their own names, making it difficult for rules to share definitions in practice, and difficult to apply a rule more than once to the same tree. The present approach uses an improved version of this scheme which addresses these issues:

1. rule application is achieved by adding path prefixes (specifying rules to be applied) to queries on the basic tree definition, rather than creating a new node and 'plumbing';
2. rule definitions are no longer dependent on the name of the rule they define, they are properly modular, making it easier to generalise across rules;
3. as a consequence it is now possible to apply a rule more than once to the same tree if required;

These improvements make it more feasible to consider a more realistic set of rules with more complex

interactions, as required for a large scale grammar such as the LEXSYS grammar.

2 Application to LEXSYS

2.1 The rules

There are at present 35 rules in the grammar that we are developing as part of the LEXSYS system. Roughly half of these rules are movement rules, because there is currently a different movement rule for each possible extraction site: e.g. there is a rule for *wh*-questions on the subject, another one for questions on the first object, different rules for extraction of prepositional objects depending of whether or not the preposition is stranded, etc. Other rules are the passive rules (with or without a *by*-phrase), the rules concerning the order of particles and complements, the inversion rule, the rule deriving VP complements from S, etc.; there are also rules for nouns, determiners, adjectives and adverbs.

Rules which share common characteristics are coded as a hierarchy, which allows them to share much of their structure. For example, for movement rules[1] the top of the hierarchy is the *topic* rule, which specifies the top structure of the derived tree (where the 'extracted' element is localised). The rule *topobj1* (topicalization of the first object) inherits the information in *topic* and specifies the position in the tree of the null category coindexed with the 'extracted' element. Finally, the rules *whobj1* and *relobj1* inherit from the rule *topobj1* and specify the type of the topicalized category: *wh*-word or relative word.

This organization in an inheritance hierarchy allows to capture linguistic generalizations: the wh-movement rules[2](topicalization, *wh*-questions, relative clauses) 'move' a constituent to the same position, the front of the clause; the fronted constituent can be a NP (if the 'extracted' element is the subject, a direct object or the object of a preposition), a PP (a prepositional object with no preposition stranding), an AP (adjective phrase) or an AdvP (adverb phrase). This constituent is associated with a gap corresponding to one of the arguments of the verb, and it shares the syntactic and semantic information of the gap: for example, a *wh*-pronoun can be an accusative form only if it corresponds to the object of the verb or of a preposition.

[1]We are only discussing the rules referred to in the HPSG literature as *filler-gap* constructions or strong unbounded dependency constructions.

[2]These constructions are discussed as a separate class of unbounded dependencies in the literature (Pollard and Sag 1996, see previous footnote).

The only information which is not shared is the type of the preposed constituent (unmarked, *wh*-word or relative word), which determines the type of the unbounded dependency.

Another example of rule organization is given by the passive rules: information is inherited along two different dimensions. First, the rule for simple passive, defined at the tree for transitive verbs, is inherited by trees lower in the hierarchy: for example, the tree for verbs with prepositional objects (V+PP, such as *look after*), inherits the information provided by the general passive rule, and need only specify idiosyncratic information (about the preposition of the original complement).

Second, the rule for passive with *by* inherits the information provided by the rule for simple passive and adds information relative to the prepositional phrase.

This hierarchical organization of rules captures the fact that there is one passive rule, which can vary depending on the object of the original transitive verb, and whether the agent is expressed or not. This cannot be captured if the different passive trees are represented independently of each other, with no more connection between them than between a passive tree and, for example, the gerund tree.

2.2 Application of rules

Not all rules are applicable to all trees, and not all orders of rule application are valid for all trees. There are three ways in which we constrain rule application:

- by the rule's position in the main hierarchy – as discussed above, rules applying to just a subset of trees can be located at the most general node defining that subset, and no other trees will be able to access the rule. The fact that inheritance is non-monotonic allows the expression of exceptions to rules: for example, transitive verbs which cannot passivize inherit from the general definition for transitive verbs, but add that the passive rule does not apply.

- by specifying conditions directly within the rule – for example, the passive rule can check that the first complement really is a noun phrase and fail to apply if not. Note that this may not be achievable pnrely by method (1) due to the possibility of applying other rules first: a transitive verb from which the direct object has been extracted will still be within the scope of the passive rule, but the rule will not be able to apply to it because the object has disappeared. Another example is the *wh*-question on the subject: the

rule should apply only if the subject is not an expletive pronoun, and this has to be checked by the rule itself.

- by explicitly specified constraints – although the previous two methods provide theoretically adequate mechanisms for all constraints, for efficiency reasons we also maintain a separate model of which rules can apply in which combination. The rules defined at each node are first grouped into sets if they enter into a paradigmatic relationship (if they cannot apply simultaneously on the same tree): this is the case in English for the extractions rules discussed earlier, and for the passive rules, for example. Rules and sets of rules are then ordered, according to a partial ordering of the rules, and all possible rule application sequences which respect that ordering are computed off-line. Not all these sequences will apply in all cases (due to the constraints of type (1) and (2)) but this is still much more efficient than blind search through *all* possible rule combinations.

This situation is reminiscent of the debate about rule ordering which took place in transformational grammar in the seventies (Soames and Perlmutter (1979)). One position defended an ordering of transformations, the other position maintained that ordering the rules is unnecessary, because rules should be allowed to apply whenever their structural description is met. In practical applications, however, this means computing and testing all possible rule combinations, which in the case at hand is impractical[3].

2.3 Grammar expansion and parsing

The LEXSYS grammar currently includes 44 basic trees and 35 rules which together expand to 619 trees. This is work in progress, and we predict that the number of trees will quickly grow. Also, we do not allow disjunctive feature values, but use multiple instances of the same tree, and this will also increase the number of trees. We currently expand the grammar as an off-line process before parsing. The high number of trees resulting from this expansion might be seen as a drawback for parsing, but techniques described in Evans and Weir (1998) can be applied to optimise the parsing of such large grammars by converting them to automata which can be merged and minimised.

[3]Rule application is an issue also in computational applications of lexicalist grammars which use rules, such as HPSG (Meurers and Minnen 1995).

3 Related work

Other work in this general area includes Becker (1993, 1994) and Vijay-Shanker and Schabes (1992). Somewhat more recently, Candito (1996) presented an approach which is somewhat different in spirit. In her approach, LTAG is viewed as the compilation of what she calls a *metagrammar*. This metagrammar is based on the notion of syntactic function and hierarchically organizes information along three dimensions: initial predicate-argument structure, redistribution of functions and surface realization of syntactic functions. These three types of information are combined to yield cross-classes, and there is a step of translation of these resulting classes into trees. Inheritance is monotonic, except for functional information (the redistribution of functions can overwrite the initial distribution of functions). This scheme does not provide for an efficient way to handle exceptions or subregularities: if a predicate does not select some trees belonging to its tree family, these trees have to be stipulated in the lexical entry of the predicate.

On the other hand, trees, in our approach, are directly organized in a non-monotonic inheritance hierarchy, so that there is no translation step. Our use of nonmonotonicity enables us to capture exceptions easily, but also contributes to the succinctness of some of our generalisations. A detailed comparison of the two approaches on a significant grammar fragment would therefore be very interesting.

References

Tilman Becker. 1993. HyTAG: *A new type of Tree Adjoining Grammar for hybrid syntactic representation of free word order languages.* Ph.D. thesis, Univ. des Saarlandes.

Tilman Becker. 1994. Patterns in metarules. In *Proceedings of the Third International Workshop on Tree Adjoining Grammars*, 9–11.

Marie-Hélène Candito. 1996. A principle-based hierarchical representation of LTAGs. *COLING-96*, Copenhagen, 194-199.

John Carroll, Nicolas Nicolov, Olga Shaumyan, Martine Smets and David Weir. 1998. The LEXSYS Project. In *Proceedings of the fourth TAG+ workshop.*

Roger Evans and Gerald Gazdar. 1996. DATR: A language for lexical knowledge representation. *Computational Linguistics* **22.2**, 167-216.

Roger Evans, Gerald Gazdar and David Weir. 1995. Encoding lexicalized tree adjoining grammars with a nonmonotonic inheritance hierarchy. *33rd*

Annual Meeting of the Association for Computational Linguistics, 77-84.

