Proceedings
Actes

Université Paris 7, Jussieu
May 25 - 27, 2000

TAG+5

5th International Workshop on
Tree Adjoining Grammars
and Related Formalisms

5e Atelier international
sur les grammaires d'arbres adjoints
et les formalismes apparentés

Les articles de ce volume ont été présenté lors du cinquième atelier international sur les Grammaires d'Arbres Adjoints (TAG) et autres formalismes proches, qui s'est déroulé à l'Université de Paris 7 du 25 au 27 mai 2000.

Les ateliers précédents se sont tenus à Dagstuhl en Allemagne (1990), à l'Université de Pennsylvanie (1992 et 1998) et à l'Université de Paris 7 (1994).

Nous espérons avoir respecté la traditionnelle représentativité de domaines variés liés aux TAG (linguistique, analyse syntaxique, développement et maintenance de systèmes à large couverture, etc.). Nous somme également heureux que soient inclus des articles portant sur des domaines proches comme LFG, HPSG ou les CCG.

Nous voudrions remercier tous les auteurs pour leur travail. Nous remercions également les membres du comité de programme qui ont pris de leur temps pour relire les articles soumis. Leurs commentaires nous ont été particulièrement utiles.

Nous remercions également l'ATALA (Association pour le Traitement Automatique des Langues) , LexiQuest, l'IUF, SineQua, l'Université de Paris 7 et l'INRIA pour leur concours financier.

Nous remercions chaleureusement les deux conférenciers invités John Carroll et Mark Steedman.

Enfin, nous remercions particulièrement Sylvain Kahane ainsi que Owen Rambow, Aravind Joshi, Jennifer MacDougall et les organisateurs des précédentes éditions de TAG+ pour leurs précieux conseils.

Le comité d'organisation de TAG+5

The papers in this volume were presented at the fifth international workshop on Tree Adjoining Grammars and related frameworks (TAG+5), held at the University of Paris 7 in May 2000.

Previous TAG workshops were held at Dagstuhl in Germany (1990), at the University of Pennsylvania (1992,1998) and at the University of Paris 7 (1994).

We have tried to maintain the tradition of representing a broad spectrum of interests (linguistics, parsing, development and maintenance of large systems ...). We are also happy that this collection includes papers relating TAG to other frameworks such as CCG, HPSG and LFG.

We would like to thank all the authors for their work in preparing their papers, as well as all the members of the programme committee, for devoting time to reviewing, and for being very helpful in providing additional comments which allowed to ease our task.

We would also like to thank ATALA, LexiQuest, IUF, SINEQUA, University Paris 7 and INRIA for their financial support.

We are also grateful to our two invited speakers : John Carroll and Mark Steedman.

Finally, special thanks go to Sylvain Kahane and to Owen Rambow, Aravind Joshi, Jennifer MacDougall as well as to the other organizers of the previous TAG workshops for their precious advice.

The TAG+5 organizing committee

PROGRAMME COMMITTEE - COMITÉ DE PROGRAMME

Anne Abeillé (Univ. Paris 7 and IUF)
Tilman Becker (DFKI, Saarbrücken)
Tonia Bleam (Upenn, Philadelphia)
Pierre Boullier (Inria, Rocquencourt)
Marie-Hélène Candito (Univ. Paris 7 and LexiQuest)
John Carroll (Univ. Sussex)
Eric de la Clergerie (Inria, Rocquencourt)
Laurence Danlos (Univ. Paris 7 and Loria)
Christy Doran (ITRI, Brighton)
Hemant Darbari, (C-DAC, Pune Univ.,India)
Christophe Fouqueré (Univ. Paris-Nord)
Robert Frank (Johns Hopkins Univ.)
Ariane Halber (Nuance Paris)
Beth Ann Hockey (RIACS, California)
Aravind Joshi (UPenn, Philadelphia)
Sylvain Kahane (Univ. Paris 7 and Paris 10)
Patrice Lopez (DFKI, Saarbrücken)
Gertjan Van Noord (Rijksuniversiteit, Groningen)
Martha Palmer (UPenn, Philadelphia)
Hyun-Seok Park (Sungshin Women's Univ., Korea)
Owen Rambow (AT&T Research)
Christian Rétoré (IRISA/INRIA, Rennes)
James Rogers (Univ. of Central Florida)
Giorgio Satta (Univ. of Padova)
Bangalore Srinivas (AT&T Research, Bell Labs)
Mark Steedman (Univ. of Edinburgh)
Yuka Tateisi (Univ. of Tokyo)
K. Vijay-Shanker (Univ. of Delaware)
Bonnie Webber (Univ. of Edinburgh)
David Weir (Univ. of Sussex)

LOCAL ORGANIZING COMMITTEE – COMITÉ LOCAL D'ORGANISATION

Anne Abeillé (Paris 7)
Nicolas Barrier (Paris 7)
Sébastien Barrier (Paris 7)
Marie-Hélène Candito (Paris 7 and Lexiquest)
Lionel Clément (Paris 7)
Kim Gerdes (Paris 7)
Alexandra Kinyon (Paris 7 and U. Penn)
Patrice Lopez (DFKI, Saarbrücken)

INVITED SPEAKERS – CONFÉRENCIERS INVITÉS

John Carroll : "Engineering Parsers and Generators for Large Lexicalised Grammars"

Mark Steedman : "The Syntactic Process"

SPONSTORS – PARRAINAGES

L'Association pour le Traitement Automatique des Langues (ATALA)

L'Institut Universitaire de France (IUF)

La Société LexiQuest

La Société SineQua

L'Université Paris 7 – Denis Diderot

L'Institut National de Recherche en Informatique et en Automatique (INRIA)

AUTHORS - INDEX DES AUTEURS

Anne Abeillé 11
Miguel A. Alonso 19, 27, 67
Ciprian Bacalu 237
Srinivas Bangalore 33
Nicolas Barrier 41
Sébastien Barrier 41
Tilman Becker 47
Raffaella Bernardie 229
Tonia Bleam 199, 215, 241
Marie-Hélène Candito 11, 115
Vicente Carrillo 67
John Carroll 55
David Chiang 61
Luminita Chiran 237
Lionel Clément 233
Eric de la Clergerie 19, 27
Hoa Trang Dang 147
Rodolfo Delmonte 237
Christine Doran 73, 199
Víctor J. Díaz 67
Mark Dras 61, 241
Jason Eisner 79
Robert Frank 85
Chung-hye Han 93, 221
Karin Harbusch 101, 245
Beth Ann Hockey 185
Frankie James 185
Aravind K. Joshi 107
Sylvain Kahane 115, 123
Laura Kallmeyer 129
Martin Kappes 135
Gerard Kempen 101
Yannick de Kercadio 115
Anne Kilger 249
Nari Kim 221
Meesook Kim 221
Alexandra Kinyon 11, 41, 141

Karin Kipper 147
Patrice Lopez 47, 155, 253
Jens Michaelis 163
Uwe Mönnich 163
Frank Morawietz 163
Nicolas Nicolov 55
Adi Palm 171
Martha Palmer 147, 199, 265
Guy Perrier 177
Peter Poller 249
Owen Rambow 33, 93
Manny Rayner 185
David Roussel 253
Anoop Sarkar 193
Giorgio Satta 79
William Schuler 61, 147
Djamé Seddah 27
Olga Shaumyan 55
Martine Smets 55
Matthew Stone 199
Oliver Streiter 257
Manuel Vilares 19
Christian Wartena 207
Andy Way 261
David Weir 55
Jens Woch 245
Juntae Yoon 221
Fei Xia 215, 265

CONTENTS – SOMMAIRE

PRESENTATIONS – COMMUNICATIONS ORALES

Anne Abeillé, Marie-Hélène Candito, Alexandra Kinyon
The current status of FTAG 11

Miguel A. Alonso, Eric de la Clergerie, Manuel Vilares
A redifinition of Embedded Push-Down Automata 19

Miguel A. Alonso, Djamé Seddah, Eric de la Clergerie
Practical aspects in compiling tabular TAG parsers 27

Srinivas Bangalore, Owen Rambow
Using TAGs, a Tree Model, and a Language Model for Generation 33

Nicolas Barrier, Sébastien Barrier, Alexandra Kinyon
Lexik: A maintenance tool for FTAG 41

Tilman Becker, Patrice Lopez
Adapting HPSG-to-TAG compilation to wide-coverage grammars 47

John Carroll, Nicolas Nicolov, Olga Shaumyan, Martine Smets, David Weir
Engineering a Wide-Coverage Lexicalized Grammar 55

David Chiang, William Schuler, Mark Dras
Some Remarks on an Extension of Synchronous TAG 61

Víctor J. Díaz, Miguel A. Alonso, Vicente Carrillo
Bidirectional Parsing of TAG without heads 67

Christine Doran
Punctuation in a Lexicalized Grammar 73

Jason Eisner, Giorgio Satta
A Faster Parsing Algorithm for Lexicalized Tree-Adjoining Grammars 79

Robert Frank
Economy in TAG 85

Chung-hye Han, Owen Rambow
The Sino-Korean Light Verb Construction and Lexical Argument Structure 93

Karin Harbusch, Gerard Kempen
Complexity of Linear Order computation in Performance Grammar, TAG and HPSG 101

Aravind K. Joshi
Relationship between strong and weak generative power of formal systems 107

Sylvain Kahane, Marie-Hélène Candito, Yannick de Kercadio
An alternative description of extractions in TAG 115

Sylvain Kahane
How to solve some failures of LTAGs 123

Laura Kallmeyer
Scrambling in German and the non-locality of local TDGs 129

Martin Kappes
Contextual Tree Adjoining Grammars 135

Alexandra Kinyon
Even Better than Supertags: introducing Hypertags ! 141

Karin Kipper, Hoa Trang Dang, William Schuler, Martha Palmer
Building a class-based verb lexicon using TAGs 147

Patrice Lopez
LTAG workbench: A General Framework for LTAG 155

Jens Michaelis, Uwe Mönnich, Frank Morawietz
Derivational Minimalism in Two Regular and Logical Steps 163

Adi Palm
A Logical Approach of Structure Sharing in TAGs 171

Guy Perrier
From Intuitionistic Proof Nets to Interaction Grammars 177

Manny Rayner, Beth Ann Hockey, Frankie James
A comparison of the XTAG and CLE Grammars for English 185

Anoop Sarkar
Practical Experiments in Parsing using Tree Adjoining Grammars 193

Matthew Stone, Tonia Bleam, Christine Doran, Martha Palmer
Lexicalized Grammar and the Description of Motion Events 199

Christian Wartena
Extending Linear Indexed Grammars 207

Fei Xia, Tonia Bleam
A Corpus-based Evaluation of Syntactic Locality in TAGs 215

Juntae Yoon, Chung-hye Han, Nari Kim, Meesook Kim
Customizing the Xtag system for efficient grammar development for Korean 221

POSTERS - AFFICHES

Raffaella Bernardie
Deriving polarity effects 229

Lionel Clément
Un outil pour calculer des arbres de dépendance à partir d'arbres de dérivation 233

Rodolfo Delmonte, Luminita Chiran, Ciprian Bacalu
Elementary trees for syntactic and statistical disambiguation 237

Mark Dras, Tonia Bleam
How Problematic are Clitics for S-TAG Translations ? 241

Karin Harbusch, Jens Woch
Reuse of Plan-Based Knowledge Sources in a Uniform TAG-based Generation System 245

Anne Kilger, Peter Poller
CDL-TAGs : A grammar formalism for flexible and efficient syntactic generation 249

Patrice Lopez, David Roussel
Predicative LTAG grammars for Term Analysis 253

Oliver Streiter
Reliability in Example Based Parsing 257

Andy Way
LFG-DOT: A Probalistic, Constraint-Based Model for Machine Translation 261

Fei Xia, Martha Palmer
Comparing and Integrating Tree Adjoining Grammars 265

PRESENTATIONS – COMMUNICATIONS ORALES

The current status of FTAG

Annne Abeillé, IUF, TALaNa & LaTTICe, Paris 7
Marie-Hélène Candito, TALaNa & LaTTICe, Paris 7 and Lexiquest
Alexandra Kinyon, TALaNa & LaTTICe, Paris 7 and IRCS, UPenn
abeille, kinyon, candito@linguist.jussieu.fr

Introduction

We describe the current status and organization of a French Lexicalized Tree Adjoining Grammar (FTAG), developped over the last 10 years at TALaNa (Abeillé 91, Candito 99). The new version grammar is generated semi-automatically, independently of any corpus or application domain. It is intended to model speaker competence, and can be used both for parsing and generation. As far as parsing is concerned, we describe a general processing module which can rank the different parses produced based on linguistic information present in FTAG.

1. General linguistic choices

Most of our linguistic analyses follow those of Abeillé 91 (except that clitic arguments are substituted and not adjoined), complemented by Candito 99. We dispense with most empty categories, especially in the case of extraction.[1] Semantically void (or non autonomous) elements, such as complementizers, argument marking prepositions or idiom chunks, are coanchors in the elementary tree of their governing predicates.

1.1 A minimal tagset

We depart from traditional part of speech wherever the modern linguistic analyses have better to propose, especially in the generative tradition. We thus distinguish a special category for Clitics (weak pronouns) following Kayne 75, and for Complementizers. We collapse proper names, common nouns and pronouns into one category N, with features. We do not have a tag for subordinating conjunctions which are either Prepositions (followed by a complementizer: *pendant que* (during)) or (full) Complementizers (*si* (if), *comme* (as)...). Sentential structures are 'flat' (no internal VP). We thus have the following tagset.
Lexical categories: D (determiners), N (nouns, names, pronouns), V (verb), Cl (clitic pronoun), Prep (preposition), A (adjective), Adv (adverb), Conj (Coordinating conjunction), C (complementizer, subordinating conjunction),
Non lexical categories: SP (prepositional phrase), S (sentence). A and N are also used for nominal or adjectival phrases.

1.2 A rich set of grammatical functions

Tree sketches of the French TAG are compiled out of the French metagrammar (Candito 99), which expresses subcategorization in terms of grammatical functions. The functions used in the French MG for verbs are the following:
subject, object, dat-object, obl-object, gen-objet, locative, source-locative, manner, goal-infinitive, perception-infinitive, interrogative clause, "predicative complement"
All these functions can be both initial functions and final functions. An additional function "agt-object" is used as final function only, and is beared by a by-phrase in the case of passive.
We use several "complement" functions for complements of adjectives, prepositions, nouns, adverbs. And these categories may bear the function "modifier" with respect to the element they modify.

1.3. A parsimonious use of features

Most of the syntactic properties handled by feature structures in unification based linguistic theories (LFG or HPSG) are directly captured by the topology of the elementary trees in LTAG.

[1] We keep some empty categories for non realized arguments, such as PRO subjects (see Abeillé 91).

No use has to be made of valence or slash features to ensure subcategorization requirements or filler-gap relations. No feature passing principles, besides unification, are needed either.
We only rely on atomic valued features (which guarantees against any cyclic structure). We distinguish between:
- Morphological features, which are used in the morphological lexicon, in the syntactic lexicon when an argument is constrained for them (eg *trouver* has only indicative sentential complement) and for agreement in the elementary tree sketches,
- Syntactic features, used in the syntactic lexicon (for a verb to disallow passive for example) and in the tree sketches (to distinguish betwen trees in the same family or to further constrain tree combinations),
- Semantic features : these are gross classifications used for arguments (human, locative etc) which should be further refined.

We are currently using about 40 features as follows:
morphological features: <det>, <card>, <case>, <el>, <mode>, <num>, <ord>, <pers>, <P-num>, <P-pers>, <tense>.
syntactic features: <ant>, <ant-s>, <ant-v>, <aux>, <cq>, <det>, <extrap>, <gen>, <inv>, <modif>, <neg>, <nom>, <passive>, <part-num>, <part-gen>, <pred>, <princ>, <pro>, <quant>, <san1>, <san2>, <suj-gen>, <suj-pers>, <suj-num>, <sym>, <tense>, <wh>.
semantic features: <conc>, <degre>, <hum>, <loc>, <man>.

2. The Internal organization of FTAG

2.1. 3 sources of information for lexicalized elementary trees

Strict lexicalization at execution time does not prevent from representing the elementary trees in a less redundant way. Indeed it is required for any reasonably sized grammar, since for instance a verbal form may anchor dozens or hundreds of elementary trees. A first level of sharing between elementary trees was proposed within the XTAG system (XTAG group 1995) : elementary trees are compiled out of three sources of information:

- a set of tree sketches ("pre-lexicalized" structures, whose anchor is not instantiated)
- a syntactic lexicon, where each lexeme is associated with the relevant tree sketches
- a morphological lexicon, where inflected forms point to a lemma plus morphological features

Lexical selection of tree sketches is controlled by features from the syntactic and morphological lexicons, and uses the notion of tree families : sets of tree sketches that share the same initial argumental structure. The tree sketches of a family show all the possible surface realization of the arguments (pronominal clitic realization, extraction, inversion...) and all the possible transitivity alternations (impersonal, passive, middle..).
A lexeme selects one or several families (corresponding to one or several initial subcat frames) and with the help of features selects exactly the relevant tree sketches : The features may rule out some tree sketches of the selected family, either because of morphological clash (eg. the passive trees are only selected by past participles) or because of "idiosyncrasies" (eg. the French transitive verb *peser* -to weigh- disallows passive).
Figure 1 shows an elementary tree anchored by *parlait* (talked) and the corresponding tree sketch.

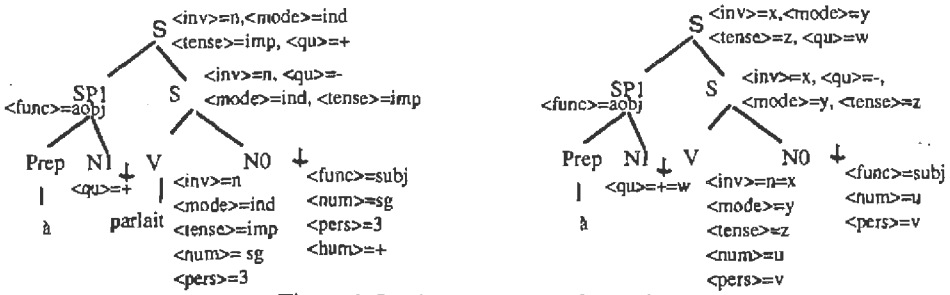


Figure 1. Lexicalized tree and tree sketch

The lexicalized tree is compiled out of the tree sketch and the following lexical entries with feature unifcation, (cf Barrier et al 00):
Morphological database:
parlait, V: [parler], {V.b:<mode>=ind, <tense>=imp, <num>=sg, <pers>=3}.
Syntactic database:
[parler], V: n0Van1 {N0:<hum>=+}.
The inflected form *parlait* points to the lemma PARLER, and the lexeme /PARLER/, that comprise the single lemma PARLER, selects in turn the n0Van1 family, where the preposition appears as a co-anchor (except in the case the argument 1 is cliticised).

2.2 The lexicons

Contrary to the English LTAG which reuses existing dictionaries (Collins 1979 for the morphological database, Oxford English Dictionary and COMLEX for the syntactic database), our French lexicons had to be done by us. They currently comprise the following items:
Morphological lexicons: over 50 000 (inflected) forms: 45800 for verbs, 3500 for nouns and pronouns, 950 for adjectives and 50 for determiners.
Syntactic lexicons : over 6000 (disambiguated) entries: 3700 for verbs, 500 for prepositions and adverbs, 800 for adjectives, 80 for determiners, 2000 for nouns, 350 for idioms
The lexical items chosen have been extracted as the most frequent ones from the frequency lists of Julliand 1970 and Catach 1984, except for idioms where one had to rely on personal intuitions. They have been disambiguated (and separated into different syntactic entries) with standard dictionaries as well as LADL lexicon-grammar tables (Gross 1975). The morphological lexicons have been automatically generated, using PC-Kimmo adapted to French. Both lexicons are organised in lexical databases, and the features normalized with templates.[2]
The morphological lexicon has nothing specific and associates lemmas, inflected forms and relevant morphological features. The syntactic lexicon associates lemmas with constructions (elementary trees or tree families with features) and performs some meaning disambiguation (based on different syntactic constructions, for example for the French verb *abattre* - knock down, shoot down) :

INDEX:	abattre/1	(physical meaning)
ENTRY:	abattre	
POS:	V	
FAM:	n0Vn1	
FS:		

INDEX:	abattre/2 (psychological meaning, possible sentential subject)
ENTRY:	abattre
POS:	V
FAM:	s0Vn1
FS:	#N1_HUM+, #N0_HUM-

Future developments include integrating a more complete full form lexicon (over 400 000 forms independently developed for our tagger; cf. Abeillé et al 1998) into the morphological database, and developing the syntactic lexicon (with shallow parsed corpora and reuse of LADL valence tables for French verbs, cf. Namer and Hathout 1998).

2.3. The metagrammar

We use an additional layer of linguistic description, called the metagrammar (MG) (Candito 1996, 99) which imposes a general organization and formalizes the well-formedness conditions for elementary tree sketches. It provides a general overview of the grammar and makes it possible for a tool to automatically generate the desired tree sketches from the combination of smaller descriptions.
MG thus represents a TAG as a multiple inheritance network, whose classes specify syntactic structures as partial descriptions of trees (Vijay-Shanker & Schabes 92, Rogers & Vijay-

[2] For unknown words, a default tree assignment is used.

Shanker, 94). Partial descriptions of trees are sets of constraints that may leave underspecified the relation existing between two nodes. The relation between two nodes may be further specified by adding constraints in sub-classes of the inheritance network. Inheritance of partial descriptions is monotonic.[3]
In order to build pre-lexicalized structures respecting the Predicate Argument Coocurrence Principle, and to group together structures belonging to the same tree family, MG makes use of syntactic functions to express either monolingual or cross-linguistic generalizations (as in LFG or Relational Grammar). Subcategorization of predicates is expressed as a list of syntactic functions, and their possible categories. The initial subcategorization is that of the unmarked case, and is modifiable by redistribution (or transitivity alternations).
Structures sharing the same initial subcategorization are grouped in a tree family. For verbal predicates, an elementary tree is partly represented with an ordered list of successive subcategorizations, from the initial one to the final one. Elementary trees sharing a final subcategorization, may differ in the surface realizations of the functions. MG represents this repartition of information by imposing a three-dimension inheritance network :

Dimension 1: initial subcategorization
Dimension 2: redistributions of functions
Dimension 3: surface realizations of syntactic functions.

Dimension 1 describes a possible initial subcategorization (and possibly frozen elements). Dimension 2 describes a list of ordered redistributions (including the case of no-redistribution) which may impose a verbal morphology (eg. the auxiliary for passive). Dimension 3 represents the surface realization of a function (independently of the initial function).
The 3 dimension hierarchy is handwritten, the elementary trees are automatically generated with a two-step process. First the compiler automatically creates additional classes of the inheritance network : the "crossing classes". Then each crossing class is translated into one or several tree sketches (the minimal structures satisfying all inherited constraints). During the first step, crossing classes are automatically built as follows (with unification):

- a crossing class inherits one terminal class of dimension 1
- then, the crossing class inherits one terminal class of dimension 2
- then, the crossing class inherits classes of dimension 3, representing the realizations of every function of the final subcategorization.

The tree sketch of figure 1, for example, has been compiled, out of an initial subcategorization with nominal subject and dative object (dimension 1), an active canonical redistribution (dimension 2), a nominal inverted realization for the subject, and a fronted interrogative realization for the dative object (dimension 3).

3. Elementary trees in FTAG

3.1 Linguistic principles for elementary trees

Within FTAG, elementary trees respect the following linguistic well-formedness principles: (Kroch & Joshi 85, Abeillé 91, Franck 92, Candito 99, Candito & Kahane 98)

- Strict Lexicalization : all elementary trees are anchored by at least one lexical element, the empty string cannot anchor a tree by itself,
- Semantic Consistency : no elementary tree is semantically void (this ensures the compositionality of the syntactic analysis),
- Semantic Minimality : no elementary tree correspond to more than one semantic unit (modulo lexicalism : lexical anchors are not broken down into morphemes).
- Predicate Argument Cooccurrence Principle (PACP): an elementary tree is the minimal syntactic structure that includes a leaf node for each realized semantic argument of the anchor(s).

Initial trees are used for arguments, verbs with non sentential arguments, auxiliary trees are used for modifiers, determiners, modals, auxiliaries and verbs with sentential complement,

Some examples of elementary trees are the following:

---

3. In MG, nodes of partial descriptions are augmented with specific feature structures, called meta-features, constraining for instance, the possible parts of speech of a node or the index in the case of argumental nodes.

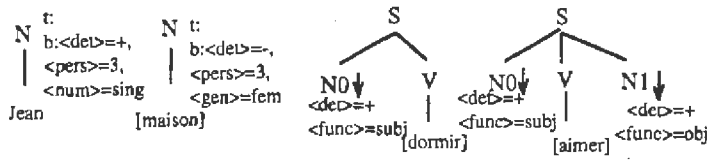


Figure 2. Initial elementary trees

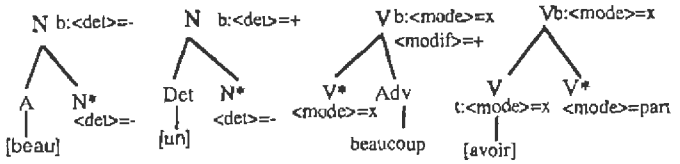


Figure 3. Auxiliary elementary trees

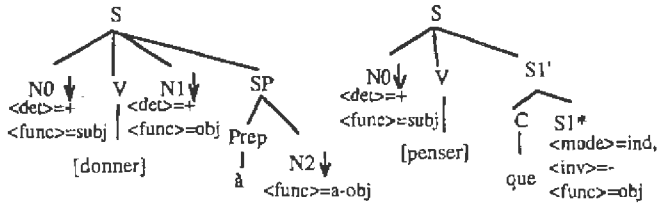


Figure 4. Elementary trees with functional co-anchors

In Figure 3, the relevant syntactic and semantic units are donner-à (give to) or penser-que (think that).

3.2. The metagrammar for FTAG

The set of tree sketches in FTAG is comprises over 5000 elementary tree sketches (not counting trees for causative constructions). Currently, all but 40 of them are compiled from the French metagrammar. The 40 remaining tree sketches are trees for determiners (plain and complex), nouns used as arguments, coordination conjunctions, clitics and "special" trees for deficient verbs such as raising verbs and auxiliaries.

The French MG comprises the description for the tree sketches anchored by full verbs, prepositions, adverbs, adjectives, and nouns (when used as modifiers).

Within dimension 1, it comprises 54 initial subcategorization frames for verbs (which means there are 54 tree families for verbs in FTAG), 4 initial subcategorizations for adjectives, 12 for adverbs, prepositions, subordinating conjunctions and nominal modifiers.

In dimension 2, primarily relevant for verbs, we have defined as redistributions the following phenomena:

- Passive (with or without agent) : additional V-headed elementary trees with auxiliary *être* susbstituted
- Causative constructions
- Reflexive
- Impersonal constructions (active and passive)
- Middle *se* (*ces robes se lavent facilement*)

In dimension 3 , we define as realizations the following phenomena:

- Infinitival and sentential arguments (treated as S-complements)
- Relatives (*qui, que, dont,* Prep *qui*, Prep *lequel*), indicative, subjunctive
- Interrogatives (direct, indirect, *est-ce-que*)
- Cleft sentences (*c'est que, c'est qui, c'est* Prep *N que*)
- Clitic pronouns
- Subject inversion (nominal or subject clitic)
- Unbounded dependencies (with island constraints)

- Participials (past or present part, NP modifiers), Past participle agreement.
- Null realization (empty subjects for infintiives, participials or imperatives)
- Factorization (for subjects in coordinated phrases)
- Word order variation (among complements)

Work remain to be done of the syntax of quantifiers (often discontinuous in French), on negation (including negative concord), coordination and comparison (including superatives).

4. Evaluating FTAG

Evaluating a wide coverage grammar is a difficult task, especially in the absence of reference tree banks for French. We performed a quantitative evaluation using the French test suite developed in the TSNLP project (Estival & Lehmann 96). Further evaluation will be done on newspaper corpora.

4.1. Evaluation using TSNLP

We have performed an external evaluation using the TSNLP multilingual data base, which aims at covering the major syntactic phenomena for each language, using a minimal vocabulary (a few hundred words). We have extracted all the French items of the TSNLP data base, classified by grammatical status (we only took 0 and 1), by length and by phenomenon (according to TSNLP original classification). For all grammatical items, the results with the 1998 version of our grammar are as follows:[4]

- over 80 % of the grammatical parsed, with an averrage of 2.9 parses per sentences
- over 82% of the agrammatical sentences have been correctly rejected.

There were no unknown words. The main failure cases are the following:

- missing lexical coding (transitive verb without object, transitive use of intransitives),
- missing elementary tree (causative trees, postverbal clitics with imperatives),
- feature unification clash (agreement with politeness forms: *vous êtes belle*, or with coordination : *deux bandes bleue et jaune*),
- missing phenomenon (tough construction, gapping...).

Cases of overanalysis either come from a disputable TSNLP coding (for example for sequence of times), or from the incompleteness of our representation (for example for coordination or negation, we overgenerate).

4.2. Comparison with other syntactic ressources

The lexicon-grammar developed at LADL for more than 20 years is an unrivaled source of knowledge reusable in the sens that it is not designed for any program and not even depedent on any special formalism. However, it cannot be directly used to analyse (or generate) a text since it only lists some basic constructions (with their lexical head). It does not code the crossing of constructions nor the productive phenomena which are not clearly lexically sensitive (such as causative, quantifier floating or argument extraction for simple verbs). Thus, even though it is crucial to know that transitive *voler* (to steal) must be distinguished from intransitive *voler* (to fly), more general grammatical rules are needed to know that it is the transitive *voler* which is instanciated in examples (1)-(2) without a postverbal NP object, or that it is the intransitive *voler* which is instanciated in examples (3)-(4) (even though there is a postverbal NP):

(1) Ils veulent tout voler
(2) les bijoux qu'ils ont finalement avoué avoir volé...
(3) Lufthansa fait voler ses avions 5 jours sur 7
(4) A une altitude à laquelle ne vole normalement aucun avion ...

M. Salkoff (1973, 79) string grammar has listed numerous grammatical strings representative of French syntax but has never been associated with a sizable lexicon and cannot be reused independently of the parsing scheme it was made for. The HPSG like grammar developed for French by Namer and Schmidt 93 suffers from the same problems and is totally dependent upon the ALEP developement platform.

---

4. A previous evaluation done in 1996 (Abeillé et al. 96), with a smaller coverage grammar (comprising about 830 elementary tree sketches), using the same lexicon, had the following results : 65% of the grammatical sentences (excluding coordination) parsed, with an ambiguity rate of 1,5 parses per parsed sentence.

The GB grammar developed for French at LATL (Wehrli 97), is more modular and associated with a sizable dictionnary. But it is not clearly separated from the program that uses it (extraction or passive phenomena are not handled as grammatical data but as types of action - attachment, trace creation... - that the program does at a certain stage in the parsing scheme) and thus cannot be reused as such for other applications.

5. Ranking parses

To be usable in pratice, our grammar must associate one best analysis per grammatical sentence. The output of a TAG parser can be viewed as a derived tree (encoding phrase structure) or as a derivation tree (encoding dependencies). Since it is both more compact and more informative, we choose the derivation tree for parse ranking (contrary to Srinivas & al. 95).

5.1. General Disambiguation principles

Our parse ranker is based on empirical (i.e. corpus-based) and psycholinguistic-based preferences (Kinyon 99). It only uses lexical and syntactic sources of information (whereas a true disambiguator should also use semantic and discourse information). Since we work on the derivation tree which exhibits the lexicalized trees used for parsing, it is easy to mix lexical and syntactic preferences. Our parse ranker thus uses 3 types of preferences : lexical preferences (such as valence preference for verbs), grammatical preferences (construction types) and general principles which are structure-based, domain, language and application independant.
The lexical preferences code either a category preference or a valence principle. They have to be computed for each word, but we rely on the general tendency in French to favor grammatical categories over lexical categories for ambiguous forms (for example weak pronouns (clitics) to strong pronouns, or auxiliaries over full valence verbs).
The grammatical preferences code a construction preference, for example active over passive or personal over impersonal. In *Il est venu une nuit*, the personal interpretation (with *il* as personal subject and *une nuit* as adjunct) is to be favored over the impersonal one (with *une nuit* as deep subject).
The general principles assume the existence of a universal preference for economy (e.g. adjunction is more costly than substitution) and therefore favor analysis that needs to perform the fewer operations. Formulating structural preference principles in terms of derivation tree allows to capture widely accepted preferences, which turn out to be difficult to formalize in terms of constituent trees : idioms are prefered over literal interpretations, arguments are prefered over modifiers.
These general principles are the following :

1- Prefer the derivation tree with the fewer number of elementary trees (=fewer nodes)
2- Prefer to attach initial trees low
3- Prefer the derivation tree with the fewer number of auxiliary trees

Principle 1 favors the idiomatic interpretation of a sentence over its literal interpretation (a), since the diferent idiom chunks belong to the same elementary tree. It also favors the attachment prepositional phrases as arguments rather than modifiers (b). Principle 2 favors the low attachment of arguments, when several alternative attachments are possible : in (c) the PP *de la manifestation* is an argument of the N *organisateur* rather than of the V *soupçonne*. In (d), the PP *à Jean* is an argument of *dit* rather than of *parle*. Principle 3 favors the derivation tree involving the fewer number of adjunctions (i.e. modifiers) : in (e) *le matin* could be a modifier, but the attachment as an argument is prefered.

(a) Jean brise la glace (Jean cuts the mustard > Jean breaks the ice)
(b) Jean pense à la reunion (Jean thinks of the reunion > Jean thinks at the reunion)
(c) Jean remercie l'organisateur de la manifestation (J. suspects the organizer of the demonstration > for the demonstration)
(d) C'est à Jean que Marie dit que Paul parle (It's to Jean that Marie says that John thinks > It's of Jean that ....)
(e) Jean attend le matin (Jean awaits the morning > J. waits in the morning)

In case of conflict, the priority is for lexical preferences, then grammatical preferences, then general principles.

5.2. Application to TSNLP

The parsed item from TSNLP had an average of 2.9 parses per item. No categorial ambiguity remained. Most feature ambiguities are handled via underspecification (eg "les enfants" feminine ou masculine). The remaining (structural) ambiguities are the following (not all of these are spurious) :

- modifier adjoined to S ou V after an intransitive verb (L'ingénieur viendra volontiers),
- prepositional phrase analysed as complement or modifier (L'ingénieur préfère le vin à l'eau; Il passe pour un spécialiste),
- passive with or without agent (the par-PP can be analysed as an agent phrase or as a modifier)
- several adjunction sites in case of multiple modifiers.

After applying the general preference principles, we are left with only 2.17 derivations / sentence (i.e. -24 %), while the number of sentences for which a "correct" parse is present only marginally decreased. After applying the language specific preferences, we are left with 1,5 derivation / sentence (i.e. - 47 % in total). It turns out that one of the main sources of spurious ambiguities lies in adverbial attachment. We are exploring how to add lexical preferences to deal with this case.

References

A. Abeillé, 1991. *Une grammaire lexicalisée d'arbres adjoints pour le français.* PhD Thesis, University Paris 7. (to appear Editions du CNRS)

A. Abeillé, B. Daille, B. Robichaud, 1996, FTAG : un analyseur de phrases françaises, Proceedings ILN, Nantes.

N. Barrier, S. Barrier, A. Kinyon, 2000, Lexik: a tool for FTAG, Proceedings TAG+5, Paris. this volume

M-H. Candito, 1996. A principle-based hierarchical representation of LTAG, *Proceedings 15th COLING*, Kopenhagen.

M-H. Candito, 1999. Représentation modulaire et paramétrable de grammaires électroniques lexicalisées. Application au français et à l'italien. PhD dissertation. University Paris 7.

M-H. Candito, S. Kahane, 1998. Can the TAG derivation tree represent a semantic graph ? an answer in the light of MTT, *Proceedings TAG+4 Workshop*, Philadelphia.

N. Catach, 1984. *Les listes orthographiques de base du français*, Paris, Nathan.

L. Danlos, 1998. GTAG : un formalisme lexicalisé pour la génération inspiré de TAG, *TAL* 39:2.

C. Doran, D. Egedi, Hockey, B., Srinivas, B., Zaidel M. 1994. The XTAG system : a wide coverage grammar for English, *Proceedings 15th COLING*, Kyoto, pp. 922-928.

C. Doran et al 2000, Evolution of the XTAG system, in Abeillé, Rambow (eds) *Tree Adjoining grammars*, CSLI.

D. Estival, S. Lehmann, 1997. TSNLP: des jeux de phrases test pour le TALN, *TAL*, 38:1, pp. 155-172.

R. Frank 1992. Syntactic locality and Tree Adjoining grammar: grammatical, acquisition and processing perspectives, PhD Dissertation, Univ. Pennsylvania, Philadelphia.

M. Gross, 1975. *Méthodes en syntaxe*, Hermann, Paris.

A. Julliand, D. Brodin, C. Davidovitch, 1970. *Frequency dictionnary of French words: 5000 words*, Mouton, La Haye.

R. Kayne, 1975. *French syntax : the transformational cycle*, Cambridge, MIT Press.

A. Kinyon 1999. Hiérarchisation d'analyses basée sur des informations dependancielles pour les LTAGs. Proceedings TALN'99. Cargèse.

A. Kroch, A. Joshi, 1985. The linguistic relevance of Tree Adjoining grammars, Technical Report, University of Pennsylvania, Philadelphia.

F. Namer, P. Schmidt, 1993. Une grammaire du français dans un formalisme à structures de traits typés, *Proceedings ILN*, Nantes.

F. Namer, N. Hathout 1998. Automatic construction and validation of French large lexical resources: reuse of verb theoretical descriptions, *Proceedings 1st LREC Conference*, Granada.

J. Rogers, K. Vijay-Shanker, 1994. Obtaining trees from their descriptions, an application to tree adjoining grammar, *Computational Intelligence*, 10:4.

M. Salkoff, 1973. *Une grammaire en chaîne du français*, analyse distributionnelle, Paris: Dunod.

M. Salkoff, 1979. *Analyse syntaxique du français, grammaire en chaîne* J. Benjamin, Amsterdam.

M. Silberztein, 1993. *Dictionnaires électroniques pour l'analyse lexicale : le système INTEX*, Masson, Paris.

B. Srinivas, Doran C., Kulick S. 1995 : Heuristics and Parse Ranking. 4th workshop on Parsing Technologies. Prag. Czech Republic.

B. Srinivas, 1997. Complexity of lexical descriptions and its relevance for partial parsing, PhD thesis, University of Pennsylvania, Philadelphia.