Roger Evans and David Weir. 1998. A structure-sharing parser for lexicalised grammars. *COLING/ACL 98.*

Walt Detmar Meurers and Guido Minnen. 1995. A Computational treatment of HPSG lexical rules as covariation in lexical entries. *Proceedings of the Fifth International Workshop on Natural Language Understanding and Logic Programming.*

K. Vijay-Shanker and Yves Schabes. 1992. Structure sharing in lexicalized tree-adjoining grammar. In *COLING-92*, 205–211.

Scott Soames and David Perlmutter. 1979. *Syntactic Argumentation and the Structure of English* University of California Press.

Formal Analyses of the Hungarian Verbal Complex

Temese Szalai and Edward Stabler
UCLA Linguistics
3125 Campbell Hall
Los Angeles, CA 90095-1543

The verbal complexes in Dutch, German, and Hungarian have interesting structures, providing good tests for formal syntactic theories. These structures have posed a problem for theories in the transformational tradition that have assumed just two, distinctly different kinds of movement operations: strictly local, morphologically motivated head movement and unbounded phrasal movement (Chomsky 1986). The problem is that while some verbal complexes seem to consist of only heads, thus allowing a head-movement analysis, there are closely related constructions which involve projections larger than bare heads, thereby requiring phrasal movements of some kind. The (synchronic) similarities between these constructions and also historical considerations suggest that we are missing a generalization by proposing both a head movement analysis and a phrasal one. The TAG formalism does not rest on any assumption of distinctly different head vs. phrasal movement operations, and so the TAG analysis of West-Germanic verb raising proposed by Kroch and Santorini (1991) fares rather well.

This paper explores a new idea from the transformational tradition: an analysis of Hungarian and Germanic verbal complexes that involves phrasal movement only (Koopman & Szabolcsi, forthcoming, hereafter K & Sz). By dropping the assumption that there are two fundamentally different kinds of movement involved, this analysis avoids the problem with earlier transformational approaches to verbal complexes. Moreover, the essence of the analysis is easily formalized in a very simple fragment of transformational grammar that has been formalized by Stabler (1996, 1997). Like the TAG formalism, this formalism involves operations on trees. The proposed analysis cannot, however, be duplicated in the TAG formalism, because it is based on extensive "remnant movements", of the kind that have gotten a lot of attention especially since Kayne's (1994) influential proposals, and "heavy pied-piping" (Nkemnji 1995, Koopman 1996). (A remnant is a constituent from which extractions have taken place.) The Hungarian verbal complexes "roll up" the tree as remnants increasing in complexity without bound.

Specifically, K & Sz consider the data in the following paradigm, all of which mean "I will not want to begin to go home."

(1) Nem fogok akarni kezdeni hazamenni
NEG will+1S want-inf begin-inf home+go-inf

(2) Nem fogok akarni hazamenni kezdeni
NEG will+1s want-inf home+go-inf begin-inf

(3) Nem fogok hazamenni kezdeni akarni
NEG will+1s home+go-inf begin-inf want-inf

"Haza" is a verbal modifier (P) that cannot appear in sentence final position. Kenesei (1989) noted that sentences like (1) and (3) are (partial) mirror images of one another. Based on this insight, K & Sz observe that Hungarian verbal constructions exhibit fully inverted orders, as in (3), non-inverted orders, as in (1), and partially inverted orders, as in (2). Thus, the following orders are possible:

V1V2 V3 P V4

V1 V2 P V4 V3

V1 P V4 V3 V2

There are restrictions, however, on the character of the partially inverted orders. Specifically, once a lower verb fails to invert its complement, this un-inverted string cannot be inverted by a higher verb. Thus, the following orders are impossible on the relevant reading:

*V1 [V2 [P [V3 [_ V4]]]]

*V1 [[V3 [P V4]] V2]
*V1 [V3[V2 [[P V4] _]]]

K & Sz propose that the acceptable patterns are derived by extracting the arguments of the verbs and then moving the VPs, now containing nothing except the verb, into larger and larger structures:

V1 V2 V3 [P V4] -> V1 V2 [[P V4] V3] ->
V1 [[P V4] V3] V2

In the linguistic literature, this type of movement is referred to as "remnant movement". This analysis also makes use of "heavy pied-piping" in which a feature of a sub-part triggers movement of a larger piece of the structure. K & Sz make a number of theoretical assumptions that dictate this type of strategy. The formalization of these assumptions forms the underpinnings of the analysis to be proposed here.

The assumptions as laid out by K & Sz are, first of all, that all languages are binary branching with underlying Spec-Head-Complement order, following Kayne (1994). Secondly, they adopt the Universal Base Hypothesis (Sportiche 1993, 1995, Cinque 1997, Koopman 1996), which requires that cross-linguistic variation be attributal to factors other than hierarchical differences. Further, they propose that certain categories (DP, CP and PredP) must be licensed by moving into the specifier of a special licensing projection (LP(DP), LP(CP), LP(PredP)). These licensing projections generalize the role of "CASE" in Case Theory. All movement must be overt and motivated by features. They further assume a number of restrictions on movement and principles that force movement. In particular, we have the COMP+ restriction, which is closely related to the Left Branch Condition.

COMP+ Restriction on Movement: A maximal projection can move if it meets either of the following two requirements.

(a) if it is the rightmost sister of a minimal projection and it has no ancestor which is the leftmost daughter of a maximal projection

(b) if it is the leftmost daughter of a maximal projection and that maximal projection is (1) the rightmost sister of a minimal projection and (2) has no ancestor which is the leftmost daughter of a maximal projection

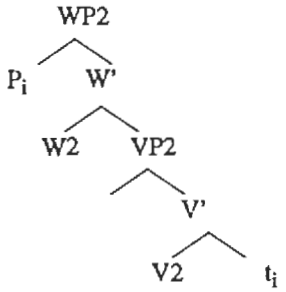
In addition, they assume the following two principles, from Koopman (1996), which force movement in a number of cases.

Principle of Projection Activation (PPA): A projection is interpretable iff it has lexical material at some stage in the derivation

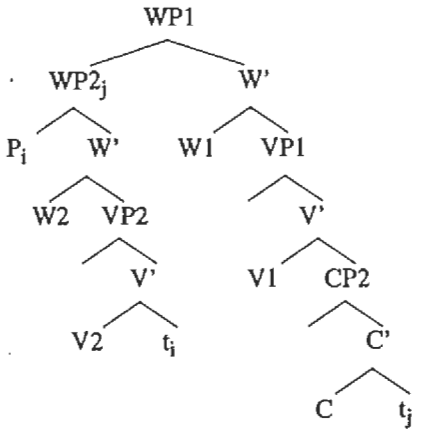
Modified LCA: No projection has both an overt Spec and an overt head at the end of the derivation.

These principles in combination with the restrictions on movement simplify the syntactic analysis of the above data quite considerably. The derivations are reduced to a more-or-less mechanical operation in which consituents "roll up" the tree. Word order differences come from limited sources of optionality. One source of optionality is the amount of material that can pied-pipe. The other source of optionality is the optionality of the functional category PredP, which is discussed in more detail below. A skeleton derivation for an inverted order involving only two verbs has been schematized below to illustrate the character of this analysis.

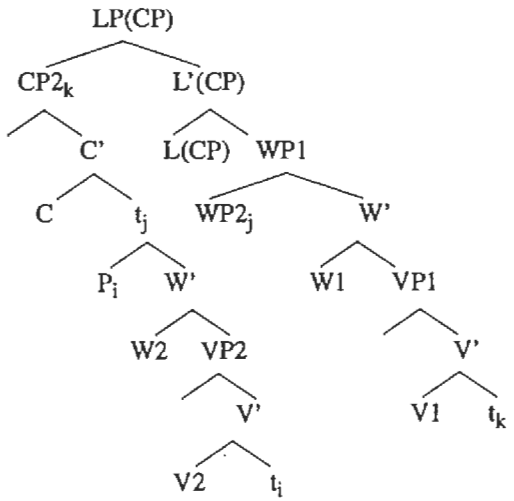
(1) WP is an extension of VP. All VPs are dominated by a WP. Spec, WP should always be filled. When there is no particle (P) or lower WP or CP to fill this position, the entire VP can move into Spec, WP.



(2) All WPs are dominated by a CP. This CP can be selected by another auxiliary (here V1). VP1 is dominated by a WP. The lower WP moves into the Spec of the higher WP.



(3) K & Sz generalize Case theory to categories not traditionally thought to require licensing. This results in licensing projections (LPs) for DPs, CPs and PredPs (to be discussed below). An LP wants an XP of the appropriate category in its Specifier. The lowest CP in this derivation did not have an LP(CP) because there was no lower CP that needed to be licensed.



Above we have a fully inverted order. To obtain an un-inverted order, the sources of optionality, namely presence of PredP and amount of pied-piped material, need to be exploited.

Inspired by Koster (1994) and Zwart (1994, 1997), PredP is really another extension of the VP which obligatorily dominates WP in certain circumstances. WP will move to Spec, PredP. PredP must then be licensed in an LP(PredP) position. WP will then cause large portions of structure to pied-pipe.

Using the type of strategy outlined above it is possible for constituents to "roll up" the tree, forming unbounded dependencies. The technology proposed by K & Sz can be used to generate $a^nb^nc^nd^ne^n$ type languages. In fact, this style of derviation derives languages well outside the class of mildly context sensitive languages. In this framework, the same kind of derivation, "rolling up" constituents by moving remnants, easily derives the language $a^nb^nc^nd^ne^n$. Roughly,

... -> eaabbccdd[ee] -> [ee]eaabbccdd ->
d[ee]eaabbccdd -> [dd]d[ee]eaabbcc -> ...
-> aaabbbcccdddeee

In fact, it is possible to obtain unboundedly many counting dependencies in this fashion.

These derivations require very large trees which make use of very little recursion, although extensive use is made of mechanical operations to ensure regularities between structures. This suggests that, if K & Sz are on the right track, TAG formalisms of their analysis would require many large elementary trees, leaving important regularities to the characterization of the elementary tree set.

It is easy to adapt Stabler's (1996, 1997) Derivational Minimalism to formalize this type of derivation, using only phrasal movement from certain structural configurations to derive the acceptable structures without allowing the unacceptable ones. This adaptation will then also allow unbounded counting dependencies to be captured in Derivational Minimalism, which has already been shown to be capable of capturing copying languages (Cornell 1996, Stabler 1997).

To adapt this derivation to Stabler's framework, certain aspects of the K & Sz proposal need to be formalized.The COMP+ restriction on movement has already been discussed. The additional principles and restrictions on movement can be formalized in terms of features in the Derivational Minimalism framework. For example, the requirement that all movement be overt is translated into Derivational Minimal-

ism by requiring that all attractor features be strong (+X features only).

The PPA will be formalized by requiring that all lexical entries bear at least one strong attractor feature. This will ensure that all minimal projections have something in their specifiers at some stage of the derivation.

To capture the Modified LCA , all empty heads will have a strong attractor feature. Additionally, a mechanism will be established to verify that when an overt lexical item licenses a constituent that constituent has additional licensee features if it is overt.

The universal base that K & Sz assume is ensured through feature selection. Lexical items will select features in the following order for the relevant domain:

lpred >> lc >> ld >> pred >> inf >> w >>v

Using these mechanisms, it is easy to formalize the basics of the K & Sz analysis in Derivational Minimalism. Because the formalism is so simple and the analysis so mechanical, the prospects here look quite good. Additionally, the type of analysis proposed here allows for any number of counting dependencies to be enforced. Derivational Minimalism can handle these dependencies quite simply, by "rolling up" constituents. Languages like these cannot be defined in standard TAGs (Vijay-Shanker and Weir 1994). The lack of recursion makes this type of analysis challenging in standard TAGs. Moreover, the regularities of the data will not be readily observable as the regularities seem to have mechanical properties.

References

Chomsky, N. 1986. *Barriers*. MIT Press, Cambridge, Massachusetts.

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

Cornell, T. 1996. A minimalist grammar for the copy language, SFB 340 Technical Report #79, University of Tubingen, http://www.sfs.nphil.uni-tuebingen.de/~cornell/"

Kayne, R. 1994. *The Antisymmetry of Syntax*. MIT Press, Cambridge, Massachusetts.

Kenesei, I. 1989. "Logikus-e a magyar szorend?" *Altalanos Nyelveszeti Tanulmanyok 17*, 105 - 152.

Koopman, H. and A. Szabolcsi. *Verbal Complexes*. forthcoming.

Koopman, H. 1996. The Spec-Head Configuration. In E. Garret and F. Lee, eds., *Syntax at Sunset, UCLA Working Papers in Syntax and Semantics* 1.

Koster, J. 1994. "Predicate Incorporation and the Word Order of Dutch" in G. Cinque, J.-Y. Pollock, L. Rizzi and R. Zannuttini, eds, *Paths Towards Universal Grammar*, 255-276, Georgetown Univ. Press: Washington, D.C..

Kroch, A.S. and B. Santorini. 1991. The derived constituent structure of the West Germanic Verb Raising construction. In Freidin, R., ed., *Principles and parameters in comparative grammar*, pages 269-338. MIT Press, Cambridge, Massachusetts.

Nkemnji, M. 1995. Heavy pied-piping in Nweh. Ph.D. Dissertation, UCLA.

Sportiche, D. 1995. Sketch of a reductionist approach to syntactic variation and dependencies. In H. Campos and P. Kempchinsky, eds., *Evolution and Revolution in Linguistic Theory*, pages 356-398. Georgetown University Press, Washington, D.C..

Stabler, E. 1996. Computing quantifier scope. In A. Szabolcsi, ed., *Ways of Scope Taking*, Kluwer, Boston.

Stabler, E. 1997. Derivational Minimalism. In C. Retore, ed., *Logical Aspects of Computational Linguistics*, pages 68-95. Springer-Verlag (Lecture Notes in Computer Science 1328), NY.

Vijay-Shanker, K. and D. Weir. 1994. The equivalence of four extensions of context free grammar formalisms. *Mathematical Systems Theory*, 27:511-545.

Zwart, C.J.-W. 1994. Dutch is Head Initial. *The Linguistic Review*, 11:377-406.

Zwart, C.J-W. 1997. *Morphosyntax of Verb Movement*. Kluwer, Dordrecht

Translating the XTAG English Grammar to HPSG

Yuka Tateisi†, Kentaro Torisawa†, Yusuke Miyao†, and Jun'ichi Tsujii†‡
†Department of Information Science, Graduate School of Science, University of Tokyo*
‡CCL, UMIST, U. K.
E-mail: {yucca, torisawa, yusuke, tsujii}@is.s.u-tokyo.ac.jp

1 Introduction

We describe the development of XHPSG, a large-scale English grammar in the HPSG formalism translated from the XTAG grammar (The XTAG Research Group, 1995). Our goal is to obtain a large-scale, linguistically sound grammar for our HPSG parser (Makino et al., 1998) with a relatively small workload. For this purpose, we try to make an HPSG grammar equivalent to the XTAG grammar in the strong sense where we preserve the structures and the linguistic analysis of the XTAG grammar.

To guarantee the equivalence of the XHPSG and XTAG grammars, the following conditions must be satisfied: 1) An XTAG elementary tree is translated to an XHPSG lexical item that translates back to the original elementary tree by applying the schemata and principles; 2) No XHPSG lexical item translates back to a tree other than the original XTAG elementary tree; 3) Substitution and adjunction allowed in the original grammar, and no other opeations, are simulated in the XHPSG parsing.

We not only use the HPSG formalism to express the linguistic analyses of the XTAG grammar, but also preserve, as much as possible, the general framework of the linguistic analyses given in the standard HPSG (Pollard and Sag, 1994). We use the standard HPSG schemata and the principles that are concerned with syntax, and translate the XTAG elementary trees into lexical feature structures so that they satisfy the conditions 1), 2) and 3) with them. Given that the XTAG features are used for controlling the substitution and adjunction, the condition 3) is reduced to the problem of whether or not all the XTAG features can be mapped to HPSG feature structures so that their values are properly propageted by the application of those schemata and principles.