K. Vijay-Shanker, Y. Schabes, 1992. Structure sharing in LTAG, Proceedings 14th COLING, Nantes.

E. Wehrli, 1997. *L'analyse syntaxique des langues naturelles : techniques et méthodes*, Paris: Masson.

Workshop TAG+5, Paris, 25-27 May 2000

A redefinition of Embedded Push-Down Automata*

Miguel A. Alonso†, Eric de la Clergerie‡ and Manuel Vilares†

†Departamento de Computación, Universidad de La Coruña
Campus de Elviña s/n, 15071 La Coruña (Spain)
{alonso,vilares}@dc.fi.udc.es
‡INRIA, Domaine de Voluceau
Rocquecourt, B.P. 105, 78153 Le Chesnay (France)
Eric.De_La_Clergerie@inria.fr

Abstract

A new definition of Embedded Push-Down Automata is provided. We prove this new definition preserves the equivalence with tree adjoining languages and we provide a tabulation framework to execute any automaton in polynomial time with respect to the length of the input string.

1. Introduction

Embedded Push-Down Automata (EPDA) were defined in (Vijay-Shanker, 1988) as an extension of Push-Down Automata that accept exactly the class of Tree Adjoining Languages. They can also be seen as a level-2 automata in a progression of linear iterated pushdowns involving nested stacks (Weir, 1994).

An EPDA consists of a finite state control, an input tape and a stack made up of non-empty stacks containing stack symbols. A transition can consult the state, the input string and the top element of the top stack and then change the state, read a character of the input string and replace the top element by a finite sequence of stack elements to give a new top stack, and new stacks can be placed above and below the top stack.

EPDA can describe parsing strategies for tree adjoining grammars in which adjunctions are recognized top-down. The same kind of strategies can be described in strongly-driven 2-stack automata (de la Clergerie & Alonso Pardo, 1998) and linear indexed automata (Nederhof, 1999), which has associated tabulation frameworks allowing those automata to be executed in polynomial time with respect to the size of the input string. In this paper we propose a redefinition of EPDA in order to provide a tabulation framework for this class of automata.

2. EPDA without states

Finite-state control is not a fundamental component of push-down automata, as the current state in a configuration can be stored in the top element of the stack of the automaton (Lang, 1991). Finite-state control can also be eliminated from EPDA, obtaining a new definition that considers a EPDA as a tuple $(V_T, V_S, \Theta, \$_0, \$_f)$ where V_T is a finite set of terminal symbols, V_S is a finite set of stack symbols, $\$_0 \in V_S$ is the initial stack symbol, $\$_f \in V_S$ is the final stack symbol and Θ is a finite set of six types of transition:

SWAP: Transitions of the form $C \overset{a}{\longmapsto} F$ that replace the top element of the top stack while scanning a. The application of such a transition on a stack $\Upsilon[\alpha B$ returns the stack $\Upsilon[\alpha C$.

* This research was partially supported by the FEDER of EU (Grant 1FD97-0047-C04-02) and Xunta de Galicia (Grant PGIDT99XI10502B).

PUSH: Transitions of the form $C \xmapsto{a} C\ F$ that push F onto C. The application of such a transition on a stack $\Upsilon[\alpha C$ returns the stack $\Upsilon[\alpha CF$.

POP: Transitions of the form $C\ F \xmapsto{a} G$ that replace C and F by G. The application of such a transition on $\Upsilon[\alpha CF$ returns the stack $\Upsilon[\alpha G$.

WRAP-A: Transitions *wrap-above* of the form $C \xmapsto{a} C, [F$ that push a new stack $[F$ on the top of the automaton stack. The application of such a transition on a stack $\Upsilon[\alpha C$ returns the stack $\Upsilon[\alpha C[F$.

WRAP-B: Transitions *wrap-below* of the form $C \xmapsto{a} [C, F$ that store a new stack $[C$ just below the top stack, and change from C to F the top element of the top stack. The application of such a transition on a stack $\Upsilon[\alpha C$ returns the stack $\Upsilon[C[\alpha F$.

UNWRAP: Transitions of the form $C, [F \xmapsto{a} G$ that delete the top stack $[F$ and replace the new top element by G. The application of such a transition on a stack $\Upsilon[\alpha C[F$ returns the stack $\Upsilon[\alpha G$.

where $C, F, G \in V_S$, $\Upsilon \in ([V_S^*)^*$, $\alpha \in V_S^*$, $a \in V_T \cup \{\epsilon\}$ and $[\notin V_S$ is a new symbol used as stack separator. It can be proved that transitions of a EPDA with states can be emulated by transitions in Θ and vice versa.

An *instantaneous configuration* is a pair (Υ, w), where Υ represents the contents of the automaton stack and w is the part of the input string that is yet to be read. A configuration (Υ, aw) derives a configuration (Υ', w), denoted $(\Upsilon, aw) \vdash (\Upsilon', w)$, if and only if there exists a transition that applied to Υ gives Υ' and scans a from the input string. We use $\vdash^*$ to denote the reflexive and transitive closure of $\vdash$. An input string is accepted by an EPDA if $([\$_0, w) \vdash^* ([\$_f, \epsilon)$. The language accepted by an EPDA is the set of $w \in V_T^*$ such that $([\$_0, w) \vdash^* ([\$_f, \epsilon)$.

3. Compiling TAG into EPDA

We consider each elementary tree γ of a TAG as formed by a set of context-free productions $\mathcal{P}(\gamma)$: a node N^γ and its g children $N_1^\gamma \ldots N_g^\gamma$ are represented by a production $N^\gamma \rightarrow N_1^\gamma \ldots N_g^\gamma$. The elements of the productions are the nodes of the tree, except for the case of elements belonging to $V_T \cup \{\varepsilon\}$ in the right-hand side of production. Those elements may have no children and can not be adjoined, so we identify such nodes labeled by a terminal with that terminal. We use $\beta \in \text{adj}(N^\gamma)$ to denote that a tree β may be adjoined at node N^γ. If adjunction is not mandatory at N^γ, then $\mathbf{nil} \in \text{adj}(N^\gamma)$. We consider the additional productions $\top^\alpha \rightarrow \mathbf{R}^\alpha$, $\top^\beta \rightarrow \mathbf{R}^\beta$ and $\mathbf{F}^\beta \rightarrow \bot$ for each initial tree $\alpha \in \boldsymbol{I}$ and each auxiliary tree $\beta \in \boldsymbol{A}$, where $\mathbf{R}^\alpha$ is the root node of α and $\mathbf{R}^\beta$ and $\mathbf{F}^\beta$ are the root node and foot node of β, respectively. After disabling $\top^\gamma$ and $\bot$ as adjunction nodes the generative capability of the grammar remains intact.

Figure 1 shows the generic compilation schema from TAG to EPDA, where symbols $\nabla_{r,s}^\gamma$ have been introduced to denote dotted productions. The meaning of each compilation rule is graphically shown in figure 2. This schema is parameterized by $\overrightarrow{N^\gamma}$, the information propagated top-down w.r.t. the node N^γ, and by $\overleftarrow{N^\gamma}$, the information propagated bottom-up. When the schema is used to implement a top-down strategy $\overrightarrow{N^\gamma} = N^\gamma$ and $\overleftarrow{N^\gamma} = \square$, where $\square$ is a fresh stack symbol. A bottom-up strategy requires $\overrightarrow{N^\gamma} = \square$ and $\overleftarrow{N^\gamma} = N^\gamma$. For a Earley-like parsing strategy, $\overrightarrow{N^\gamma} = \overline{N^\gamma}$ and $\overleftarrow{N^\gamma} = \overline{\overline{N^\gamma}}$, where $\overline{N^\gamma}$ and $\overline{\overline{N^\gamma}}$ are used to distinguish the top-down prediction from the bottom-up propagation of a node.

We can observe in figure 1 that each stack stores pending adjunctions with respect to the node placed on the top of the stack in a top-down treatment of adjunctions: when an adjunction node

[INIT]	$\$_0 \longmapsto \$_0 \left[\nabla^{\alpha}_{0,0}$	$\alpha \in \boldsymbol{I}$
[CALL]	$\nabla^{\gamma}_{r,s} \longmapsto \nabla^{\gamma}_{r,s} \left[\overrightarrow{N^{\gamma}_{r,s+1}}$	$N^{\gamma}_{r,s+1} \notin \text{spine}(\gamma)$, $\mathbf{nil} \in \text{adj}(N^{\gamma}_{r,s+1})$
[SCALL]	$\nabla^{\beta}_{r,s} \longmapsto \left[\nabla^{\beta}_{r,s}, \ \overrightarrow{N^{\beta}_{r,s+1}}$	$N^{\beta}_{r,s+1} \in \text{spine}(\beta)$, $\mathbf{nil} \in \text{adj}(N^{\beta}_{r,s+1})$
[SEL]	$\overrightarrow{N^{\gamma}_{r,0}} \longmapsto \nabla^{\gamma}_{r,0}$	
[TAB]	$\nabla^{\gamma}_{r,n_r} \longmapsto \overleftarrow{N^{\gamma}_{r,0}}$	
[RET]	$\nabla^{\gamma}_{r,s}, \left[\overleftarrow{N^{\gamma}_{r,s+1}} \longmapsto \nabla^{\gamma}_{r,s+1}$	$N^{\gamma}_{r,s+1} \notin \text{spine}(\gamma)$, $\mathbf{nil} \in \text{adj}(N^{\gamma}_{r,s+1})$
[SRET]	$\nabla^{\beta}_{r,s}, \left[\overleftarrow{N^{\beta}_{r,s+1}} \longmapsto \nabla^{\beta}_{r,s+1}$	$N^{\gamma}_{r,s+1} \in \text{spine}(\beta)$, $\mathbf{nil} \in \text{adj}(N^{\beta}_{r,s+1})$
[SCAN]	$\overrightarrow{N^{\gamma}_{r,0}} \overset{a}{\longmapsto} \overleftarrow{N^{\gamma}_{r,0}}$	$N^{\gamma}_{r,0} \to a$
[ACALL-a]	$\nabla^{\gamma}_{r,s} \longmapsto \left[\nabla^{\gamma}_{r,s}, \ \Delta^{\gamma}_{r,s}$	$\text{adj}(N^{\gamma}_{r,s+1}) \neq \{\mathbf{nil}\}$
[ACALL-b]	$\Delta^{\gamma}_{r,s} \longmapsto \Delta^{\gamma}_{r,s} \ \overrightarrow{\top^{\beta}}$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[ARET]	$\nabla^{\gamma}_{r,s}, \left[\overleftarrow{\top^{\beta}} \longmapsto \nabla^{\gamma}_{r,s+1}$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FCALL-a]	$\nabla^{\beta}_{f,0} \longmapsto \left[\nabla^{\beta}_{f,0}, \ \bot$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta}$
[FCALL-b]	$\Delta^{\gamma}_{r,s} \ \bot \longmapsto \overrightarrow{N^{\gamma}_{r,s+1}}$	
[FRET]	$\nabla^{\beta}_{f,0}, \left[\overleftarrow{N^{\gamma}_{r,s+1}} \longmapsto \nabla^{\beta}_{f,1}$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta}$, $\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FINAL]	$\$_0 \left[\nabla^{\alpha}_{0,1} \longmapsto \left[\$_f$	$\alpha \in \boldsymbol{I}$

Figure 1: Generic compilation schema from TAG to EPDA

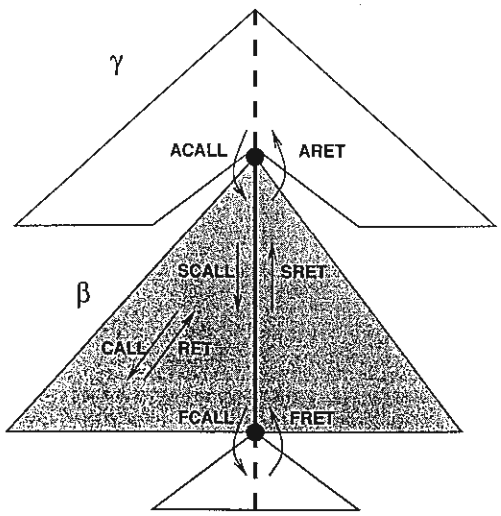


Figure 2: Meaning of compilation rules

Transition	EPDA	L-LIA
SWAP	$C \overset{a}{\longmapsto} F$	$C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ]$
PUSH	$C \overset{a}{\longmapsto} CF$	$C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ C]$
POP	$CF \overset{a}{\longmapsto} G$	$F[\circ\circ C] \overset{a}{\longmapsto} G[\circ\circ]$
WRAP-A	$C \overset{a}{\longmapsto} C, [F$	$C[\circ\circ] \overset{a}{\longmapsto} C[\circ\circ]\ F[\,]$
WRAP-B	$C \overset{a}{\longmapsto} [C, F$	$C[\circ\circ] \overset{a}{\longmapsto} C[\,]\ F[\circ\circ]$
UNWRAP	$C, [F \overset{a}{\longmapsto} G$	$C[\circ\circ]\ F[\,] \overset{a}{\longmapsto} G[\circ\circ]$
WRAP-B+PUSH	$C \overset{a}{\longmapsto} [C, X\ F$	$C[\circ\circ] \overset{a}{\longmapsto} C[\,]\ F[\circ\circ X]$
WRAP-B+POP	$X\ C \overset{a}{\longmapsto} [C, F$	$C[\circ\circ X] \overset{a}{\longmapsto} C[\,]\ F[\circ\circ]$

Figure 3: Equivalence between EPDA and L-LIA

is reached, the adjunction node is stored on the top of the stack ([**ACALL-a**]) and the traversal of the auxiliary tree is started ([**ACALL-b**]); the adjunction stack is propagated through the spine ([**SCALL**]) down to the foot node, where the traversal of the auxiliary tree is suspended to resume the traversal of the subtree rooted by the adjunction node ([**FCALL-a**]), which is eliminated of the stack ([**FCALL-b**]). To avoid confusion, we store $\Delta^{\gamma}_{r,s}$ instead of $\nabla^{\gamma}_{r,s}$ to indicate that an adjunction was started at node $N^{\gamma}_{r,s+1}$. A symbol Δ can be seen as a symbol ∇ waiting an adjunction to be completed.

4. EPDA and Left-oriented Linear Indexed Automata

Left-oriented Linear Indexed Automata (L-LIA) is a class of automata defined by Nederhof (1999) that can be used to implement parsing strategies for TAG in which adjunctions are recognized in a top-down way. Given a EPDA, the equivalent L-LIA is obtained by means of a simple change in the notations: if we consider the top element of a stack as a stack symbol, and the rest of the stack as the indices list associated to them, we obtain the correspondence shown in figure 3.

This change in notation is also useful to show that EPDA accept exactly the class of tree adjoining languages. That tree adjoining languages are accepted by EPDA is shown by the compilation schema defined previously. To prove that the languages accepted by EPDA are tree adjoined languages, we exhibit a procedure that, given an EPDA $\mathcal{A} = (V_T, V_S, \Theta, \$_0, \$_f)$, builds a linear indexed grammar (Gazdar, 1987) $\mathcal{G} = (V_T, V_N, V_I, S, P)$ that recognizes the language accepted by $\mathcal{A}$. Non-terminals in V_N are pairs $\langle A, B\rangle$, where $A, B \in V_S$, and $V_I = V_S$. Productions in P are obtained from transitions in Θ as follows:

- For each transition $C \overset{a}{\longmapsto} F$ and for each $E \in V_S$, a production $\langle C, E\rangle[\circ\circ] \rightarrow a\ \langle F, E\rangle[\circ\circ]$ is created.
- For each transition $C \overset{a}{\longmapsto} CF$ and for each $E \in V_S$, a production $\langle C, E\rangle[\circ\circ] \rightarrow a\ \langle F, E\rangle[\circ\circ C]$ is created.
- For each transition $C\ F \overset{a}{\longmapsto} G$ and for each $E \in V_S$, a production $\langle F, E\rangle[\circ\circ C] \rightarrow a\ \langle G, E\rangle[\circ\circ]$ is created.

- For each pair of transitions $C \overset{b}{\longmapsto} C, [F'$ and $C, [F \overset{a}{\longmapsto} G$, and for each $E \in V_S$, a production $\langle C, E\rangle[\circ\circ] \rightarrow b\ \langle F', F\rangle[\,]\ a\ \langle G, E\rangle[\circ\circ]$ is created.

- For each pair of transitions $C \overset{b}{\longmapsto} [C, F'$ and $C, [F \overset{a}{\longmapsto} G$, and for each $E \in V_S$, a production $\langle C, E\rangle[\circ\circ] \rightarrow b\ \langle F', F\rangle[\circ\circ]\ a\ \langle G, E\rangle[\,]$ is created.

- For each $E \in V_S$, a production $\langle E, E\rangle[\,] \rightarrow \epsilon$ is created.

The axiom of the grammar is $S = \langle \$_0, \$_f\rangle$. Applying induction in the length of derivations, we can prove that $\langle C, E\rangle[\alpha] \overset{*}{\Rightarrow} w$ if and only if $([\alpha C, w) \vdash^* ([E, \epsilon)$.

5. Tabulation

The direct execution of EPDA may be exponential with respect to the length of the input string and may even loop. To get polynomial complexity, we must avoid duplicating computations by tabulating traces of configurations called *items*. The amount of information to keep in an item is the crucial point to determine to get efficient executions.

The tabulation of EPDA using PUSH and POP transitions without restrictions seems to be difficult. By studying the compilation schema of figure 1, we observe that the compilation rules **[ACALL-a]** and **[ACALL-b]** can be combined to form a single rule **[ACALL]** generating transitions WRAP-B+PUSH of the form $C \longmapsto [C, X\ F$:

$$\textbf{[ACALL]}\quad \nabla^{\gamma}_{r,s} \longmapsto [\nabla^{\gamma}_{r,s},\ \Delta^{\gamma}_{r,s}\ \top^{\beta}$$

such that $\beta \in \mathrm{adj}(N^{\gamma}_{r,s+1})$. The **[FCALL-a]** and **[FCALL-b]** can be combined to form a single rule generating transitions WRAP-B+POP of the form $X\ C \longmapsto [C, F$:

$$\textbf{[FCALL]}\quad \Delta^{\gamma}_{r,s}\ \nabla^{\beta}_{f,0} \longmapsto [\nabla^{\beta}_{f,0},\ N^{\gamma}_{r,s+1}$$

such that $N^{\beta}_{f,0} = \mathbf{F}^{\beta}$ and $\beta \in \mathrm{adj}(N^{\gamma}_{r,s+1})$.

In this section, we consider the tabulation of a subset of EPDA consisting of transitions SWAP, WRAP-A, WRAP-B, UNWRAP, WRAP-B+PUSH and WRAP-B+POP.

In order to define items and attending to the form of the transitions, we classify derivations of EPDA into the following types:

Call derivations. Correspond to the propagation of a stack by means of WRAP-B, WRAP-B+PUSH and WRAP-B+POP transitions:

$$\begin{aligned} & (\Upsilon\ [\alpha A, a_{h+1} \ldots a_n) \\ \vdash^* & (\Upsilon\ [A\ \Upsilon_1\ [\alpha X B, a_{i+1} \ldots a_n) \\ \vdash^* & (\Upsilon\ [A\ \Upsilon_1\ [\alpha X C, a_{j+1} \ldots a_n) \end{aligned}$$

where $A, B, C, X \in V_S$, $\alpha \in V_S^*$ and $\Upsilon, \Upsilon_1 \in ([V_S^*)^*$. The two occurrences of α denote the same stack in the sense that α is neither consulted nor modified through the derivation. These derivations are independent of Υ and α, so they can be represented by items

$$[A, h \mid B, i, X, C, j, X \mid -, -, -, -]$$

Return derivations. Correspond to the bottom-up propagation of unitary stack by means of UNWRAP transitions:

$$\begin{aligned}
& (\Upsilon\ [\alpha A, a_{h+1} \ldots a_n) \\
\vdash^* & (\Upsilon\ [A\ \Upsilon_1\ [\alpha X B, a_{i+1} \ldots a_n) \\
\vdash^* & (\Upsilon\ [A\ \Upsilon_1\ [B\ \Upsilon_2\ [\alpha D, a_{p+1} \ldots a_n) \\
\vdash^* & (\Upsilon\ [A\ \Upsilon_1\ [B\ \Upsilon_2\ [E, a_{q+1} \ldots a_n) \\
\vdash^* & (\Upsilon\ [A\ \Upsilon_1\ [C, a_{j+1} \ldots a_n)
\end{aligned}$$

where $A, B, C, D, E, X \in V_S$, $\alpha \in V_S^*$, $\Upsilon, \Upsilon_1, \Upsilon_2 \in ([V_S^*)^*$ and α is passed unaffected through derivation. These derivations are independent of Υ but not with respect to the subderivation $([\alpha D, a_{p+1} \ldots a_n) \vdash^* ([E, a_{q+1} \ldots a_n)$, so they are be represented in compact form by items

$$[A, h \mid B, i, X, C, j, - \mid D, p, E, q]$$

Special point derivations. When $\alpha X = \epsilon$ we have a particular case of previous derivations:

$$(\Upsilon\ [\alpha B, a_{i+1} \ldots a_n) \vdash^* (\Upsilon\ [\alpha C, a_{j+1} \ldots a_n)$$

where $B, C \in V_S$, and $\Upsilon \in ([V_S^*)^*$. These derivations can be represented by items

$$[-, - \mid B, i, -, C, j, - \mid -, -, -, -]$$

To combine items, we use the set of inference rules shown in figures 4 and 5. Each rule is of the form $\frac{\eta_1, \ldots, \eta_k}{\eta'}$ *trans*, meaning that if all antecedents η_i are tabulated items and there exist the transitions *trans*, then the consequent item η' should be created. In order to simplify the inference rules, but without loss of generality, we have considered that scanning is only performed by SWAP transitions. The computation starts with the initial item $[-, - \mid \$_0, 0, -, \$_0, 0, - \mid -, -, -, -]$. An input string $a_1 \ldots a_n$ has been recognized if the final item $[-, - \mid \$_0, 0, -, \$_f, n, - \mid -, -, -, -]$ is present. It can be proved that handling items with the inference rules is equivalent to applying the transitions on the whole stacks.
To illustrate the relation between EPDA and L-LIA, figures 4 and 5 show the transitions of both models of automata that must be considered to apply a given inference rule. Therefore, the proposed tabulated technique can be also applied to L-LIA working with transitions SWAP, WRAP-A, WRAP-B, UNWRAP, WRAP-B+PUSH and WRAP-B+POP.

6. Conclusion

Embedded Push-Down Automata have been redefined: finite-state control has been eliminated and several kinds of transition have been defined. We have also shown that the new definition preserves the equivalence with tree adjoining languages and that tabulation techniques are possible to execute these automata in polynomial time with respect to the length of the input string.

References

DE LA CLERGERIE E. & ALONSO PARDO M. (1998). A tabular interpretation of a class of 2-Stack Automata. In *COLING-ACL'98, 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Proceedings of the Conference*, volume II, p. 1333–1339, Montreal, Quebec, Canada: ACL.

Rule	EPDA transition	L-LIA transition
$\dfrac{[A,h \mid B,i,X,C,j,X \mid -,-,-,-]}{[A,h \mid B,i,X,F,k,X \mid -,-,-,-]}$ where $k=j$ if $a=\epsilon$ and $k=j+1$ if $a \in V_T$	$C \overset{a}{\longmapsto} F$	$C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ]$
$\dfrac{[A,h \mid B,i,X,C,j,- \mid D,p,E,q]}{[A,h \mid B,i,X,F,k,- \mid D,p,E,q]}$ where $k=j$ if $a=\epsilon$ and $k=j+1$ if $a \in V_T$	$C \overset{a}{\longmapsto} F$	$C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ]$
$\dfrac{[A,h \mid B,i,X,C,j,X \mid -,-,-,-]}{[-,- \mid F,j,-,F,j,- \mid -,-,-,-]}$	$C \longmapsto C,[F$	$C[\circ\circ] \longmapsto C[\circ\circ]\ F[\,]$
$\dfrac{[A,h \mid B,i,X,C,j,- \mid D,p,E,q]}{[-,- \mid F,j,-,F,j,- \mid -,-,-,-]}$	$C \longmapsto C,[F$	$C[\circ\circ] \longmapsto C[\circ\circ]\ F[\,]$
$\dfrac{[A,h \mid B,i,X,C,j,X \mid -,-,-,-]}{[A,h \mid F,j,X,F,j,X \mid -,-,-,-]}$	$C \longmapsto [C,F$	$C[\circ\circ] \longmapsto C[\,]\ F[\circ\circ]$
$\dfrac{[A,h \mid B,i,X,C,j,- \mid D,p,E,q]}{[-,- \mid F,j,-,F,j,- \mid -,-,-,-]}$	$C \longmapsto [C,F$	$C[\circ\circ] \longmapsto C[\,]\ F[\circ\circ]$
$\dfrac{[A,h \mid B,i,X,C,j,X \mid -,-,-,-]}{[C,j \mid F,j,X',F,j,X' \mid -,-,-,-]}$	$C \longmapsto [C,X'F$	$C[\circ\circ] \longmapsto C[\,]\ F[\circ\circ X']$
$\dfrac{[A,h \mid B,i,X,C,j,- \mid D,p,E,q]}{[C,j \mid F,j,X',F,j,X' \mid -,-,-,-]}$	$C \longmapsto [C,X'F$	$C[\circ\circ] \longmapsto C[\,]\ F[\circ\circ X']$
$\dfrac{\begin{array}{c}[A,h \mid B,i,X,C,j,X \mid -,-,-,-] \\ [M,m \mid N,t,X',A,h,X' \mid -,-,-,-]\end{array}}{[M,m \mid F,j,X',F,j,X' \mid -,-,-,-]}$	$XC \longmapsto [C,F$	$C[\circ\circ X] \longmapsto C[\,]\ F[\circ\circ]$
$\dfrac{\begin{array}{c}[A,h \mid B,i,X,C,j,X \mid -,-,-,-] \\ [M,m \mid N,t,X',A,h,- \mid D,p,E,q]\end{array}}{[-,- \mid F,j,-,F,j,- \mid -,-,-,-]}$	$XC \longmapsto [C,F$	$C[\circ\circ X] \longmapsto C[\,]\ F[\circ\circ]$

Figure 4: Tabulation rules

GAZDAR G. (1987). Applicability of indexed grammars to natural languages. In U. REYLE & C. ROHRER, Eds., *Natural Language Parsing and Linguistic Theories*, p. 69–94. D. Reidel Publishing Company.

LANG B. (1991). Towards a uniform formal framework for parsing. In M. TOMITA, Ed., *Current Issues in Parsing Technology*, p. 153–171. Norwell, MA, USA: Kluwer Academic Publishers.

NEDERHOF M.-J. (1999). Models of tabulation for TAG parsing. In *Proc. of the Sixth Meeting on Mathematics of Language (MOL 6)*, p. 143–158, Orlando, Florida, USA.

VIJAY-SHANKER K. (1988). *A Study of Tree Adjoining Grammars*. PhD thesis, University of Pennsylvania. Available as Technical Report MS-CIS-88-03 LINC LAB 95 of the Department of Computer and Information Science, University of Pennsylvania.

WEIR D. J. (1994). Linear iterated pushdowns. *Computational Intelligence*, **10** (4), p. 422–430.

Rule	EPDA transition	L-LIA transition
$\dfrac{[-,-\mid F',j,-,F,k,-\mid -,-,-,-] \quad [A,h\mid B,i,X,C,j,X\mid -,-,-,-]}{[A,h\mid B,i,X,G,k,X\mid -,-,-,-]}$	$C \longmapsto C,[F'$ $C,[F \longmapsto G$	$C[\circ\circ] \longmapsto C[\circ\circ]\ F'[\,]$ $C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]$
$\dfrac{[-,-\mid F',j,-,F,k,-\mid -,-,-,-] \quad [A,h\mid B,i,X,C,j,-\mid D,p,E,q]}{[A,h\mid B,i,X,G,k,-\mid D,p,E,q]}$	$C \longmapsto C,[F'$ $C,[F \longmapsto G$	$C[\circ\circ] \longmapsto C[\circ\circ]\ F'[\,]$ $C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]$
$\dfrac{[A,h\mid F',j,X,F,k,-\mid D,p,E,q] \quad [A,h\mid B,i,X,C,j,X\mid -,-,-,-]}{[A,h\mid B,i,X,G,k,-\mid D,p,E,q]}$	$C \longmapsto [C,F'$ $C,[F \longmapsto G$	$C[\circ\circ] \longmapsto C[\,]\ F'[\circ\circ]$ $C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]$
$\dfrac{[-,-\mid F',j,-,F,k,-\mid -,-,-,-] \quad [A,h\mid B,i,X,C,j,-\mid D,p,E,q]}{[A,h\mid B,i,X,G,k,-\mid D,p,E,q]}$	$C \longmapsto [C,F'$ $C,[F \longmapsto G$	$C[\circ\circ] \longmapsto C[\,]\ F'[\circ\circ]$ $C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]$
$\dfrac{[C,j\mid F',j,X',F,k,-\mid D,p,E,q] \quad [A,h\mid B,i,X,C,j,X\mid -,-,-,-] \quad [A,h\mid D,p,X,E,q,-\mid O,u,P,v]}{[A,h\mid B,i,X,G,k,-\mid O,u,P,v]}$	$C \longmapsto [C,X'F'$ $C,[F \longmapsto G$	$C[\circ\circ] \longmapsto C[\,]\ F'[\circ\circ X']$ $C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]$
$\dfrac{[C,j\mid F',j,X',F,k,-\mid O,u,P,v] \quad [A,h\mid B,i,X,C,j,-\mid D,p,E,q] \quad [-,-\mid O,u,-,P,v,-\mid -,-,-,-]}{[A,h\mid B,i,X,G,k,-\mid D,p,E,q]}$	$C \longmapsto [C,X'F'$ $C,[F \longmapsto G$	$C[\circ\circ] \longmapsto C[\,]\ F'[\circ\circ X']$ $C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]$
$\dfrac{[M,m\mid F',j,X',F,k,-\mid D,p,E,q] \quad [A,h\mid B,i,X,C,j,X\mid -,-,-,-] \quad [M,m\mid N,t,X',A,h,X'\mid -,-,-,-]}{[A,h\mid B,i,X,G,k,-\mid F',j,F,k]}$	$XC \longmapsto [C,F'$ $C,[F \longmapsto G$	$C[\circ\circ X] \longmapsto C[\,]\ F'[\circ\circ]$ $C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]$
$\dfrac{[-,-\mid F',j,-,F,k,-\mid -,-,-,-] \quad [A,h\mid B,i,X,C,j,X\mid -,-,-,-] \quad [M,m\mid N,t,X',A,h,-\mid D,p,E,q]}{[A,h\mid B,i,X,G,k,-\mid F',j,F,k]}$	$XC \longmapsto [C,F'$ $C,[F \longmapsto G$	$C[\circ\circ X] \longmapsto C[\,]\ F'[\circ\circ]$ $C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]$

Figure 5: Tabulation rules

Workshop TAG+5, Paris, 25-27 May 2000

Practical aspects in compiling tabular TAG parsers

Miguel A. Alonso[†], Djamé Seddah[‡], and Éric Villemonte de la Clergerie[*]

[†]Departamento de Computación, Universidad de La Coruña
Campus de Elviña s/n, 15071 La Coruña (Spain)
alonso@dc.fi.udc.es
[‡]LORIA, Technopôle de Nancy Brabois,
615, Rue du Jardin Botanique - B.P. 101, 54602 Villers les Nancy (France)
Djame.Seddah@loria.fr
*INRIA, Domaine de Voluceau
Rocquencourt, B.P. 105, 78153 Le Chesnay (France)
Eric.De_La_Clergerie@inria.fr

Abstract

This paper describes the extension of the system DyALog to compile tabular parsers from Feature Tree Adjoining Grammars. The compilation process uses intermediary 2-stack automata to encode various parsing strategies and a dynamic programming interpretation to break automata derivations into tabulable fragments.

1. Introduction

This paper describes the extension of the system DyALog in order to produce tabular parsers for Tree Adjoining Grammars [TAGs] and focuses on some practical aspects encountered during the process. By tabulation, we mean that traces of (sub)computations, called *items*, are tabulated in order to provide computation sharing and loop detection (as done in Chart Parsers).

The system DyALog[1] handles logic programs and grammars (DCG). It has two main components, namely an abstract machine that implements a generic fix-point algorithm with subsumption checking on objects, and a bootstrapped compiler. The compilation process first compiles a grammar into a Push-Down Automaton [PDA] that encodes the steps of a parsing strategy. PDAs are then evaluated using a Dynamic Programming [DP] interpretation that specifies how to break the PDA derivations into elementary tabulable fragments, how to represent, in an optimal way, these fragments by *items*, and how to combine items and transitions to retrieve all PDA derivations. Following this DP interpretation, the transitions of the PDAs are analyzed at compile time to emit application code as well as to build code for the skeletons of items and transitions that may be needed at run-time.

Recently, (Villemonte de la Clergerie & Alonso Pardo, 1998) has presented a variant of 2-stack automata [2SA] and presented a DP interpretation for them. These 2SAs allow the encoding of many parsing strategies for TAGs, ranging from pure bottom-up ones to valid-prefix top-down ones. For all strategies, the DP interpretation ensures worst-case complexities in time $O(n^6)$ and space $O(n^5)$, where n denotes the length of the input string.

This theoretical work has been implemented in DyALog with minimum effort. Only a few files have been added to the DyALog compiler and no modification was necessary in the DyALog machine. Several extensions and optimizations were added: handling of Feature TAGs,

[1] Freely available at http://atoll.inria.fr/~clerger

use of more sophisticated parsing strategies, use of meta-transitions to compact sequences of transitions, use of more efficient items, and possibility to escape to logic predicates.

2. Tree Adjoining Grammars

We assume the reader to be familiar with TAGs (Joshi, 1987) and with the basic notions in Logic Programming (substitution, unification, subsumption, ...). Let us just recall that Feature TAGs are TAGs where a pair of first-order arguments `top` T_ν and `bottom` B_ν may be attached to each node ν labeled by a non-terminal.

We have chosen a Prolog-like linear representation of trees. For instance, the grammar `count` (Fig. 1) recognizes the language $a^n b^n e c^n d^n$ with $n > 0$ and returns the number n of performed adjunctions. It corresponds (omitting the top and bottom arguments) to the trees on the right side. By default, the nodes are adjoinable, except when they are leaves or are prefixed with $-$. Obligatory Adjunction [OA] nodes are prefixed with ++ and foot nodes by $*$. Node arguments are introduced with the operators **at**, **and**, **top**, and **bot** and escapes to Prolog are enclosed with {} (as done in DCGs).

```
tree top=s(X) and bot=s(0) at ++s("e").
auxtree top=s(XpI)
     at -s("a",
            top=s(X) and bot=s(Y)
        at s("b", bot=s(Y) at *s, "c"),
          {XpI is X+1},
          "d").
```

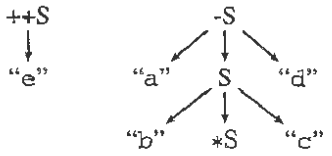


Figure 1: Concrete representation of grammar `count` and corresponding trees

3. Compiling into 2SAs following a modulated Call/Return Strategy

2SAs (Becker, 1994) are extensions of PDAs working on a pair of stacks and having the power of a Turing Machines. We restrict them by considering asymmetric stacks, one being the Master Stack MS where most of the work is done and the other being the Auxiliary Stack AS only used for restricted "bookkeeping" (Villemonte de la Clergerie & Alonso Pardo, 1998). When parsing TAGs, MS is used to save information about the different elementary tree traversals that are under way while AS saves information about adjunctions, as suggested in figure 2.

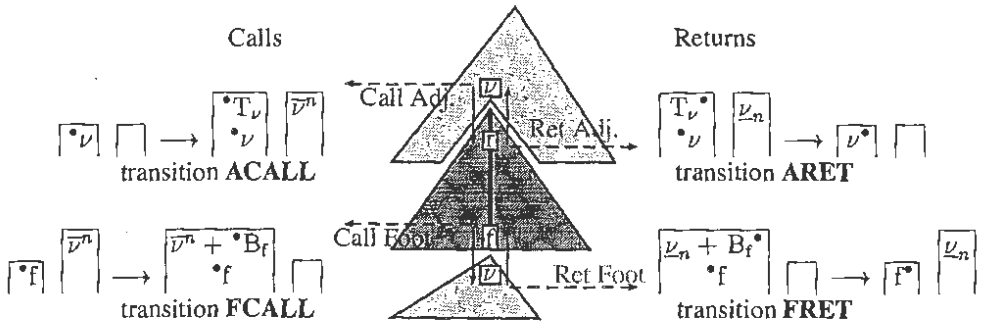


Figure 2: Illustration of some steps

Figure 2 also illustrates the notion of *modulated Call/Return strategy*: an elementary tree α is traversed in a pre-order way (skipping Null Adjunction [NA] internal nodes) and when predicting an adjunction on node ν, the traversal is suspended and some *prediction information*

${}^{\bullet}\mathrm{T}_\nu$ relative to the top argument T_ν of ν is pushed on MS (step **Call**) and used to select some auxiliary tree β (step **Select**). Some information $\overline{\nu^n}$ (partially) identifying ν is also pushed on AS and propagated to the foot of the auxiliary tree. Then $\overline{\nu^n}$ is popped, combined with some information ${}^{\bullet}\mathrm{B}_\mathrm{f}$ and pushed on MS in order to select the traversal of the subtree α_ν rooted at ν. Once α_ν has been traversed, we pop MS and resume the suspended traversal of β. We also push propagation information $\underline{\nu_n}$ on AS about the adjunction node. Once β has been traversed, we publish some propagation information about β (step **Publish**). We then pop both MS and AS and resume the suspended traversal of α, checking with $\underline{\nu_n}$ and $\mathrm{T}_\nu{}^{\bullet}$ that the adjunction has been correctly handled (step **Return**).