*This work is partially founded by Japan Society for the Promotion of Science (JSPS-RFTF96P00502).

2 Translation

We start with the standard HPSG feature structure and schemata[1] with slight modification and addition. As for principles, we use phonology-, head-feature-, valence-, non-local feature-, spec- and marker-principles.

We separate the translations to two steps. First, we translate the tree structure of elementary trees to HPSG feature structures. Second, we map the XTAG feature into the HPSG structure.

2.1 Translation of the tree structure

In most initial trees, labels of the nodes on the trunk (the path from the anchor to the root) are the projections of that of the lexical anchor. On the other hand, in HPSG, labels are expressed by a part of the HEAD and the VALENCE features. The HEAD feature corresponds to the projection of a category. For example, VP is expressed as a structure whose HEAD is *verb* and COMPS is saturated and S is expressed as a structure whose HEAD is *verb* and both COMPS and SUBJ is saturated (Figure 1). Thus, if no features are concerned, the nodes on the trunk corresponds to the HEAD feature[2] and we can construct the lexical feature structure corresponding to an initial tree by translating the label of the nodes on

[1]Refer to (Pollard and Sag, 1994) for the meaning of the standard HPSG features and schemata. We use the following abbreviations for feature names: SS=SYNSEM, LOC=LOCAL, NONLOC=NONLOCAL, VAL=VALENCE, MARK=MARKING.

[2]In a few initial trees whose label of the root is different from that of the anchor, we set the HEAD feature according to the root node because of the substitution.

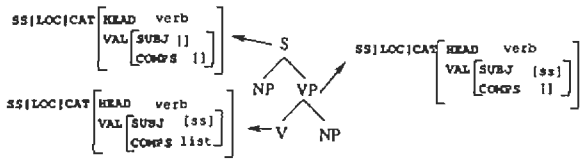


Figure 1: Correspondence between node labels and feature structures

the trunk into the HEAD features and remaining nodes into SUBJ, COMPS or SLASH features according to the syntactic role of the nodes.

For example, α_1 in Fig 2 shows the tree for a transitive verb *like*. As an HPSG schema corresponds to one branching of the tree, the tree can be re-constructed if a proper schema is selected for each branching. In this case, applying head-complement, and head-subject in this order will restore the shape of a tree and properly project the information contained in the anchor to the root of the tree[3].

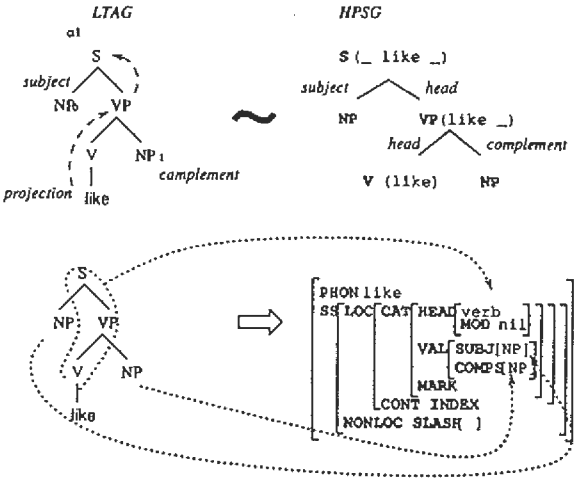


Figure 2: Translation of verb *like*

In auxiliary trees, we translate the anchor to the HEAD and leaf node other than the foot node to an element of either SUBJ, COMPS, or SLASH list, as in initial trees. However, as an auxiliary tree adjoins to a node of another tree, the label and the structure of the adjoined tree must be preserved after the adjunction. Therefore, in the application of a schema corresponding to a branching just above the foot node, the HEAD,VAL and SLASH must be preserved (Figure 3). Considering this property and the fact that

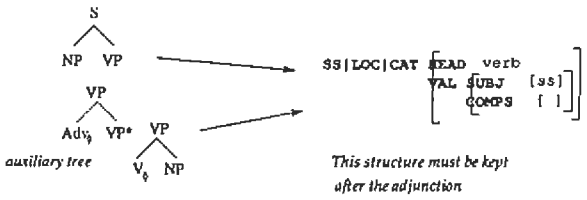


Figure 3: Adjunction

the auxiliary tree selects the node it adjoins, it seems natural to put the foot node into the MOD or SPEC feature so that the head-adjunct or head-marker schema will be applied to form the branching just above the foot node in a way that the auxiliary tree becomes the adjunct or the marker (Figure 4). As adjunction involves the propagation of other feature values, the translation of an auxiliary trees is re-examined in section 2.2.

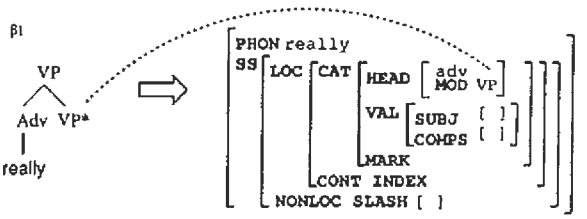


Figure 4: Translation of modifiers

2.2 Translation of the feature and the auxiliary trees

Generally, the HPSG features propagate from the daughters to the mother as in Table 1[4].

We put the XTAG features that have an equivalent in the HEAD features of the standard HPSG into XHPSG HEAD features. We call these XTAG features HEAD' features. We observed that the values of HEAD' features are propagated from the node defined as the head-daughter in section 2.1 on the branching that is

[3]The specification of schemata presented in (Pollard and Sag, 1994) ensures that the schemata are applied in proper order. For example, the condition on head-subject schema that the COMPS of the head daughter must be empty ensures that the subject comes above the complements on the tree structure.

[4]In exceptional cases, the propagation is explicitly marked in the lexical item

Table 1: Propagation of HPSG features

Feature	Schema head-subj	head-comp	head-adj	head-mark	head-fill
HEAD	H	H	H	H	H
MARKING	H	H	H	n	H
INDEX	A	H	e	H	H
SLASH	e	e	e	e	U

H:the value propagates from head-daughter to the mother n:the value propagates from head-daughter to the mother e:the value propagates from either daughter A:the value must agree between both daughters U:the value of the feature on the head-daughter unifies the non-head daughter

not the one just above the foot node of an auxiliary tree. We put the *agr* feature into INDEX feature and *trace* into the SLASH feature.

Among the remaining features, the ones whose value propagates in the same way as the HEAD' feature values are also put into the XHPSG HEAD feature, and the others are put into the MARKING feature. Table 2 shows where the XTAG features are put into in the XHPSG feature structure. Now, we re-examine

Table 2: Correspondence of XTAG and XHPSG features

XTAG feature	XHPSG feature
assign-case, assign-comp, case*, extracted*, inv*, mainv*, mode*, passive, perfect, pred, progressive, pron, tense*	HEAD
card, comp, const, decrease, definite, gen, neg, quan, sub-conj, wh	MARKING
agr	INDEX
trace	SLASH

The features marked with * has a counterpart HEAD feature in the standard HPSG analysis.

the translation of tree structure regarding foot nodes. We determine the schema to be applied to the branching above the foot node as follows: If the auxiliary tree changes the value of the HEAD feature on adjunction, we apply the head-complement schema on the branching just above the foot node. In this case, the foot node becomes a complement just as in the case of substitution node[5] (Figure 5); If the auxiliary tree does not change the value of the HEAD node but changes the value of the MARKING feature on

[5]The tree structure of the tree that this kind of auxiliary tree adjoins to is kept by letting the appropiriate part of the VALENCE feature structure-share betweeu the head and the complement.

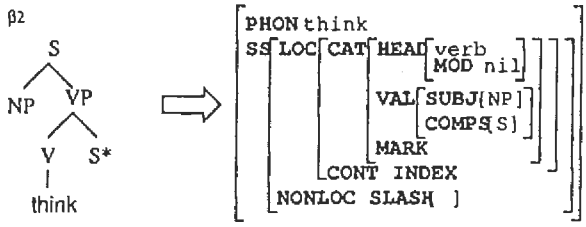


Figure 5: Translation of verbs that take a sentential complement

adjunction, we apply the head-marker schema on the branching just above the foot node. In this case, the foot node becomes the head to be selected by the SPEC feature of the lexical feature structure corresponding to the auxiliary tree (Figure 6); Otherwise, we apply the head-

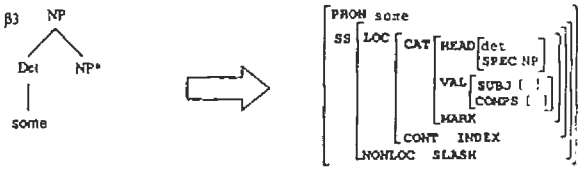


Figure 6: Translation of determiners

modifier schema on the branching just above the foot node. In this case, the foot node becomes the head to be selected by the MOD feature of the lexical feature structure corresponding to the auxiliary tree [6] (Figure 4).

3 Problems

Though in most cases the abovementioned translation works, there are a few exceptional cases. In this section, we mention two cases. The first one is a treatment of bar level, the second is predicatives.

3.1 Bar level

As mentioned in section 2.1, bar-level is not explicitly marked in the standard HPSG (see Figure 1), but implicitly stated in VALENCE features. In consequence, there is no distinction between a word who has no arguments and the phrase just consists of that word.

[6]In the latter two cases, the tree structure of the tree that the auxiliary tree adjoins to is kept by the valence principle and the head-marker (head-adjunct) schema.

This caused a problem when modifiers are involved. For example, there is no way to prevent a noun-modifying adjective from modifying an NP as there is no distinction between N with no arguments and NP.

To solve the problem we introduced features named XP and ASSIGN_XP. XP is used for restrict the modifiee's bar-level. ASSIGN_XP is used by a modifier to assign a bar-level to a phrase generated as a result of modification.

3.2 Predicatives and small clause

In XTAG analysis, a predicative noun[7] has a tree whose root is labeled S and the copula *be* has an auxiliary that adjoins to the tree. We assigned a head feature verb to a predicative noun (see the footnote in section 2.1). However, we could not allow the extraction of the predicative noun, because it would be the head that is extracted. We splitted the lexical entry of *be* to handle the extraction.

4 Implementation

We have translated the syntactic lexicon of the XTAG grammar version 1.1 and implemented the translated grammar in LiLFeS language (Makino et al., 1998). We assumed only binary branching, and splitted the schemata according to whether the head is on the left or on the right.

Currently we have verified our grammar partially in the sense that XHPSG grammar generates the structures equivalent to the elementary trees and the trees constructed with one or less adjunction. For the general cases, we are currently working on constructing a structure equivalent to the derivation tree for XTAG parsing in XHPSG. The derivation trees will enable us to easily compare the parsing results between the original and the translated grammars to check the validity of XHPSG in a practical sense.

We optimized the grammar by pre-compiation (Torisawa and Tsujii, 1996) and measured the parsing time of the ATIS corpus using the two-phased parsing of the precompiled XHPSG. The average user time was 1.12 seconds on Alpha Station (400MHz CPU, 4GB main memory). We expect a futher speed-up of the parsing by packing feature structures (Miyao et al., 1998).

[7] adjective and preposition also

5 Conclusion and Future Work

We translated the XTAG grammar to get a wide-coverage grammar in the HPSG formalism. By assigning an HPSG schema to a branching of XTAG trees, we have shown that the branching in XTAG trees can be licensed by the standard HPSG schemata and principles.

We are also interested in comparing our result to the HPSG English grammar being developed at Stanford University (CSLI, 1998) and to the CCG English grammar converted from XTAG grammar (Doran and Srinivas, to appear).

6 Acknowledgment

We are grateful to Prof. Aravind Joshi and the members of the XTAG group for their valuable helps and suggestions.

References

CSLI. 1998. English Resource Grammar Online. available in http://hpsg.stanford.edu/hpsg/ergo.html.

Christine Doran and B. Srinivas. to appear. Developing a wide-coverage CCG system. CSLI lecture notes edited by A. Abeille and O. Rambow.

Takaki Makino, Minoru Yoshida, Kentaro Torisawa, and Jun'ichi Tsujii. 1998. LiLFeS—towards a practical hpsg parser. to appear in Proc. COLING—ACL '98.

Yusuke Miyao, Kentaro Torisawa, Yuka Tateisi, and Jun'ichi Tsujii. 1998. Packing of feature structures for optimizing the hpsg-style grammar translated from TAG. submitted to TAG+ workshop.

Carl J. Pollard and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press.

The XTAG Research Group. 1995. A lexicalized tree adjoining grammar for english. Technical Report IRCS Research Report 95-03, IRCS, University of Pennsylvania. available in http://www.cis.upenn.edu/~xtag/tech-report/tech-report.html.

Kentaro Torisawa and Jun'ichi Tsujii. 1996. Computing phrasal-signs in HPSG prior to parsing. In *Proc. Coling 96*, pages 949–955.

CONSISTENT DENDRIFICATION: TREES FROM CATEGORIES

A.M. Wallington
Department of Language Engineering, UMIST, Manchester M60 1QD, UK.

1. Introduction

I shall start by taking a fairly simple Combinatory Categorial Grammar (CCG) of the type developed by Steedman over the past decade or so (e.g. Steedman 1996) including rules of functional application, and functional composition. I shall have nothing to say about functional substitution in this paper, and shall assume that there are type-raised categories in the lexicon (e.g. S/(S\NP)). I shall also assume, following Steedman, that syntactic symbols such as S, NP, S\NP are in fact abbreviations for feature bundles.

From a Phrase Structure Grammar (PSG) perspective, a CCG derivation that uses functional composition, if interpreted as building a structural level of representation, can give rise to some very strange looking trees containing some very unusual node labels. Whereas certain labels correspond to PSG ones (e.g. VP = S\NP), others do not (e.g. S/NP), Furthermore, because certain analyses require a rule of composition, such trees and labels will be required. If there is anything at all "real" about traditional PSG categories for languages such as English, then on the face of it, CCG fails to capture them. There is a related point. If these strange categories such as S/NP need to be assembled, then one would expect that some lexical items would require such a category either as an argument or as the result. But, there seem to be curiously few such words and possibly no verbs.

What we shall do in this paper is examine how CCG categories can correspond to trees (cf. Joshi & Kulick 1996 and Henderson 1992 for other approaches). We shall see that interpreting a lexical CCG category as a partial description of a tree using a number of very simple principles will allow a number of "natural" distinctions to fall out without being stipulated. In particular, subjects but not objects will be immediately dominated by the S, different types of "empty" categories will be predicted; and structural differences between raising and control verbs will be observed. If the lexicon is constrained so that the categories can be interpreted as trees in the manner we shall describe, and if during the course of a successful derivation such trees can then be combined with other trees, then we shall say that the lexicon is constrained by a principle of "Consistent Dendrification".

2. Hypothesising Trees

As a start towards interpreting a lexical CCG category (e.g. X\Y) as a partial description of a tree, we shall assume that a category does a maximum of three things: it "names" certain nodes within a subtree (a crucial point we shall return to is that these may not be unique nodes); it describes a minimum of dominance relations (not necessarily immediate dominance); and where appropriate it describes relative precedence relations. For example, in the example given, X and Y would be two named nodes, X would dominate a subtree (is the root) which would contain the node Y and also a node dominating the lexical item (general principles which we shall spell out later determine how this item is named). Finally, because of the direction of slash, the Y argument subtree must be to the left of another node.

At this point the tree will be very under specified. However, we shall also assume a set of very general principles that can be applied to the minimum information specified in the category and these will allow other nodes to be hypothesised, named, and related to still more nodes in the tree. Finally, when a tree combines with another tree during the course of a derivation the resulting tree will be further specified.

2.1 *Principles and Conventions of Tree Building*

I shall first give two principles governing how nodes that have been hypothesised are labelled, then give two mechanisms for hypothesising nodes in a tree, and finally state a principle of economy that limits the number of nodes that can be hypothesised.

Principle of Full Correspondence: All (non-slash (and brackets)) labels in a category correspond to, i.e. they label, (not necessarily different) nodes in a tree.