For each kind of suspension that may occur during a tree traversal, we get a pair of Call/Return transitions and a related pair of Select/Publish transitions. For TAGs, we have three kinds of suspension, occurring at substitution, adjunction, and foot nodes. We explicit here the transitions relative to an adjunction on node ν and to an auxiliary tree of root r and foot f. A transition τ of the form $(\Xi, \xi) \longmapsto (\Theta, \theta)$ applies on any *configuration* $(\mathbf{MS}, \mathbf{AS}) = (\Psi\Xi', \psi\xi')$ and returns $((\Psi\Theta)\sigma, (\psi\theta)\sigma)$ whenever the most general unifier $\sigma = \mathrm{mgu}(\Xi\xi, \Xi'\xi')$ exists. [2]

ACALL	$({}^{\bullet}\nu, \epsilon) \longmapsto ({}^{\bullet}\nu\ {}^{\bullet}\mathrm{T}_\nu, \overline{\nu^n})$	**ARET**	$({}^{\bullet}\nu\ \mathrm{T}_\nu{}^{\bullet}, \underline{\nu_n}) \longmapsto (\nu^{\bullet}, \epsilon)$
ASEL	$({}^{\bullet}\mathrm{T}_\mathrm{r}, \epsilon) \longmapsto ({}^{\bullet}\mathrm{r}, \epsilon)$	**APUB**	$(\mathrm{r}^{\bullet}, \epsilon) \longmapsto (\mathrm{T}_\mathrm{r}{}^{\bullet}, \epsilon)$
FCALL	$(.\mathrm{f}, X) \longmapsto (.\mathrm{f}\ (.\mathrm{B}_\mathrm{f} + X), \epsilon)$	**FRET**	$(.\mathrm{f}\ (\mathrm{B}_{\mathrm{f}\bullet} + Y), \epsilon) \longmapsto (\mathrm{f}_\bullet, Y)$
FSEL	$((.\mathrm{B}_\nu + \overline{\nu^n}), \epsilon) \longmapsto (.\nu, \epsilon)$	**FPUB**	$(\nu_\bullet, \epsilon) \longmapsto ((\mathrm{B}_{\nu\bullet} + \underline{\nu_n}), \epsilon)$

In these transitions, X, Y denote free variables and *dotted atoms* ${}^{\bullet}\nu$, $\nu^{\bullet}$ (resp. $.\nu$, $\nu_\bullet$) denote computation points during a traversal that are just left and right of ν including (resp. not including) adjunction. The *prediction atoms* ${}^{\bullet}\mathrm{T}_\nu$, $.\mathrm{B}_\nu$ and *propagation atoms* $\mathrm{T}_\nu{}^{\bullet}$, $\mathrm{B}_{\nu\bullet}$ are built using *modulations* from the node arguments T_ν and B_ν completed by position variables $({}^{\bullet}\mathrm{P}_\nu, \mathrm{P}_\nu{}^{\bullet})$ and $(.\mathrm{P}_\nu, \mathrm{P}_{\nu\bullet})$ used to delimit the span of ν including or not adjunction. [3] A *modulation* (Barthélemy & Villemonte de la Clergerie, 1998) is formalized as a pair of projection morphisms $(\overline{\ }, \underline{\ })$ such that, for all atoms A, B, $\mathrm{mgu}(A, B) = \mathrm{mgu}(\overline{A}\,\underline{A}, \overline{B}\,\underline{B})$. For TAGs, we offer the possibility to have distinct modulations $(\overline{\ }^t, \underline{\ }_t)$ and $(\overline{\ }^b, \underline{\ }_b)$ for top and bottom arguments, as well as a third one $(\overline{\ }^n, \underline{\ }_n)$ for nodes, leading to the following definitions: [4]

$$
\begin{array}{ll}
{}^{\bullet}\mathrm{T}_\nu = \overline{T_\nu[{}^{\bullet}\mathrm{P}_\nu;\ \mathrm{P}_\nu{}^{\bullet}]}^{t} & .\mathrm{B}_\nu = \overline{B_\nu[.\mathrm{P}_\nu;\ \mathrm{P}_{\nu\bullet}]}^{b} \\
\mathrm{T}_\nu{}^{\bullet} = \underline{T_\nu[{}^{\bullet}\mathrm{P}_\nu;\ \mathrm{P}_\nu{}^{\bullet}]}_{t} & \mathrm{B}_{\nu\bullet} = \underline{B_\nu[.\mathrm{P}_\nu;\ \mathrm{P}_{\nu\bullet}]}_{b}
\end{array}
$$

Modulation is useful to tune the top-down prediction of trees and the bottom-up propagation of recognized trees. It allows an uniform description of a wide family of parsing strategies, ranging from pure bottom-up ones to prefix-valid top-down ones and also including mixed strategies such as Earley-like. In practice, a directive **tag_mode**(s/1,top,+(−),+,−) states that, for a node argument $T = \mathrm{s}(X)$ and position variables $({}^{\bullet}\mathrm{P}, \mathrm{P}^{\bullet}) = (L, R)$, we get ${}^{\bullet}\mathrm{T} = \mathtt{call\_s\_1}(L)$ and $\mathrm{T}^{\bullet} = \mathtt{ret}(R, X)$.

In practice, the transitions built following a Call/Return strategy may be grouped in *meta-transitions* by (a) grouping pairs of related call and return transition and (b) considering dotted nodes as *continuations*. For instance, figure 3 shows the skeleton of a meta-transition representing the traversal of an auxiliary tree with root r and some adjoinable node ν. [5]

[2] Transitions and configurations have been simplified in this paper for sake of clarity and space.

[3] Of course, there are many congruence relations between the position variables. For instance, if ν immediately precedes μ in the traversal, we have $\mathrm{P}_\nu{}^{\bullet} \equiv {}^{\bullet}\mathrm{P}_\mu$. When ν is not adjoinable, we have ${}^{\bullet}\mathrm{P}_\nu \equiv .\mathrm{P}_\nu$ and $\mathrm{P}_\nu{}^{\bullet} \equiv \mathrm{P}_{\nu\bullet}$.

[4] There is also a specific modulation for substitution nodes.

[5] Actually, we have considered the case of a mandatory adjunction at ν. To handle non mandatory ones, we add disjunction points in the meta-transitions and share common continuations between alternatives.

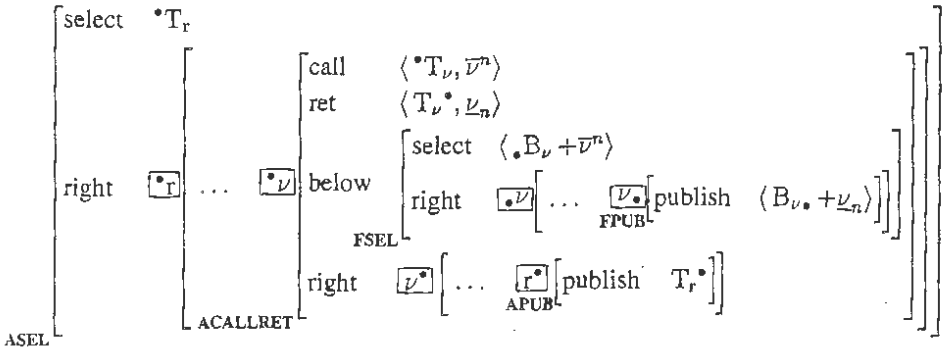


Figure 3: Sketch of a meta-transition for an auxiliary tree

4. Preparing the Dynamic Programming evaluation

The next compilation phase unfolds the meta-transitions and identifies which objects (items or transitions) may arise at run-time. This analysis is based on a DP interpretation for 2SAs.

Items We consider two kinds of items, namely Context-Free [CF] Items $\boldsymbol{ABC}$ (also represented by $\boldsymbol{AB}[\diamond\diamond]\boldsymbol{C}$) and Escaped Context-Free [xCF] Items $\boldsymbol{AB}[\boldsymbol{DE}]\boldsymbol{C}$ representing subderivations passing by configurations $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, $\mathcal{D}$ and $\mathcal{E}$. [6] Computation sharing stems from the fact that we don't save a full configuration $\mathcal{X} = (\Xi X, \xi x)$ but rather a *mini configuration* $\langle X, x\rangle$ or a *micro configuration* $\langle X\rangle$. Better, it is possible to keep, in some cases, only a fraction $\epsilon(X)$ of the information available in a stack element X. For instance, we take $\epsilon(^\bullet\nu) = {}^\bullet\mathrm{T}_\nu$ for an adjoinable node ν, and $\epsilon(_\bullet\nu) = {}_\bullet\mathrm{B}_\nu$ for a foot node. Finally, $\boldsymbol{A} = \langle\epsilon A\rangle$, $\boldsymbol{B} = \langle\epsilon B, b\rangle$ (when $\boldsymbol{B} \neq \boldsymbol{A}$), and $\boldsymbol{D} = \langle\epsilon D, d\rangle$ (when $\boldsymbol{D} \neq \diamond$). Table 1 shows some items relative to an adjunction at ν. If ν dominates some foot g in an adjunction on μ, $[\boldsymbol{DE}] = [\langle {}_\bullet\mathrm{B}_\mathrm{g}, a\rangle\langle \mathrm{B}_{\mathrm{g}_\bullet} + b\rangle]$ and $(a, b) = (\overline{\mu^n}, \underline{\mu_n})$; otherwise $[\boldsymbol{DE}] = [\diamond\diamond]$ and $(a, b) = (\diamond, \diamond)$.

	after CALL	before RET
on ADJ ν	$\langle {}^\bullet\mathrm{T}_\nu\rangle\langle {}^\bullet\mathrm{T}_\nu\rangle\langle {}^\bullet\mathrm{T}_\nu, \overline{\nu^n}\rangle$	$\langle {}^\bullet\mathrm{T}_\nu\rangle\langle {}^\bullet\mathrm{T}_\nu\rangle[\boldsymbol{DE}]\langle \mathrm{T}_\nu{}^\bullet, \underline{\nu_n}\rangle$
on FOOT f	$\langle {}^\bullet\mathrm{T}_\nu\rangle\langle {}_\bullet\mathrm{B}_\mathrm{f}, \overline{\nu^n}\rangle\langle {}_\bullet\mathrm{B}_\mathrm{f} + \overline{\nu^n}, a\rangle$	$\langle {}^\bullet\mathrm{T}_\nu\rangle\langle {}_\bullet\mathrm{B}_\mathrm{f}, \overline{\nu^n}\rangle[\boldsymbol{DE}]\langle \mathrm{B}_{\mathrm{f}_\bullet} + \underline{\nu_n}, b\rangle$

Table 1: Refined items at adjunction and foot nodes

Application rules Figure 4 shows (some of) the application rules used to combine items and transitions. The antecedent transition and items are (implicitly) correlated using unification with the resulting most general unifier applied to the consequent item. Component that need not be consulted are replaced by holes $\star$. Similar items occurring in different rules have been supscripted by I, J, K and L. Note that these rules derive from more abstract ones, independent of any strategy, that we have instantiated, to be more concrete, for the Call/Return strategies.

Projections Time complexity may be reduced by removing from objects the components that are not consulted (marked by $\star$), leading to tabulate one or more *projected* objects instead of the original one. In particular, instead of tabulating $K = \langle {}^\bullet\mathrm{T}_\nu\rangle\langle {}_\bullet\mathrm{B}_\mathrm{f}, \overline{\nu^n}\rangle[\boldsymbol{DE}]\langle \mathrm{B}_{\nu_\bullet} + \underline{\nu_n}, b\rangle$

[6] The different conditions satisfied by these configurations are outside the scope of this paper.

$$\text{(ACALL)}\quad \frac{(\,^{\bullet}\nu,\epsilon)\longmapsto(\,^{\bullet}\nu\;{}^{\bullet}\mathrm{T}_\nu,\overline{\nu}^n)\quad A\star\langle\,^{\bullet}\nu,\star\rangle^I}{\langle\,^{\bullet}\mathrm{T}_\nu\rangle\langle\,^{\bullet}\mathrm{T}_\nu\rangle\langle\,^{\bullet}\mathrm{T}_\nu,\overline{\nu}^n\rangle}$$

$$\text{(FCALL)}\quad \frac{(\,_{\bullet}\mathrm{f},\overline{\nu}^n)\longmapsto(\,_{\bullet}\mathrm{f}\,(\,_{\bullet}\mathrm{B}_\mathrm{f}+\overline{\nu}^n),\epsilon)\quad \begin{array}{c}A\star\langle\,^{\bullet}\nu,a\rangle^I\\ \langle\,^{\bullet}\mathrm{T}_\nu\rangle\star(\,_{\bullet}\mathrm{f},\overline{\nu}^n\rangle^J\end{array}}{A\langle\,_{\bullet}\mathrm{B}_\mathrm{f},\overline{\nu}^n\rangle\langle\,_{\bullet}\mathrm{B}_\mathrm{f}+\overline{\nu}^n,a\rangle}$$

$$\text{(FRET)}\quad \frac{(\,_{\bullet}\mathrm{f}\,(\,\mathrm{B}_{\mathrm{f}\bullet}+\underline{\nu}_n),\epsilon)\longmapsto(\,\mathrm{f}_\bullet,\underline{\nu}_n)\quad \begin{array}{c}\langle\,^{\bullet}\mathrm{T}_\nu\rangle O\langle\,_{\bullet}\mathrm{f},\overline{\nu}^n\rangle^J\\ A\star\langle\,^{\bullet}\nu,a\rangle^I\\ A\langle\,_{\bullet}\mathrm{B}_\mathrm{f},\overline{\nu}^n\rangle[D\star]\langle\,_{\bullet}\mathrm{B}_\mathrm{f}+\underline{\nu}_n,\star\rangle^K\end{array}}{\langle\,^{\bullet}\mathrm{T}_\nu\rangle O[\langle\,_{\bullet}\mathrm{B}_\mathrm{f},\overline{\nu}^n\rangle\langle\,_{\bullet}\mathrm{B}_\mathrm{f}+\underline{\nu}_n\rangle]\langle\,\mathrm{f}_\bullet,\underline{\nu}_n\rangle}\qquad D=\langle\star,a\rangle$$

$$\text{(dxCF)}\quad \frac{\begin{array}{c}\star\langle\,_{\bullet}\mathrm{B}_\mathrm{f},\overline{\nu}^n\rangle[DE]\langle\,\mathrm{B}_{\mathrm{f}\bullet}+\underline{\nu}_n,\star\rangle^K\\ \langle\,^{\bullet}\mathrm{T}_\nu\rangle\langle\,^{\bullet}\mathrm{T}_\nu\rangle[\langle\,_{\bullet}\mathrm{B}_\mathrm{f},a\rangle\langle\,\mathrm{B}_{\mathrm{f}\bullet}+\underline{\nu}_n\rangle]\langle\,\mathrm{T}_\nu{}^{\bullet}\rangle\end{array}}{\langle\,^{\bullet}\mathrm{T}_\nu\rangle\langle\,^{\bullet}\mathrm{T}_\nu\rangle[[DE]]\langle\,\mathrm{T}_\nu{}^{\bullet}\rangle^L}$$

$$\text{(ARET)}\quad \frac{(\,^{\bullet}\nu\;\mathrm{T}_\nu{}^{\bullet},\underline{\nu}_n)\longmapsto(\,\nu^{\bullet},\epsilon)\quad \begin{array}{c}AN\langle\,^{\bullet}\nu,a\rangle^I\\ A\star[DE]\langle\star,b\rangle^K\\ \langle\,^{\bullet}\mathrm{T}_\nu\rangle\langle\,^{\bullet}\mathrm{T}_\nu\rangle[[DE]]\langle\,\mathrm{T}_\nu{}^{\bullet},\underline{\nu}_n\rangle^L\end{array}}{AN[DE]\langle\,\nu^{\bullet},b\rangle}\qquad D=\langle\,_{\bullet}\mathrm{B}_\mathrm{f},a\rangle$$

Figure 4: Some application rules for the Dynamic Programming interpretation

corresponding to the traversal of the subtree rooted at ν, we tabulate 3 projections used with Rules (FRET), (dxCF), and (ARET). The effectiveness of projections is still to be validated!

Partial and immediate applications To further reduce worst-case complexity, DyALog is configured to combine, at each step, a single item with a single transition. It is therefore necessary to decompose the application rules to follow this scheme, which is done by introducing intermediate pseudo transitions materializing partial applications. Subsumption checking can be done on these intermediate transitions, leading to better computation sharing. We get cascades of partial applications as illustrated by figure 5. An object $(Rule)\alpha_1\ldots\alpha_k$ represents the (intermediate) structure associated to the partial application with *(Rule)* of $\alpha_1,\ldots,\alpha_k$ and α_i being either the transition τ or the ith item (from the top) in the rule.
Another advantage of partial application w.r.t. meta-transitions is the possibility to do *immediate partial application*, with no tabulation of some items. For instance, when reaching an adjunction node ν, we should tabulate item $I = A\star\langle\,^{\bullet}\nu,a\rangle$ and wait for other items to apply Rules (FCALL), (FRET), and (ARET). Instead, we immediately perform partial applications and tabulate the intermediate objects (see the underlined objects in fig. 5). We also apply Rule (ACALL) and tabulate "Call Aux Item" $\mathbf{CAI} = \langle\,^{\bullet}\mathrm{T}_\nu\rangle\langle\,^{\bullet}\mathrm{T}_\nu\rangle\langle\,^{\bullet}\mathrm{T}_\nu,\overline{\nu}^n\rangle$. Immediate applications are also done when reaching a foot f with item $J = \langle\,^{\bullet}\mathrm{T}_\nu\rangle O\langle\,_{\bullet}\mathrm{f},\overline{\nu}^n\rangle$.

Shared Derivation Forest They are extracted from tabulated objects by recursively following *typed backpointers* to their parents and are expressed as Context-Free Grammars(Vijay-Shanker & Weir, 1993). Parsing `aabbeccdd` with the grammar of figure 1 returns the following forest:

```
s(2)(0,9)              1 <-- 2 % Der. of elem tree with adj.
s(2)(0,9) * s(0)(4,5)  2 <-- 3 % Der. of aux. tree with adj.
s(1)(1,8) * s(0)(3,6)  3 <--   % Der. of aux. tree
```

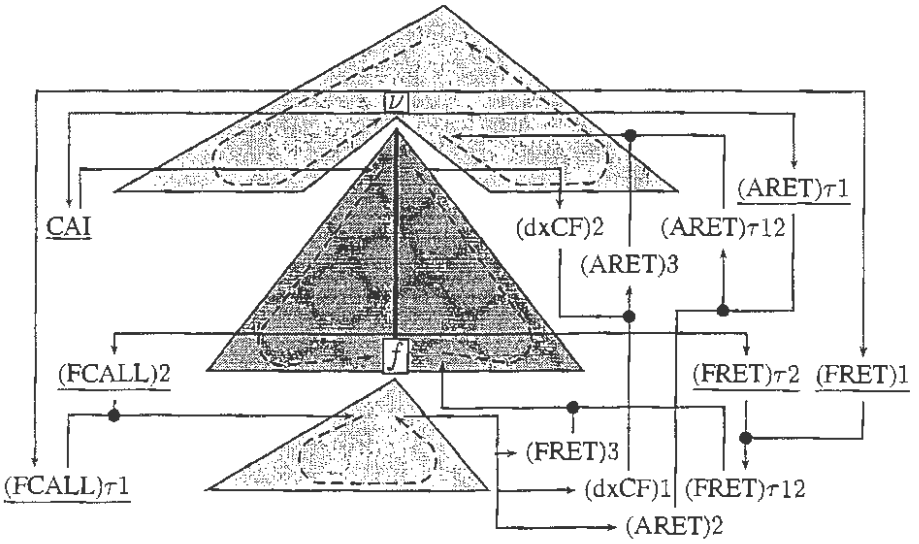


Figure 5: Cascades of partial evaluations related to an adjunction

5. Analysis and conclusion

In the worst case and in the case of TAGs without features, the number of created objects (using subsumption checking) is in $O(n^5)$ and the number of tried applications is in $O(n^6)$. These complexities come from the number of different instantiated position variables which may occur in objects or be consulted (for applications). [7] These complexities remain polynomial when dealing with DATALOG features (no symbol of functions) and may be exponential otherwise. Time complexity is directly related to the number of tried applications and created objects if objects can be accessed and added in constant time. The indexing scheme of DyALog based on trees of hashed tables ensures this property only for pure and DATALOG TAGs.

These different remarks about complexity have been confirmed for small "pathological" grammars. However, some recent experimentations done with a prefix-valid top-down parser compiled from a French XTAG-like non lexicalized grammar of 50 trees (with DATALOG features) have shown a much better behavior (0.5s to 2s for sentences of 3 to 15 words on a Pentium-II 450Mhz). We hope to improve these figures by factorizing tree traversals and using (when possible) specialized left and right adjunctions.

References

BARTHÉLEMY F. & VILLEMONTE DE LA CLERGERIE E. (1998). Information flow in tabular interpretations for generalized push-down automata. *Theoretical Computer Science*, **199**, 167–198.

BECKER T. (1994). A new automaton model for TAGs: 2-SA. *Computational Intelligence*, **10** (4).

JOSHI A. K. (1987). An introduction to tree adjoining grammars. In A. MANASTER-RAMER, Ed., *Mathematics of Language*, p. 87–115. Amsterdam/Philadelphia: John Benjamins Publishing Co.

VIJAY-SHANKER K. & WEIR D. J. (1993). The use of shared forest in tree adjoining grammar parsing. In *Proc. of the 6th Conference of the European Chapter of ACL*, p. 384–393: EACL.

VILLEMONTE DE LA CLERGERIE E. & ALONSO PARDO M. (1998). A tabular interpretation of a class of 2-stack automata. In *Proc. of ACL/COLING'98*.

[7] Note that uninstantiated or duplicated instantiated position variables may occur.

Using TAGs, a Tree Model, and a Language Model for Generation

Srinivas Bangalore et Owen Rambow

AT&T Labs-Research, B233
180 Park Ave, PO Box 971
Florham Park, NJ 07932-0971, USA
srini, rambow@research.att.com

Abstract

Previous stochastic approaches to sentence realization do not include a tree-based representation of syntax. While this may be adequate or even advantageous for some applications, other applications profit from using as much syntactic knowledge as is available, leaving to a stochastic model only those issues that are not determined by the grammar. In this paper, we present three results in the context of surface realization: a stochastic tree model derived from a parsed corpus outperforms a tree model derived from unannotated corpus; exploiting a hand-crafted grammar in conjunction with a tree model outperforms a tree model without a grammar; and exploiting a tree model in conjunction with a linear language model outperforms just the tree model.

1. Introduction

Most sentence realizers (systems that take a fairly shallow semantic or lexico-syntactic representation and return a surface string in the target language) are entirely grammar-based, including quite a few based on TAG (starting with (McDonald and Pustejovsky1985)). Generators using hand-crafted grammars are useful for constrained applications, when strict control over the output is needed, and when a sufficiently large grammar is available. Recently, (Langkilde and Knight1998a) and (1998b) have used stochastic techniques in NLG, by mapping semantic primitives to a set of possible ordered sequences of tokens, and assembling theses into a lattice. They then use a linear language model to select the best path through the lattice. Stochastic generators are useful when a large grammar is not available, or when the range of generated utterances is large.

To date, generators are either fully hand-crafted or entirely syntax-free, and use a stochastic model only at the level of linear strings. In this paper we present FERGUS (Flexible Empiricist/Rationalist Generation Using Syntax). FERGUS follows Knight and Langkilde's seminal work in using an n-gram language model, but we augment it with a tree-based stochastic model and a TAG grammar. We argue that the combination of all three key modules of our approach – tree model, TAG grammar, linear model – is crucial and improves over models using only a subset of these modules.

The structure of the paper is as follows. In Section 2, we start out by describing a modification to standard TAG that we have followed for the sake of generation. In Section 3, we describe the architecture of the system, and some of the modules. In Section 4 we discuss three experiments. We conclude with a summary of on-going and future work.

2. γ-Trees

We depart from standard TAG practice in our treatment of trees for adjuncts (such as adverbs or adjectives), and instead follow (McDonald and Pustejovsky1985) and (Rambow et al.1995). While in XTAG the elementary tree for an adjunct contains phrase structure that attaches the adjunct to a node in another tree with the specified label (say, VP) from the specified direction (say, from the left), in our system the trees for adjuncts simply express their own phrase and argument structure (active valency), but not how they connect to the lexical item they modify (passive valency). Information about passive valency is kept in the *adjunction table* which is associated with the grammar. We call trees that can adjoin to other trees (and have entries in the adjunction table) γ-*trees*, the other trees (which can only be substituted into other trees) are α-*trees*, while β trees are now restricted to predicative auxiliary trees. Note that each γ-tree corresponds to a set of predicative auxiliary trees in a traditional TAG grammar (which share common phrase structure but attach differently).

3. System Overview

FERGUS is composed of three modules: the stochastic Tree Chooser, the grammar-based Unraveler, and the stochastic Linear Precedence (LP) Chooser. The input to the system is a dependency tree as shown in Figure 1 on the left. Note that the nodes are labeled only with lexemes, not with supertags. The Tree Chooser then uses a stochastic tree model to choose TAG trees for the nodes in the input structure. This step can be seen as analogous to supertagging (Bangalore and Joshi1999), except that now supertags (i.e., names of trees) must be found for words in a tree rather than for words in a linear sequence. The Unraveler then uses the XTAG grammar (XTAG-Group1999) to produce a lattice of all possible linearizations that are compatible with the supertagged tree and the XTAG grammar. The LP Chooser then chooses the most likely traversal of this lattice, given a language model. We discuss the input representation and the three components in turn. For a more detailed overview over the system, see (Bangalore and Rambow2000b).

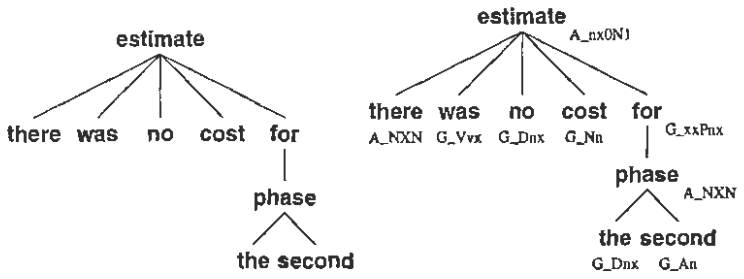


Figure 1: 80

3.1. *The Input to* FERGUS

As mentioned, the input to FERGUS is a dependency tree. We make three remarks.

First: we see the task of FERGUS as one of incremental specification. Clearly, a TAG derivation tree fully specifies a derivation. It consists of three types of information for each node: the supertag, the lexical anchor, and the address at which this tree is attached at the tree of the mother node (except of course for the root). In FERGUS, we assume that the input contains

only the lexeme, and the other information is added during the generation process. As a result, the input tree is actually semantically underspecified – for example, from the tree on the left in Figure 1, we could in theory obtain a sentence such as *cost was no estimate for the second phase there*, by choosing an α-tree for *cost* and an adverbial auxiliary tree for *there*. Thus we leave it to the corpus to determine how the lexemes relate to each other. Clearly, for many applications, we know which role the dependents of a lexeme play in the argument structure of their head, and FERGUS allows us to annotate dependents with a role feature (adj for adjuncts, func for function words, or for arguments an integer referring to the numbering of argument slots in the XTAG grammar). However, the option of leaving the role underspecified is useful in machine translation applications (when the parser cannot fully determine the syntactic roles in the source language), and for all applications because often it is difficult to determine whether a dependent is an argument or an adjunct (for example, *Baton Rouge* in *he disappeared from Baton Rouge*). In realizers that do not allow for underspecification, it is necessary to consult the linguistic data base (lexicon) of the realizer in order to construct valid inputs; FERGUS allows us to leave the role of some dependents open.

Second: in the system that we used in the experiments described in Section 4, all words (including function words) need to be present in the input representation, fully inflected. This is of course unrealistic for applications. In this paper, we only aim to show that the use of our three modules improves performance of a generator.

Third: as is well known, because of lexicalization, the derivation tree of TAG is a dependency tree. However, because of the definition of adjunction, there are cases in which the derivation tree is not the dependency tree as commonly assumed, in particular cases of clausal embedding using predicative auxiliary trees (Rambow and Joshi1996). Because our training corpus is annotated with standard dependency trees and not derivation trees, we assume standard dependency trees as input, and treat the footnode of predicative auxiliary trees as a substitution node. As a consequence, we do not currently exploit the full formal power of TAG and we are not able to generate long-distance dependencies. We intend to address this issue in future work.

3.2. The Tree Chooser

In general, for a given dependency tree, each node can be given more than one supertag in order to turn the tree into a valid derivation tree. If the syntactic roles of the daughter nodes are not fixed, then the subcategorization frame of the mother node needs to be chosen, but even if they are fixed, choices remain such as voice, and how to realize the arguments (for example, dative shift or topicalization). Ideally, we would have a correct set of rules for each choice and enough data in the generation process so that we can make the decision. (Stone and Doran1997) have shown how to integrate such rules into a TAG framework. However, the required research to find the correct rules is nowhere near completed and the data required in order to make such decisions is not always available in generation. An alternative is to assume a default ordering of choices, as does (Becker1998). This cuts back on the required off-line theoretical work and on-line data, but represents a rather inflexible solution. We have chosen to use a stochastic tree model sensitive to the lexemes of the mother and daughter nodes to make this choice.

The Tree Chooser draws on a tree model, which is a representation of an XTAG derivation for 1,000,000 words of the Wall Street Journal. It makes the simplifying assumptions that the choice of a tree for a node depends only on its daughter nodes, thus allowing for a top-down dynamic programming algorithm. Specifically, a node η in the input structure is assigned a supertag s so that the probability of finding the treelet composed of η with supertag s and all of its daughters (as found in the input structure) is maximized, and such that s is compatible with η's mother and her supertag s_m. Here, "compatible" means that the tree represented by s can be adjoined or

substituted into the tree represented by s_m, according to the XTAG grammar. For our example sentence, the input to the system is the tree shown in Figure 1 on the left, and the output from the Tree Chooser is the tree as shown in Figure 1 on the right. Note that while a derivation tree in TAG fully specifies a derivation and thus a surface sentence, the output from the Tree Chooser does not, because for us adjunct auxiliary trees are γ-trees and thus underspecified with respect to the adjunction site and/or the adjunction direction (from left or from right) in the tree of the mother node, and they may be unordered with respect to other adjuncts (for example, the famous adjective ordering problem). (See Section 2 above.) Furthermore, supertags may have been chosen incorrectly or not at all.

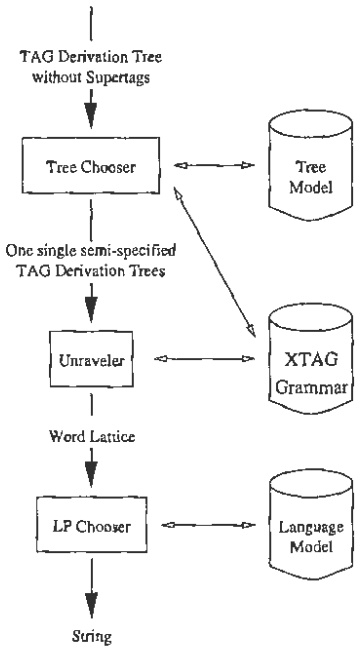


Figure 2: Architecture of FERGUS

3.3. The Unraveler

The Unraveler takes as input the semi-specified derivation tree (Figure 1 on the right) and produces a word lattice. Each node in the derivation tree consists of a lexical item and a supertag. The linear order of the daughters with respect to the head position of a supertag is specified in the XTAG grammar. This information is consulted to order the daughter nodes with respect to the head at each level of the derivation tree. In cases where a daughter node can be attached at more than one place in the head supertag (as is the case in our example for *was* and *for*), a disjunction of all these positions are assigned to the daughter node. A bottom-up algorithm then constructs a lattice that encodes the strings represented by each level of the derivation tree. The lattice at the root of the derivation tree is the result of the Unraveler.

3.4. The Linear Precedence Chooser

The lattice output from the Unraveler encodes all possible word sequences permitted by the derivation structure. Again, it might be possible to develop rules to choose among possible adjunction sites for adverbs, or for choosing possible orderings of adjuncts at the same adjunction site (such as for the notorious adnominal adjective ordering problem). However, such research is not completed, and we instead propose to use a stochastic model in order to make this choice. We rank the word sequences encoded by the lattice in the order of their likelihood by composing the lattice with a finite-state machine representing a trigram language model. This model has been constructed from 1,000,0000 words of Wall Street Journal corpus. We pick the best path through the lattice resulting from the composition using the Viterbi algorithm, and this top ranking word sequence is the output of the LP Chooser.

4. Experiments and Results

In order to show that the use of a tree model, a grammar, and a linear model does indeed help performance, we performed four experiments:

- For the **baseline** experiment, we impose a random tree structure for each sentence of the corpus and build a Tree Model whose parameters consist of whether a lexeme l_d precedes or follows her mother lexeme l_m. We call this the Baseline Left-Right (LR) Model. This model generates *There was estimate for phase the second no cost* . for our example input.

- In the second experiment (**TM-LM**), we derive the parameters for the LR model from an annotated corpus, in particular, the XTAG derivation tree corpus. Thus, we use a tree model and a linear language model, but not the TAG grammar. This model generates *There no estimate for the second phase was cost* . for our example input.

- In the third experiment (**TM-XTAG**), we use a tree model which has been trained on a corpus annotated with traditional TAG derivation trees (using β-trees rather than γ-trees). Except in very rare cases, this entirely determines linear order. So in this experiment we use a tree model and the XTAG grammar, but no linear language model.[1]

- In the fourth experiment (**TM-XTAG-LM**), we use the system as described in Section 3. Specifically, we employ the supertag-based tree model whose parameters consist of whether a lexeme l_d with supertag s_d is a dependent of l_m with supertag s_m. Furthermore we use the supertag information provided by the XTAG grammar to order the dependents, but using γ-trees rather than β-trees. This model generates *There was no cost estimate for the second phase* . for our example input, which is indeed the sentence found in the WSJ.

The test corpus is a randomly chosen subset of 100 sentences from the Section 20 of WSJ. The dependency structures for the test sentences were obtained automatically from converting the Penn TreeBank phrase structure trees, in the same way as was done to create the training corpus. The average length of the test sentences is 16.7 words with a longest sentence being 24 words in length.

As in the case of machine translation, evaluation in generation is a complex issue. We use a metric suggested in the MT literature (Alshawi et al.1998) based on string edit distance between

[1] In fact, we use the linear language model in those rare cases when a β trees can be adjoined in more than one position.

the output of the generation system and the reference corpus string from the WSJ. This metric, *generation accuracy*, allows us to evaluate without human intervention, automatically and objectively. Clearly, the metric does not provide a complete assessment of the quality of a generator since often there is more than one "good" result, but we assume that the requirement is to model the corpus as closely as possible (as is the case in some, but not all, applications). We have also independently verified the metric by asking human subjects for subjective judgments; the judgments show significant correlation with the metrics (Bangalore and Rambow2000a).
Generation accuracy, shown in Equation (1), is the number of insertion (I), deletion (D) and substitutions (S) errors between the target language strings in the test corpus and the strings produced by the generation model except that it treats deletion of a token at one location in the string and the insertion of the same token at another location in the string as one single movement error (M). This is in addition to the remaining insertions (I') and deletions (D').

$$GenerationAccuracy = (1 - \frac{M + I' + D' + S}{R}) \quad (1)$$

The average generation accuracy for the four experiments are tabulated in Table 1. As can be seen, the use of a tree model improves results over the baseline, but the use of a linear model also improves results if the XTAG grammar is used: the best results are obtained when the tree model, the XTAG grammar, and the linear model are used.

Tree Model	Generation Accuracy
Baseline	56.2%
TM-LM	66.8%
TM-XTAG	68.4%
TM-XTAG-LM	72.4%

Table 1: Performance results from the three tree models.

5. Featurization of Supertags

5.1. Features

As pointed out by (Candito1996) and (Xia et al.1998), a supertag is a composite representation of a few orthogonal linguistic dimensions such as the subcategorization (argument list) of the head (Subcat) and the way in which specific arguments are realized syntactically (Transformation). These dimensions can be represented as features that can potentially be assigned independently of one another. A featurized representation of supertags helps in a more fine-grained error analysis and may allow for better stochastic supertag assignment models. In this section, we will describe our attempt to represent supertags as features and some preliminary results of error analysis using featurized supertags.
Table 2 shows the list of features and their values used in representing the supertags. (The Modifiee features only are used if ADJ is T.) Although the set of features are directly based on those proposed in (Candito1996) and (Xia et al.1998), we have made a few additions, most notably, FRR2, SGP1 and SGP2. While FRR (for "Function Reassignment Rule") is used to represent changes in the valency of a supertag, FRR2 is used to represent the linear order variations of arguments in the supertag such as dative shift and particle shift. Note that FRR, FRR2, and Transformation are all orthogonal to each other. SGP features are used to represent strongly governed prepositions for supertags that use a preposition in the realization of an argument

Features		Possible Values
POS		10 different part-of-speech tags
Subcat		Different argument frames (eg. NP, NP_NP, NP_S ...)
Transformation	Type	Declarative, WH, Relative, Resumptive_Relative, Gerund, Imperative, Inversion
	Argument	NIL,0,1,2
FRR	Type	NIL,Ergative,Equative,Passive,Passive_by,Predicative
	Argument	NIL,1,2
Modifiee	Type	NIL,NP,S,VP,N,Ad,PP,A,D,AP,P,DetP,V
	Direction	NIL,left,right
FRR2		DativeShift, ParticleShift1,ParticleShift2
SGP1		Strongly Governed prepositions for objects
SGP2		Strongly Governed prepositions for indirect objects
ADJ		Flag to indicate adjunct status

Table 2: The set of features and their values used to represent the supertags.

(SGP1 for the direct object, SGP2 for the indirect object). The value of this feature is derived from PP complements present in the corpus.

5.2. Error Analysis

We replaced the supertags in the **TM-XTAG-LM** model with the featurized representation treated as a single string. Since the featurized representation is just a notational variant for supertags, we got the same performance figures. However, the feature representation allows us to analyze the errors with respect to each of the features. We see that the most error occur in the features ADJ, SUBCAT, and POS (with about equal frequency). Errors also occur in TRANS and FRR, but much less frequently, and even less frequently in the other features. A sample of the individual errors with frequency is shown in Table 3.

Correct	FERGUS Assigned	Number
ADJ=NIL	ADJ=T	134
ADJ=T	ADJ=NIL	49
SUBCAT=NP	SUBCAT=NIL	46
SUBCAT=NIL	SUBCAT=NP	30
POS=N	POS=D	16
TRANSARG=NIL	TRANSARG=0	16
POS=V	POS=N	14
POS=G	POS=NIL	14
FRR=NIL	FRR=Predicative	14
TRANS=decl	TRANS=REL	13

Table 3: List of most frequent individual errors by features

We are working on developing models that better predict each of the individual features using modeling techniques from the Machine Learning community such as Bayesian Nets. The use of "featurized" supertags also has the advantage that they allow us to use FERGUS even when the TAG grammar is much less complete than the English XTAG grammar.

6. Future Work

FERGUS as presented in this paper is not ready to be used as a module in applications. Specifically, we will add a morphological component, a component that handles function words (auxiliaries, determiners), and a component that handles punctuation. In all three cases, we will provide both knowledge-based and stochastic components, with the aim of comparing their behaviors, and using one type as a back-up for the other type.

References

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

Srinivas Bangalore and Aravind Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2).

Srinivas Bangalore and Owen Rambow. 2000a. Evaluation metrics for generation. In *Proceedings of the First International Natural Language Generation Conference (INLG2000)*, Mitzpe Ramon, Israel.

Srinivas Bangalore and Owen Rambow. 2000b. Exploiting a probabilistic hierarchical model for generation. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)*, Saarbrücken, Germany.

Tilman Becker. 1998. Fully lexicalized head-driven syntactic generation. In *Proceedings of the 8th International Workshop on Natural Language Generation*, Niagara-on-the-Lake, Ontario.

Marie-Helene Candito. 1996. A Principle-Based Hierarchical Representation of LTAGs. In *Proceedings of COLING-96*, Copenhagen, Denmark.

Irene Langkilde and Kevin Knight. 1998a. Generation that exploits corpus-based statistical knowledge. In *36th Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL'98)*, pages 704–710, Montréal, Canada.

Irene Langkilde and Kevin Knight. 1998b. The practical value of n-grams in generation. In *Proceedings of the Ninth International Natural Language Generation Workshop (INLG'98)*, Niagara-on-the-Lake, Ontario.

David D. McDonald and James D. Pustejovsky. 1985. Tags as a grammatical formalism for generation. In *23rd Meeting of the Association for Computational Linguistics (ACL'85)*, pages 94–103, Chicago, IL.