For example, with the category S/(S/NP) ("whom"), nodes must have been hypothesised that can be labelled with an S_1 an S_2 and an NP, but crucially, the argument (i.e. S/NP) will not be used to label a node, because it has been separated into an S and an N. Suppose we were to an S/NP label; then, the tree will contain an S/NP node which does not correspond to any standard PSG node. If we wanted to relate CCG to standard trees, then we would have to give an

alternative category to words such as "whom" and a different analysis to long distance dependencies.

Naming Principle: Any node that has been hypothesised and does not correspond to a label in the category will be labelled with the label of the dominating node as the result part of the label and with the label of the other daughter of the dominating node as the argument part of the label. The position of this other daughter on the left or right will determine the direction of the slash.

Note that the Principle of Full Correspondence entails that functional nodes in the tree e.g. X\Y must be labelled by the Naming Principle. It will often be the case that the dominating node referred to in the Naming Principle is the nodes mother, and the other daughter is the nodes sister.

Lexical Anchor: A node is hypothesised that immediately dominates the lexical item.

Argument and Result Correspondence: (Not necessarily different) Nodes will be hypothesised to correspond to every argument (i.e. the right-hand-side of a slash), and to every result (i.e. left-hand-side of a slash) in a category.

Note the important difference between this mechanism governing the hypothesis of nodes in a tree and the Principle of Full Correspondence, governing the labelling of nodes. A node will be hypothesised for the argument S/NP in the S/(S/NP) category (and for the NP argument and the S and S results). However, it will not be labelled with a S/NP label.

We might also note the importance of the lexical anchor. Trees hypothesised from categorial grammar categories will be binary branching. Consequently, a minimal subtree will consist of three nodes. Of these three, the root node will correspond to the result part of the category and one of the daughter nodes will correspond to the argument part of the category. In higher reaches of the tree, the second daughter node will correspond to the root of a lower subtree. However, there are two situations in which this will not be the case. One such situation will be when the (functional) lexical category is split into the result and argument categories. A root node and a sister node will be hypothesised to correspond to this division, but a second daughter node will not have been hypothesised. The Lexical Anchor forms this node. The second situation can arise when an argument is itself a functional category. This will be the situation with the category of "whom" S/(S/NP). In this case, an S node will by hypothesised; an NP node, which must be on the right of some other node, will also be hypothesised, and a relation of dominance, although not necessarily immediate dominance, will be assumed between the two nodes. According to the Principle of Economy that we will introduce next, no other nodes can be hypothesised on the basis of the category of "whom". And, this is what we want, since if another daughter of S were hypothesised as a sister of the NP, then by the Naming Principle it would receive the label S/NP. It would not correspond to any conventional PSG category, and nor would it be found in trees hypothesised from simple transitive verbs, so preventing combination of trees. Finally, the NP is the object NP being questioned and such an NP can be arbitrarily low in the tree. We do not want to hypothesise exactly what this NPs sister is until the tree for "whom" has combined with trees hypothesised from other categories.

Principle of Economy: The smallest number of hypotheses about nodes, and dominance and precedence relations are made.

This principle entails that nodes and relations between nodes are not hypothesised without evidence. It also entails that if there is reason to hypothesise two nodes and these two nodes will receive the same label, then all things being equal the two labels will refer to the same node.

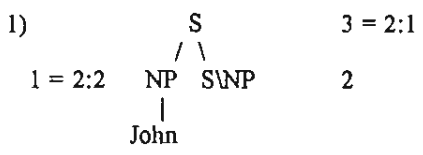
3. Sample Analyses

3.1 Type-Raised Subjects

Let us assume that the lexicon gives the following category for the proper noun "John" for use as a subject S/(S\NP). The assumption of a Lexical Anchor leads to the hypothesis of a node dominating "John" although at the moment it cannot be named. Let us call this node 1. By Argument and Result Correspondence, we can hypothesise two further nodes by splitting the category into a result part and an argument part. We shall call the node corresponding to the result node 3. Turning now to the argument, the right slash entails that there will be a node to the right of node 1 corresponding to the subtree hypothesised from the S\NP. Let us call this node 2. This subtree can also be split into an argument and a result. Consequently, we can at this point hypothesise two nodes for the subtree. By the Principle of Full Correspondence, we can label these an S and an NP. Let us call these nodes 2:1 and 2:2. Because of the left slash we also know that node 2:2 must appear on the left of some other, as yet unknown, node. Can we equate nodes 2 and 2:1, i.e. is the sister of the lexical anchor an S? At this point, this question cannot be answered since node 2:1 could also be a higher node that dominates node 2. At this stage we cannot choose between these two options, so we will leave the node unlabelled.

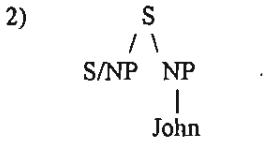
We can now turn to node 3, i.e. the node corresponding to the result part of the S/S\NP category. Since the result cannot be split into a result and an argument, we can label it with an S by the Principle of Full Correspondence.

We can return to the earlier hypotheses. The node corresponding to the S\NP argument (i.e. node 2) was required to be dominated by an S (node 2:1). The just hypothesised root node (node 3) will dominate this node and so by the Principle of Economy we shall equate nodes 3 and 2:1. Node 2:1 dominates an NP, node 2:2, which must appear on the left. We have equated nodes 3 and 2:1. There is an as yet unlabelled node on the left that is dominated by node 3 and that is node 1, the lexical anchor. Consequently, we shall equate nodes 1 and 2:2. Node 2 has not yet been labelled. However, its sister is labelled NP, and its mother is labelled S. Consequently, by the Naming Principle, node 2 will be labelled S\NP. In other words, the tree corresponding to a type-raised subject is the following:



Assuming a correspondence between an S\NP and a VP, this is the correct result.

Suppose that the lexicon contained an S\(S/NP) category for a type-raised object. It should be clear that if this were the case the resulting tree would be as depicted in 2.

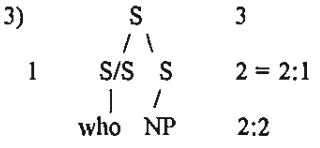


Not only does such a tree contain the S/NP label that does not correspond to a PSG label, it will not be able to combine with any tree that does not also include a S/NP label as the daughter of the S. In particular it will not be able to combine with the tree hypothesised from a simple transitive verb. In other words, if the categories in the lexicon will be interpreted as trees, then the type of category that may occur will be constrained. We can say that the lexicon is constrained by a requirement of consistent dendrification.

Wh-words

We shall assume that categories for the question words "who" and "whom" are S/(S\NP) and S/(S/NP) respectively. Notice that in terms of major features the category of a subject wh-word and that of a type-raised subject are identical. However, we have assumed, following Steedman, that labels are in fact feature bundles, and we shall assume that an S label with interrogative force has a +int feature. Consequently, a fuller description of these categories would be: S+int/(S-int\NP) and S+int/(S-int/NP).

I shall take the subject wh-word first. In the previous example, we assumed that the two Ss referred to the same node. However, in these examples, they differ with respect to the `int` feature. Much of the procedure for hypothesising a tree proceeds as before, but since the two Ss are no longer identical nodes 3 and 2:1 cannot be equated. If nodes 3 and 2:1 cannot be equated, then one S will be dominated by the other S and it will be nodes 2 (i.e. the node corresponding to the S\NP argument) and 2:1 (i.e. the result part of the S\NP argument) that will be equated. Node 2:1 dominates an NP, node 2:2. This time no other node has been hypothesised that can be equated with node 2:2. In particular, node 2:2 will not be equated with the lexical anchor node 1. A consequence of this is that no node has been hypothesised as a sister of the NP node. As discussed earlier, such a node will only be introduced when this tree combines with another tree that has an S root, an NP on the left (or right if the category is the object we-word) and a sister of the NP. Again as discussed earlier, the absence of a sister node means that the NP may be arbitrarily far from the S. Finally, if the lexical anchor (node 1) is not equated with node 2:2, then it must be named by the Naming Principle. Its mother is an S node (node 3) and its sister is also an S node (node 2:1). Consequently, the node dominating the word "whom" has the category S/S. The tree then consists of the wh-word chomsky-adjoined on the left side of a declarative sentence as depicted in 3. This again is the result we want.



3.3 Subject Raising Verbs

I shall assume that if we restrict ourselves to major features, then the category for a raising verb such as "seem" and the category of a control verb such as "try" is the same: S\NP/(S\NP) (cf. Jacobson 1990 for an alternative view).

We shall proceed as usual. A lexical anchor will be hypothesised (node 1). The category splits into an argument corresponding to the S\NP (node 2) and result corresponding to another S\NP (node 3). The argument also splits into a result (node 2:1) and an argument (node 2:2). Node 2:1 will dominate node 2:2. Since both of the categories corresponding to these nodes are atoms, these nodes will be labelled with an S and an NP respectively.

In this case, the result node (node 3) corresponds to a functional category and so node 3 will not be immediately named, and will be dominated by a node corresponding to the result (node 3:1) which will also dominate a node corresponding to the argument (node 3:2). The result of the result (i.e. the node corresponding to the S) cannot be split into an argument and result and so by the Principle of Full Correspondence, it will be labelled with an S. This is the root of the tree. Similarly, the argument of the result cannot be split, and so node 3:2 will be labelled with an NP. By the Naming Principle, node 3, which has an S mother and NP sister will be labelled S\NP.

If we have hypothesised nodes and labels for the result part of the lexical item, we can turn to the argument part. The node corresponding to this is node 2. The subtree corresponding to this node is dominated by an S (node 2:1). In this case node 3:1 is labelled with an S and dominates (although not immediately dominates) node 2. There appears to be no reason in terms of features not to equate nodes 3:1 and 2:1. However, if their daughter nodes 3:2 and 2:2 were labelled differently, then these could not be equated and as a consequence their mothers could not be equated. In this instance both are NPs and on the left. However, we might ask whether they differ in terms of minor features. In a raising construction, the subject NP has no independent theta-role projected by main verb. Its theta-role is projected from that of the subordinate verb. If we examine the lexical category, it is the subtree hypothesised from the result that will combine with the tree hypothesised from an adjacent verb. In other words NP 2:2 will be marked as taking an independent theta-role, and NP 3:2 marked as not having an independent theta-role. In such a situation, I shall assume that there is no possibility of theta-roles clashing, node 2:2 equates with node 3:2. If on the other hand, both NPs had been marked as taking independent theta-roles, then I will assume that the nodes could not be equated.

What about the label for node 2? Since node 2 was hypothesised to be dominated by an S (node 2:1) which also dominates an NP (node 2:2), it will also be labelled S\NP. We can finally return to the lexical anchor. The node corresponding to the result (node 3) that dominates it is labelled with an S\NP, and the node corresponding to the argument is also labelled with an S\NP, and so this node is labelled with an S\NP\(S|NP).

```
4)                  S                    3:1 = 2:1
                   / \
  3:2 = 2:2  NP    S\NP                  3
                  / \
  1      S\NP/(S\NP)  S\NP                2
                  |
                seem
```

Control Verbs

I shall assume that a verb such as "try" has the same category as "seem", the only difference being that the two NPs have independent theta-roles. A consequence of this difference is that nodes 3:2 and 2:2 cannot be equated. This in turn entails that the two S nodes (3:1 and 2:1) cannot be equated. Instead, node 2 will be equated with node 2:1, and will dominate node 2:2, which will have no hypothesised node yet as a sister. Finally, the label of the lexical anchor will be different from that given to it in the case of "seem". It will be dominated by an S\NP and its sister will be an S. Hence the label will be S\NP/S.

```
5)                S                    3:1
                 / \
  3:2         NP   S\NP                 3
                  / \
  1          S\NP/S  S                  2 = 2:1
                 |   /
               tried NP                 2:2
```

Henderson, J. 1992. "A Structural Interpretation of Combinatory Categorial Grammar." Technical Report MS-CIS-92-49, CIS, University of Pennsylvania.

Jacobson, P. 1990. "Raising as Function Composition." Linguistics & Philosophy 13, 423-476.

Joshi, A and S Kulick. 1997. "Partial Proof Trees as Building Blocks for a Categorial Grammar." Linguistics & Philosophy 20 637-669.

Steedman, M. 1996. "Surface Structure and Interpretation". Cambridge, Mass.: MIT Press.

Consistent Grammar Development Using Partial-Tree Descriptions for Lexicalized Tree-Adjoining Grammars

Fei Xia, Martha Palmer, K. Vijay-Shanker, Joseph Rosenzweig
Institute for Research in Cognitive Science
University of Pennsylvania
400A, 3401 Walnut Street
Philadelphia, PA 19104,USA
fxia/mpalmer/vshanker/josephr@linc.cis.upenn.edu

1 Introduction

An important characteristic of an FB-LTAG is that it is lexicalized, i.e., each lexical item is anchored to a tree structure that encodes subcategorization information. Trees with the same canonical subcategorizations are grouped into tree families. The reuse of tree substructures, such as wh-movement, in many different trees creates redundancy, which poses a problem for grammar development and maintenance (Vijay-Shanker and Schabes, 1992). To consistently implement a change in some general aspect of the design of the grammar, all the relevant trees currently must be inspected and edited. Vijay Shanker and Schabes suggested the use of hierarchical organization and of tree descriptions to specify substructures that would be present in several elementary trees of a grammar. Since then, in addition to ourselves, Becker, (Becker, 1994), Evans et al. (Evans et al., 1995), and Candito(Candito, 1996) have developed systems for organizing trees of a TAG which could be used for developing and maintaining grammars.

Our system is based on the ideas expressed in Vijay-Shanker and Schabes, (Vijay-Shanker and Schabes, 1992), to use partial-tree descriptions in specifying a grammar by separately defining pieces of tree structures to encode independent syntactic principles. Various individual specifications are then combined to form the elementary trees of the grammar. Our paper begins with a description of our grammar development system and the process by which it generates the Penn English grammar as well as a Chinese TAG. We describe the significant properties of both grammars, pointing out the major differences between them, and the methods by which our system is informed about these language-specific properties. We then compare our approach to other grammar development approaches for LTAG such as the specification of TAGs in DATR (Evans et al., 1995) and Candito's implementation (Candito, 1996).

2 System Overview

In our approach, three types of components – subcategorization frames, blocks and lexical redistribution rules – are used to describe lexical and syntactic information. Actual trees are generated automatically from these abstract descriptions. In maintaining the grammar only the abstract descriptions need ever be manipulated; the tree descriptions and the actual trees which they subsume are computed deterministically from these high-level descriptions.

2.1 Subcategorization frames

Subcategorization frames specify the category of the main anchor, the number of arguments, each argument's category and position with respect to the anchor, and other information such as feature equations or node expansions. Each tree family has one canonical subcategorization frame.

2.2 Blocks

Blocks are used to represent the tree substructures that are reused in different trees, i.e. blocks subsume classes of trees. Each block includes a set of nodes, dominance relation, parent relation, precedence relation between nodes, and feature equations. This follows the definition of the tree descriptions specified in a logical language patterned after Rogers and Vijay-Shanker(Rogers and Vijay-Shanker, 1994).

Blocks are divided into two types according to their functions: subcategorization blocks and transformation blocks. The former describes structural configurations incorporating the various information in a subcategorization frame. For example, some of the subcategorization blocks used in the development of the English grammar are shown in Figure 1.[1]

When the subcategorization frame for a verb is given by the grammar developer, the system will automatically create a new block (of code) by essentially selecting the appropriate primitive subcategorization blocks corresponding to the argument information specified in that verb frame.

The transformation blocks are used for various transformations such as wh-movement. These transformation blocks do not encode rules for modifying trees, but rather describe the properties of a particular syntactic construction. Figure 2 depicts our representation of phrasal extraction. This can be specialized to give the blocks for wh-movement, topicalization, relative clause formation, etc. For example, the wh-movement block is defined by further specifying that the ExtractionRoot is labeled S, the NewSite has a +wh feature, and so on.