Owen Rambow and Aravind Joshi. 1996. A formal look at dependency grammars and phrase-structure grammars, with special consideration of word-order phenomena. In Leo Wanner, editor, *Current Issues in Meaning-Text Theory*. Pinter, London.

Owen Rambow, K. Vijay-Shanker, and David Weir. 1995. D-Tree Grammars. In *33rd Meeting of the Association for Computational Linguistics (ACL'95)*, pages 151–158. ACL.

Matthew Stone and Christine Doran. 1997. Sentence planning as description using tree adjoining grammar. In *35th Meeting of the Association for Computational Linguistics (ACL'97)*, pages 198–205, Madrid, Spain.

Fei Xia, Martha Palmer, K.Vijay-Shanker, and Joseph Rosenzweig. 1998. Consistent grammar development using partial-tree descriptions for lexicalized tree-adjoining grammars. In *Proceedings of the Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+4)*, number 98–12 in IRCS Report. Institute for Research in Cognitive Science, University of Pennsylvania.

The XTAG-Group. 1999. A lexicalized Tree Adjoining Grammar for English. Technical Report `http://www.cis.upenn.edu/~xtag/tech-report/tech-report.html`, The Institute for Research in Cognitive Science, University of Pennsylvania.

Lexik : a maintenance tool for FTAG

Nicolas Barrier, Sébastien Barrier, Alexandra Kinyon

TALaNa – LaTTice
UFRL, University Paris 7
2, pl. Jussieu F-75251 Paris Cedex 05
{nbarrier, sbarrier, kinyon}@linguist.jussieu.fr

Abstract

In this paper we present LEXIK, a tool which allows to maintain and gather data on wide coverage grammars based on the XTAG format. We present the tool, show how it is used within the FTAG project (Abeillé & al. 2000a), and compare it to similar work done on the Xtag grammar for English (Sarkar & Wintner 99).

Introduction

Over the past ten years, FTAG, a wide coverage LTAG has been developed at Talana, building up on the work of (Abeillé, 91). Thanks to the MetaGrammar developed by (Candito 96,99), which allows to generate semi-automatically an LTAG, the number of trees has augmented drastically: from 650 trees for the manually written grammar, we have now reached 5000 elementary trees (cf. Abeillé & al 99,00). Although this has improved the coverage of the grammar, new maintenance issues have appeared :
To remedy this problem, we have developed a tool which we call Lexik. The goal is twofold :

- Insuring consistency in the grammar
- Easily extracting information on a large scale

In the first part of this paper, we review the main characteristic of FTAG and present the problems encountered for maintaining and updating the Grammar. In a second part, we present our tool, as well as its utility. Especially, we compare it to the work presented in (Sarkar & Wintner 99). Finally, we show how this tool is used in other projects.

1. Main characteristics of FTAG

We assume some familiarity with the LTAG formalism. We recall that elementary units of an LTAG are lexicalized constituent trees, which encode all the surface constructions available for a given language. Within FTAG, elementary trees respect the following linguistic well-formedness principles: (Kroch and Joshi 1985, Abeillé 1991, Frank 1992) :

- Strict Lexicalization : all elementary trees are anchored by at least one lexical element, the empty string cannot anchor a tree by itself,
- Surfacism: an elementary tree encodes all word order variations, all basic syntactic phenomena (passive, extraction...) and crossing of phenomena.
- Semantic Consistency : no elementary tree is semantically void (this ensures the compositionality of the syntactic analysis),
- Semantic Minimality : elementary trees correspond to no more than one semantic unit
- Predicate Argument Cooccurrence Principle : the elementary tree is the minimal syntactic structure that includes a leaf node for each realized semantic argument of the anchor(s).

Semantic minimality and consistency imply that function words appear as co-anchors (cf. Figure 1, the relevant syntactic and semantic units are donner-à (give to) and penser-que (think that)).
The elementary trees are combined by substitution or adjunction, and the features of nodes in contact must unify. They thus directly represent all the syntactic rules of the language.

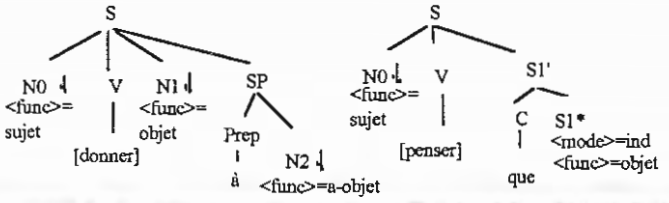


FIGURE 1 : Elementary trees with functional coanchors

1.1. Factorization of Lexicalized Elementary trees

Strict lexicalization at execution time does not prevent from internally compacting the common parts of the elementary trees. This compacting is required for any reasonably sized grammar, since for instance a verbal form may anchor dozens or hundreds of elementary trees. In practice, lexicalized elementary trees are compiled out of three sources of information:

- a set of tree sketches ("pre-lexicalized" structures, whose lexical anchor(s) is not instantiated)
- a syntactic lexicon, where each lemma is associated with the relevant tree sketches
- a morphological lexicon, where inflected forms point to a lemma associated to morphological features

Lexical selection of tree sketches is controlled by features from the syntactic and morphological lexicons, and uses the notion of tree families, grouping sets of tree sketches that share the same (initial) subcategorization frame. The tree sketches of a family show all possible surface realizations of arguments (pronominal clitic realization, extraction, inversion...) as well as all possible transitivity alternations (impersonal, passive, middle..).

A lemma selects one or several families (corresponding to one or several initial subcat frames) and with the help of features selects exactly the relevant tree sketches.

Figure 3 shows the canonical elementary tree anchored by *parlait* (talked)[1] and Figure 2 the three sources of information associated with its internal representation.

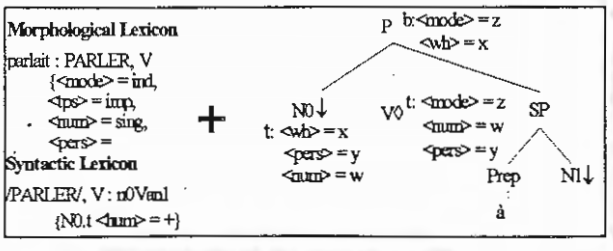


FIGURE 2 : 3 sources of information

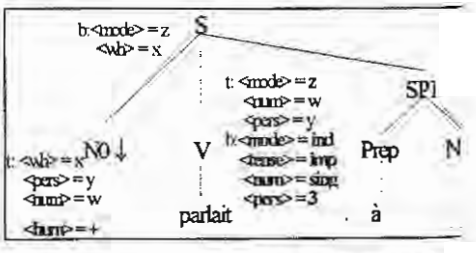


FIGURE 3 : Corresponding elementary trees

[1] Information coming from the lexicon appears in bold characters.

The inflected form *parlait* points to the lemma PARLER, and the lexeme /PARLER/ selects in turn the n0Vanl family, where the preposition appears as a co-anchor (except when argument 1 is cliticized).

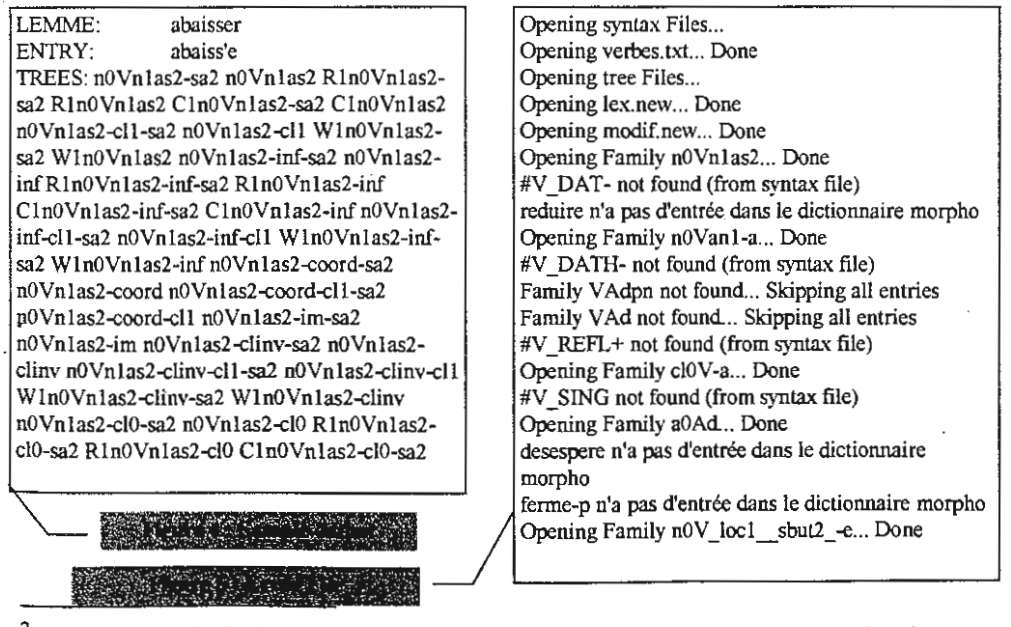
Currently, our morphological lexicon comprises 50000 inflected for ms, our syntactic lexicon has more than 6000 entries, and the bulk of the grammar consist in 5280 tree sketches. Concretely, each family is a file were a set of trees is stored.

Maintaining and updating such a large database is difficult : for example, one can generate a large grammar using Candito's tool, but integrate it with manually written tree sketches for idioms (since trees for idioms are not automatically generated). Then one needs to make sure that the features used in those 2 parts of the grammar are identical. Also, while the automatic generation of the grammar insures consistency (i.e. all features are generated automatically from a hand written hierarchy), errors may still propagate in the grammar, but on a larger scale : if a feature has a typo in the hand written hierarchy (ex : aggreement instead of agreement), then this error will be propagated in hundreds of trees when the grammar is generated (with dramatic effects if it remains undetected). Also, consistency between the grammar and the lexicons is an important issue : for example one would like to detect lexical items which refer to trees that do not exist in the grammar (either because of an error or of an update).

Also, we just said that that a verb can anchor several dozens of trees, but we would like to have a more precise measure of this, and be able to answer questions such as : how many trees does verb X anchor ? How many trees on average are anchored for French verbs ?
This is were Lexik comes in.

2. Lexik : presentation of the tool

Lexik allows to lexicalize tree sketches, that is it takes the morphological lexicon, the syntactic lexicon and the tree sketches as input (e.g. figure 2) and outputs on the one hand fully lexicalized trees (figure 3) anchored by each inflected form[2], and on the other hand, if necessary, an error file. A sample output file can be seen on figure 4, a sample error file can be seen on figure 5.



[2] This is done at runtime by the Xtag parser, but in an opaque manner, which prevents error detection and repair

2.1. Consistency issues

The error file outputed by Lexik allows to detect 4 types of errors :

1. Inconsistencies between the morphological and syntactic lexicons (i.e. lemma with no corresponding inflected forms and vice-versa)
2. Organization problems in the grammar (e.g. missing trees or families)
3. Feature problems (e.g. unknown features)

A simple script allows to extract the most common (and hence damaging) errors, which can then be repaired (cf figure 5)

This work on consistency can be compared with that of (Sarkar and Wintner 99), who validate the consistency of feature structures by imposing type discipline. Contrary to us, their approach focuses on features to detect the 4 following kinds of problems :

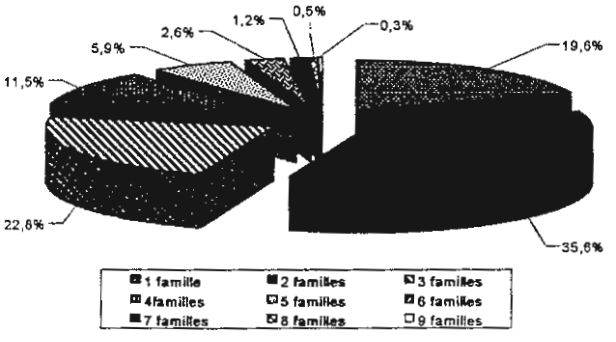
1- ambiguous features (e.g. gen : genitive or gender ?)
2- typos : relpro instead of rel-pro
3- Undocumented features (i.e. used in previous versions of the grammar)
4- type errors : e.g. assign-case is relevant only for verbs, not for nouns

Their tool runs on a wide-coverage LTAG for English (cf Xtag group 95), while ours runs on FTAG for French (cf Abeille & al 99). Since the 2 grammars resort to similar formats, it would be interesting to couple the 2 approaches.

2.2. Gathering information

In addition of detecting errors in the grammar, Lexik allows to gather information that was unavailable previously.

FIGURE 6 : Repartition of verbal lemmas by number of families anchored



Up to now, we could only gather data at the level of families. This allowed to know for instance that the two tree families n0Vn1 (transitive) and n0Vn1-à-n2 (ditransitive) are anchored by two thirds of lemmas (cf NBarrier 99). To have a clearer idea, we extracted 1060 inflected forms of verbs from the 1 million word corpus LeMonde (cf Abeillé & Clément 99) and found that verbs anchor on average 2.8 families / verb (Figure 6), whereas other parts of speech (i.e. nouns, adverbs, adjectives) only anchor between 1 & 2 trees. Only 7 of these verbs anchor 8 families or more[3] (cf SBarrier 99) and only 2 out of these 7 verbs are among the most 100 frequent ones (être *(be)* most often used as an auxiliary, and parler *(talk)*). Intuitively, one could expect that verbs anchoring the more families will also be anchoring the

[3] These verbs are : amuser (amuse), être (be), parler(talk), répendre(spill), revenir (come back), heurter (bump into), dresser (put up)

more trees, and conversely that verbs anchoring the more trees will be verbs anchoring the more families, despite the fact that some verbs anchor only some of the trees contained in a family[4].

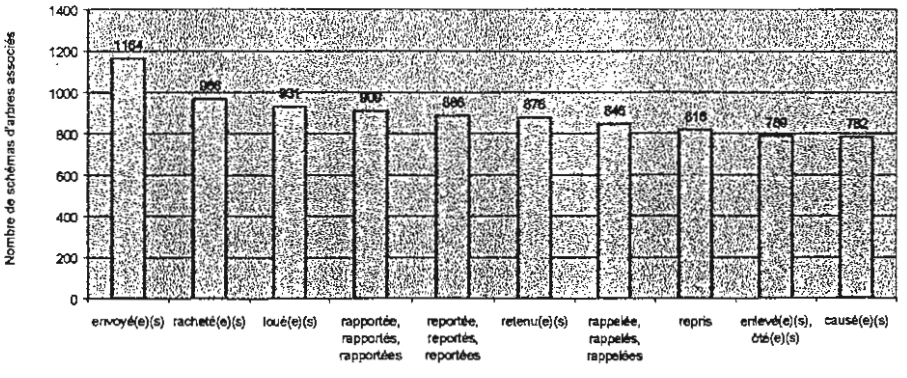
But by going down to the level of trees, Lexik allows to show that this is not the case : it turns out that the inflected form anchoring the more trees (1164) is "envoyés" (past participle for the verb "envoyer"/send) whereas it selects only 3 families. More generally, we have reached the conclusion that the number of families anchored by a given lexical item does not indicate how many trees this item will anchor. Figure 7 illustrates this phenomenon for a few common verbs. We also found that the morphological properties of the item (e.g. past-participles ...) are actually important to predict how many trees an item can anchor.

Lemme	Nombre de familles associées	Forme fléchie retenue	Nombre de schémas d'arbres associés
Amuser	9	Amusés	112
Parler	9	Parlés	333
Répondre	8	Répondus	595
Revenir	8	Revenus	210
Rendre	7	Rendus	452
Parier	6	Parier	93
Louer	5	Loués	931
Envoyer	3	Envoyés	1164
Visiter	1	Visités	78

FIGURE 7 : families and trees anchored by a few common inflected forms

On average, each of the 1060 inflected verbs from LeMonde anchors 139.17 tree schemata (ranging from 1 to 1164). Figure 8 shows the inflected forms which anchor the most trees. It is noticable that all the examples on Figure 8 are past-participles: for exemple for "envoyer" the past-participle anchors 1164 trees, but other inflected forms of this verb (e.g. "envoyons" : Present 1st person plural) anchors only 596 trees. Similarly, if we examine the 2nd most ambigous form (racheté(es) / rebuy), it anchors 966 trees. But "rachetez", which is the 2nd person plural for the same verb in the present, anchors only 498 trees.

FIGURE 8 : Number of trees anchored by the most ambiguous inflected forms



[4] E.g. couter (cost) is a transitive verb which does not passivize, hence it will select all elementary trees in the transitive family, excluding trees for passive.

We also ran Lexik on partial data : we used the same 1060 inflected verbs but kept in the grammar only one tree family n0vn1 for transitive verbs. This family consists in 78 trees. We than ranked the 1060 forms by the number of trees they anchor. It turned out that classes of items bearing morphological similarities appeared : past-participles were at the top of the list (anchoring all 78 trees), followed by infinitivals (anchoring approximately 46 of these trees) and by past participles (anchoring roughly 12 of these trees).

Conclusion

We have presented Lexik : a tool which allows to detect inconsistencies in a wide coverage LTAG for French, and which allows to extract information on a large scale.

It is a first step towards online disambiguation, similarly to what was done for English in (Srinivas & al 94), by allowing to refine a first-pass strategy during parsing (cf Kinyon 99a), and by coupling it with a parse-ranker for TAGs (cf Kinyon 99b,c)

Also, Lexik is being extended to serve as a front end to a function annotation tool, in order to create a large treebank for French (cf. Abeillé & al 00b).

It is also used as the front end of a rule-based supertagger for French, and to collect data in order to build a psycholinguistically relevant processing model for TAGs (cf Kinyon 99d,00)

References

Abeillé A., 1991. Une grammaire lexicalisée d'arbres adjoints pour le français. PhD Thesis, University Paris 7.

Abeillé A. Clément L., 1999. A reference tagged corpus for French. Proc. LINC99, EACL. Bergen.

Abeillé A., Candito M.H., Kinyon A. 2000a : Current status of FTAG. Proc. TAG+5. Paris.

Abeillé A., Clément L. Kinyon A., 2000b : Building a treebank for French. Proc. LREC'2000. Athens

Abeillé A., Candito M.H., Kinyon A. : FTAG : current status and parsing scheme. Proc. Vextal'99. Venice.

Barrier S. 1999. Classification et repérage des valences verbales en français : experiences avec FTAG. MS thesis. Univ. Paris 7.

Barrier N. 1999. Lexik : un outil de lexicalisation des TAGs. Application à la désambiguation syntaxique du français MS thesis. Univ. Paris 7.

M-H Candito, 1996. A principle-based hierarchical representation of LTAG, Proc. 15th COLING. Kopenhagen.

M.-H. Candito, 1999. Représentation modulaire et paramétrable de grammaires électroniques lexicalisées. Application au français et à l'italien. PhD dissertation. University Paris 7.

M-H. Candito, S. Kahane, 1998. Can the TAG derivation tree represent a semantic graph ? an answer in the light of MTT, Proceedings TAG+4 Workshop, Philadelphia.

Frank R. 1992 Syntactic Locality and Tree Adjoining Grammar : Grammatical Acquisition and Processing Perspectives. PhD dissertation. University of Pennsylvania.

Kinyon A. 1999a : Distinction entre Regard en avant et Première Passe pour l'analyse des LTAGs. Proc. Recital'99. Cargèse.

Kinyon A. 1999b : Parsing preferences and lexicalized Tree Adjoining Grammars : exploiting the derivation tree, Proc. ACL'99. Maryland.

Kinyon A. 1999c : Hiérarchisation d'analyses basée sur des informations dependancielles pour les LTAGs, Proc. TALN'99. Cargèse.

Kinyon A. 1999d : Some remarks about the psycholinguistic relevance of LTAGs. Proc. CLIN'99. Utrecht.

Kinyon A. 2000: Towards a psycholinguistically relevant processing model for LTAGs. Proc. Cogsci'2000. Philadelphia.

Kroch A., Joshi, A. 1985. The linguistic relevance of Tree Adjoining grammars, Technical Report, Univ. of Pennsylvania.

Sarkar A., Wintner S. 1999 : Typing as a means for validating feature structures. Proc. CLIN'99. Utrecht.

Srinivas B., Doran C., Kulick S. 1995 : Heuristics and Parse Ranking. Proc. IWPT'95. Prag. Czech Republic.

Xtag group 1995 A LTAG for English. Technical Report IRCS 95-03. University of Pennsylvania.

Adapting HPSG-to-TAG compilation to wide-coverage grammars

Tilman Becker and Patrice Lopez

DFKI GmbH
Stuhlsatzenhausweg 3
D-66123 Saarbrcken, Germany
{becker,lopez}@dfki.de

Abstract

The HPSG-to-TAG compilation algorithm proposed in (Kasper et al., *1995) has been the basis of large scale experiments in VerbMobil, a speech–to–speech dialouge translation system in the scheduling and travel domain. The results here refer to the English HPSG grammar developed at CSLI. Several non-trivial theoretical problems have been discovered by the practical application of this algorithm. This paper presents these experiments, the main shortcomings of the initial algorithm and some of the solutions we have developed in order to use the resulting compiled LTAG grammar in a real world system.*

1. Introduction

The LTAG formalism is a mathematical tool that has proven to be attractive for the modeling of natural language syntax. In parallel to pure-LTAG grammar developments, some researches have addressed the relation between LTAG and existing formalisms both for theoretical and practical reasons. In particular, compiling a LTAG grammar from a HPSG grammar has been proposed by (Kasper, 1992). Such a compilation is interesting for several reasons:

- **Sharing of resources** between the two formalisms, in particular the syntactic lexicon. For instance, since both formalisms are lexicalized, the syntactic lexicon which gives all possible predicative frames for each lemma is very costly to write.
- **Speed efficiency:** The precompilation process allows to identify substructures of the HPSG grammar that are not context-dependent. The extracted partial backbones can be tabulated (chart parsing, memoization) which results in more time efficient systems than a direct HPSG parser/generator.
- **Capturing dependencies:** An LTAG elementary tree directly encodes a full syntactic context by the way of an extended domain of locality. Elementary trees are combined in order to realize dependency relations between the syntactic contexts they represent. Thus the construction of a sentence can be obtained very easily just with a dependency tree indicating the elementary trees that are involved and their mutual dependencies. This information is represented only indirectly in an HPSG derivation.
- **Exploiting HPSG's** expressivity as well as utilizing existing HPSG grammars is interesting for the LTAG community. HPSG grammars usually include the syntax-semantics interface and a semantic level that is ignored in existing LTAG grammars. HPSG grammars also define explicitely all dependency relations (Pollard & Sag, 1994) while LTAG

grammars are limited by a tree structure which is problematic for, e.g. coordination and equi-verbs. Finally, there is a large amount of linguistic research which done in the HPSG framework.

Moreover, studying how such a compilation can be performed is an opportunity to identify the assets and the limits of the LTAG formalism. Which relations given in a HPSG grammar should be localized in the LTAG elementary trees in order to obtain a grammar that is either linguistically meaningful or computationally efficient?
We first recapture the basic principles of the compilation algorithm as described in (Kasper *et al.*, 1995). Then we present the various problems and limits of this initial algorithm and the adaptations that have been necessary for the practical HPSG-to-TAG compilation of a wide-coverage grammar.

2. The initial compilation algorithm

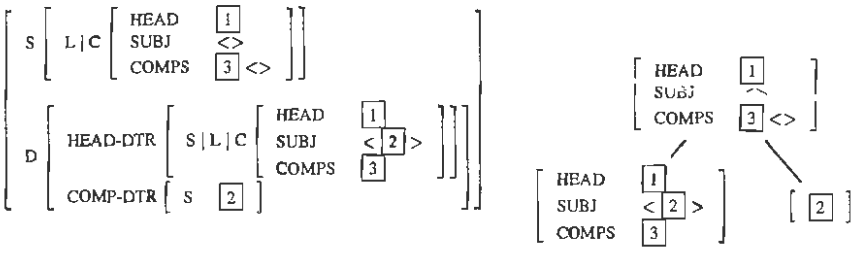


Figure 1: HPSG Head-Subj Schema and its representation as a local tree.

We assume that the rule schemata in the HPSG grammar only correspond to binary or unary rules. For instance, the *Head-Subj-Schema* given in figure 1 can be represented by a partial tree. The algorithm presented in (Kasper *et al.*, 1995) is based on the following mechanisms:

- **Selection/Reduction process:** The features which constrain a possible argument are called *Selection Features* (SF). Given a binary schema S, if some SF are expressed in S, we say that the daughter which contains these features is the *Selection Daughter* (SD), the other one is the non-SD. The single daughter of unary rules is the SD. Given the SF of the SD, we say that a schema reduces the SF, if the value of at least one of the features that select the non-SD for this schema is not contained in the feature value of the mother node. In the example figure 1, we see that the SF of attribute SUBJ is reduced.

- **Tree production iteration:** The basic algorithm starts with the creation of a node for the lexical type. A root node n is first added to this initial node with a copy of all its features. Then we instantiate each schema S which actually reduces at least one SF of n when n is unified as the SD of S. Finally, we add an additional root node dominating the instantiated schema. This step is repeated until the *termination condition* is met (see below).

- **Raising Features Across Domination Links:** this principle determines which features are raised (copied) into the additional root nodes. In the first phase of the algorithm, all and only the SF are raised. In the additional phases, some SF are not raised (see below).

- **Detecting foot nodes:** A tree is an auxiliary tree if the root node and one of the leaf nodes (non-anchor) have some non-empty SF value in common. This leaf node becomes the foot node of the auxiliary tree.

- **Termination:** A SF is not reduced anymore if its value is an empty list or it shares its value with a feature at a leaf node other than the foot node.

- **Additional phases:** Systematically raising all possible SF across domination links (i.e., considering only complete projections) results in redundant projections for multiple dependency structures as raising verbs or equi-verbs and consequently corresponding trees that can not be combined. In order to avoid these redundant projections, (Kasper *et al.*, 1995) propose additional phases in order to create new trees without redundant projections for double dependencies. Their decision is to keep the redundant dependencies in the auxiliary trees and consequently re-compile all initial trees, ignoring the SF which are responsible for the redundancy.

At the end of the process, the SF can be deleted from the resulting trees since they express constraints that have been captured in the tree structure. The next section will show that this initial algorithm raises both practical and theoretical problems.

3. Algorithmical problems

3.1. Choice of Selection Features

A given phrase structure (derived tree) can be obtained with different LTAG grammars, where the derivation trees might differ. A lot of choices in the compilation process (SF & SD) depend on the kind of derivation tree we want to obtain and its role given a particular task (generation or parsing). Moreover, the algorithm proposed in (Kasper *et al.*, 1995) aims to capture the phrase structure of the HPSG grammar in the LTAG structure, which is only one choice among other possibilities.

However, even the original algorithm leaves open the choice of SF. This choice, together with the termination criteria influences the resulting elementary trees (and thus the dependency structures) while the derived trees are still isomorphic to the HPSG derivation. In theory, the SF must be chosen such that at least one of them is reduced in every HPSG schema. In practice no such set of SF can be determined and some schemata must be applied in the compilation algorithm with less strict criteria such as a mere change (without reduction) of SF or even (non-recursive) applications with both daughters as possible SD.

The interface between deep syntax and derivation trees highly depends on the LTAG grammar resulting from the compilation algorithm and thus from the choice of SF and termination criteria. Since HPSG is based on lexical projections as expressed, e.g., in the *head feature principle*, there is a certain straightforward choice of SF. However, the HPSG schemata localize syntactic dependencies and not semantic dependencies as classically in LTAG grammars. Especially in generation, when mapping from semantic dependencies, this mismatch becomes apparent, e.g. in auxiliaries (see section 5), raising and equi-verbs, modifier extraction, etc.

As an example for how the choice of selector feature can change the selector daughter for an HPSG schema and thus the resulting elementary TAG trees, we look at the Head-Specifier schema in the HPSG grammar we use. Figure 2 shows how choosing either SPR or SPEC as a SF results in an auxiliary tree anchored at DET (β_1) or an initial tree anchored at N (β_2) respectively. The surprisingly different structures are possible due to a case of double dependencies, where the SPR and SPEC features mutually constrain each other.

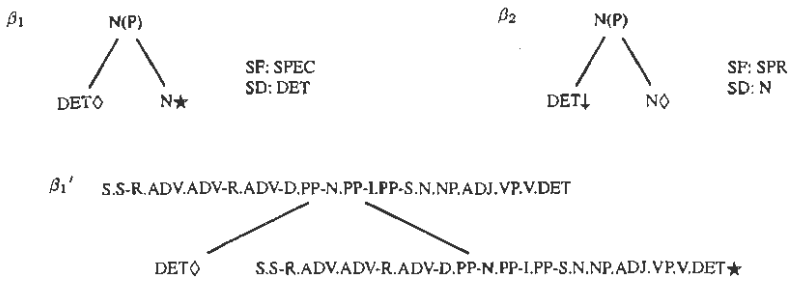


Figure 2: Possible projections for the HPSG Head-Specifier Schema.

Whenever the selector daughter is *not* the head daughter, the property of HPSG that it is *head driven* becomes important: because almost all information (i.e., features) that is raised comes from the head (the non-selector daughter in this case), the root node is very much underspecified in these cases. Thus, e.g. the category of root and foot node is highly underspecified. See the example β_1' in figure 2 (the dot in the node labels indicates a disjunction of categories).
Note however, that in the case of the Adjunct-Head schema, where the selector daughter is clearly not the head daughter, the MOD feature supplies a lot of constraints about the head. Thus the MOD feature is used as a selector feature. However, the HPSG grammar often encodes constraints in the semantics of the MOD feature which does not immediately constrain the category.

3.2. *Adequacy of HPSG and LTAG categories*

The HPSG grammar does not define statically the syntactic categories of nodes as in the elementary trees of LTAG grammars. The original algorithm assumes that all possible values of the SF appear during the compilation and thus can be mapped (collapsed) to a finite set of not obviously meaningful categories. In the HPSG framework, syntactic categories are usually computed only for complete derived trees and they are represented as (disjoint) underspecified feature structures (typically with values only for the SF) such that in a derived tree only one syntactic category (feature structure) unifies with a node. Using these HPSG categories, one gets a set of meaningful categories.
But since elementary trees resulting from the compilation algorithm correspond to *partial* parsing trees obtained by the application of several HPSG schema, it is not possible to determine a unique syntactic category per node in the compiled LTAG elementary trees. As a consequence, nodes of elementary trees must often be labeled with a disjunction of syntactic categories. Duplicate trees in order to avoid such disjunctions would result in a critical explosion of the number of trees. Note that we can observe in our resulting compiled grammar disjunction of more than twenty syntactic categories.

3.3. *Raising of non-SF features*

Following (Kasper *et al.*, 1995), only SFs are raised across dominance links. In practice, non-SFs features are very important for the selection and the filtering of the HPSG schema that are applicable in production of an elementary tree. Without raising them across dominance link, too many HPSG schema would be applied, resulting in an dramatically overgenerating and much larger grammar. Naturally, raising non-SF features can result in an *undergenerating* grammar. But in generation, this is often less of a problem than overgeneration. Also, by extending (relaxing) the termination criteria, we can ensure the generation of all necessary elementary

trees.

3.4. Anchoring the Projection

In HPSG as in LTAG, there is a separation between the grammar and the lexicon such that a lexical entry specifies the word, its semantics and a *lexical type* or *tree family*. If this separation is clean in the HPSG grammar, as it is in our case, the compilation process can start from the lexical types and becomes independent from the lexicon.
However, many lexical types only differ wrt. semantics and the compilation process only extracts the syntactic part, so either (i) for a set of lexical types that generate the same tree-families, we must determine the most specific subsuming type in the type hierarchy or (ii) eliminate redundancies in a post-process.
Note that since the HPSG grammar has the option to locate constraints either in syntax or semantics, some of the syntactic features are highly underspecified. This leads to the above-mentioned redundancies between the syntax of lexical-types and also to the underspecifications in the syntactic categories.
We also have found another source of redundancy: a sizable number of trees appear in more than one tree-family and a further reduction could be achieved by storing them only once an introduce pointers to the tree-families.
Note that this kind of underspecification seems very undesirable but it is inherent in the specifications of the HPSG grammar and therefore cannot be avoided easily. The only principled solution is a change of the HPSG grammar.

4. Specific linguistic phenomena

4.1. Coordination

Coordination is problematic in the LTAG formalism since the tree structures are not able to localize the multiple dependencies that this phenomena introduces. The HPSG analysis of this phenomena exploits at the syntactic level the type hierarchy of features, particularly by introducing new morpho-syntactic ones. Feature coindexing at the level of the semantics are also used for crossed-dependencies for instance. These techniques are really far from the existing ones in the LTAG-world based on explicit structural dominance link (Sarkar & Joshi, 1996). The processing solutions of this phenomena in the two respective formalisms are too specific to expect the capture of the HPSG approach by TAG.

4.2. Double Dependencies

The double dependencies, where a given phrase structure is the argument of two different predicates, are a problem in the LTAG formalism which can only capture one of these dependencies in the structure of elementary tree. For equi-verbs for instance, as the verb *want*, the classical choice corresponds to the elementary trees given in figure 3. These trees are obtained by the initial HPSG-to-TAG compilation algorithm (β_1 and α_1) but the additional phases generate also some other trees ignoring some SF, i.e. some predicate-argument relations (β_2 and α_2). Considering that the elementary trees generated for the main verbs localize all possible set of predicate-argument relations (see figure 3), we obtain redundant projections of substitution nodes. One can see that we obtain the same derived tree by combining β_1 and α_2 or β_2 and α_1, but in both cases only a part of the dependencies are captured.
Consequently we can question the relevance of the multiple phase part of the algorithm. Fully executing the additional phases as described in the original algorithm is impractical since it generates far too many trees. Therefore, we have added by hand those extensions of the termination and raising criteria that are needed to obtain the elementary trees needed in our domain.

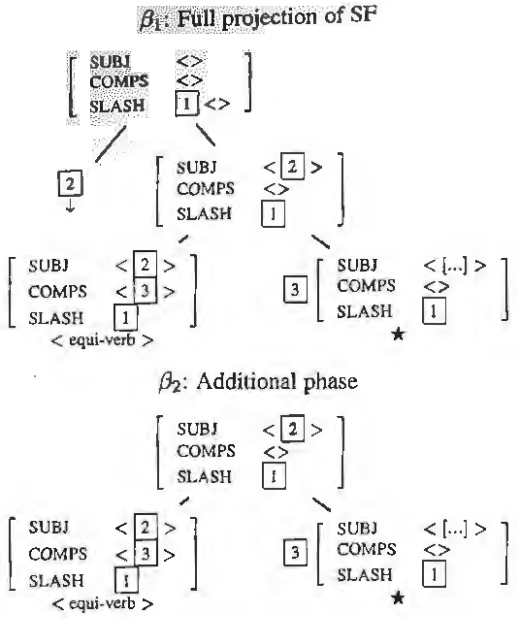


Figure 3: Elemenary trees for equi-verbs.

In general, the construction of elementary trees stops when SF are reduced to empty features or lists. In practice however, SF often are never empty and only a detailed analysis of the content (e.g., the type) of the SF can determine whether the projection must stop, i.e., that the SF can/must not be reduced further. Also, in order to mimic the effect of the additional phases (see paragraph below), we have to relax the termination criteria to apply even when some SF are not reduced. E.g., the projection of auxiliaries results in VP substitution nodes, thus adding VP nodes (with a reduced COMP feature and an unreduced SUBJ feature) to the list of terminating nodes.

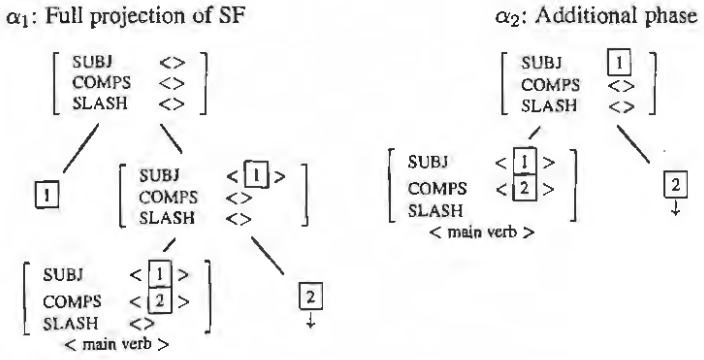


Figure 4: Elemenary trees for main verbs.

4.3. Idioms

LTAG can represent idioms directly, including the fact that idioms have a single (non-compositional) semantic representation. Since HPSG grammars cannot do this directly, this is also true for the compiled TAG grammar. Even multiple anchors (like in particle verbs and the (semantically empty) prepositions of prepositional arguments) are not incorporated into the resulting trees of the compilation algorithm. One solution would be the extension of the compilation algorithm to expand those leaf nodes (e.g., prepositional complements) that include semantically empty, syntactic arguments. I.e., instead of a PP substitution node, expand it to its anchored P daughter and its NP substitution daughter. Since this would add even more trees, we have instead chosen to include these expansions either in the microplanning or the preprocessing (see section below) phases.

4.4. Futher issues

The LTAG formalism can not capture all long dependencies that can be represented easily in HPSG with feature percolation. One way to capture these phenomena is to compile the HPSG grammar into an extension of the LTAG formalism as the DTG formalism (Rambow *et al.*, 1995).

In the current version of the algorithm, semantics is not taken into account. We have conducted some experiments by compiling the semantic level in a specific LTAG grammar that could be synchronized to the classical compiled LTAG grammar. The interest of this approach highly depends on the compositionality of the resulting semantic grammar which still needs some futher investigations.[1]

One can also note that some linguistic constraints are not represented in the usual HPSG grammars, such as modifier, e.g. adjective, ordering and topic/focus distinctions.

5. Interface to the non-syntactic level of HPSG

As discussed above, the dependency structure of the resulting TAG grammar depends mainly on the dependencies that are specified in the HPSG grammar and the choice of selector features only has a limited effect. This is especially important when generating with the resulting TAG grammar. Typically, the input to a syntactic realizer that is based on TAG will be a dependency structure that can be interpreted as the derivation structure. As an example from our system, the HPSG grammar specifies auxiliaries as the lexical heads of sentences, taking a subject and a VP as arguments. So the microplanning step in the generator that maps from the semantic input to the syntactic dependencies must not only plan word choice and map the semantic roles to syntactic arguments (e.g., the *giver* to a *subject*), it also must be prepared to insert an auxiliary (e.g., *have*) and rearrange syntactic arguments (e.g., ensure that the *giver* becomes the subject of *have* and not *give* in *We have given..*). In order to keep a more general interface between microplanner and syntactic realizer, we have chosen not to include the auxiliaries in the microplanner but rather add a preprocessing module to the syntactic realizer which adapts the dependency structure to the specifics of the HPSG/TAG grammar. Thus we can switch to other syntactic realizers (based on other TAG grammars) more easily.