[1]In order to focus on the use of tree descriptions and to make the figures less cumbersome, we show only the structural aspects and do not show the feature value specification. The parent, (immediate dominance), relationship is illustrated by a plain line and the dominance relationship by a dotted line. The arc between nodes shows the precedence order of the nodes are unspecified. The nodes' categories are enclosed in parentheses.

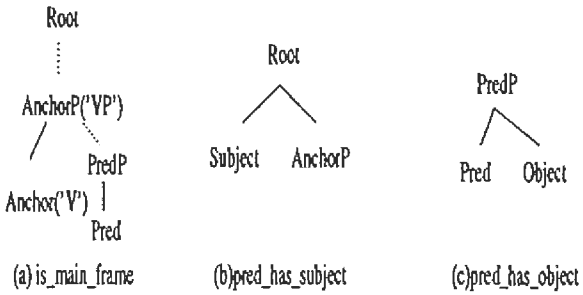


Figure 1: Some subcategorization blocks

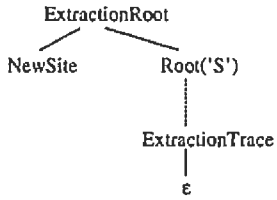


Figure 2: Transformation block for extraction

2.3 Lexical Redistribution Rules (LRRs)

The third type of machinery available for a grammar developer is the Lexical Redistribution Rule (LRR). An LRR is a pair (r_l, r_r) of subcategorization frames, which produces a new frame when applied to a subcategorization frame s, by first *matching*[2] the left frame r_l of r to s, then combining information in r_r and s. LRRs are introduced to incorporate the connection between subcategorization frames. For example, most transitive verbs have a frame for active(a subject and an object) and another frame for passive, where the object in the former frame becomes the subject in the latter. An LRR, denoted as passive LRR, is built to produce the passive subcategorization frame from the active one. Similarly, applying dative-shift LRR to the frame with one NP subject and two NP objects will produce a frame with an NP subject and an PP object.

Besides the distinct content, LRRs and blocks also differ in several aspects:

[2]Matching occurs successfully when frame s is compatible with r_l in the type of anchors, the number of arguments, their positions, categories and features. In other words, incompatible features etc. will block certain LRRs from being applied.

- They have different functionalities: Blocks represent the substructures that are reused in different trees. They are used to reduce the redundancy among trees; LRRs are introduced to incorporate the connections between the closely related subcategorization frames.
- Blocks are strictly additive and can be added in any order. LRRs, on the other hand, produce different results depending on the order they are applied in, and are allowed to be non-additive, i.e., to remove information from the subcategorization frame they are being applied to, as in the procedure of passive from active.

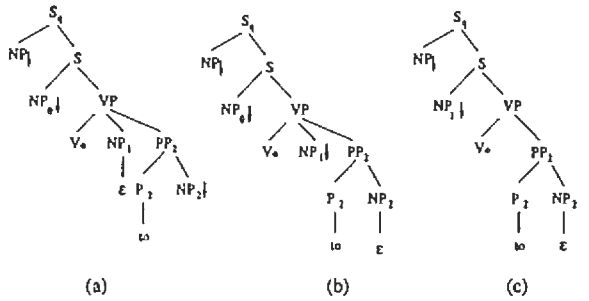


Figure 3: Elementary trees generated from combining blocks

2.4 Tree generation

To generate elementary trees, we begin with a canonical subcategorization frame. The system will first generate related subcategorization frames by applying LRRs, then select subcategorization blocks corresponding to the information in the subcategorization frames, next the combinations of these blocks are further combined with the blocks corresponding to various transformations, finally, a set of trees are generated from those combined blocks, and they are the tree family for this subcategorization frame. Figure 3 shows some of the trees produced in this way. For instance, the last tree is obtained by incorporating information from the ditransitive verb subcategorization frame, applying the dative-shift and passive LRRs, and then combining them with the wh-non-subject extraction block.

3 Generating grammars

We have used our tool to specify a grammar for English in order to produce the trees used in the current English XTAG grammar. We have also used our tool to generate a large grammar for Chinese. In designing these grammars, we have tried to specify the grammars to reflect the similarities and the differences between the languages. The major features of our specification of these two grammars are summarized in Table 1.

	English	Chinese
examples of LRRs	passive dative-shift ergative	bei-construction object fronting ba-construction
examples of transformation blocks	wh-question relativization declarative	topicalization relativization argument-drop
# LRRs	6	12
# subcat blocks	34	24
# trans blocks	8	15
# subcat frames	43	23
# trees generated	638	280

Table 1: Major features of English and Chinese grammars

By focusing on the specification of individual grammatical information, we have been able to generate nearly all of the trees (91.3% - 638 out of the 699) from the tree families used in the current English grammar developed at Penn[3]. Our approach, has also exposed certain gaps in the Penn grammar. We are encouraged with the utility of our tool and the ease with which this large-scale grammar was developed.

We are currently working on expanding the contents of subcategorization frame to include trees for other categories of words. For example, a frame which has no specifier and one NP complement and whose predicate is a preposition will correspond to PP → P NP tree. We'll also introduce a modifier field and semantic fea-

[3]We have not yet attempted to extend our coverage to include punctuation, it-clefts, and a few idiosyncratic analyses that are included in the sixty trees we are not generating.

tures, so that the head features will propagate from modifiee to modified node, while non-head features from the predicate as the head of the modifier will be passed to the modified node.

4 Comparison to Other Work

Evans, Gazdar and Weir (Evans et al., 1995) also discuss a method for organizing the trees in a TAG hierarchically, using an existing lexical representational system, DATR (Evans and Gazdar, 1989). Since DATR can not capture directly dominance relation in the trees, these must be simulated by using feature equations.

There are substantial similarities and significant differences in our approach and Candito's approach, which she applied primarily to French and Italian. Both systems have built upon the basic ideas expressed in (Vijay-Shanker and Schabes, 1992) for organizing trees hierarchically and the use of tree descriptions that encode substructures found in several trees. The main difference is how Candito uses her dimensions in generating the trees. Her system imposes explicit conditions on how the classes appearing in the hierarchy can be combined, based on which dimension they are in. For example, one condition states that only a terminal node (leaf node of a hierarchy) of the second dimension can be used in constructing a tree. Therefore two redistributions (such as passive and causative) can be used in a single tree only when a new passive-causative terminal node is first created manually. In contrast, our approach automatically considers all possible applications of LRRs, and discards those that are inconsistent.

5 Conclusion

We have described a tool for grammar development in which tree descriptions are used to provide an abstract specification of the linguistic phenomena relevant to a particular language. In grammar development and maintenance, only the abstract specifications need to be edited, and any changes or corrections will automatically be proliferated throughout the grammar. In addition to lightening the more tedious aspects of grammar maintenance, this approach also allows a unique perspective on the general characteristics of a language. Defining hierarchical blocks for the grammar both necessitates and facilitates an examination of the linguistic assumptions that have been made with regard to feature specification and tree-family definition. This can be very useful for gaining an overview of the theory that is being implemented and exposing gaps that remain unmotivated and need to be investigated. The type of gaps that can be exposed could include a missing subcategorization frame that might arise from the automatic combination of blocks and which would correspond to an entire tree family, a missing tree which would represent a particular type of transformation for a subcategorization frame, or inconsistent feature equations. By focusing on syntactic properties at a higher level, our approach allows new opportunities for the investigation of how languages relate to themselves and to each other.

References

Tilman Becker. 1994. Patterns in metarules. In *Proceedings of the 3rd TAG+ Conference*, Paris, France.

Marie-Helene Candito. 1996. A principle-based hierarchical representation of ltags. In *Proceedings of COLING-96*, Copenhagen, Denmark.

Roger Evans and Gerald Gazdar. 1989. Inference in datr. In *EACL-89*.

Roger Evans, 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 Association for Computational Linguistics(ACL '95)*, Cambridge, MA.

James Rogers and K. Vijay-Shanker. 1994. Obtaining Trees from their Descriptions: An Application to Tree Adjoining Grammars. *Computational Intelligence*, 10(4).

K. Vijay-Shanker and Yves Schabes. 1992. Structure sharing in lexicalized tree adjoining grammar. In *Proceedings of the* 15^{th} *International Conference on Computational Linguistics (COLING '92)*, Nantes, France.