This touches on a more general point which is not really discussed in the original work: The interface between the extracted subgrammar and the full HPSG grammar. As proposed, the compiled TAG grammar actually overgenerates since it represents all possible phrase structures

[1] Since the extraction of just a subset of the features of the HPSG grammar amounts to an (overgenerating) approximation, it is very important to include as many of the constraining features as possible. Many of them are entangled into the semantics though, so a clearer separation in the HPSG grammar is needed. See also the work on context–free approximation of HPSG in (Kiefer & Krieger, 2000).

but omits some of the constraints, especially those of semantics. Ideally, the semantics of the HPSG grammar would be purely compositional, thus the compiled TAG language would be identical. However, in practice there are non-compositional elements in the HPSG semantics and it turned out to be impractical to extract the semantics for every compiled elementary tree and use these partial semantic expressions for microplanning.[2] Thus we have developed the microplanning rules only semi-automatically which also allowed the inclusion of a large subset of planning rules that deal robustly with all kinds of problems in our input (Becker *et al.*, 2000).

6. Practical Results and Conclusion

In the context of the generation module of the Verbmobil project (Becker *et al.*, 1998), we have implemented our adaptation of the compilation algorithm in Common Lisp as an addition to the PAGE system which is used to specify and parse the HPSG grammars. We currently cover an English and a Japanese grammar; the English grammar has around 350 lexical types and 40 schemata and a lexicon with around 6,800 entries. The Japanese grammar has a similar size, with a smaller lexicon. Compilation takes about 15 minutes CPU time on a 400MHz Ultrasparc resulting in around 2,500 elementary trees.

We found the adapted compilation process to be useful in a real system, since we could influence the design of the HPSG grammars, which is an important factor. Also, work on a German HPSG grammar is under way. Given the growth in computational power, we hope to be able to explore a complete application of the original algorithm in the near future.

References

BECKER T., FINKLER W., KILGER A. & POLLER P. (1998). An efficient kernel for multilingual generation in speech–to–speech dialogue translation. In *Proceedings of COLING/ACL-98*, Montreal, Quebec, Canada.

BECKER T., KILGER A., LOPEZ P. & POLLER P. (2000). An extended architecture for robust generation. International Natural Language Generation Conference (INLG), Mitzpe Ramon, Israel.

KASPER R. (1992). Compiling Head-Driven Phrase Structure Grammar into Lexicalized Tree Adjoining Grammar. In *Proceedings of the TAG+ workshop 1992*, University of Pennsylvania, Philadelphia.

KASPER R., KIEFER B., NETTER K. & VIJAY-SHANKER K. (1995). Compilation of HPSG to TAG. In *Proceedings of ACL'95*, p. 92–99, Cambridge, Mass.

KIEFER B. & KRIEGER H.-U. (2000). A context–free approximation of head–driven phrase structure grammar. In J. CARROLL, Ed., *Proceedings of the Sixth International Workshop on Parsing Technologies, IWPT2000*, p. 135–146, Trento, Italy.

POLLARD C. & SAG I. (1994). Head-driven phrase structure grammar. In *CSLI series*. University of Chicago Press.

RAMBOW O., SHANKER K. V. & WEIR D. (1995). D-Tree Grammars. In *33rd Conference of the Association of Computational Linguistics (ACL'95)*, p. 151–158.

SARKAR A. & JOSHI A. (1996). Coordination in tree adjoining grammars: Formalization and implementation. In *COLING'96, Copenhagen*, p. 610–615.

[2]To make things worse, the HPSG grammar only includes an intermediate semantic representation (MRS) in its declarative feature structures and the semantic representation that is actually used in our system is derived by procedural code.

Engineering a Wide-Coverage Lexicalized Grammar

J. Carroll, N. Nicolov, O. Shaumyan, M. Smets & D. Weir
School of Cognitive and Computing Sciences
University of Sussex
Brighton, BN1 6RG
UK

Abstract

We discuss a number of practical issues that have arisen in the development of a wide-coverage lexicalized grammar for English. In particular, we consider the way in which the design of the grammar and of its encoding was influenced by issues relating to the size of the grammar.

1. Introduction

Hand-crafting a wide-coverage grammar is a difficult task, requiring consideration of a seemingly endless number of constructions in an attempt to produce a treatment that is as uniform and comprehensive as possible. In this paper we discuss a number of practical issues that have arisen in the development of a wide-coverage lexicalized grammar for English: the LEXSYS grammar. In particular, we consider the way in which the design of the grammar and of its encoding—from the viewpoint both of the grammar writer and of the parsing mechanism—was influenced by issues relating to the size of the grammar.

One criterion that is often used as a judge of grammar quality is the extent to which 'linguistic generalizations' have been captured. Generally speaking, concern over this issue leads to a preference for smaller rather than larger grammars. A second reason for preferring smaller grammar sizes is on the basis of parsing efficiency, since the running time of parsing algorithms generally depends on the size of the grammar.

However, a rather different criterion determining grammar quality has to do with the analyses that the grammar assigns to sentences: in particular, the extent to which they provide a good basis for further, perhaps deeper processing. It is not necessarily the case that this criterion is compatible with the desire to minimize grammar size.

In developing the LEXSYS grammar we have explored the consequences of giving the grammar writer the freedom to write a grammar that maximizes analysis quality without any regard for grammar size. In the next three sections we present detailed statistics for the current LEXSYS grammar that give an indication of what the grammar contains, its current size, and why it has grown to this size.

In order to ease the process of engineering such a large grammar, we have made use of the lexical knowledge representation language DATR (Evans & Gazdar, 1996) to compactly encode the elementary trees (Evans *et al.*, 1995; Smets & Evans, 1998). In Section 5 we present some figures that show how the size of the encoding of the grammar has increased during the grammar development process as the number and complexity of elementary trees has grown.

We have addressed problems that result from trying to parse with such a large grammar by using a technique proposed by (Evans & Weir, 1997) and (Evans & Weir, 1998) in which all the trees that each word can anchor are compactly represented using a collection of finite state automata.

In Section 6 we give some data that shows the extent to which this technique is successful in compacting the grammar.

2. Coverage of the LEXSYS Grammar

The LEXSYS grammar has roughly the same coverage as the Alvey NL Tools grammar (Grover *et al.*, 1993), and adopts the same set of subcategorization frames as in the Alvey lexicon. There are at present 143 families in the grammar. Each family contains the base tree of the family, and definitions of lexical rules which derive trees from the base tree. There are currently 88 lexical rules. Possible rule combinations are determined automatically (see (Smets & Evans, 1998)).

There are 7 **noun** and **pronoun** families. The noun families include trees for bare nouns, for small clauses headed by a noun, for noun-noun modifiers and for coordination. Coordination can be at the N, $\bar{\text{N}}$ or NP levels. There are 19 **adjective** families, distinguished according to the position of the adjective and its subcategorization frames. Trees derived by lexical rules include small clauses headed by an adjective, comparative constructions, trees with unbounded dependencies for adjectives which subcategorize for a complement (***wh***-questions, relative clauses, topicalization), a tree for *tough*-movement, and trees for coordination.

Numerals also anchor adjective trees. Rules derive from the base tree uses of numerals as pronouns and nouns, and coordination of cardinal numbers (for example, *hundred and ten*). However, the grammar does not as yet have a complete account of complex numerals. For ordinals, there are rules to derive fractions with complement, fractions without complement, and the use of ordinals as degree specifiers.

Adverbs are distinguished according to whether they are complements or modifiers. Modifier trees differ according to the modified category and the relative position of the adverb and its argument. Rules derive coordinated structures headed by adverbs, and also adverb distribution. Long distance dependencies possibly involving adverbs (for example, *How did he behave*) are handled in the PP modifier family.[1]

The grammar contains an account of constituent and sentential negation (but in the latter disregarding scope issues arising when an adverb comes in between the auxiliary and the negation).

Specifier families include families for determiners, quantifying pre-determiners and genitive determiners. There is also a family for adjective and adverb specifiers.

Prepositions followed by an NP are divided into two families: a family for case-marking prepositions and a family for predicative prepositions. These two types of prepositions differ in their semantic content, and syntactically also: case-marking prepositions do not head PP-modifiers. The case-marking preposition family includes trees for long-distance dependencies with preposition stranding (*wh*-questions, relative clause, *tough*-constructions) and trees for coordination. The family of predicative prepositions inherits these trees, and also contains trees for adjunct preposition phrases and long-distance dependencies involving adjunct PPs. There are also families for prepositions introducing Ss, VPs, PPs and AP. There are two families for **complementizers** (introducing an S or a VP).

The 94 **verb** families constitute the bulk of the grammar. Verb families include trees[2] for gerunds (nominal and verbal), long-distance dependencies (topicalization, relative clause and *wh*-questions), VP complements, VP complements for *tough*-constructions, small clauses (headed by a present participle or a passive verb), *for-to* clauses, extraposition, imperative, passive with or without *by*, inversion (for auxiliaries and modals), VP-ellipsis (after auxiliaries and modals), dative alternation, movement of particles, and coordination (at V, VP and S).

Finally, we have recently extended the grammar to include semantic features capturing predicate

[1] It would be redundant also to have such a rule in the adverb family.

[2] Of course, these constructions are not relevant for every single family.

argument structures. We have not implemented quantification yet. The grammar adopts a semantic representation inspired by the Minimal Recursion Semantics (MRS) framework (Copestake *et al.*, unpublished). MRS representations are flat lists of elementary predications, with relations between predications being expressed through coindexation.

3. Localization of Syntactic Dependencies

The LEXSYS grammar has been designed to localize syntactic dependencies, not only unbounded dependencies between filler and gap, but agreement relations, case and mode of the clause, etc. (Carroll *et al.*, 1999). One immediate advantage is that there is no need for feature percolation during parsing: all syntactic features are grounded during anchoring. There are, however, a few cases where all syntactic features cannot be localized in the same tree. This happens when the values of syntactic features are determined by more than one constituent.

This is the case, for example, in raising constructions: the subject raising verb agrees with its syntactic subject but the complement of the raising verb (adjective or verb) determines the category of the subject. In such cases, feature percolation is needed, unless one define trees for all the possible feature combinations. This is what we have done in the grammar, and 9 more trees are needed to that effect.

In *there*-constructions, the NP following the verb (*be*) determines the agreement of the verb. This does not represent a problem if the dependency is local. However, if a subject raising verb comes in between *there* and the rest of the sentence, agreement cannot be determined locally anymore. We need one more tree to cover both possible instantiations of agreement features.

Finally, PP phrases can involve a *wh*-NP or a *rel*-NP, thus must be specified as such. Because the head of PPs does not set that feature, feature percolation would be needed between the NP and the root of the PP. In the grammar, we define three PP trees, one for each possible instantiation of that feature. Thus, two more trees are needed than if we had feature percolation.

In all the above cases, the specification of all possible feature combinations does not involve the creation of many more trees. However, from a linguistic point of view, we do miss generalizations.

With coordination, however, the problem is not the loss of linguistic generalizations, but the substantial increase in the number of trees. Indeed, coordination[3] trees are anchored by the head of one of the coordinated constituents. The advantage of this is that constraints on the coordination phrase are defined at anchoring. But the disadvantage is that this doubles the number of trees in the grammar: every structure can occur in coordination.

4. Anchored Trees

The previous two sections discussed the coverage of the grammar, and how some decisions have increased the number of *unanchored* trees. Another important property of the grammar is the number of trees that result from *anchoring* with lexical items.

We find that some verbs induce a very large number of anchored trees: for example, *get* results in 2847 trees, *put* 3465, *come* 2656, and *turn* 1425. To illustrate why, consider *get*. First, *get* has 17 different subcategorization frames (it can be transitive, ditransitive, it can have a prepositional complement, be followed by one or more particles, etc.). It therefore belongs to 17 different families, and each family contains a number of trees (for example, the V_PP family, selected by *get*, has 33 trees, and the V_NP_PPto family contains 146 trees).

Moreover, when a *lexical item* anchors a tree, features get grounded, and different feature instantiations characterize different trees. For example, *get* can be followed by one of 12 different prepositions which means that there are at least 12 × 33 trees for the single subcategorization

[3]Only same constituent coordination has been implemented so far.

# trees in set	# sets	# merged states (mean)	# minimized states (mean)	ratio merged / minimized
1–10	112	17.9	6.9	2.6
11–20	83	53.9	13.1	4.1
21–50	69	133	18.1	7.4
51–100	47	364	28.1	13.0
101–200	68	687	33.0	20.8
201–500	56	1815	42.8	42.4
501–1000	23	3654	48.9	74.7
1001–5000	16	10912	60.1	181.5
Totals	474	927.7	23.5	39.4

Table 1: Grammar compaction statistics

frame V_PP. Similarly, there are 16 different particles which can follow *get*, and this also multiplies the number of trees.

Finally, there are other features that get instantiated and are responsible for the creation of new trees, such as agreement features of the anchor, verb form feature of the anchor and of its verbal complement. Thus the different instantiations of features together with the various subcategorization frames that a word selects explain the very high number of trees anchored by some individual words.

5. Encoding for Grammar Development

Following (Evans *et al.*, 1995) and (Smets & Evans, 1998) the LEXSYS grammar is encoded using DATR, a non-monotonic knowledge representation language.

In 1998, the grammar contained 620 trees organized into 44 tree families and produced using 35 rules. This grammar was encoded in 2200 DATR statements, giving an average of 3.55 DATR statements per tree. The grammar currently contains around 4000 trees in 143 families produced with 88 rules. This grammar is encoded with around 5300[4] DATR statements, giving an average of 1.325 statements per tree. Thus, as the grammar has grown the number of DATR statements needed to encode it has grown, but not as rapidly.

6. Encoding for Parsing

Following (Evans & Weir, 1997) and (Evans & Weir, 1998) each elementary tree is encodec as a finite state automaton that specifies an accepting traversal of the tree from anchor to root For each input word, the set of all the alternative trees that can anchor an input word can b captured in just one such automaton, which can be minimized in the standard way, and the used for parsing.

In order to assess the extent to which this technique alleviates the problem of grammar size we produced automata for the words appearing in the 1426 sentences (mean length 5.70 words forming the Alvey NL Tools grammar development test suite. Each sentence was processe by a morphological analyser, and the result was then used in conjunction with the lexicon t determine for each word in the sentence the complete set of anchored trees, feature valu being determined by the morphological analyser or lexicon as appropriate. 474 distinct sets anchored trees ('tree sets') were produced in this way, ranging in size from 1 to 3465 tree The total number of anchored trees was 24198, with a mean of 175.5 trees in each tree s

[4]We have excluded from this figure around 700 DATR statements that specify the semantics associated w elementary trees.

# occurrences	# sets	# trees in sets (mean)	# minimized states in sets (mean)
1	98	256	25.8
2–5	178	205	26.6
6–10	68	182	23.0
11–20	56	83	19.6
21–50	48	64	16.7
51–100	12	84	21.2
101–200	9	54	11.2
201–500	3	21	13.0
501–1000	2	5	6.0

Table 2: Occurrences of tree sets in test sentences

Before parsing, the trees in each tree set are stripped of their anchor, merged into a single automaton and minimized; at parse time the relevant automaton is retrieved and the appropriate anchoring lexical item inserted. Table 1 shows what happens when the tree sets are converted into automata and minimized, giving figures for the distribution of tree sets, mean numbers of merged and minimized states in each tree set, and ratios of numbers of merged and minimized states.

What is not clear from Table 1 is how often each of the 474 distinct tree sets occurred in the test sentences. This is shown in Table 2 which gives the numbers and mean sizes of tree sets (number of trees and minimized states) relative to the number of times they occurred in the test suite sentences. This shows that the larger tree sets tend to occur less often than small ones, and that very few of those tree sets containing more than 100 trees anchored more than 10 of the more than 8100 word tokens in the test sentences.

The results we have presented in this section appear to show that by encoding the anchoring possibilities for words with minimized automata we are able to alleviate the grammar size problem to a considerable extent.

References

CARROLL J., NICOLOV N., SHAUMYAN O., SMETS M. & WEIR D. (1999). Parsing with an extended domain of locality. In *Proceedings of the Eighth Conference of the European Chapter of the Association for Computational Linguistics*, p. 217–224.

COPESTAKE A., FLICKINGER D., SAG I. & POLLARD C. (unpublished). Minimal recursion semantics: An introduction.

EVANS R. & GAZDAR G. (1996). DATR: A language for lexical knowledge representation. *Computational Linguistics*, **22** (2 p.), 167–216.

EVANS R., GAZDAR G. & WEIR D. (1995). Encoding lexicalized Tree Adjoining Grammars with a nonmonotonic inheritance hierarchy. In *Proceedings of the 33rd Meeting of the Association for Computational Linguistics*, p. 77–84.

EVANS R. & WEIR D. (1997). Automaton-based parsing for lexicalized grammars. In *Proceedings of the Fifth International Workshop on Parsing Technologies*, p. 66–76.

EVANS R. & WEIR D. (1998). A structure-sharing parser for lexicalized grammars. In *Proceedings of the 36th Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics*, p. 372–378.

GROVER C., CARROLL J. & BRISCOE E. (1993). *The Alvey Natural Language Tools grammar (4th*

release). Technical Report 284, Cambridge University, Computer Laboratory.

SMETS M. & EVANS R. (1998). A compact encoding of a DTG grammar. In *Proceedings of the Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks*.

Some remarks on an extension of synchronous TAG*

David Chiang†, William Schuler†, and Mark Dras‡

†University of Pennsylvania
Dept of Computer and Information Science
200 S 33rd St
Philadelphia PA 19104 USA
{dchiang,schuler}@linc.cis.upenn.edu

‡University of Pennsylvania
Inst for Research in Cognitive Science
Suite 400A, 3401 Walnut St
Philadelphia PA 19104 USA
madras@linc.cis.upenn.edu

Abstract

We explore some properties of the synchronous formalism introduced in Dras (1999), showing that it handles an interaction, noted in Schuler (1999), between bridge and raising verbs which is problematic for synchronous TAG. We also show that it has greater formal power than synchronous TAG and discuss its computational complexity.

1. Introduction

Synchronous TAG (S-TAG), as defined by Shieber (1994), defines relations between languages by assembling paired elementary structures into isomorphic derivations. This isomorphism requirement is formally and computationally attractive, but for practical applications somewhat too strict. For this reason, Shieber suggests relaxing this requirement by treating bounded subderivations as elementary, but there are a few cases which remain problematic because they involve unbounded non-isomorphisms.

One such case is described by Schuler (1999). If a predicate is analyzed as a VP-adjunct in one language but an S-adjunct in another, then an unbounded non-isomorphism will arise when this predicate interacts with other VP-adjuncts. Consider the following sentences from English and Portuguese:

(1) X is supposed to (be going to ...) have to fly.

(2) É pressuposto que X (vai ...) tem/ter que voar.

We might analyze these sentences with the trees in Figure 1, but the resulting derivations for (1) and (2) would be non-isomorphic (see Figure 2).

Shieber (1994) describes this situation as "elimination of dominance"; in this case the non-isomorphism is potentially unbounded because the tree for *supposed to* adjoins into the lowest VP-adjunct on the derivation tree in English, but into the highest tree (that is, the initial tree) in Portuguese.

Schuler (1999) describes a solution to this problem based on a compositional semantics for TAG (Joshi & Vijay-Shanker, 1999) which relies on a mapping of contiguous ranges of scope in source and target derivations, but because it does not map subderivations in the source to subderivations in the target, this solution can only be used on individual

*This research is partially supported by ARO AASERT grant N00014-97-1-0603, ARO grant DAAG55971-0228, and NSF grant SBR-89-20230-15.

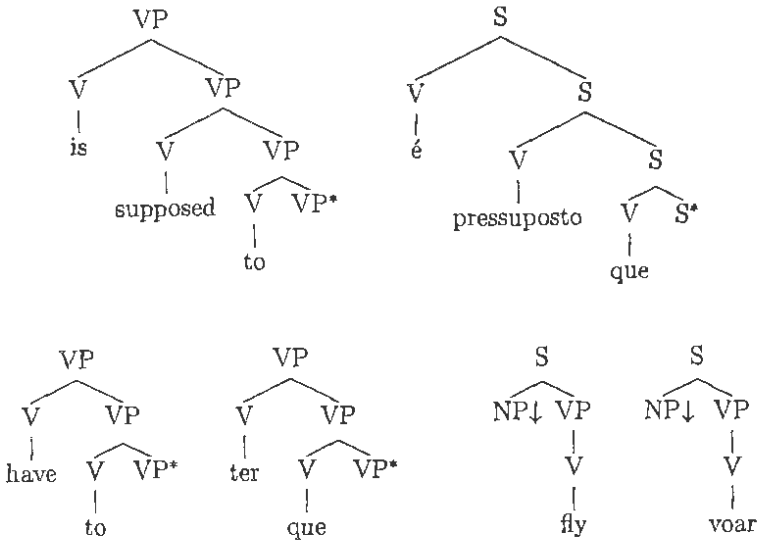


Figure 1: Elementary trees for sentences (1) and (2).

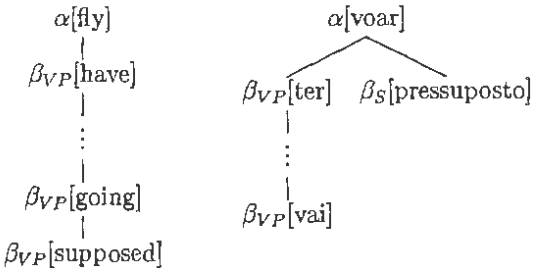


Figure 2: Derivation trees demonstrating *supposed/pressuposto* non-isomorphism.

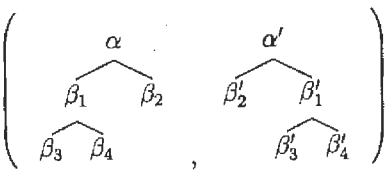


Figure 3: Paired derivation trees

derivation trees and not (tractably) on entire shared forests of possible derivations (Vijay-Shanker & Weir, 1993). Thus, for example, it is not directly possible to parse a natural language question and prune the chart using constraints on a semantic target.[1]
This paper shows that Schuler's example of unbounded non-isomorphism can be handled by the use of a meta-grammar, as in Dras (1999); specifically, by using a TAG meta-grammar in the regular form of Rogers (1994). (We will refer to this formalism as RF-2L(evel)TAG.) In addition, this paper explores how synchronous RF-2LTAG is more powerful than S-TAG: even though the weak generative capacity of the component TAGs is not altered by the synchronisation, the extra strong generative capacity of synchronous RF-2LTAG (that is, the extra structural descriptions it can produce) enables it to describe more *relations* between languages (that is, languages of pairs of strings). We also discuss the computational complexity of this formalism.

2. Using a meta-grammar

Dras (1999) describes what is in effect a relaxation of the requirement in the standard definition of S-TAG that paired derivation trees be isomorphic (as unordered trees). Since TAG derivation trees can be thought of as generated by context-free rules (Weir, 1988), we can likewise think of isomorphic derivation trees as generated by paired context-free rules (Aho & Ullman, 1969). For example, the derivation trees of Figure 3 would be generated by the following:

$$\langle \alpha \rightarrow \beta_1 \boxed{1}\ \beta_2 \boxed{2} \quad , \quad \alpha' \rightarrow \beta'_2 \boxed{2}\ \beta'_1 \boxed{1} \rangle$$
$$\langle \beta_1 \rightarrow \beta_3 \boxed{1}\ \beta_4 \boxed{2} \quad , \quad \beta'_1 \rightarrow \beta'_3 \boxed{1}\ \beta'_4 \boxed{2} \rangle$$

The relaxation proposed by Dras (1999) is to allow some other type of grammar to specify the pairings,[2] namely, TAG: with its greater domain of locality than CFGs, it can specify relationships between nodes of a derivation tree pair which are arbitrarily far apart. A meta-grammar thus pairs substructures in the derivation tree, rather than individual nodes; there is consequently an isomorphism between the trees representing the derivations of the derivations (the 'meta-derivations').
If the TAG meta-grammar is in the regnlar form of Rogers (1994), then the set of derivation trees is recognizable, and the weak generative capacity of the formalism is unchanged (Dras, 1999). Nevertheless, the additional strong generative capacity allows more mappings to be specified.
For example, a TAG meta-grammar can resolve the English-Portuguese mismatch noted above. If we use the same elementary tree pairs from Figure 1, the resulting derivation tree

[1] Ordinary synchronous TAG could use semantic target expressions to filter parse forests, but only if the target grammar were designed to accommodate a particular source grammar, with artificial notions of 'bridge' and 'raising' logical forms.

[2] Shieber's suggestion of treating bounded subderivations as elementary would be analogous to using a tree substitution grammar instead of a CFG to specify the pairings.

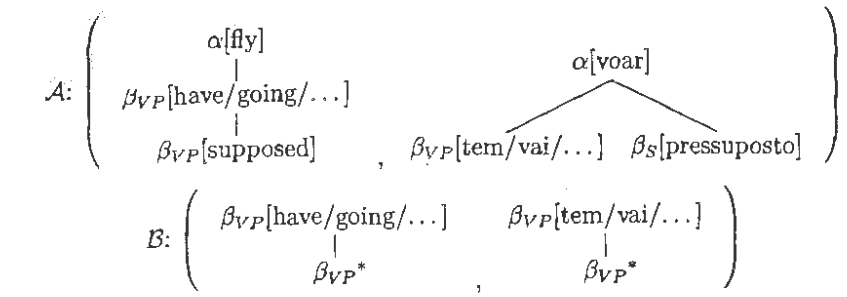


Figure 4: One possible meta-grammar for the *supposed/pressuposto* translation

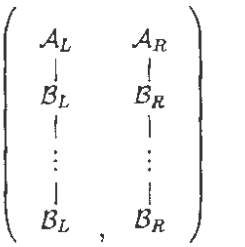


Figure 5: Meta-derivation trees.

structures (Figure 2) are non-isomorphic: in the English case, β[*fly*] and β[*supposed*] get stretched apart by an unbounded number of raising verbs, whereas in the Portuguese case, β[*pressuposto*] attaches directly to α[*voar*] and does not get stretched away. A TAG meta-grammar can be used to factor out the recursive material with pairs of auxiliary trees, like the pair $\mathcal{B}$ in Figure 4. An initial tree pair $\mathcal{A}$ specifies the difference between the English 'linear' derivation structure versus the Portuguese 'branching' derivation structure. The meta-derivation trees are as in Figure 5, with $\mathcal{A}_L$ and $\mathcal{A}_R$ being the left and right projections respectively of $\mathcal{A}$, and similarly for $\mathcal{B}$; they are clearly isomorphic, as desired.

3. Formal properties

Synchronous RF-2LTAG has the weak language preservation property (Rambow & Satta, 1996)—that is, the left and right projection languages of synchronous RF-2LTAGs are all TALs. However, as we have suggested, synchronous RF-2LTAG can specify relations between TALs which synchronous TAG cannot, as the following two claims show:

Claim (synchronous pumping lemma). If L is a language of pairs defined by a synchronous TAG, then there is a constant n such that if $\langle z, z' \rangle \in L$ and $|z| \geq n$ and $|z'| \geq n$, then $\langle z, z' \rangle$ may be written as $\langle u_1 v_1 w_1 v_2 u_2 v_3 w_2 v_4 u_3, u'_1 v'_1 w'_1 v'_2 u'_2 v'_3 w'_2 v'_4 u'_3 \rangle$, with $|v_1 v_2 v_3 v_4 v'_1 v'_2 v'_3 v'_4| > 0$, $|v_1 w_1 v_2 v_3 w_2 v_4| \leq n$, $|v'_1 w'_1 v'_2 v'_3 w'_2 v'_4| \leq n$, such that for all $i \geq 0$, $\langle u_1 v_1^i w_1 v_2^i u_2 v_3^i w_2 v_4^i u_3, u'_1 v'^{i}_1 w'_1 v'^{i}_2 u'_2 v'^{i}_3 w'_2 v'^{i}_4 u'_3 \rangle \in L$.

The proof is similar to that of the normal pumping lemma for TALs (Vijay-Shanker, 1987). The intuition is that the pumping lemma for local sets is applied to the derivation trees, and since paired derivation trees are isomorphic, the pumping constant can be chosen so that the pumping lemma holds for both sides simultaneously.

Claim. $L = \{\langle \mathrm{a}^i 1^j 2^j \mathrm{b}^i \mathrm{c}^i 3^j 4^j \mathrm{d}^i, 1^j \mathrm{a}^i \mathrm{b}^i 2^j 3^j \mathrm{c}^i \mathrm{d}^i 4^j \rangle \mid i, j \geq 0\}$ is not definable by a syn-

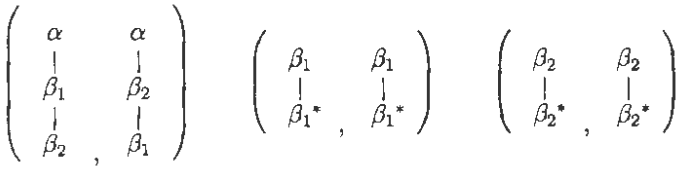


Figure 6: TAG meta-grammar for defining L

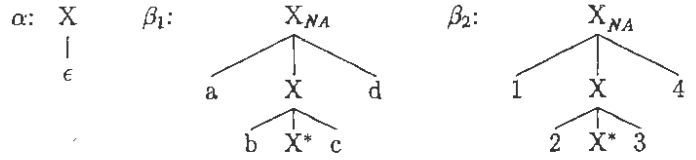


Figure 7: Object level trees for defining L

chronous TAG.

Proof. Assume that L is definable by a synchronous TAG. If n is the constant given by the pumping lemma, let $\langle z, z' \rangle = \langle \mathrm{a}^n 1^n 2^n \mathrm{b}^n \mathrm{c}^n 3^n 4^n \mathrm{d}^n, 1^n \mathrm{a}^n \mathrm{b}^n 2^n 3^n \mathrm{c}^n \mathrm{d}^n 4^n \rangle$ Then z and z' have to be written so that the v_i and v_i' are all letters or all numerals, or else the "pumped" pairs will not be in L. But if they are all letters, then $|v_1 w_1 v_2 v_3 w_2 v_4| > n$; if they are all numerals, then $|v_1' w_1' v_2' v_3' w_2' v_4'| > n$. Since $\langle z, z' \rangle$ cannot be rewritten in the manner indicated by the pumping lemma, L must not be definable by a synchronous TAG.

L can, however, be defined by the synchronous RF-2LTAG in Figure 6, where α, β_1, and β_2 are the same for both sides, shown in Figure 7.

So synchronous RF-2LTAG is more powerful than synchronous TAG; however, just as RF-TAG can be parsed in $\mathcal{O}(n^3)$ time like CFG, RF-2LTAG can be parsed in $\mathcal{O}(n^6)$ time like TAG. We can do this by keeping track of meta-adjunctions using stacks inside the chart items (Rogers, 1994). Because of the regular-form condition, the stacks will have bounded depth.

If we wish to transfer entire shared forests of derivations (Vijay-Shanker & Weir, 1993) rather than single parses, we may incur additional complexity, but this problem can still be solved in polynomial time, because there is a subderivation in the target grammar for every subderivation in the source. In contrast, the method of (Schuler, 1999) would require exponential time because it is defined only on completed parses.

One remaining question is, is it sufficient to use a TAG as a meta-grammar? For any k, define a language over the alphabet $\{\mathrm{a}_1, \mathrm{a}_2, \ldots, \mathrm{a}_k\}$: SEPARATE-$k$ $= \{\langle w, \mathrm{a}_1^{i_1} \mathrm{a}_2^{i_2} \cdots \mathrm{a}_k^{i_k} \rangle \mid$ w has exactly i_j occurrences of $\mathrm{a}_j\}$. SEPARATE-8 can be generated by a synchronous RF-2LTAG (the grammar is not complicated, but large), but SEPARATE-9 cannot. This can be seen by left-intersecting with $(\mathrm{a}_1 \mathrm{a}_2 \cdots \mathrm{a}_9)^*$ (this can be done without disrupting the synchronization): the right projection of the result will be $\{\mathrm{a}_1^n \mathrm{a}_2^n \cdots \mathrm{a}_9^n\}$, which is not generable by any 2LTAG.

More generally, SEPARATE-2^{k+1} can be generated by a synchronous k-level TAG, but SEPARATE-$(2^{k+1}+1)$ cannot. These are all well-behaved relations between *regular* languages; thus the weak language preservation property does not provide a natural ceiling on how powerful a meta-grammar can be. It remains to be seen what kinds of meta-grammars are actually practically useful, and what bounds can be placed on their computational

complexity.

4. Conclusion

In the future we hope to explore the possibility of using meta-level structures for linguistic description (in particular, shifting the Condition on Extended Tree Minimality (Frank, 1992) to meta-level elementary trees); in such an approach it becomes possible to eliminate the *supposed/pressuposto* non-isomorphism entirely.

Under the present approach, however, we have shown that a synchronous TAG meta-grammar provides the extra strong generative capacity needed to localize certain unbounded non-isomorphisms, overcoming some of the limitations of standard synchronous TAG while preserving the essential idea of local synchronization and its attendant advantages.

References

AHO A. V. & ULLMAN J. D. (1969). Syntax directed translations and the pushdown assembler. *J. Comp. Syst. Sci.*, **3** (1 p.), 37–56.

DRAS M. (1999). A meta-level grammar: redefining synchronous TAG for translation and paraphrase. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL '99)*.

FRANK R. (1992). *Syntactic locality and tree adjoining grammar: grammatical acquisition and processing perspectives*. PhD thesis, Computer Science Department, University of Pennsylvania.

JOSHI A. & VIJAY-SHANKER K. (1999). Compositional Semantics with Lexicalized Tree-Adjoining Grammar (LTAG): How Much Underspecification is Necessary? In *Proceedings of the 2nd International Workshop on Computational Semantics*.

RAMBOW O. & SATTA G. (1996). Synchronous Models of Language. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL '96)*.

ROGERS J. (1994). Capturing CFLs with tree adjoining grammars. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics (ACL '94)*.

SCHULER W. (1999). Preserving semantic dependencies in synchronous tree adjoining grammar. *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL '99)*.

SHIEBER S. M. (1994). Restricting the weak-generative capability of synchronous tree adjoining grammars. *Computational Intelligence*, **10** (4 p.).

VIJAY-SHANKER K. (1987). *A study of tree adjoining grammars*. PhD thesis, Department of Computer and Information Science, University of Pennsylvania.

VIJAY-SHANKER K. & WEIR D. (1993). The use of shared forests in tree adjoining grammar parsing. In *Proceedings of EACL '93*, p. 384–393.

WEIR D. (1988). *Characterizing Mildly Context-Sensitive Grammar Formalisms*. PhD thesis, Department of Computer and Information Science, University of Pennsylvania.

Bidirectional parsing of TAG without heads

Víctor J. Díaz[†], Miguel A. Alonso[‡] and Vicente Carrillo[†]

[†]Facultad de Informática y Estadística, Universidad de Sevilla
Avda. Reina Mercedes s/n, 41012 Sevilla (Spain)
{vjdiaz,carrillo}@lsi.us.es
[‡]Departamento de Computación, Universidad de La Coruña
Campus de Elviña s/n, 15071 La Coruña (Spain)
alonso@dc.fi.udc.es

Abstract

We present a bottom-up bidirectional parser for Tree Adjoining Grammars that is an extension of the parser defined by De Vreught and Honig for Context Free Grammars. Although this parser does not improve the complexity of the parsers defined in the literature, it presents several characteristics that can be of interest for practical parsing of natural languages.

1. Introduction

Several algorithms have been proposed for parsing tree adioining grammars (TAGs), most of them derived from context-free tabular parsers, ranging from simple bottom-up algoritinms, like CYK, to sophisticated extensions of Earley's algorithm (Alonso *et al.*, 1999). However, some of the bidirectional parsers proposed are not applicable in all the cases. Lavelli and Satta parser (1991) is restricted to elementary trees with only one anchor. Van Noord parser (1994) introduces several improvements to Lavelli and Satta parser: the substitution operation, the foot-driven recognition of auxiliary trees and the notion of headed elementary trees in order to take advantage of lexicalization.

According to Van Noord, a headed TAG is a TAG in which each elementary tree is a headed tree. For each internal node in a headed tree, there must be a daughter which is the head of the subtree rooted in that node. The reflexive and transitive closure of the head relation is called the head-corner relation. In order to establish the head-corner relation we must fulfill the following two constraints: (i) the anchor of an initial tree must be a head-corner of the root node of the initial tree and (ii) the foot node of an auxiliary tree must be head-corner of the root of the auxiliary tree. Since there exists the notion of anchor in the context of lexicalized TAG, it seems that the notion of head, as defined by Van Noord, is redundant. Moreover, in the case of anchor siblings the definition of head requires to select only one anchor as the head.

In this paper we present a bidirectional bottom-up parser for TAG, called dVH, derived from the context-free parser defined by de Vreught and Honig (de Vreught & Honig, 1989; Sikkel, 1997), which presents several interesting characteristics: (i) the bidirectional strategy allows us to implement the recognition of the adjoining operation in a simple way, (ii) the bottom-up behavior allows us to take advantage of lexicalization, reducing the number of trees under consideration during the parsing process, (iii) in the case of ungrammatical input sentences, the parser is able to recover most of partial parsings according to lexical entries, and (iv) the parser can be applied to every kind of anchored elementary trees without introducing the notion of head

1.1. Notation

Let $G = (V_T, V_N, S, \boldsymbol{I}, \boldsymbol{A})$ be a TAG where V_T and V_N are the terminal and non-terminal alphabets, $S \in V_N$ the axiom symbol, and $\boldsymbol{I}$ and $\boldsymbol{A}$ the set of initial and auxiliary trees respectively. As usual, $\boldsymbol{I} \cup \boldsymbol{A}$ consist of the set of elementary trees.

Parsing algorithms for context-free grammars usually denote partial recognition of productions by dotted productions. We can extend this approach to the case of TAG by considering each elementary tree γ as formed by a set of context-free productions $\mathcal{P}(\gamma)$: a node N^γ in γ and its g children $N_1^\gamma \ldots N_g^\gamma$ are represented by a production $N^\gamma \rightarrow N_1^\gamma \ldots N_g^\gamma$. The elements of the productions are the nodes of the tree, except for the case of elements belonging to $V_T \cup \{\varepsilon\}$ in the right-hand side of productions. Those elements may not have children and are not candidates to be adjunction nodes, so we identify such nodes labeled by a terminal or ε with the label[1]. We use $\beta \in \text{adj}(N^\gamma)$ to denote that $\beta \in \boldsymbol{A}$ may be adjoined at node N^γ. If adjunction is not mandatory at N^γ, then $\textbf{nil} \in \text{adj}(N^\gamma)$. With respect to substitution, we use $\alpha \in \text{sub}(M^\gamma)$ to denote that $\alpha \in \boldsymbol{I}$ can be substituted at node M^γ.

To simplify the description of parsing algorithms we consider additional productions: $\top \rightarrow \mathrm{R}^\alpha$, $\top \rightarrow \mathrm{R}^\beta$ and $\mathrm{F}^\beta \rightarrow \bot$ for each $\alpha \in \boldsymbol{I}$ and each $\beta \in \boldsymbol{A}$, where R^α is the root node of α and R^β and F^β are the root node and foot node of β, respectively. After disabling $\top$ and $\bot$ as adjunction nodes the generative capability of the grammars remains intact.

2. The parser dVH

The definition of the parser is based on deductive systems similar to *Parsing Schemata* (Sikkel, 1997). Given the input string $w = a_1 \ldots a_n$ with $n \geq 0$ and a TAG grammar, the formulas (called items in this context) in the deductive system will be of the form:

$$[N^\gamma \rightarrow \nu \bullet \delta \bullet \omega, i, j, p, q]$$

where $N^\gamma \rightarrow \nu\delta\omega \in P(\gamma)$ is a production decorated with two dots indicating the part of the subtree dominated by N^γ that has been recognized. When ν and ω are both empty, the whole subtree has been recognized. The two indices i and j denote the substring of w spanned by δ. If $\gamma \in \boldsymbol{A}$, p and q are two indices with respect to w indicating the substring spanned by the foot node of γ. In other case $p = q = -$, representing they are undefined.

With respect to deduction steps, we have that

$$\mathcal{D}_{\text{dVH}} = \mathcal{D}_{\text{dVH}}^{\text{Ini}} \cup \mathcal{D}_{\text{dVH}}^{\varepsilon} \cup \mathcal{D}_{\text{dVH}}^{\text{Inc}} \cup \mathcal{D}_{\text{dVH}}^{\text{Conc}} \cup \mathcal{D}_{\text{dVH}}^{\text{Foot}} \cup \mathcal{D}_{\text{dVH}}^{\text{Adj}} \cup \mathcal{D}_{\text{dVH}}^{\text{Subs}}$$

The initializer steps deduce those items associated to productions whose right hand side includes a terminal that matches with an input symbol. The position of the terminal in the input string determines the values of the indices. Empty-productions are considered to match any position in the input string. The indices associated to the foot node in the consequent of both deduction steps are undefined since no foot has been recognized yet:

$$\mathcal{D}_{\text{dVH}}^{\text{Ini}} = \frac{}{[N^\gamma \rightarrow \nu \bullet a \bullet \omega, j-1, j, -, -]} \; a = a_j$$

$$\mathcal{D}_{\text{dVH}}^{\varepsilon} = \frac{}{[N^\gamma \rightarrow \bullet\bullet, j, j, -, -]}$$

Once the subtree dominated by a node M^γ has been recognized completely, a include step in $\mathcal{D}_{\text{dVH}}^{\text{Inc}}$ continues the bottom-up recognition of the supertree dominated by M^γ when no adjoining

[1] Without lost of generality, we assume that if a node is labeled by ε then it has no siblings.

is mandatory on that node. Indexes are not modified when a step of this type is applied:

$$\mathcal{D}_{\text{dVH}}^{\text{Inc}} = \frac{[M^\gamma \rightarrow \bullet\delta\bullet, i, j, p, q]}{[N^\gamma \rightarrow \nu \bullet M^\gamma \bullet \omega, i, j, p, q]} \ \mathbf{nil} \in \text{adj}(M^\gamma)$$

Given a node N^γ such that $N^\gamma \rightarrow \nu\delta_1\delta_2\omega$, the concatenate steps in $\mathcal{D}_{\text{dVH}}^{\text{Conc}}$ try to combine two partial analysis spanning consecutive parts of the input string, in order to recognize δ_1 and δ_2. The indices i and j in the consequent cover the whole recognized substring. The values of the indices p and q, corresponding to the foot node, are propagated bottom-up:

$$\mathcal{D}_{\text{dVH}}^{\text{Conc}} = \frac{\begin{array}{c}[N^\gamma \rightarrow \nu \bullet \delta_1 \bullet \delta_2\omega, i, j', \mathbf{p}, q], \\ [N^\gamma \rightarrow \nu\delta_1 \bullet \delta_2 \bullet \omega, j', j, \mathbf{p}', q']\end{array}}{[N^\gamma \rightarrow \nu \bullet \delta_1\delta_2 \bullet \omega, i, j, p \cup p', q \cup q']}$$

where $p \cup q$ is equal to p if q is undefined and is equal to q if p is undefined, being undefined in other case.

The foot steps $\mathcal{D}_{\text{dVH}}^{\text{Foot}}$ introduce in a bottom-up way a new instance of an auxiliary tree β in an adjunction node M^γ where $\beta \in Adj(M^\gamma)$. The recognition of the auxiliary tree begins with the introduction of the foot node. The string spanned by the node M^γ between position k and l determines the values of the indices in the consequent. The indices p and q in the antecedent are ignored in the consequent because a new adjoining has been introduced. The values of these indices will be considered by adjoining steps in order to conclude the adjoining of β in M^γ:

$$\mathcal{D}_{\text{dVH}}^{\text{Foot}} = \frac{[M^\gamma \rightarrow \bullet\delta\bullet, k, l, p, q]}{[\mathrm{F}^\beta \rightarrow \bullet\bot\bullet, k, l, k, l]} \ \beta \in \text{adj}(M^\gamma)$$

When the recognition of the auxiliary tree β reaches the root node, the adjoining steps $\mathcal{D}_{\text{dVH}}^{\text{Adj}}$ conclude the adjoining on M^γ, continuing the bottom-up recognition of the supertree of γ with respect to M^γ. This step is only applied when the string spanned by the foot node of β is equal to the string spanned by the adjunction node M^γ. Indices p and q in the consequent are obtained from the antecedent associated to the adjunction node. Now, the string spanned by the adjunction node M^γ corresponds with the string spanned by the root of the auxiliary tree β:

$$\mathcal{D}_{\text{dVH}}^{\text{Adj}} = \frac{\begin{array}{c}[\top \rightarrow \bullet\mathrm{R}^\beta\bullet, j, m, k, l], \\ [M^\gamma \rightarrow \bullet\delta\bullet, k, l, p, q]\end{array}}{[N^\gamma \rightarrow \nu \bullet M^\gamma \bullet \omega, j, m, p, q]} \ \beta \in \text{adj}(M^\gamma)$$

A substitution is performed when an initial tree α has been completely recognized. The initial tree establishes the string spanned by the node M^γ where α can be substituted.

$$\mathcal{D}_{\text{dVH}}^{\text{Subs}} = \frac{[\top \rightarrow \bullet\mathrm{R}^\alpha\bullet, i, j, -, -]}{[N^\gamma \rightarrow \nu \bullet M^\gamma \bullet \omega, i, j, -, -]} \ \alpha \in \text{sub}(M^\gamma)$$

The input string must belong to the language defined by the grammar, given $\alpha \in I$ rooted with the axiom symbol, whenever an item $[\top \rightarrow \bullet\mathrm{R}^\alpha\bullet, 0, n, -, -]$ is deduced. The algorithm so described is just a recognizer. However, it is not difficult to construct an actual chart parser based on the specification presented above. From the set of derived items (the chart), a parser of the input can be constructed retracing the recognition steps in reverse order or annotating the items computed by the recognition algorithm with information about how they were obtained.

The time complexity of the algorithm with respect to the length n of the input is $O(n^6)$. This complexity is due to deduction steps in $\mathcal{D}_{\text{dVH}}^{\text{Adj}}$ since they present the maximum number of relevant input variables (j, m, k, l, p, q). The space-complexity of the parser is $O(n^4)$ since every item is composed of four input positions ranging on the length of the input string.

The number of items deduced by the parser, as stated before, can be reduced if we apply a filter on the concatenate steps. We can note that these steps produce redundant derivations when the trees are not binary branching. If we do so, the parser obtained will not actually be bidirectional. We will not consider this version because of clarity in the exposition.

3. A new parser dVH′

In the context of parsing lexicalized grammars for natural languages, the parser dVH can be slightly modified in order to speed up the recognition process. In this way, we will consider the characteristics of the English grammar defined in (XTAG, 1999). The study will be mainly centered on $\mathcal{D}^{\text{Init}}_{\text{dVH}}$, $\mathcal{D}^{\varepsilon}_{\text{dVH}}$ and $\mathcal{D}^{\text{Subs}}_{\text{dVH}}$ deduction steps. We will call dVH′ the new parser obtained after modifications.

First of all, we must note that $\mathcal{D}^{\text{Init}}_{\text{dVH}}$ steps can be applied on anchors as well. In the case of multi-anchors, the step will deduce one item for every anchor in the elementary tree with the suitable positions respect to the input. Furthermore, this step implies an important reduction in the search space since only those elementary trees with anchors matching the input will be consider in the recognition. In this way, $\mathcal{D}^{\text{Foot}}_{\text{dVH}}$ and $\mathcal{D}^{\text{Subs}}_{\text{dVH}}$ deduction steps will not introduce any elementary tree except for those trees considered by $\mathcal{D}^{\text{Init}}_{\text{dVH}}$ deduction steps.

When substitution nodes are siblings of no-substitution nodes the application of $\mathcal{D}^{\text{Subs}}_{\text{dVH}}$ steps can introduce items that are not necessary in order to recognize the input string. The reason is that $\mathcal{D}^{\text{Subs}}_{\text{dVH}}$ deduction steps always try to include an initial tree when this tree is completed. To avoid these redundant substitution operations we can introduce a filter as follows: this step will only applied when M^{γ} is a substitution node whose daughters do not dominate a terminal or the foot node of γ. In other cases, the substitution operation will be performed by the new deduction steps $\mathcal{D}^{\text{SubsR}}_{\text{dVH}}$ and $\mathcal{D}^{\text{SubsL}}_{\text{dVH}}$.

$$\mathcal{D}^{\text{SubsR}}_{\text{dVH}'} = \frac{\begin{array}{c}[\top \rightarrow \bullet \mathbf{R}^{\alpha} \bullet, j, k, -, -], \\ [N^{\gamma} \rightarrow \nu \bullet \delta \bullet M^{\gamma} \omega, i, j, p, q]\end{array}}{[N^{\gamma} \rightarrow \nu \bullet \delta M^{\gamma} \bullet \omega, i, k, p, q]}$$

$$\mathcal{D}^{\text{SubsL}}_{\text{dVH}'} = \frac{\begin{array}{c}[\top \rightarrow \bullet \mathbf{R}^{\alpha} \bullet, i, j, -, -], \\ [N^{\gamma} \rightarrow \nu M^{\gamma} \bullet \delta \bullet \omega, j, k, j, p, q]\end{array}}{[N^{\gamma} \rightarrow \nu \bullet M^{\gamma} \delta \bullet \omega, i, k, p, q]}$$

With respect to ε productions, the number of items deduced by $\mathcal{D}^{\varepsilon}_{\text{dVH}}$ steps can be an important drawback in the application of the parser when the grammar has a lot of elementary trees with ε productions. As an example, in the English grammar (XTAG, 1999), it is usual that left hand sides of empty productions present a null adjoining constraint. The practical behavior of the parser can be improved if we filter the steps dealing with empty productions. Given a production $M^{\gamma} \rightarrow \varepsilon$ such that $\{\mathbf{nil}\} = \text{adj}(M^{\gamma})$, where $\{\mathbf{nil}\} = \text{adj}(M^{\gamma})$ represents a null-adjoining constraint on M^{γ} and M^{γ} has at least a daughter that dominates a terminal or the foot node of γ, the following deduction steps

$$\mathcal{D}^{\varepsilon\text{R}}_{\text{dVH}'} = \frac{[N^{\gamma} \rightarrow \nu \bullet \delta \bullet M^{\gamma} \omega, i, j, p, q]}{[N^{\gamma} \rightarrow \nu \bullet \delta M^{\gamma} \bullet \omega, i, j, p, q]}$$

$$\mathcal{D}^{\varepsilon\text{L}}_{\text{dVH}'} = \frac{[N^{\gamma} \rightarrow \nu M^{\gamma} \bullet \delta \bullet \omega, i, j, p, q]}{[N^{\gamma} \rightarrow \nu \bullet M^{\gamma} \delta \bullet \omega, i, j, p, q]}$$

drastically reduce the number of items generated. When the above constraints are not satisfied, a $\mathcal{D}^{\varepsilon}_{\text{dVH}}$ step must be applied.

Input	dVH	dVH′	VN	E	Ned
Transitives and Ditransitives					
1	0.16	0.05	0.10	0.33	0.33
2	0.27	0.05	0.16	0.38	0.44
Arguments and Adjuncts					
3	0.38	0.11	0.22	0.49	0.55
4	0.33	0.05	0.16	0.44	0.49
5	0.16	0.01	0.05	0.27	0.33
Ergatives and Intransitives					
6	0.33	0.11	0.22	0.38	0.44
7	0.16	0.05	0.11	0.27	0.27
8	0.16	0.05	0.11	0.33	0.33
9	0.16	0.05	0.11	0.27	0.27
Sentential Complements					
10	0.16	0.05	0.11	0.55	0.44
11	0.22	0.05	0.17	0.66	0.49
Relative Clauses					
12	0.60	0.16	0.38	0.77	0.88
13	0.55	0.16	0.38	0.66	0.77

Input	dVH	dVH′	VN	E	Ned
Auxiliary Verbs					
14	0.60	0.22	0.44	0.66	0.77
Extraction					
15	0.16	0.05	0.16	0.33	0.33
16	0.22	0.05	0.11	0.33	0.33
17	0.16	0.05	0.11	0.38	0.33
Unbounded Dependencies					
18	0.22	0.05	0.11	0.22	0.27
19	0.39	0.11	0.22	0.71	0.61
20	0.28	0.11	0.16	0.55	0.49
21	0.82	0.16	0.49	1.54	1.26
Adjectives					
22	0.11	0.05	0.05	0.22	0.27
23	0.16	0.05	0.11	0.27	0.27
24	0.22	0.05	0.11	0.27	0.27
25	0.33	0.05	0.16	0.33	0.33

Table 1: Parsing time in seconds

4. Experimental results

The results we are going to discuss have been obtained using a naive implementation in Prolog of the deductive parsing machine presented in (Shieber *et al.*, 1995) running on a Pentium II. We have implemented and tested the following parsers: E is an Earley-based parser without prefix valid property (Alonso *et al.*, 1999), Ned is an Earley-based parser with prefix valid property (Nederhof, 1999), VN is the bidirectional parser defined by Van Noord, and dVH and dVH′ are the parsers defined in this paper.

The study is based on the English grammar presented in (XTAG, 1999). From this document we have selected a subset of the grammar consisting of 27 elementary trees that cover a variety of English constructions: relative clauses, auxiliary verbs, unbounded dependencies, extraction, etc. In order to compare only the behavior of the parsers, we have not consider the feature structures of elementary trees. In this way, we have simulated the features using local constraints. Also, we have selected from the document 25 correct and incorrect sentences grouped with respect to the aspect treated. Every sentence has been parsed without previous filtering of elementary trees. Table 1 shows the time in seconds used for every algorithm and sentence.

From table 1, we can observe that VN, dVH and dVH′ obtain better time results than predictive parsers E and Ned. However, in terms of the more expensive step, the adjoining operation, predictive parsers perform equal or less adjoining operations than bottom-up parsers. Therefore, we can argue that this result is a consequence of the implicit filtering of elementary trees of bottom-up strategies.

On the other hand, we can also note that although dVH presents worse time than VN, we can see dVH′ improves the results of VN. Since the adjoining operations performed by all the bottom-up parsers are practically the same, we can conclude that this improvement is basically due to the reduction of items removed by the filter in the rules related to ε productions.

5. Conclusion

A bottom-up bidirectional parser for TAG has been defined based on the parser defined by De Vreught and Honig for CFG. The parser does not improve the worst-case bounds of already known parsing methods for TAG but the experiments show similar or better time results than classical parsers. Other benefits can be argued to consider this algorithm of interest in the context of bidirectional parsers. In particular, with respect to Lavelli-Satta parser the dVH schema can be applied to multi-anchor auxiliary trees. With respect to Van Noord parser, this new approach does not introduce the concept of head and it is applicable to every kind of anchored elementary tree.

As further work, it would be interesting to investigate the effects of compacting elementary trees, as performed by Lopez (2000), in the real performance of the parser.

6. Acknowledgments

This research was partially supported by the FEDER of EU (Grant 1FD97-0047-C04-02) and Xunta de Galicia (Grant PGIDT99XI10502B).

References

ALONSO M. A., CABRERO D., DE LA CLERGERIE E. & VILARES M. (1999). Tabular algorithms for TAG parsing. In *Proc. of EACL'99, Ninth Conference of the European Chapter of the Association for Computational Linguistics*, p. 150–157, Bergen, Norway: ACL.

DE VREUGHT J. P. M. & HONIG H. J. (1989). *A Tabular Bottom-up Recognizer*. Technical Report 89-78, Department of Applied Mathematics and Informatics, Delft University of Technology, Delft, The Netherlands.

LAVELLI A. & SATTA G. (1991). Bidirectional parsing of lexicalized tree adjoining grammars. In *Proceedings of the 5th Conference of the European Chapter of the Association for Computational Linguistics (EACL'91)*, Berlin, Germany: ACL.

LOPEZ P. (2000). Extended partial parsing for lexicalized tree grammars. In *Proc. of the Sixth International Workshop on Parsing Technologies (IWPT 2000)*, p. 159–170, Trento, Italy.

NEDERHOF M.-J. (1999). The computational complexity of the correct-prefix property for TAGs. *Computational Linguistics*, **25** (3), p. 345–360.

SHIEBER S. M., SCHABES Y. & PEREIRA F. (1995). Principles and implementation of deductive parsing. *Journal of Logic Programming*, **24** (1–2), p. 3–36.

SIKKEL K. (1997). *Parsing Schemata — A Framework for Specification and Analysis of Parsing Algorithms*. Texts in Theoretical Computer Science — An EATCS Series. Berlin/Heidelberg/New York: Springer-Verlag.

VAN NOORD G. (1994). Head-corner parsing for TAG. *Computational Intelligence*, **10** (4), p. 525–534.

THE XTAG RESEARCH GROUP (1999). A lexicalized tree adjoining grammar for English. `http://www.cis.upenn.edu/~xtag`. Technical Report IRCS 95-03, IRCS, Institute for Research in Cognitive Science, University of Pennsylvania

Workshop TAG+5, Paris, 25-27 May 2000

Punctuation in a Lexicalized Grammar

Christine Doran*

The MITRE Corporation
202 Burlington Road
Bedford, MA 01730

Abstract

Just as people make use of information from punctuation to structure and understand text, NLP systems can use information from punctuation in processing texts automatically. The aim of the research presented here was to explore the feasibility of treating a sizable core of punctuation phenomena at the level of the sentence grammar. A large set of punctuation rules were manually derived from naturally occurring data, and added to the XTAG English grammar. Our results confirm that punctuation can be used in analyzing sentences to increase the coverage of the grammar, reduce the ambiguity of certain word sequences and facilitate later processing of larger text units, without either adversely impacting the existing grammar or deriving analyses which would be incompatible with later levels of processing.

1. Motivation

Punctuation helps us to structure, and thus to understand, texts. Many uses of punctuation straddle the line between syntax and discourse, because they serve to combine multiple propositions within a single orthographic sentence. They allow us to insert discourse-level relations at the level of a single sentence. Just as people make use of information from punctuation in processing what they read, natural language processing systems can use information from punctuation in processing texts automatically.

Most current NLP systems fail to take punctuation into account at all, losing a valuable source of information about the text. Those which do mostly do so in a superficial way, again failing to fully exploit the information conveyed by punctuation. To be able to make use of such information in a computational system, we must first characterize its uses and find a suitable representation for encoding them.

Previous work on punctuation was mostly of the descriptive variety, of which Quirk et al. (1985) and Sampson (1995) are particularly good instances. Some linguistic work has been done by Chafe (1988), Schmidt (1995), Jones (1996b) and Meyer (1987). Nunberg (1990) offers the most comprehensive linguistic discussion of punctuation to date, with an extensive analysis of the interactions of different punctuation marks. He is primarily interested in characterizing punctuation as a formal system, independent from syntax. Briscoe (1994) presents an treatment of punctuation within the Alvey Natural Language Tools grammar. He and Carroll (1995) show that this analysis considerably reduces ambiguity in parsing the SUSANNE corpus (a subset of the Brown corpus) and Jones shows similar results.

The work discussed here differs from previous work in a number of ways. It includes an analysis of the syntax of punctuation which has been implemented and integrated into a large English

This work was done while the author was a graduate student at the University of Pennsylvania, and was partially supported by NSF Grant SBR8920230 and ARO Grant DAAH0404-94-G-0426. Thanks to Aravind Joshi and Ted Briscoe for their helpful comments at all stages of this work, and to the members of the XTAG Project.

grammar that is being used on an everyday basis. In addition, the analysis differs considerably from those of Jones and Briscoe in treating punctuation within a framework which allows for more concise characterization of the non-local aspects of certain uses of punctuation. Furthermore, neither of their implementations cover the range of punctuated constructions our treatment does.

2. Analysis

Many parsers require that punctuation be stripped out of the input. Where punctuation is optional, as is often the case, this may have no effect. However, there are a number of constructions where punctuation is obligatory. Adding analyses of these to the grammar without the punctuation can lead to severe over-generation, possibly to the point where it is better to not add the constructions at all.

The work here focuses on extending a lexicalized syntactic grammar to handle phenomena occurring within a single sentence which have punctuation as an integral component. The main job of the sentence grammar, then, is to produce a structure that makes the appropriate units easily accessible to later levels of processing–not just basic grammatical elements like subject noun and verb group, but more complex relations like nominal apposition as well. Punctuation marks are treated as full-fledged lexical items in a Feature-Based Lexicalized Tree Adjoining Grammar (Joshi, 1985; Schabes, 1990; Vijay-Shanker & Joshi, 1991). The localization of both syntactic and semantic dependencies provides an elegant framework for encoding punctuation in the sentence grammar. The elementary units of LTAG are of a suitable size for stating most of the constraints we are interested in, and the derivation histories it produces contain information that later stages of processing will need about which elementary units have been used and how they have been combined. Each punctuation mark or pair of marks anchors its own elementary trees and imposes constraints on the surrounding lexical items. The TAG adjunction operation is advantageous in handling paired punctuation marks, because it allows us to keep both pieces of the complex object, e.g. a pair of parentheses or commas, in the same elementary tree, regardless of the size of the constituent they enclose.

We have analyzed naturally-occurring data (primarily from the Brown Corpus) representing a wide variety of constructions, and added treatments of them to the XTAG English grammar. The new trees are of two types. The first have the punctuation marks as anchors, reflecting the fact that they do not strongly constrain the lexical content of the constructions they participate in. For example, any NP except a pronoun can be an appositive, and this is reflected in the analysis by having the NP position as a substitution site in the NP appositive tree (Figure 1). The second type of tree has the punctuation marks as substitution sites, for instance the tree for parenthetical adverbs, where the lexical material may vary, some punctuation mark is required, but any of several types of punctuation mark is permissible. This is illustrated by the tree for a quoting clause shown in Figure 2. There are a total of 47 trees containing punctuation marks in the current implementation. Doran (1998) discusses all of the trees in more detail.

The full set of punctuation marks is divided into three classes: **balanced**, **structural** (term from (Meyer, 1987)) and **terminal**. The balanced punctuation marks are quotes and parentheses, structural are commas, dashes, semi-colons and colons, and terminal are periods, exclamation points and question marks. These three types of punctuation are essentially independent subsystems, and a given constituent will typically have only one of each type. Structural and terminal punctuation marks do not occur adjacent to other members of the same class, but may occasionally occur adjacent to members of the other class, e.g. a question mark on a clause which is separated by a dash from a second clause. Balanced punctuation marks are sometimes adjacent to one another, e.g. quotes immediately inside of parentheses as in example

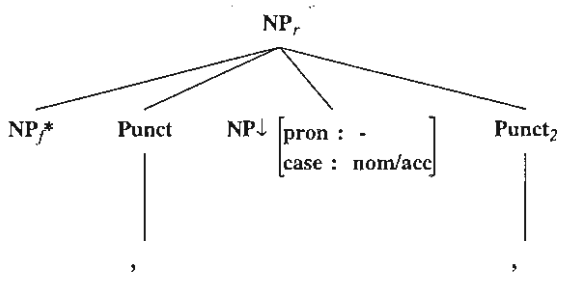


Figure 1: The non-peripheral NP appositive tree, showing relevant features.

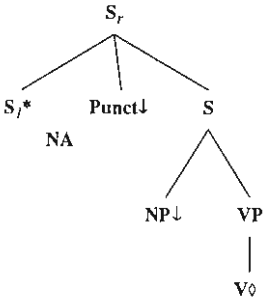


Figure 2: Tree for a quoting clause which follows the quote; the tree would be anchored by e.g. *mutter* in a sentence such as *Liver again, Mary muttered.*

(1). Features allow us to control these local interactions. We also use these features to control non-local phenomenon such as quote alternation, whereby single and double quotes alternate when embedded, and also to control the embedding of colons and semi-colons.

(1) Each enjoys seeing the other hit home runs ("I hope Roger hits 80", Mantle says), and each enjoys even more seeing himself hit home runs ("and I hope I hit 81"). [Brown:ca39]

2.1. How punctuation improves the grammar

There are two primary ways in which adding punctuation improved the coverage and performance of the XTAG English grammar. First, it allowed us to add some syntactically "exotic" constructions which would have previously been considered too unconstrained in their unpunctuated forms. Many such constructions occur with great frequency in naturally occurring texts. As an example, consider noun appositives, where an NP modifies another NP. Example (2) has two appositives. Without access to punctuation, the parser would derive every combinatorial possibility of NPs in noun sequences, which is obviously undesirable (especially since there is already unavoidable noun-noun compounding ambiguity). These phrases must be "bracketed" by punctuation, which provides precisely the sort of additional constraint we need to make the parsing task manageable. By adding a treatment of punctuation to the grammar, we can recognize and correctly analyze appositive constituents. Other similar such constructions include parenthetical elements, reported speech, compound sentences, comma coordination and vocatives. None of these constructions were handled by our English grammar before it was extended to treat punctuation.

(2) But Tony Robinson, *the current sheriff of Nottingham* – **a job that really exists** – rejected the theory, saying that "as far as we are concerned, Robin Hood was a Nottinghamshire lad." [clari.living.celebrities]

Second, punctuation provides additional constraints for parsing constructions already handled by the grammar. In developing a large grammar for any language, one of the fundamental concerns is the increase in ambiguity of derivations which invariably accompanies any increase in coverage of the language's constructions. Adding punctuation to the grammar reduces the ambiguity of analyses by marking the boundaries of clauses and phrases. Adding analyses of subordinate clauses, the majority of whose variants include punctuation, was found in (Doran, 1996) to improve the coverage of the XTAG English grammar by 6.6% on Brown corpus data.

2.2. *Previous Work*

Information from punctuation has only recently been taken into consideration in parsing and grammar development (see (Briscoe, 1994; Jones, 1996b)). The only other such grammar to treat punctuation integrally is a POS-tag sequence grammar developed by (Briscoe & Carroll, 1995) using the Alvey Natural Language Tools as a starting point, which includes Briscoe's analysis of punctuation. Unlike the present work, they do not look at the particular lexical items in the input string, only the POS sequence. However, they do treat punctuation "lexically" to a certain extent, in the sense that each punctuation mark occurs in a range of (discourse) grammar rules.

3. Evaluation

Ideally, we would evaluate the punctuation rules using full parsing—take a corpus of sufficiently complex sentences, parse it both with and without the punctuation marks, and measure the improvements in coverage and accuracy when the punctuation is taken into consideration. However, such an experiment proved impossible for practical reasons because our current parser runs out of memory on sentences of any interesting length with their punctuation stripped.[1]

Another way to measure the improvement in the grammar is to use the supertagging technique developed by (Srinivas, 1997). Supertagging takes the trees of an LTAG, and uses them as complex part-of-speech tags. To evaluate the LTAG punctuation analysis, we used a supertagger trained on just over 1 million words of Wall Street Journal data whose supertags were derived by conversion from the (hand-corrected) Treebank parses. We first trained the tagger on the data with all punctuation stripped, and tested it on 2012 held-out sentences, also with punctuation stripped. We then retrained the tagger on the full million words, and tested it on the same test data with punctuation retained. The performance is shown in Table 1. The most important line is the middle one, showing performance of both sets of training data on exclusively non-punctuation tokens. We achieved an error reduction of 10.9% on non-punctuation tokens, showing that the presence of punctuation does indeed improve the accuracy of analysis of the surrounding texts. Our result reflects an increase in the number of non-punctuation tokens to which the correct structural tag was assigned only when punctuation was present. This figure is not directly comparable to the coverage improvement obtained by (Briscoe & Carroll, 1995) of 8%, which reflects an increase in the number of sentences for which some parse (not necessarily correct) was obtained. Nor can it be compared with their improved crossing brackets performance on SUSANNE sentences, which looks at the number of correct constituents. Supertagging accuracy is measured on a per word basis, and always assigns a tag to every word,

[1] Jones (1996a) encounters the same problem in attempting to evaluate his grammar. His chart-based parser cannot enumerate the number of parses possible for many of the unpunctuated sentences in his test set, and he has to turn to a special estimation process which interrupts the parser before it actually builds any parses.

so there is no notion of complete failure on a sentence. In that sense, supertagging does assign a structure to every sentence, but without assembling the supertag sequence assigned, you do not know what the hypothesized constituents are. The most appropriate comparison is with the evaluation presented in (Briscoe, 1994), where he finds a 2% improvement in "rule application" on SUSANNE sentences (i.e. the correct derivational step applied at a given point) since we can think of each LTAG tree as a rule (or possible several rules) to be applied.

	Trained and tested on text without punctuation	with punctuation
% Correct Overall	87.1%	88.0%
On non-punct tokens	**87.1%**	**88.5%**
On punct tokens	—	83.7%

Table 1: Accuracy of supertagging with and without punctuation

One important thing to remember is that the supertagger has only a three-token window in assigning tags, and constructions involving punctuation often span a fairly large number of tokens (e.g. the comma around a relative clause, parentheses around sentences). This suggests that performance might be much more dramatically improved if we were able to use the full parser. The baseline performance for supertagging punctuation marks (i.e. assigning simply the most likely tag to each mark) is 65.9%. This is considerably lower than regular part-of-speech tagging at around 90% and supertagging overall at 77.2% for this corpus. The baseline for punctuation is lower because the average number of candidate supertags per token is higher: 6.5 supertags per punctuation mark compared with 1.5 parts-of-speech per word in standard part-of-speech tagging.

The difference in performance can be seen on example (3). When the comma preceding the lexical conjunction *and* is removed, the supertagger incorrectly assigns a relative clause tag to the verb *gave*. With the comma present, the verb correctly gets a main verb tag for *gave*.

(3) He left his last two jobs at Republic Airlines and Flying Tiger with combined stock-option gains of about $22 million $\begin{matrix}\text{and UAL gave\_N0nx0Vnx1nx2}\\ \text{, and UAL gave\_nx0Vnx1nx2}\end{matrix}$ him a $15 million bonus when he was hired .

4. Conclnsions

Our aim in undertaking this research was to find out how feasible it was to handle a sizable core of punctuation phenomena at the level of the sentence grammar, without either adversely impacting the existing grammar or deriving analyses which would be incompatible with later levels of processing, in particular at the discourse level. Our results confirm that punctuation can be used in analyzing sentences to increase the coverage of the grammar, reduce the ambiguity of certain word sequences and facilitate discourse-level processing of the texts.[2] We have implemented quite an extensive grammar for punctuation which has been incorporated into the XTAG English grammar, and found that the punctuation rules do indeed improve the coverage of the existing grammar with no negative impact on the rest of the grammar.

[2]In (Doran, 1998), we also show that the LTAG analysis of the text adjunct variant is fully compatible with a discourse grammar of the sort proposed by Webber and Joshi (1998).

References

BRISCOE T. (1994). *Parsing (with) Punctuation etc.* Technical Report MLTT-TR-002, Rank Xerox Research Centre, Grenoble, France.

BRISCOE T. & CARROLL J. (1995). Developing and Evaluating a Probabilistic LR Parser of Part-of-Speech and Punctuation Labels. In *Proceedings of the Fourth International Workshop on Parsing Technologies (IWPT'95)*, p. 48–57, Prague/Karlovy Vary, Czech Republic.

CHAFE W. (1988). Punctuation and the prosody of written language. *Written Communication*, **5** (4 p.), 395–426.

DORAN C. (1996). Punctuation in Quoted Speech. In *Proceedings of the SIGPARSE96*, Santa Cruz, California.

DORAN C. (1998). *Incorporating Punctuation into the Sentence Grammar: A Lexicalized Tree Adjoining Grammar Perspective*. PhD thesis, University of Pennsylvania.

JONES B. (1996)a. Towards a Syntactic Account of Punctuation. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*, Copenhagen, Denmark.

JONES B. (1996)b. *What's the Point? A (Computational) Theory of Punctuation*. PhD thesis, University of Edinburgh.

JOSHI A. K. (1985). Tree Adjoining Grammars: How much context Sensitivity is required to provide a reasonable structural description. In D. DOWTY, I. KARTTUNEN & A. ZWICKY, Eds., *Natural Language Parsing*, p. 206–250. Cambridge, U.K.: Cambridge University Press.

MEYER C. F. (1987). *A Linguistic Study of American Punctuation*, volume 5 of *American University Studies, Series XIII - Linguistics*. New York: Peter Lang.

NUNBERG G. (1990). *The Linguistics of Punctuation*. CSLI Lecture Notes No. 18. Stanford: Center for the Study of Language and Information.

QUIRK R., GREENBAUM S., LEECH G. & SVARTVIK J. (1985). *A Comprehensive Grammar of the English Language*. London: Longman.

SAMPSON G. (1995). *English for the Computer:The SUSANNE Corpus and Analytic Scheme*. Oxford: Clarendon Press.

SCHABES Y. (1990). *Mathematical and Computational Aspects of Lexicalized Grammars*. PhD thesis, Computer Science Department, University of Pennsylvania.

SCHMIDT M. (1995). *Acoustic Correlates of Encoded Prosody in Written Conversation*. PhD thesis, The University of Edinburgh.

SRINIVAS B. (1997). *Complexity of Lexical Descriptions and its Relevance to Partial Parsing*. PhD thesis, Department of Computer and Information Sciences, University of Pennsylvania.

VIJAY-SHANKER K. & JOSHI A. K. (1991). Unification Based Tree Adjoining Grammars. In J. WEDEKIND, Ed., *Unification-based Grammars*. Cambridge, Massachusetts: MIT Press.

WEBBER B. & JOSHI A. (1998). Anchoring a Lexicalized Tree-Adjoining Grammar for Discourse. In *Proceedings of the Workshop on Discourse Relations and Discourse Markers (at COLING98)*, p. 86–92, Montreal, Canada.

A Faster Parsing Algorithm for Lexicalized Tree-Adjoining Grammars

Jason Eisner† and Giorgio Satta‡

†Dept. of Computer Science
University of Rochester
P.O. Box 270226
Rochester, NY 14627-0226 USA
jason@cs.rochester.edu

‡Dip. di Elettronica e Informatica
Università di Padova
via Gradenigo 6/A
I-35131 Padova, Italy
satta@dei.unipd.it

Abstract

This paper points out some computational inefficiencies of standard TAG parsing algorithms when applied to LTAGs. We propose a novel algorithm with an asymptotic improvement.

Introduction

Lexicalized Tree-Adjoining Grammars (LTAGs) were first introduced in (Schabes *et al.*, 1988) as a variant of Tree-Adjoining Grammars (TAGs) (Joshi, 1987). In LTAGs each elementary tree is specialized for some individual lexical item. Following the original proposal, LTAGs have been used in several state-of-the-art, real-world parsers; see for instance (Abeillé & Candito, 2000) and (Doran *et al.*, 2000).

Like link grammar (Sleator & Temperley, 1991) and lexicalized formalisms from the statistical parsing literature (Collins, 1997; Charniak, 1997; Alshawi, 1996; Eisner, 1996) LTAGs provide two main recognized advantages over more standard non-lexicalized formalisms:

- subcategorization can be specified separately for each word; and
- each word can restrict the anchors (head words) of its arguments and adjuncts, encoding lexical preferences as well as some effects of semantics and world knowledge.

To give a simple example, consider the verb *walk*, which is usually intransitive but can take an object in some restricted cases. An LTAG can easily specify the acceptability of sentence *Mary walks the dog* by associating *walk* with a transitive elementary tree that selects for an indirect object tree anchored at word *dog* (and some other words within a limited range).

LTAGs are large because they include separate trees for each word in the vocabulary. However, parsing need consider only those trees of the grammar that are associated with the lexical sym-

bols in the input string. While this strategy reduces parsing time in all practical cases, since the input string length tends to be considerably smaller than the grammar size, it also introduces an additional factor in the runtime that depends on the input string length. This problem was recognized early in (Schabes *et al.*, 1988), but a precise computational analysis has not been provided in the literature, up to the authors' knowledge. See (Eisner, 1997; Eisner, 2000) for related discussion on different lexicalized formalisms.

In this work we reconsider LTAG parsing in the above perspective. Under standard assumptions, we show that existing LTAG parsing methods perform with $\mathcal{O}(tg^2\,|w|^8)$ worst case running time, where t and g are smallish constants depending on the grammar and w is the input string. As our main result we present an $\mathcal{O}(tg\,|w|^6\,\max\{g,|w|\})$ time algorithm, a considerable improvement over the standard LTAG parsing methods.

1. The problem

We assume the reader is familiar with TAGs, LTAGs and related notions (Joshi, 1987; Joshi & Schabes, 1992). Each node n in an elementary tree is associated with a selectional constraint $Adj(n)$ representing a (possibly empty) set of elementary trees that can be adjoined at n (we treat substitution as adjunction at a childless node). We define the size of n as $1 + |Adj(n)|$. The size of an LTAG G, written $|G|$, is the sum of the sizes of all nodes in the elementary trees of G.

Standard parsing algorithms for TAGs have running time $\mathcal{O}(|G|\,|w|^6)$, where G is the input grammar and w is the input string. As already mentioned in the introduction, LTAGs allow more selective parsing strategies, resulting in $\mathcal{O}(f(G,w)\,|w|^6)$ running time, for some function $f(G,w)$ that is independent of the size of the vocabulary treated by G (hence typically much less than $|G|$). In order to give an estimate of the factor $f(G,w)$, let us define t as the maximum number of nodes in an elementary tree (of G), and g as the maximum number of elementary trees that are anchored in a common lexical item. We argue below that $f(G,w)$ is $\mathcal{O}(tg^2\,|w|^2)$.

We need to introduce some additional notation. We write $w_{i,j}$ to denote the substring of w from position i to position j, $0 \leq i \leq j \leq |w|$. (Position i is the boundary between the i-th and the $(i+1)$-th symbols of w.) We write w_i for $w_{i-1,i}$. In the grammar, assume some arbitrary ordering for the elementary trees with a given anchor and for the nodes of each elementary tree, with the root node always being the first. Then $\langle h,k\rangle$ denotes the k-th elementary tree anchored at w_h, $\langle h,k,1\rangle$ denotes its root node, and $\langle h,k,m\rangle$ denotes its m-th node (for $1 \leq h \leq |w|$, $1 \leq k \leq g$, $1 \leq m \leq t$).

By "tree" we now mean an elementary or derived tree that may contain a foot-node. The most time-expensive step in TAG and LTAG tabular parsing is the recognition of adjunction at nodes dominating a foot-node. Say that we have constructed a subtree that is rooted at the node $\langle h,k,m\rangle$, which may be an *internal* node of some elementary tree, and covers substrings $w_{i,p}$ and $w_{q,j}$. Say also that we have constructed a complete tree β rooted at $\langle h',k',1\rangle$, covering substrings $w_{i',i}$ and $w_{j,j'}$. In a tabular method these two analyses can be represented, respectively, by the items $[\langle h,k,m\rangle,i,p,q,j]$ and $[\langle h',k',1\rangle_{\mathsf{T}},i',i,j,j']$.[1] In items, the subscript T on a node indicates that no further adjunction is allowed to take place there (i.e., adjunction has already occurred or has been explicitly declined). Adjunction of β at the node $\langle h,k,m\rangle$ is then carried out as illustrated by the following abstract inference rule (see for instance (Vijay-Shanker &

[1] Top-down tabular algorithms, and those that enforce the valid-prefix property, might use more indices in item representations, in addition to those shown in our example. In some cases this may damage the asymptotic runtime.

Joshi, 1985; Vijay-Shanker & Weir, 1993)):

$$\frac{[\langle h,k,m\rangle, i, p, q, j] \quad [\langle h',k',1\rangle_\top, i', i, j, j']}{[\langle h,k,m\rangle_\top, i', p, q, j']} \; \langle h',k'\rangle \in Adj(\langle h,k,m\rangle) \tag{1}$$

Item $[\langle h,k,m\rangle_\top, i', p, q, j']$ represents a new partial analysis spanning $w_{i',p}$ and $w_{q,j'}$; no further adjunction is possible at node $\langle h,k,m\rangle$ in this analysis.

In order to bound $f(G,w)$, let us fix positions i', i, p, q, j and j'. Then step (1) can be executed a number of times bounded by $((i-i')+(j'-j))(p+|w|-q)tg^2$. This is because $w_{h'}$ can freely range within $w_{i',i}$ or $w_{j,j'}$, w_h can freely range within $w_{0,p}$ or $w_{q,|w|}$, since the anchor w_h of tree $\langle h,k\rangle$ might not be dominated by node $\langle h,k,m\rangle$; also, k, k' and m can assume any values within their respective ranges. We therefore conclude that $f(G,w) = \mathcal{O}(tg^2\,|w|^2)$.

Note that a better upper bound would be given by $\mathcal{O}(tg^2 \min\{|V_\mathrm{T}|^2, |w|^2\})$, V_T the terminal alphabet (vocabulary) of G, since each anchor can assume no more than $|V_\mathrm{T}|$ different values. However, in practical applications we have $|w| \ll |V_\mathrm{T}|$, and therefore in this paper we will always use the former bound. We then conclude that standard LTAG parsing algorithms run with a worst case time of $\mathcal{O}(tg^2\,|w|^8)$.

2. A novel algorithm

This section improves upon the time upper bound reported in §1. The result is achieved by splitting step (1) into three substeps. (A similar method may be applied to speed up parsing of lexicalized context-free grammars (Eisner & Satta, 1999).)

We start by observing that at step (1) we simultaneously carry out two tests on the trees under analysis:

- we check that the tree $\langle h',k'\rangle$ is found in the selectional constraint $Adj(\langle h,k,m\rangle)$; and
- we check that the tree yield $w_{i',i}$, $w_{j,j'}$ "wraps around" the tree yield $w_{i,p}$, $w_{q,j}$, i.e., that the two copies of i match and likewise j.

To some extent, the two computations above can be carried out independently of each other. More precisely, the result of the check on the selectional constraint does not depend on the value of positions p and q. Furthermore, once the check has been carried out, we can do away with the anchor position h', since this information is not used by the wrapping test or mentioned in the result of step (1).

In order to implement the above idea, we define two new kinds of items, which we write as $[\![\langle h,k,m\rangle, i, j]\!]$ and $[\![\langle h,k,m\rangle_\top, i', i, j, j']\!]$. Item $[\![\langle h,k,m\rangle, i, j]\!]$ packages together all items of the form $[\langle h,k,m\rangle, i, u, v, j]$. Similarly, item $[\![\langle h,k,m\rangle_\top, i', i, j, j']\!]$ packages together all items of the form $[\langle h',k',1\rangle_\top, i', i, j, j']$ such that $\langle h',k'\rangle \in Adj(\langle h,k,m\rangle)$. We can then replace step (1) with the following three steps:

$$\frac{[\langle h,k,m\rangle, i, p, q, j]}{[\![\langle h,k,m\rangle, i, j]\!]} \tag{2}$$

$$\frac{[\![\langle h,k,m\rangle, i, j]\!] \quad [\langle h',k',1\rangle_\top, i', i, j, j']}{[\![\langle h,k,m\rangle_\top, i', i, j, j']\!]} \; \langle h',k'\rangle \in Adj(\langle h,k,m\rangle) \tag{3}$$

$$\frac{[\langle h,k,m\rangle_{\top}, i', i, j, j'] \quad [\langle h,k,m\rangle, i, p, q, j]}{[\langle h,k,m\rangle_{\top}, i', p, q, j']} \tag{4}$$

A computational analysis similar to the one carried out in §1 shows the following overall time costs: step (2) takes time $\mathcal{O}(tg\,|w|^5)$, step (3) takes time $\mathcal{O}(tg^2\,|w|^6)$ and step (4) takes time $\mathcal{O}(tg\,|w|^7)$. Thus the overall time cost for all the above steps is $\mathcal{O}(tg\,|w|^6 \max\{g, |w|\})$.

All the remaining steps in standard LTAG tabular parsing algorithms that have not been considered here can easily be accommodated within the indicated upper bound. Thus, steps (2) to (4) can be integrated into a standard LTAG parser, providing a new parsing algorithm for LTAG with worst case running time $\mathcal{O}(tg\,|w|^6 \max\{g, |w|\})$.

3. Discussion

We have discussed standard LTAG, in which every elementary tree has exactly one lexical anchor. Multiply anchored trees can be handled straightforwardly and without additional cost: for the analysis, simply consider one anchor to be primary when defining the grammar constant g and when naming the tree $\langle h, k\rangle$. The parse table should be seeded with all of a tree's anchor nodes if and only if all those anchor words appear in the input w in the correct order. (Recall that it was always possible to construct subtrees over substrings that do not include the primary anchor.)

Our inference rules enforce the traditional prohibition against multiple adjunctions at the same node (Vijay-Shanker & Joshi, 1985). This prohibition has been questioned on linguistic grounds (Schabes & Shieber, 1994), since for example a verb may need to select lexically for each of its multiple PP adjuncts. To relax the prohibition it is sufficient to drop the symbol $\top$ throughout the rules.

Our algorithm is an asymptotic improvement for any values of g, t, and $|w|$. However, we really have in mind grammars where g is a smallish constant, much smaller than the vocabulary size. In particular, we do not expect a word to anchor multiple elementary trees that have the same labeled internal structure as one another, differing only in their selectional constraints. Thus, the selectional constraints at each node in an elementary tree only depend on the tree's head and the internal structure of the tree itself. Grammars satisfying this requirement have been called node-dependent or SLG(2) in (Carroll & Weir, 1997), and bilexical in (Eisner, 1997; Eisner & Satta, 1999; Eisner, 2000). If we drop the above assumption, the grammar can capture lexical relations of arity larger than two. For instance, in an LTAG which is not bilexical, a verb V_1 could anchor many instances of the basic transitive-sentence elementary tree, in each of which the selectional constraint at the object node required a specific object tree (with a specific head). In this case, the selectional constraint at each V_1 tree's subject node would depend on both V_1 *and* its required object, thus establishing a relation between three lexical elements. Moreover, an upstairs verb V_0 could select for certain of these V_1 trees and thereby restrict both V_1 and the head of V_1's object, again establishing a relation between three lexical elements. This style of grammar can dramatically increase g as a function of the vocabulary size. To overcome this one would again have to substitute some factor that depends on the input string length.

Even in the bilexical grammars we expect, where g is unrelated to vocabulary size, g can still be somewhat large in broad-coverage grammars such as those cited in the introduction, which include large tree families for each word. The literature describes some further tricks for efficiency in this case. Similar trees in the same family may be made to share structure (Evans

& Weir, 1997; Carroll *et al.*, 1998). "Supertagging" techniques (Srinivas & Joshi, 1999; Chen *et al.*, 1999) use contextual probabilities to eliminate some elementary trees heuristically before parsing begins. Alternatively, under a stochastic LTAG (Resnik, 1992; Schabes, 1992), one may prune away unpromising items, such as those with low inside probability. It should be possible to combine any of these tricks with our technique.

References

ABEILLÉ A. & CANDITO M.-H. (2000). FTAG: A lexicalized tree-adjoining grammar for french. In A. ABEILLÉ & O. RAMBOW, Eds., *Tree-Adjoining Grammar*. Stanford, CA: CSLI Lecture Notes. To appear.

ALSHAWI H. (1996). Head automata and bilingual tiling: Translation with minimal representations. In *Proc. of the 34th ACL*, p. 167–176, Santa Cruz, CA.

CARROLL J., NICOLOV N., SHAUMYAN O., SMETS M. & WEIR D. (1998). Grammar compaction and computation sharing in automaton-based parsing. In *Proceedings of the 1st Workshop on Tabulation in Parsing and Deduction (TAPD'98)*, p. 16–25, Paris, France.

CARROLL J. & WEIR D. (1997). Encoding frequency information in lexicalized grammars. In *Proceedings of the 5th Int. Workshop on Parsing Technologies*, MIT, Cambridge, MA.

CHARNIAK E. (1997). Statistical parsing with a context-free grammar and word statistics. In *Proc. of AAAI-97*, Menlo Park, CA.

CHEN J., SRINIVAS B. & VIJAY-SHANKER K. (1999). New models for improving supertag disambiguation. In *Proc. of the 9th EACL*, p. 188–195, Bergen, Norway.

COLLINS M. (1997). Three generative, lexicalised models for statistical parsing. In *Proc. of the 35th ACL*, Madrid, Spain.

DORAN C., HOCKEY B., SARKAR A., SRINIVAS B. & XIA F. (2000). Evolution of the XTAG system. In A. ABEILLÉ & O. RAMBOW, Eds., *Tree-Adjoining Grammar*. Stanford, CA: CSLI Lecture Notes. To appear.

EISNER J. (1996). Three new probabilistic models for dependency parsing: An exploration. In *Proc. of the 16th COLING*, p. 340–345, Copenhagen, Denmark.

EISNER J. (1997). Bilexical grammars and a cubic-time probabilistic parser. In *Proceedings of the 5th Int. Workshop on Parsing Technologies*, MIT, Cambridge, MA.

EISNER J. (2000). Bilexical grammars and a cubic-time probabilistic parser. In H. C. BUNT & A. NIJHOLT, Eds., *New Developments in Natural Language Parsing*. Kluwer.

EISNER J. & SATTA G. (1999). Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proc. of the 37th ACL*, p. 457–464, College Park, Maryland.

EVANS R. & WEIR D. (1997). Automata-based parsing for lexicalized grammars. In *Proceedings of the 5th Int. Workshop on Parsing Technologies*, MIT, Cambridge, MA.

JOSHI A. K. (1987). An introduction to tree adjoining grammars. In A. MANASTER-RAMER, Ed., *Mathematics of Language*. Amsterdam: John Benjamins.

JOSHI A. K. & SCHABES Y. (1992). Tree adjoining grammars and lexicalized grammars. In M. NIVAT & A. PODELSKY, Eds., *Tree Automata and Languages*. Amsterdam, The Netherlands: Elsevier.

RESNIK P. (1992). Probabilistic tree-adjoining grammar as a framework for statistical natural language processing. In *Proc. of the 14th COLING*, p. 418–424, Nantes, France.

SCHABES Y. (1992). Stochastic lexicalized tree-adjoining grammars. In *Proc. of the 14th COLING*, p. 426–432, Nantes, France.

SCHABES Y., ABEILLÉ A. & JOSHI A. K. (1988). Parsing strategies with 'lexicalized' grammars: Application to tree adjoining grammars. In *Proc. of the 12th COLING*, p. 578–583, Budapest, Hungary.

SCHABES Y. & SHIEBER S. M. (1994). An alternative conception of tree-adjoining derivation. *Computational Linguistics*, **20** (1), p. 91–118.

SLEATOR D. & TEMPERLEY D. (1991). *Parsing English with a Link Grammar.* Computer Science Technical Report CMU-CS-91-196, Carnegie Mellon University.

SRINIVAS B. & JOSHI A. (1999). Supertagging: An approach to almost parsing. *Computational Linguistics*, **20** (3), p. 331–378.

VIJAY-SHANKER K. & JOSHI A. K. (1985). Some computational properties of tree adjoining grammars. In 23rd *Meeting of the Association for Computational Linguistics*, p. 82–93, Chicago, Illinois.

VIJAY-SHANKER K. & WEIR D. J. (1993). Parsing some constrained grammar formalisms. *Computational Linguistics*, **19** (4), p. 591–636.

Workshop TAG+5, Paris, 25-27 May 2000

Economy in TAG*

Robert Frank

Department of Cognitive Science
Johns Hopkins University
3400 N. Charles Street
Baltimore, MD 21218 USA

1. On the role of economy in grammatical derivations

Much recent work within generative grammar has made use of the idea that grammatical derivations exhibit a certain type of economy. The intuition behind this application of economy is a familiar one: that the well-formed sentences of a language are as simple as they can be (given the demands of expressiveness), and do not involve any unnecessary lexical items or dislocations. There have been a variety of formalizations of the relevant notion of economy, with a range of empirical consequence. Let us look at one of these, proposed by Chomsky (1995, ch.4) to account for the contrast between the examples in (1).

(1) a. There seems [*t* to be [a unicorn in the garden]]
b. * There seems [a unicorn to be [*t* in the garden]]

From a certain perspective, the derivations of both of these sentences are equally complex: both involve a single instance of syntactic movement. In (1a), it is *there* which raises from the subject of the infinitival to the subject of *seems*. In (1b), *a unicorn* undergoes raising, from within the small clause to the subject position of the infinitival clause. Why, then, should (1b) be blocked? Chomsky adopts a derivational model in which phrase structure is built in a bottom-up fashion. In such a model, the derivation of both examples in (1) will begin by constructing the following representation:

(2) $[_{\overline{T}}$ to be [a unicorn in the garden]]

Chomsky assumes that every T(ense) head (for example, *to*) has a feature that must be checked during the derivation by the insertion of a DP subject in its specifier position, an instantiation of the Extended Projection Principle (EPP). At the point in the derivation depicted in (2), then, some element must be inserted into the specifier of TP position. Under the assumption that merging a new lexical item into a structure is a simpler operation that syntactic movement, Chomsky formulates the following principle of derivational economy:

(3) Prefer Merge over Move

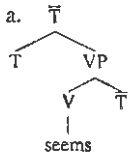
By (3), we are forced to merge *there* into the specifier of the TP in (2), rather than moving *a unicorn*. When we reach the matrix clause, however, the fact that no additional lexical items remain to merge forces us to employ the more costly move operation. (Note that the presence or absence of *there* in these examples is, for Chomsky, determined prior to the onset of the derivation. Further, on Chomsky's theory structures with distinct numerations are not compared for economy. See Chomsky (1995; 1998) for further discussion.)

* Thanks to Colin Wilson, and Paul Hagstrom for helpful discussion, to two anonymous reviewers for comments, and to the National Science Foundation for their monetary support in the form of grant SBR-9710247.

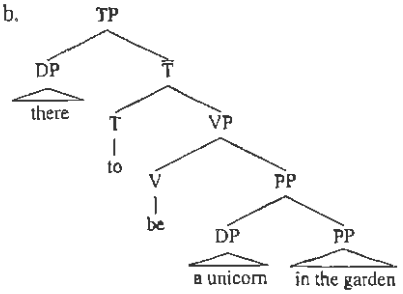
2. Eliminating the need for economy with TAG

What becomes of the contrast in (1) in a TAG context? Under the assumptions of Frank (1992; to appear) concerning elementary trees, example (1a) derives from the adjoining of the *seems*-headed auxiliary in (4a) to the $\overline{T}$ node of the tree in (4b).

(4) a.

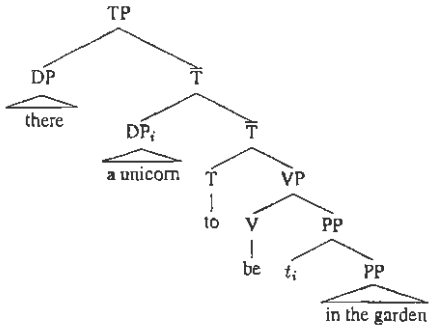


b.



What about the example in (1b), then? On analogy with the derivation of (1a), we might derive (1b) by adjoining (4a) into the initial tree in (5) at the higher $\overline{T}$ node.

(5)



The ill-formedness of this example would then derive from the impossibility of elementary trees like (5), which I take to derive from the absence in English of so-called transitive expletive constructions (TECs), in which both an expletive and lexical DP appear in VP-external subject positions.

(6) a. * There a cat has eaten the mice.

b. * There has a cat eaten 3 mice.

This analysis makes the immediate prediction that a language that permits TECs, and therefore elementary structures like (5), ought to permit examples like (1b). This prediction is confirmed in Icelandic. As seen in (7), Icelandic permits both transitive expletive constructions and the partial raising construction (examples from Bobaljik and Thráinsson (1998) and Jonas (1996)).

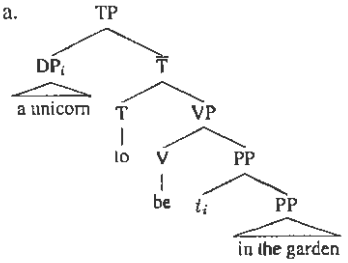
(7) a. Það hefur einhver köttur étið mýsnar
there has some cat eaten mice-the
'A cat has eaten mice.'

b. Það virðast margir menn vera í herberginu
there seem many men be-inf in the room

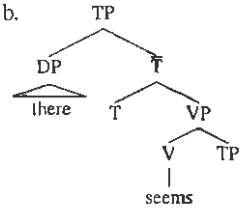
3. Expletives and the return of economy

It seems then that by using TAG we are able to explain the contrast motivating Chomsky's principle of derivational economy in (3) without resort to any such principle. This constitutes another case in which the use of TAG allows us to eliminate otherwise needed stipulations from the grammar. There remains a hole in this line of argument, however, as there exists an alterative derivation for the example (1b) that we have not yet ruled out. This derivation involves the combination (either by substitution or adjoining) of the elementary tree in (8a) with the *seems*-headed tree in (8b).

(8) a.



b.

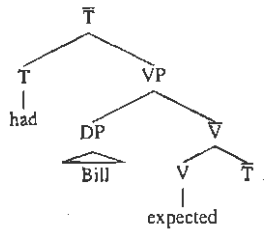


Clearly, there is nothing wrong with the elementary tree in (8a), as we take this tree to participate in the derivation of well-formed examples like the following:

(9) A unicorn seems to be in the garden.

The culprit, therefore, must be the elementary tree in (8b). What then is wrong this tree?
To answer this question, we must first face the issue of what licenses the presence of *there* within an elementary tree. For Chomsky, the insertion of *there* is driven by the need to check the EPP feature of T, which guarantees the insertion of a specifier. The proposal that T always demands a specifier is not easily incorporable into a TAG context, at least not as a constraint on elementary trees: otherwise we would exclude trees like (4a) whose T heads lack specifiers. Nonetheless, there are situations in which we will need to invoke some form of the EPP to constrain elementary trees. For example, we will want to prevent the possibility of an auxiliary tree like the following, in which the subject *Bill* has not raised to the specifier of TP position:

(10)



Such an auxiliary tree, if allowed in the grammar, could adjoin into a TP infinitival elementary tree like (8a), just as a raising auxiliary like (4a) would. In this case, however, the result would be anomalous:

(11) * A unicorn had Bill expected to be in the garden
(meaning 'Bill had expected a unicorn to be in the garden')

I suggest that TAG elementary trees are in fact subject to an EPP requirement along the lines that Chomsky suggests. That is, I assume that elementary trees are constructed in a derivational

process along the lines proposed by Chomsky, but one which is restricted in the size of the structures that it may construct. Every T head that occurs in such a derivation will include an EPP feature that can be checked only by a DP in its specifier position. However, while Chomsky assumes that all such EPP features must be checked at the conclusion of the derivation, I assume that the checking of such features is subject to the following economy condition that constrains the process of elementary tree formation:

(12) **Maximal Checking Principle (MCP)**: Check as many features (i.e., satisfy as many grammatical requirements) as possible within an elementary tree.

The MCP renders violable within an elementary tree domain the requirement that features that need to be checked, if there is no way for them to be satisfied within an elementary tree.[1] This means that the unchecked EPP feature in the tree in (10) is fatal since there is an element within the elementary tree, the DP *Bill*, that could be raised to check this feature. The elementary tree in (10) is therefore blocked by the alternative elementary tree in which the subject is raised to specifier of TP.

Under the MCP, what becomes of the elementary tree in (4a)? One might reasonably expect that this tree would be blocked by the tree in (8b), since the latter lacks an unchecked EPP feature (having been checked by the insertion of *there*).[2] I suggest, however, that the set of elementary trees that are compared for the purposes of the MCP is restricted to those that are constructed from the same set of lexical resources, or numeration in Chomsky's terms. In the TAG context, I assume that a numeration will also include the non-projected non-terminals that become the foot nodes of auxiliary trees and sites for substitution. Since the elementary trees in (4a) and (8b) are derived, respectively, from the distinct pair of numerations given below, the MCP does not choose between these trees.

(13) a. { T, seems, $\overline{\text{T}}$ }

b. { there, T, seems, TP }

This leaves us in the position of correctly allowing the tree in (4a), but incorrectly allowing (8b) as well. To rule out the latter tree, I assume that feature checking in elementary trees abides by the following principle:

(14) **All or Nothing Checking Regimen (ANCR)**: In an elementary tree, if some of the features of head are checked, they must all be checked.

I assume that T possesses not only its EPP feature, but also contains agreement features that must be checked. Thematic subjects in specifier of TP will typically check both of these features, satisfying the ANCR. Since *there* does not determine agreement, as seen in (15), I will assume that its insertion into specifier of TP does not suffice to check T's agreement features.

(15) a. There is a unicorn in the garden.

b. There are three unicorns in the garden.

[1] I maintain Chomsky's original intuition that all uninterpretable features must eventually be checked, though the relevant point here is the conclusion of the TAG derivation. To ensure this, we will translate all features that remain unchecked within an elementary tree into constraints on adjoining. One can do this in terms of the unification-based system of adjoining constraints of Vijay-Shanker (1988), though alternatives are possible that more directly link up with the feature checking machinery discussed here. See Frank (to appear, ch.4) for more discussion.

[2] For the purposes of simplicity, I assume that expletives can be present in a verbally-headed elementary tree, without inducing a violation of the Condition of Elementary Tree Minimality (Frank, 1992; Frank, to appear). Alternatively, we can assume the presence of a DP frontier node containing features that restrict substitution to expletive-headed DPs.

As a result, after insertion of *there*, only the EPP features of the T head in the elementary tree in (8b) are checked. Since there are no lexical DPs within this elementary tree that could check the agreement features of T, as occurs with the post-copular DPs in (15), the agreement features will necessarily remain unchecked in this elementary tree, leading to a violation of the ANCR. An anonymous review suggests that the ill-formedness of (8b) receives a simpler explanation under a constraint I gave in Frank (1992) that was called, perhaps misleadingly, the Projection Principle:

(16) If α is a non-terminal which appears along the frontier of an elementary tree τ, then α is part of a chain whose tail is selected in τ, either through theta role assignment or predication.

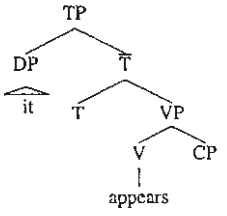
Under this constraint, T cannot project past $\overline{\text{T}}$ in an elementary tree headed by a raising predicate because there is no thematic role or predication relation that could be assigned to (the chain) of an element in the specifier of TP position. While the lack thematic role is clear enough, it is less clear that there is no licensing predication relation. In Frank (1992), I discussed two instances of predication relations, the first between a modifier and the XP foot node of its elementary tree, irrelevant to current concerns, and the second between a T head and an expletive subject. This was meant to allow for the presence of expletive *it* in subject position in constructions like the following:

(17) a. It appears that Gabriel has finally fallen asleep.

b. Il a été tiré sur la bateau
it has been fired upon the boat
'The boat was fired upon.'

To generate (17a), we will need an elementary of something like the following form:

(18)



This tree is strikingly similar to the illicit one in (8b), and differs only in the content of the expletive. Since the projection principle in (16) imposes no restriction on the content of elements that can enter into a predication relation, and indeed there seems no principled reason for assuming that *there* cannot enter in a predication relation with T, it leaves unexplained the contrast between (8b) and (18).

The ANCR, in contrast, allows us to explain why the elementary tree in (18) is well-formed. To see how, observe first that *it*, unlike *there*, systematically induces third person singular agreement on the verb, even in the face of a clausal conjunct that could induce plural agreement on the verb when in subject position (McCloskey, 1992).

(19) a. It seems/*seem equally likely at this point that the president will be reelected and that he will be impeached.

b. That the president will be reelected and that he will be impeached seem equally likely at this point.

Consequently, we will assume that *it*, unlike *there* is able to check T's agreement features. As a result, T's EPP and agreement features are both checked in the elementary tree in (18), with no ANCR violation.[3]

If the preceding discussion is correct, we must assume that dative experiencers, as occur in raising examples like (20), are incapable of moving to subject position to check the EPP feature of T.

(20) A unicorn seems [to Gabriel] to be in the garden.

If such movement were possible, the presence of a dative in a *seem*-headed elementary tree would affect the potential satisfaction of T's EPP features (putting aside for the moment questions about checking of agreement features and the ANCR). And as a result, the MCP would rule out an auxiliary tree in which this dative was not raised to specifier of TP position, effectively blocking raising past experiencer arguments as in (20). In a language in which datives could move to subject position, checking EPP and agreement features, we would expect to find just this pattern, where raising without experiencers is grammatical, but raising across experiences, as in (20), is impossible. In fact, this is exactly what is observed in Icelandic (Sigurðsson, 1996).[4] It has been convincingly demonstrated that Icelandic allows dative arguments to surface in subject position (see, among others, Zaenen *et al.* (1985)).

(21) Strákunum leiddist
the boys-dat bored-3sg
'The boys were bored.'

As seen in (22), Icelandic allows raising when the raising verb has no experiencer argument.

(22) Margir menn virðast vera í herberginu
many men seem-3pl to be in the room

However, when the raising verb projects an experiencer, such raising is impossible, with the grammatical form having the experiencer in subject position.[5]

(23) a. * Margir menn virðast mér vera í herberginu
many men seem-3pl to me be-inf in the room

b. Mér virðast margir menn vera í herberginu
to me seem-3pl many men be-inf in the room

4. Further implications of the MCP: superiority effects

The effects of the MCP can also be observed in the context of wh-movement. Let us assume that wh-movement is driven by a wh-feature in the C head to whose specifier movement takes place. This means that in the standard TAG derivation of examples like (24), the auxiliary tree representing the matrix clause will contain an C head with an unchecked wh-feature.

[3]Though space considerations prevent me from demonstrating this here, the ANCR has a number of consequences, allowing us to predict the differing distributions of *it* and *there*, as well as deriving Burzio's generalization that the possibility of structural case assignment by a verb implies the existence of an external argument (Burzio, 1986). See Frank (to appear, ch.4) for details.

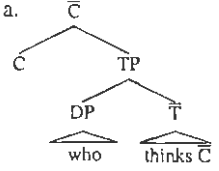
[4]See also Boeckx (1999) for extensions to Romance.

[5]For reasons of space, I omit discussion of how the matrix T's agreement features are checked on the DP in the lower subject position, and how the elementary tree with the dative experiencer subject satisfies the ANCR. In brief, I assume that T enters into an agreement relation with the dative subject and also (at least optionally) with the raising verb's TP complement, into which the agreement features of the embedded nominative subject have percolated. Evidence in favor of this view comes from the optionality of such agreement, and the locality conditions on such agreement. See Frank (to appear, ch.4) for extended discussion.

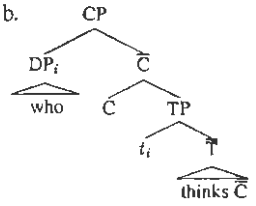
(24) Which song did Daniel think that Gabriel was playing?

As before, the presence of this unchecked feature, per se, is not problematic for the well-formedness of this elementary tree, since there is no element within this tree capable of checking the feature. If however such an elementary tree included a wh-phrase capable of checking this feature, the MCP would rule out any elementary tree in which the C feature remains unchecked, for example (17a), in favor of one where it is checked, as in (17b).

(25) a.



b.



This leads us to predict the impossibility of long-distance extraction of a wh-element into the specifier of CP of a clause which itself contains a wh-phrase. Such extractions are, in fact, impossible, as shown in the following English and German examples (the latter from Heck and Müller (2000)):

(26) a. * Which song does who think that Gabriel was playing?

b. Who thinks that Gabriel was playing which song?

(27) a. * Wen hat wer gesagt, daß Maria liebt?
whom has who said that Maria loves

b. Wer hat gesagt, daß Maria wen liebt?
who has said that Maria whom loves

This explanation does not extend to local "superiority" cases, in which one wh-phrase moves across another within a single clause, as the MCP does not dictate which element must move when there are two local possibilities. Consequently, all else being equal, we would expect that such cases to be well-formed, an expectation that is borne out for German:[6]

~~(28) a. Wen hat wer getroffen?~~
whom has who met

b. * Which song was who playing?

As seen in (28b), however, even these local cases are ill-formed in English. This does not falsify the MCP, but merely renders its effects untestable. One might fear that there is redundancy between the principle responsible for the ill-formedness of (28b) and that underlying the ill-formedness of (26a). However, there is evidence that these are distinct. As noted originally by Baker (1970), local superiority violations are obviated in multiple wh-questions so long as the in-situ wh-phrase, *who* in the example below, is interpreted in a higher clause.

(29) Q: Who asked which song who was playing?
A: Alice asked which song Gabriel was playing/* Alice did.

The effect of this higher interpretation is that both the matrix and subordinate occurrences of *who* must be answered. This avoidance of superiority effects is not possible, however, when the superiority violation is of the long-distance sort governed by the MCP. Thus, the following example is not possible, regardless of the scopal interpretation of the in-situ wh-phrase *who*.

[6]The German pattern of well-formed local superiority, and ill-formed long-distance superiority is replicated in Serbo-Croatian (Richards, 1997, p.32).

(30) * Who asked which song who thought that Gabriel was playing?

5. Conclusions

I take the range of data discussed here to provide substantial support for the role of economy in determining the well-formedness of TAG elementary trees, particularly in the form of the MCP and ANCR. The fact that these economy principles apply to TAG elementary trees enforces a certain locality on the process of determining which structures are most economical. Such a local notion of economy has in fact been proposed by a number of authors including Collins (1997) and Chomsky (1999) on rather different empirical grounds. I would like to suggest that we are seeing a convergence to the idea, familiar from work in the TAG tradition, that syntactic structure is composed from non-recursive structural elements whose well-formedness is independently determined.

References

BAKER C. L. (1970). Notes on the description of English questions: the role of an abstract question morpheme. *Foundations of Language*, **6**.

BOBALJIK J. D. & THRÁINSSON H. (1998). Two heads aren't always better than one. *Syntax*, **1** (1 p.), 37–71.

BOECKX C. (1999). Raising in Romance. In *Chomsky Celebration Website*. MIT Press. Available at http://mitpress.mit.edu/chomskydisc/Boeckx2.html.

BURZIO L. (1986). *Italian Syntax*. Dordrecht: D. Reidel.

CHOMSKY N. (1995). *The Minimalist Program*. Cambridge, MA: MIT Press.

CHOMSKY N. (1998). Minimalist inquiries: the framework. MIT Occasional Papers in Linguistics 15, Department of Linguistics and Philosophy, MIT.

CHOMSKY N. (1999). Derivation by phase. MIT Occasional Papers in Linguistics 18, Department of Linguistics and Philosophy, MIT.

COLLINS C. (1997). *Local Economy*. Cambridge, MA: MIT Press.

FRANK R. (1992). *Syntactic Locality and Tree Adjoining Grammar: Grammatical, Acquisition and Processing Perspectives*. PhD thesis, University of Pennsylvania, Philadelphia, PA.

FRANK R. (to appear). *Structural Composition and Syntactic Dependencies*. Cambridge, MA: MIT Press.

HECK F. & MÜLLER G. (2000). Repair-driven movement and the local optimization of derivations. Talk presented at Department of Cognitive Science, Johns Hopkins University.

JONAS D. (1996). Clause structure, expletives and verb movement. In W. ABRAHAM, S. D. EPSTEIN, H. THRÁINSSON & J.-W. ZWART, Eds., *Minimal Ideas*, p. 167–188. Amsterdam: John Benjamins.

MCCLOSKEY J. (1992). *There, it*, and agreement. *Linguistic Inquiry*, **22** (3 p.), 563–567.

RICHARDS N. (1997). *What Moves Where When in Which Language*. PhD thesis, MIT, Cambridge, MA.

SIGURÐSSON H. A. (1996). Icelandic finite verb agreement. *Working Papers in Scandinavian Syntax*, **57**, 1–46.

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

ZAENEN A., MALING J. & THRÁINSSON H. (1985). Case and grammatical functions: the Icelandic passive. *Natural Language and Linguistic Theory*, **3**, 441–483.

The Sino-Korean Light Verb Construction and Lexical Argument Structure

Chung-hye Han† and Owen Rambow‡

†IRCS, University of Pennsylvania
3401 Walnut St., Suite 400
Philadelphia, PA 19104
chunghye@linc.cis.upenn.edu

‡ATT Labs-Research, B233
180 Park Ave., PO Box 971
Florham Park, NJ 07932
rambow@research.att.com

Abstract

In Korean, a class of lexemes of Chinese origin exhibit both nominal and verbal behavior. Specifically, they can assign lexically idiosyncratic case, but require a semantically vacuous light verb in order to form a sentence and are themselves marked with accusative case. In this paper, we propose a TAG-based account of this behavior, and propose some generalizations towards a pure representation of lexical argument structure.

1. Linguistic Facts and Issues

In this paper, we provide a syntactic analysis of Sino-Korean light verb constructions (LVC henceforth) that are composed of the light verb *ha* and an activity-denoting noun of Chinese origin.[1] We will refer to this activity-denoting noun as the 'base' of the LVC. The argument structure of LVCs come from the base, and the light verb is semantically vacuous and does not assign any theta roles. This is shown by the fact that although the examples in (1) all contain *ha*, they have different argument structures.

(1) a. John-i swuhak-ul yenkwu-lul ha-yess-ta.
John-Nom math-Acc research-Acc HA-Past-Decl
'John researched math.'

b. Kicha-ka Seoulyek-ey tochak-ul ha-yess-ta.
train-Nom Seoul-station-at arrival-Acc HA-Past-Decl
'The train arrived at Seoul station.'

c. Kicha-ka Seoulyek-eyse chwulpal-ul ha-yess-ta.
train-Nom Seoul-station-from departure-Acc HA-Past-Decl
'The departed from Seoul station.'

For instance, the arguments in (1a) are agent and goal, those in (1b) are patient and goal, and those in (1c) are patient and source.

If, however, the theta roles in LVCs are assigned by the base, it is puzzling why the argument NPs are syntactically realized outside of the base NP. The case postpositions such as *Acc*, *-ey* and *-eyse* on the argument NPs indicate that they are daughters of VP, and not the base NP. An

[1]Han has been partially funded by the Army Research Lab via a subcontract from CoGenTex, Inc., and by NSF Grant SBR 8920230. We would like to thank Aravind Joshi, Tony Kroch, Martha Palmer, and Anoop Sarkar for useful discussions.

NP that is a daughter of another NP requires genitive or null case postposition in Korean. We will refer to the first kind of case as VERBAL CASE, and the second as NOMINAL CASE. Moreover, as noted by (Grimshaw & Mester, 1988), there are restrictions on argument realization, which can be clearly shown with ditransitive LVCs, as in (2).

(2) a. John-i Mary-eykey inhyung-ul senmwul-ul ha-yess-ta.
John-Nom Mary-to doll-Acc gift-Acc HA-Past-Decl
'John gave a gift of a doll to Mary.'

b. John-i Mary-eykey inhyung(-uy) senmwul-ul ha-yess-ta.
John-Nom Mary-to doll(-Gen) gift-Acc HA-Past-Decl

c. * John-i inhyung-ul Mary-eykey-uy senmwul-ul ha-yess-ta.
John-Nom doll-Acc Mary-to-Gen gift-Acc HA-Past-Decl

d. * John-i Mary-eykey-uy inhyung(-Gen) senmwul-ul ha-yess-ta.
John-Nom Mary-to-Gen doll(-Gen) gift-Acc HA-Past-Decl

The base *senmwul* ('gift') assigns agent, goal and theme. In (2a), all the argument NPs are realized outside of the base NP. In (2b), the agent and goal arguments are realized outside of the base NP, but the theme argument is realized inside the base. However, it is not possible to realize theme argument outside of the base when the goal argument is realized inside the base, as shown in (2c), and it is not possible to realize both theme and goal arguments inside the base, as shown in (2d).

(Grimshaw & Mester, 1988) (G&M henceforth) summarize the restrictions on argument realization as follows: (i) the subject argument must always be outside the base NP; (ii) at least one argument apart from the subject must be outside the base NP; and (iii) for nouns that take a theme and a goal, if the theme argument is realized outside the base NP, the goal must also be realized outside the base NP. In what follows, we first briefly discuss some previous analyses and their shortcomings, and present our own analysis using the framework of Feature Based Lexicalized Tree Adjoining Grammar. We discuss English data in comparison, and conclude with a discussion of noun phrases.

2. Previous Analyses

According to G&M, a light verb such as *ha* has no argument structure on its own and it occurs with a noun which is 'theta-transparent.' Theta-transparent nouns can transfer some or all of their arguments to the argument structure of the light verb. This mechanism allows the light verb to directly assign theta roles to the argument NPs in syntax and such argument NPs are realized outside the base NP. They further assume (following much previous work) that arguments have a hierarchy according to prominence. For instance, the agent is more prominent than the goal, which is more prominent than the theme. Based on this assumption, they propose that when a theta role is transfered (e.g., the theme), any theta roles that are higher in prominence must transfer as well (i.e, the agent and goal). This explains the ungrammaticality of (2c). G&M also stipulate that the base noun must transfer at least one internal argument in order to be licensed. Otherwise, the theta-criterion is violated, since the base noun does not receive a theta role from anywhere. This is why (2d) is ungrammatical under G&M's system.

G&M wrongly predict that intransitive LVCs do not exist, since there is no internal argument to participate in the transfer. But intransitive LVCs clearly do exist, as shown in (3) and (4). Note that while (4) may be ambiguous between a heavy and light verb reading of *ha*, (3) is not, since the subject is not an agent.

(3) John-i samang-ul ha-yess-ta.
John-Nom death-Acc HA-Past-Decl
'John died.'

(4) John-i swuyuong-ul ha-yess-ta.
John-Nom swimming-Acc HA-Past-Decl
'John was swimming.'

For this reason, (Yoon, 1991) rejects G&M's argument transfer theory and proposes 'argument sharing' mechanism. He argues that the light verb is thematically underspecified and so unsaturated. This forces the base noun which has theta structure and the light verb to undergo the operation of Theta Identification, allowing the argument structure of the base noun and that of the light verb to be shared. This sharing is viewed as the unification of the argument structure of the base noun into the underspecified argument structure of the light verb. Yoon's theory predicts that when there are more than one internal arguments, they must all be realized outside of the base NP. But this is an incorrect prediction: in ditransitive LVCs, while the goal argument is realized outside the NP, the theme argument can be realized inside NP, as shown in (2b).
The same problem persists in (Park, 1992). He argues that the categorial status of the base is not a noun, but a verb. Thus, it assigns theta-roles just as any other verbs. The light verb is simply an auxiliary verb that supports inflection. But if the base is simply a verb, then (2b) is wrongly predicted to be ungrammatical.

3. TAG Analysis

The key to our analysis is the assumption that the base is underspecified with respect to word class (verb or noun). We propose that this base is the anchor of an elementary tree with all its arguments and that it acquires a noun status only after the light verb adjoins into the elementary tree. The assumption that the category of the base is unspecified is well-motivated for two reasons: (i) The base form originates from Chinese, in which the same form is used both as a noun and a verb, (ii) there is no consensus in the literature as to what the category of the base is and positing that it is either a noun or a verb leads to difficulties, as discussed in §2. We represent this by using the label X for its category (which projects to XP). We also assume that each node in a tree is associated with a category feature CAT with values such as V(ERB) and N(OUN). The CAT feature of nodes labeled V, VP, or S is necessarily V for both the top and bottom feature structures, while nodes labeled N or NP necessarily have [CAT:N].[2] But the CAT feature of the base of LVC is unspecified. In addition, we assume that nodes in a projection have a full set of morpho-syntactic features. In this paper we use only the binary feature [TENSED:]. We assume that the base is [TENSED:-] (since it carries no tense morphology), that the S node is marked [TENSED:+], and that the TENSED feature is shared among the nodes of a projection. We assume that when a lexeme (of any category) forms a syntactic predication structure it projects to a maximal verbal projection (VP) and we refer to this VP as the PREDICATE. Furthermore, following (Heycock & Lee, 1989), we assume that in Korean, nominative case is assigned by the predicate, not by Infl. (Heycock & Lee, 1989) use as evidence the presence of multiple nominative constructions and the fact that infinitivals can have nominative case-marked subject. As a result, all clausal structures need a VP node as a sister to the subject argument to license nominative case.[3] We also assume that the lexeme projects all of its argument positions in canonical order according to theta hierarchy. That is, the most prominent argument attaches

[2]The node labels are not actually used in our analysis, and we could also label all nodes XP. We retain the traditional labels for clarity.

[3]This is compatible with the XTAG analysis of the predicative use of nouns and adjectives in English, the trees for which project from N (or A) to S via NP (AP) and VP (though perhaps the VP is less motivated in English than in Korean because the adjoined auxiliary provides the nominative case in English, not the predication structure itself).

to the highest projection, and the least prominent attaches to the lowest projection.[4] We assume that each lexeme idiosyncratically fixes a case grid for its arguments,[5] which is only realized in appropriate syntactic contexts. (Thus, rather than speak of unified case assignment, we will henceforth speak of case assignment by the lexical head and subsequent *case realization* in a particular syntactic context.)

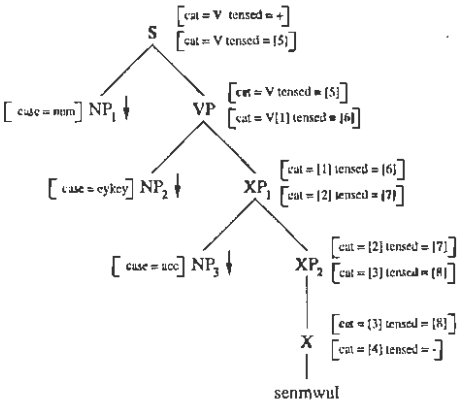


Figure 1: Sino-Korean base lexeme *senmwul* 'gift' projecting to a predicative structure

In Korean, a verbal case such as *Nom, Acc, eykey* is realized when the head has feature [CAT:V], while if the head has feature [CAT:N], *Nom* and *Acc* are realized as *Gen* or null, while any other postpositional case is realized as that postposition followed by *Gen*.[6] As an example, the elementary tree for the base *senmwul* 'gift' is shown in Figure 1, which is a ditransitive structure. Assuming the argument hierarchy agent – goal – theme, as in G&M, agent (as indicated by [case:nom]) is attached to S, goal (as indicated by [case:eykey]) is attached to VP and theme (as indicated by [case:acc]) is attached to XP_2. (The subscripts on nodes are used only for distinguishing different nodes, they play no role in the analysis.)

We now turn to the light verb. Its properties can be best explained in comparison to heavy *ha*. Heavy *ha* (Figure 2, left) is a standard transitive verb: it has two arguments (i.e., theta-marked dependents), to which it assigns nominative and accusative case, respectively. (Nominative case is realized in a syntactic predication environment, while accusative case is realized whenever the lexical head is verbal, which it is by assumption.) The light verb *ha* (Figure 2, right) differs from the heavy *ha* in that the light *ha* loses its ability to assign theta roles: it has no arguments of its own. Furthermore, it has lost its ability to create a predication structure. Thus it can no longer assign nominative case. It therefore does not project to a VP after taking its complement, but only to an XP, with [CAT:V]. However, light *ha* retains its ability to assign accusative case as well as the feature [CAT:N] to its complement. Since there is only one substitution node left, and since both root and substitution node are labeled XP, the tree is optionally an auxiliary tree (as is the case for English predicative auxiliary trees).

[4]We do not deal with the issue of optional arguments in this paper.

[5]Alternatively, we could assume each lexeme idiosyncratically chooses a set of theta-roles and then devise a functional mapping that derives the cases of a lexeme from the set of theta roles. Such an approach is only a notational variant of ours, and, as it has no additional content, we do not pursue it here.

[6]In other languages, the mapping between verbal and nominal case may not be as straightforward and each may be marked idiosyncratically from the head.

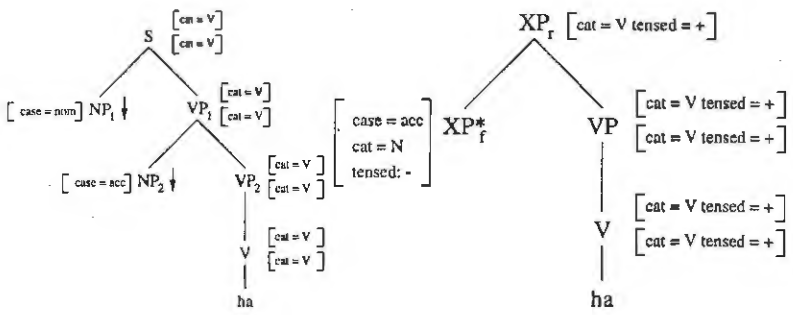


Figure 2: Heavy *ha* (left) and light *ha* (right)

The feature clash between [TENSED:-] on the base and [TENSED:+] at the root of the tree in Figure 1 forces the adjunction of light *ha*. We can adjoin this tree at either of the two XP nodes in the tree in Figure 1: if we adjoin at XP_1, the lowest argument NP_3 is realized with genitive/null case (as in (2b)); if we adjoin at XP_2, the lowest argument is realized with verbal case (as in (2a)). In both cases, NP_1 and NP_2 are realized with verbal cases. Our analysis predicts the pattern of data introduced in §1, while specifically avoiding the less appealing aspect of G&M's and Yoon's analyses, namely the cumbersome mechanisms of argument transfer or theta-identification, and the stipulation that agent and at least one internal argument must be transferred from the base. In fact, our analysis correctly predicts the existence of intransitive LVCs such as (3). The unique argument is the most prominent argument trivially and so it simply attaches to S, and receives nominative case from the predicate VP.[7]

4. Comparison to English Light Verb Constructions

Unlike in Korean, in English LVCs, all the internal arguments are realized within an NP. For instance, in a ditransitive light verb construction such as *make a donation*, the theme and goal arguments are nominal, as shown in (5). The theme *10,000 dollars* requires *of*, indicating that it is a sister of a noun, not a verb. Although it is rather difficult to tell whether the goal *to the charity* is nominal or verbal, it clearly has the possibility of being nominal, as shown in (6).

(5) a. * John made a donation 10,000 dollars to the charity last year.
b. John made a donation of 10,000 dollars to the charity last year.

(6) Twenty donations of 10,000 dollars to the charity occurred last year.

In the spirit of (Larson, 1988), we assume the structure given in Figure 3 (left) for ditranstives with a dative NP. We do not postulate a privileged predicate VP for English, in contrast to Korean, since nominative case assignment in English is done through a tensed verb. This is supported by the fact that infinitivals in English cannot have a nominative case-marked subject NP. We propose that the light verb *make* anchors the auxiliary tree given in Figure 3 (right). The light verb tree is similar to the Korean light verb tree in that the root node has [CAT:V] and the foot node has [CAT:N]. This tree can only adjoin to XP1, constrained by the English SVO word

[7] In the English XTAG grammar (The XTAG-Group, 1998), the light verb and the base noun are anchors of a single elementary tree. Although this analysis works for English for all practical purposes, extending it to Korean forces us to postulate multiple elementary trees for a single Sino-Korean lexeme, failing to account for the systematic variation in the syntactic realization of its argument structure. In §4, we will show that our analysis can also be extended to English, allowing a uniform anaylsis of LVCs in both Korean and English.

order. Once the adjunction takes place at XP1, the [CAT:] feature on XP2, X2′, X2, X1′ and X1 will all come to have the value N. Thus, NP2 must be realized with a preposition *of*, and X1 must be realized as a nominal form *donation*. Further, the adjoined light verb *make* assigns a nominative case to subject NP_1, and an accusative case to its NP complement *a donation of 10,000 dollars to the charity*.

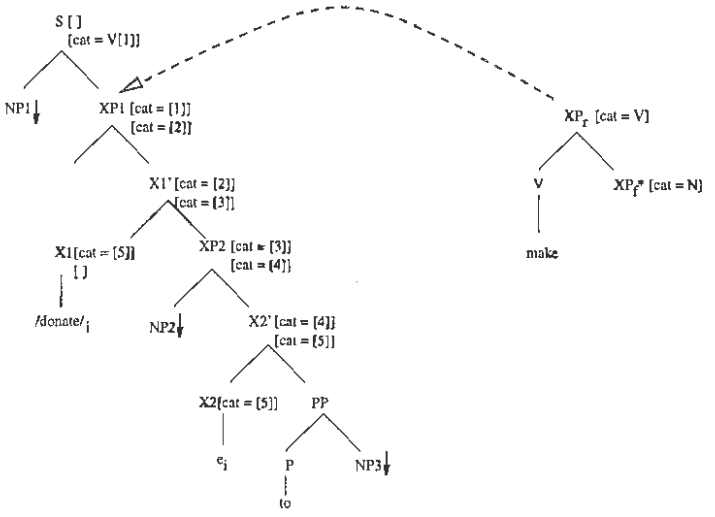


Figure 3: Ditransitive with Dative NP (left) and Light Verb *make* (right)

According to Larson, ditransitive sentences can undergo dative shift deriving a double object construction. If dative shift applied to the ditransitive structure in Figure 3, one might expect to derive from (5b) a string as in (7).

(7) * John made the charity a donation of 10,000 dollars last year.

If we blindly apply dative shift to the ditransitive structure in Figure 3, abstracting away from details, the case marking *to* on the goal argument NP_3 (*the charity* in (5b)) would disappear, and so it could in principle move up to [Spec, XP1] to receive case. But once the adjunction of the light verb takes place at XP1, all the projections of X2 and X1 would become nominal, disallowing any case assignment to NP_3. This leads us to conclude that dative shift cannot apply to sentences such as (5b), which means that the only way to derive (7) is through a ditransitive full verb *make* as in *John made Mary a cake*. But then, this ends up in semantic conflict between full ditransitive *make* and *donation*. In (7), *make* requires a direct object who is a beneficiary of John's action, but the direct object *the charity* is behaving as a recipient due to the presence of the noun *donation*.

Our analysis on the Korean LVC therefore can be extended to the English LVC, allowing for a unified account of LVCs in both languages as well as accounting for their differences.

5. Towards a Pure Representation of Lexical Argument Structure

In Korean, the Sino-Korean lexemes that we have discussed in §1 can also project to an NP. In this case, all arguments are obligatorily realized using genitive case marking.

(8) John-uy Mary-eykey-uy inhyung(-uy) senmwul
John-Gen Mary-to-Gen doll(-Gen) gift

'John's gift of a doll to Mary'

The question arises how these NPs are represented. We assume that the same Sino-Korean lexeme can project to an NP or to an S with the same argument structure, and thus we need a unique representation of lexical argument structure. Our current representation is fixed to project to S. We propose to extend our analysis by assuming that there is an underlying *lexical argument frame* (LAF), a representation of pure argument structure in which syntactic categories (i.e., node labels and the CAT feature) are not yet fixed. Node labels are added (as features) during the lexical phase of a derivation, when an LAF is instantiated with syntactic features prior to the syntactic derivation involving other trees (also see (Chomsky, 1970)).

Specifically, we will now represent *all* node labels as XP, X', or X. The difference between the verbal and nominal node labels will now be represented at all nodes using the feature [CAT:]. But [CAT:] does not yet account for the difference between VP and S, so we need to introduce new features in order to represent our analysis of the LVC (which crucially relies on the VP/S distinction).

As discussed in §3, in Korean, the VP represents an unsaturated syntactic predication structure, the nominal argument is the subject of predication, and the S represents a saturated predication structure. We will capture this analysis with two new binary features, PRED and SUBJPRED, which indicate the presence or absence of an unsaturated predication structure and of a saturated predication structure, respectively. Note that [PRED:- SUBJPRED:+] does not make sense and is assumed not to occur.

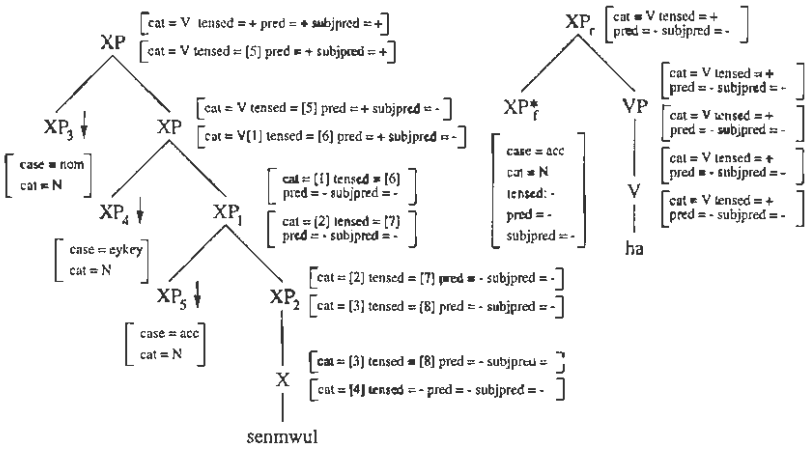


Figure 4: Base lexeme *senmwul* 'gift' (left) and light verb *ha* (right), not using node labels

We now show how our new way of representing node labels accounts for the LVC data by assigning the new features to the nodes in our example (Figure 1). Clearly, none of the nodes labeled XP in Figure 1 form predication structures, so all feature structures associated with them are [PRED:-, SUBJPRED:-]. The subtree anchored at the VP node represents the predicate, so the bottom feature structure of the VP node is [PRED:+, SUBJPRED:-]. Since no further adjunction at the VP node can alter the fact that a predication structure exists, the top feature structure is also [PRED:+, SUBJPRED:-]. Finally, the subtree rooted at the S node (even if adjuncts are adjoined to it later) is the saturated predication structure, so both bottom and top feature structures get [PRED:+, SUBJPRED:+]. The new tree for *semnwul* is shown in Figure 4 on the left.

We now turn to light *ha*. Since it is a light verb, it contributes the [CAT:V] information, but does not create a syntactic predication structure on its own (since it is semantically vacuous). Furthermore, it cannot be adjoined into an already existing predication structure, because predication structures are necessarily verbal (by assumption) and *ha*'s footnode is labeled [CAT:N]. Therefore, the root and foot nodes of light *ha* have top and bottom features labeled [PRED:-, SUBJPRED:-]. The new tree for light *ha* is shown in Figure 4 on the right. It is clear that our previous analysis of the light verb construction facts carries over essentially unchanged to the new representation.

Thus, we have shown that we can represent the information contained in node labels as features in a motivated manner. We can now define an LAF (i.e., a syntactically neutral representation of lexical argument structure) as a tree projected from a lexeme with substitution nodes for all its arguments, in which all syntactic features (CAT, PRED, SUBJPRED, TENSED) are undefined. Specifically, if we take the representation in Figure 4 on the left and set all syntactic features to undefined, then we obtain the LAF for *semnwul*. This LAF is the starting point for lexical derivations. Not all assignments of values for the four syntactic features are valid. In fact, as mentioned above, in Korean, only verbal structures (with [CAT:V]) can create nodes with [PRED:+], [SUBJPRED:+], or [TENSED:+] — the features simply don't make sense for [CAT:N]. Thus, if the choice of projecting to a verbal predication structure is made, then the analysis presented in §3 follows. If instead we choose [CAT:N] at the root node, then we do not get a predication structure, light *ha* cannot be adjoined, and all arguments are realized in the genitive, as desired.

6. Conclusion

We have shown how we can derive the Sino-Korean LVC by assuming that the base lexemes have a single entry in the lexicon and a single light verb *ha* is adjoined into them to obtain the LVC. We have suggested that our analysis extends to English light verb constructions as well. Finally, we have shown how this analysis points to a TAG-based representation of lexical argument structure independent of syntactic categories such as lexical class. In future work, we intend to investigate more cross-linguistic data on LVCs in order to verify that our approach carries over to different types of LVCs, and we intend to verify our TAG-based representation of lexical argument structure by investigating nominalization in different languages.

References

CHOMSKY N. (1970). Remarks on nominalization. In R. JACOBS & P. ROSENBAUM, Eds., *Readings in English Transformational Grammar*. Waltham, Mass.: Ginn.

GRIMSHAW J. & MESTER A. (1988). Light verbs and theta-marking. *Linguistic Inquiry*, **19**, 205–232.

HEYCOCK C. & LEE Y.-S. (1989). Subjects and predication in Korean and Japanese. In H. HOJI, Ed., *Japanese/Korean Linguistics*, volume 2, p. 239–254. Stanford: CSLI.

LARSON R. (1988). On the double object construction. *Linguistic Inquiry*, **19** (3 p.), 335–392.

PARK K. (1992). *Light Verb Constructions in Korean and Japanese*. PhD thesis, University of North Carolina at Chapel Hill.

THE XTAG-GROUP (1998). *A Lexicalized Tree Adjoining Grammar for English*. Technical Report IRCS 98-18, UPenn.

YOON J. (1991). Theta operation and the syntax of multiple complement constru ctions in Korean. In S. KUNO, I.-H. LEE, J. WHITMAN, J. MALING, Y.-S. KANG & Y. JOO KIM, Eds., *Harvard Studies in Korean Linguistics*, volume 4. Dept. of Linguistics, Harvard University.

Complexity of Linear Order Computation in Performance Grammar, TAG and HPSG

Karin Harbusch
Computer Science Dept., Univ. Koblenz
Rheinau 1, D-56075 Koblenz/DE
harbusch@informatik.uni-koblenz.de

Gerard Kempen
Dept. of Psychology, Leiden Univ.
PO Box 9555, NL-2300 RB Leiden/NL
kempen@fsw.leidenuniv.nl

Abstract This paper investigates the time and space complexity of word order computation in the psycholinguistically motivated grammar formalism of Performance Grammar *(PG). In PG, the first stage of syntax assembly yields an unordered tree ('mobile') consisting of a hierarchy of lexical frames (lexically anchored elementary trees). Associated with each lexical frame is a* linearizer*—a Finite-State Automaton that locally computes the left-to-right order of the branches of the frame. Linearization takes place after the* promotion *component may have raised certain constituents (e.g. Wh- or focused phrases) into the domain of lexical frames higher up in the syntactic mobile. We show that the worst-case time and space complexity of analyzing input strings of length* n *is* $O(n^5)$ *and* $O(n^4)$*, respectively. This result compares favorably with the time complexity of word-order computations in Tree Adjoining Grammar (TAG). A comparison with Head-Driven Phrase Structure Grammar (HPSG) reveals that PG yields a more declarative linearization method, provided that the FSA is rewritten as an equivalent regular expression.*

1. Performance Grammar

Performance Grammar (PG; Kempen, 1999) is a psycholinguistically motivated grammar formalism for analysis and generation. Somewhat simplified, and in the terminology of TAGs (cf. Joshi & Schabes, 1997), PG defines lexically anchored initial trees and generates derived trees synchronously linked to conceptual structures described in the same formalism (as in Synchronous TAGs; Shieber & Schabes, 1990) and it factors dominance relationships and linear precedence in surface structure trees (Joshi, 1987). PG differs from recent TAG versions in that there are no auxiliary trees, and that adjunction is replaced by a combination of substitution—the only composition operation—and finite-state linearizers that take care of vertical movement (*'promotion'*) of phrases and of the linear order of branches of derived trees.

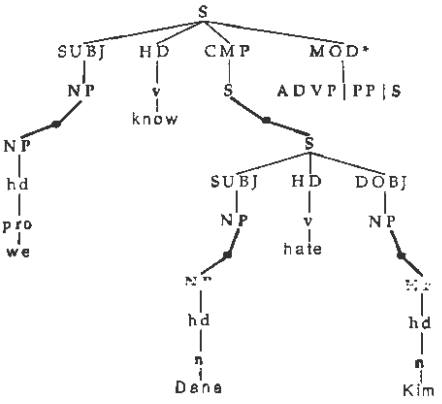


Fig. 1. Simplified lexical frames underlying the sentences We know Dana hates Kim *and* Kim we know Dana hates *(example from Sag & Wasow, 1999). Order of branches is arbitrary. The lines containing filled circles denote substitution (feature unification).*

More precisely, PG's initial trees, called *lexical frames*, are 4-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 (cf. the Kleene star: MOD*). Every functional node dominates exactly one phrasal node ('foot') in the third layer, except for HD which immediately dominates a lexical (part-of-speech) node. Each lexical frame is 'anchored' to exactly one lexical item: a 'lemma' printed in the fourth layer below the lexical node serving as the frame's HeaD (Fig. 1).

Associated with nodes in the first and the third layer are *feature matrices* (not discussed any further here), which can be *unified* with other matrices as part of the substitution process. The unification operation is non-recursive and always involves one root and one foot node of two different lexical frames (see the filled circles in Fig. 1). Only local information can prevent a substitution. No feature information is percolated through the derived tree.

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 (slots, 'landing sites') for constituents. 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 the 12 slots where constituents of English verb frames can go. The positions numbered F1 through F3 make up the Forefield (from Ger. *Vorfeld*) M1 through M7 belong to the Midfield (*Mittelfeld*); B1 and B2 are the Backfield (*Nachfeld*). The annotations at the arcs denote possible fillers of the slots. For example, slot F1 can be occupied by one constituent: either a focus carrying constituents (in Main clauses only), a subordinating conjunction (in an adverbial MODifier clause), a Wh-phrase 'promoted' out of a lower lexical frame (see below), or a non-promoted Wh-phrase. The HeaD verb of a clause is assigned the first Midfield slot (M1), possibly preceded by the complementizer *to* and followed by a particle. Lexical frames anchored to other parts of speech than verbs (e.g. NP- or PP-frames) have their own specialized linearization arrays.

A key property of linearization in PG is that certain constituents may move out of their 'own' array and get 'promoted' to a position in an array located at a higher level in the hierarchy of lexical frames. *Promotion* takes place when, due to subcategorization constraints, a linearization array is 'truncated', that is, instantiated incompletely. For instance, if a verb takes a non-finite complement clause, the whole Forefield (slots F1 through F3) will be missing from the complement's array. Due to incomplete instantiation of the linearization array of a lexical frame, one or more constituents of that lexical frame may be deprived of its landing site. In that case, these constituents move up the hierarchy of lexical frames, looking for an instantiation of their landing site in a higher array. The first (i.e. lowest) landing site is always chosen as the final destination.

Truncation of linearization arrays only affects *lateral* (i.e. left- or right-peripheral) slots. The slot occupied by the head of the phrase is never truncated away, which implies that the head of a lexical frame is never promoted. How many slots at either side of the head are actually instantiated, is determined strictly locally, i.e. depends only on information contained by the lexical frame the array belongs to, and its parent frame (its unification partner).

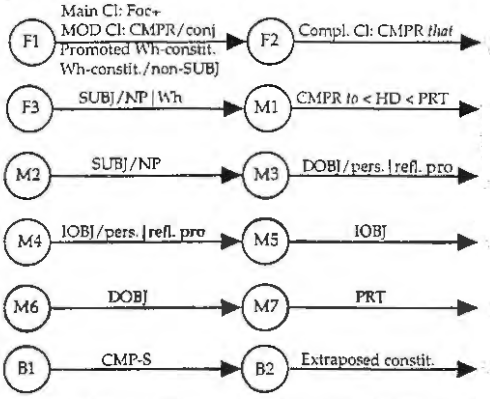


Fig. 2. Linearization array for constituents of S-frames. Placement conditions are annotated on the arcs. E.g., „SUBJ/NP|Wh" at slot F3 means: SUBJect, provided it is an NP or a Wh-phrase"; „<" indicates the precedence relation between constituents sharing a slot. MODifiers have not been depicted.

The mechanism controlling the distribution of constituents over the instantiated slots of a linearization array, is modeled as a *Finite State Automaton* (FSA). The FSA associated with a lexical frame traverses the instantiated slots of its array from left to right. At each slot, it inspects the set of constituents that are waiting for placement in the array, and in-

serts there any constituents meeting the placement conditions (arc labels in Fig. 2).

Fig. 3 illustrates promotion of a focused Direct Object. Examples (1)-(4), taken from Haegeman (1994), demonstrate some subtle consequences of PG's word ordering scheme for *Wh*-questions[1].

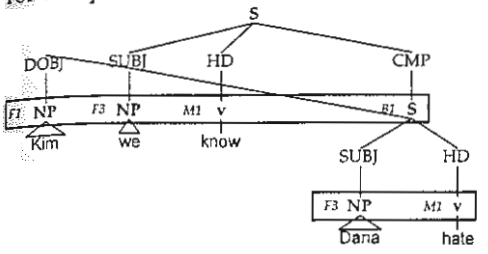


Figure 3. Promotion/linearization at work for the sentence Kim we know Dana hates. *The Direct OBJect of hates carries focus and therefore needs an F1 slot as landing site. Because the linearization array of* hates *has been instantiated incompletely,* Kim *is promoted into the array of the main clause.*

(1) Who do you think left?
[S[*F1* Who *M1* do *M2* you *B1*
[S[*M1* think *B1*
[S[~~*F1 F2 F3*~~ *M1* left]]]]]]

(2) *Who do you think that left?
[S[*F1* Who *M1* do *M2* you *B1*
[S[*M1* think *B1*
[S[~~*F1*~~ *F2* that *F3 M1* left]]]]]]

(3) Who do you think Bill saw?
[S[*F1* Who *M1* do *M2* you *B1*
[S[*M1* think *B1*
[S[~~*F1 F2*~~ *F3* Bill *M1* saw]]]]]]

(4) Who do you think that Bill saw?
[S[*F1* Who *M1* do *M2* you *B1*
[S[*M1* think *B1*
[S[~~*F1*~~ *F2* that *F3* Bill *M1* saw]]]]]]

As outlined in Kempen & Harbusch (1998) and Kempen (1999), the PG's word ordering scheme enables generating the mildly context-sensitive language $a^n b^n c^n$, as well as to account for the movement and word order patterns in English, German and Dutch, including certain rather complicated scrambling phenomena in German. The complexity of these phenomena in contrast with the relative simplicity of this scheme suggests that PG may give rise to very efficient methods of analyzing linear order. Below we show that the worst-case time and space complexity is $O(n^5)$ and $O(n^4)$, respectively.

2. Time and Space Complexity

Consider input string $w=w_1,...,w_n$ of length n. The overall analysis is divided into two steps:

1. Enumerating the complete set of lexical frame hierarchies dominating all permutations of w (henceforth called the *set of dominance structures*), and
2. Checking linear order on the basis of the FSA, taking into account the possibility of promotion of phrases in valid dominance structures.

Step 1. Any lexical frame is rewritable in terms of a context-free rule because the functional nodes in the second layer of a lexical frame can be viewed simply as annotations on edges descending from the root node. Every word in w is associated with one or (in case of word-class ambiguity) several (O(1)) lexical frames, and every lexical frame has exactly one lexical anchor.

Since a lexical frame is an unordered tree, it can be viewed as an Immediate Dominance rule with an empty set of Linear Precedence rules (ID/LP); and parsing with lexical frames could proceed as outlined in Shieber (1984). However, this method would not take the full set of valid dominance structures into account. For instance, the sentence *Kim we know Dana hates* cannot be analyzed by an ID/LP grammar because *Kim* has moved outside the locality scope of *hate*.

Therefore we follow an indirect course. We interpret the input string as a multiset, i.e. as the set of all permutations of input words, so that any scope of locality is included. Moreover, we 'freeze' the lexical frames into an arbitrary but fixed left-to-right order of branches, which gives a context-free gram-

[1] Our promotion scheme differs from the 'lifting' scheme recently proposed by Kahane, Nasr & Rambow (1998) in that we allow promotion exclusively along *lateral* (i.e. truncated) regions of a linearization array (thus ruling out, e.g., the promotion pattern in example (2) above). Lifting does not seem to embody an non-ad-hoc equivalent restriction.

mar[2]. This guarantees, for instance, that the valid dominance structure is built for the example in Fig. 3 (as one of the permutations of *We know Dana hates Kim).* Hence, the first step enumerates all locality domains[3].

In order to deal efficiently with multisets in the input, we use a slightly extended version of Earley parsing which overgenerates with respect to repetitions of the same input symbol. The reason is that we do not check here whether any symbol occurs more than once.

First, a subgrammar G' is constructed which only provides the lexical frames of any input symbol w_i, $i=1,...,n$. The only modification of the Earley algorithm concerns the *scanning* step. Instead of exploiting only the items (X, $\alpha \bullet t\beta$) where $t=w_{i+1}$ in the original input string, the parser scans all items and produces (X, $\alpha t \bullet \beta$) according to subgrammar G'. Obviously, this modification performs as bad as ordinary scanning does in the worst case, without introducing additional time and space requirements are introduced. Moreover, the modified scanning method implies that all permutations of the input string are explored. Consequently, given the extended Earley algorithm for subgrammar G', the time complexity and the space complexity for the construction of all dominance structures of the multiset of w remains $O(n^3)$ time and $O(n^2)$ space units[4].

Step 2 is based on the linearizer FSAs and linearization arrays associated with the phrases ('items') in the dominance structures. An array represents a hypothetical order of the input elements $w_i..w_n$ under the assumption that the input elements $w_1...w_{i-1}$ have been ordered successfully. These orderings are licensed by the finite number of slots in the FSA. As the grammar is in Greibach normal form, one ordered symbol must equal the terminal in the rule. All other symbols may go to the finite set of *promotion sites* provided by the FSA. Therefore, the task of step 2 can be reformulated as follows: For any derivation, compute all bijective functions from the terminals in the context-free rules to the input symbols.

In order to deal efficiently with the $O(n^2)$ items that are provided as input to step 2, ordinary Earley processing is assumed along the backpointers inserted in step 1. Initially, this yields all items of the form (S, $tX_j...X_k\bullet$) in I_n. These $O(n)$ items have successfully passed step 1. Now, each of these items is associated with arrays each representing one of the following hypotheses:

$t=w_j$ and no landing site is selected, or

$t=w_k$ and the sequence $w_i,...,w_{k-1}$ of promoted symbols is licensed by a sequence of landing sites to the left of w_k according to the item's FSA ($k=2, ...,n$).

Exploring the number of resulting items, we have to consider $O(n)$ context-free rules in I_n. Moreover, the order in the original input string determines a finite sequence of landing sites according to the currently considered FSA (w_j, w_jw_2, $w_jw_2w_3$, ..., $w_jw_2w_3...w_n$). Hence, the space is $O(n^3)$. With each array we associate a pair containing (1) the *'list of promoted symbols' LPS* and (2) the *'fixed-order marker' FOM* which provides the index in w that t takes. Notice the length of LPS$\leq n$; |FOM|=1.

Now, all substructures of these items (completions) are evaluated, taking the already analyzed input symbols into account (this index with respect to w is provided by the FOM). Hence, the context-free rules applied here can only order $O(n-1)$ times $O(n-1)$ symbols. In general, assuming FOM=i, there exist $O(n-i+1)$ times $O(n-i+1)$ potential orders for the remaining elements $w_{i+1},...,w_n$. Hence, the overall space complexity is $O(n^4)$.

Now consider the general case for an item ($X,t_j\beta \bullet Y\gamma$) in I_j with LPS=$a_i,...,a_p$, and

[2]Without loss of generality, we assume that the left-most branch contains the head of the frame. Hence we deploy a context-free grammar in *Greibach normal form:* (X, t Y_1 ... Y_k), with X and Y_1 ...Y_k non-terminal, and t terminal.

[3]Throughout the paper we assume a condensed representation of the set of potential dominance structures; cf. 'items' in Earley parsing (Earley, 1970).

[4] Since the unification operation in PG is non-recursive, it only involves testing a finite list of constraints. Hence, it does not increase time complexity.

FOM=i ($p<i$; $j \leq i$; β, γ, δ possibly empty sequences of non-terminals; $a_1,...,a_p=w_j...w_{i-1}$ with missing elements):

For any item $(Y,t_{j+1}\delta\bullet)$ in I_{j+1} the following hypotheses are generated: $w_{i+1}...w_{i+k-1}$ is licensed by a sequence of landing sites to the left of w_{i+1} according to the local FSA. Furthermore, one of the following situations holds: $t_{j+1}=w_{i+k}$ or $t_{j+1} \in$ LPS. Consequently, FOM=$i+k$, LPS=LPS+$w_{i+1},...,w_{i+k-1}$. If $t_{j+1} \in$ LPS, the rightmost w_{i+1} in LPS is erased without loss of generality[5].

If we assume that all items are revisited according to their backpointers, an ordinary Earley parser is capable of performing step 2. (Initially, LPS=nil and FOM=0; finally, an item $(S,\alpha\bullet)$ with LPS=nil, FOM=n must exist.) Hence, the input of size $O(n^2)$—the output of step 1—leads to an overall time complexity of $O(n^2)$ times $O(n^3)$, i.e. $O(n^5)$.

Because this result compares favorably with other grammar formalisms (see below), we conclude that PG provides an efficient method for linear order computation. This advantage derives basically from the deployment of the promotion/linearization scheme, which allows for non-local ordering effects of local ordering decisions, in particular the partial instantiation of linearizers.

3. PG, TAG, and HPSG

For reasons of space we only address the two broadly applied formalisms of *Tree Adjoining Grammar* (TAG, cf. Joshi & Schabes, 1997; and *Head-Driven Phrase Structure Grammar* (HPSG, cf. Sag & Wasow, 1999).

For TAGs, various definitions of dominance and linear order have been proposed in the literature (cf. Joshi, 1987; Vijay-Shanker, 1992 for the definition of quasi-trees; Rambow, 1994 for V-TAGs). They all have in common that long-distance movements are structurally realized by adjoining, thus yielding the extended domains of locality characteristic of all TAGs.

Linear ordering in *Local Dominance/(Tree) Linear Precedence (LD/(T)LP) TAGs* proceeds very much like the ID/LP framework defined for context-free grammars. Since local dominance structures are provided where 'moved' constituents feature at the structural level (i.e. adjoining stretches the distance between nodes of the same elementary tree), the cost of linear ordering is at least $O(n^6)$ time units—as for ordinary TAGs (cf. Joshi & Schabes, 1997).

As is well-known, scrambling cannot be described by a simple (LD/(T)LP) TAG. *Quasi-trees* represent partial descriptions of trees. This definition allows for underspecified ordering of moved elements. Loosely speaking, in this framework the spine for promotion is specified declaratively. Similarly, *V-TAGs* (a specific kind of Multi-component TAG) provide a method for manipulating different portions of the same overall derivation tree. Both formalisms are able to handle scrambling phenomena. However, the individual readings are spelled out as different derived trees which are computed on the basis of adjoining in an ordinary TAG parser; hence, this costs at least $O(n^6)$ time units.

The essential difference between the PG and TAG formalisms can be summarized as follows. In both PG and TAG, dominance structures—consisting of lexical frames and elementary trees, respectively—describe linguistically motivated domains of locality. In TAG, the adjoining operation which moves constituents apart, affects the dominance structure. In PG, the linearization component leaves the dominance structure intact. The linearizer FSA associated with lexical frames can accommodate constituents originating from other constituents—a behavior that is less costly, as shown above.

In HPSG (Sag & Wasow, 1999), the *PHON* and *GAP* features, the *GAP principle* and the *argument realization principle* are basically responsible for word ordering and long-distance movement. The *PHON* feature of phrasal types enumerates the linear order on

[5]This reflects the linguistic observation that a promoted phrase chooses the lowest possible landing site.

the basis of list addition (⊕, i.e. a non-commutative sum). Furthermore, movement phenomena are handled by the *GAP feature*, the *GAP principle* and the *argument realization principle*. The *GAP feature* contains a list of elements to be moved. The *argument realization principle*, which says that a word structure tree is well-formed only if the valence lists (SPR and COMPS) add up to the argument structure (ARG-ST), is extended to instantiate gaps freely; i.e. some elements of ARG-ST are neither on the SPR nor on the COMPS list, but on the GAP list instead. The *GAP principle* tests whether the GAP values of all daughters add up to be the GAP value of the mother, unless the rule sanctioning the structure is the Head-Filler Rule. In order to ultimately get all gaps filled, the initial symbol must have an empty GAP list.

This method, like PG's linearization scheme, computes linear order without manipulating the dominance structure (i.e., the daughters' feature descriptions). Loosely speaking, the specification in the PHON feature can be interpreted as a regular expression equivalent to a FSA (although the PHON feature does not provide the definition of the Kleene Star; the infinity of licensed orderings is provided by the recursive application of schemata, i.e. A⊕B, where B has the PHON feature C⊕D—cf. Sag & Wasow, *o.c.*, p. 374). Furthermore, the realization of movement phenomena corresponds directly to promotion, i.e., the gap is percolated along the spine. The definition of landing sites is defined differently, however. In PG, landing sites are enumerated declaratively whereas HPSG terminates the percolation procedurally in terms of the GAP principle. As computation of feature specifications is, in general, NP-complete (Hegner, 1995), the cost of linear order computation is of no particular interest to HPSG. However, HPSG aims at describing linguistic phenomena *declaratively*. Our description, we claim, is more declarative than the current HPSG realization. The linearizer FSA of a lexical frame can be rewritten as an equivalent regular expression and becomes associated with the referring phrasal type in HPSG.

4. Conclusions

We have described an approach to linear ordering that involves a non-local precedence mechanism which does not rely on a definition and scope of movement as in terms of the GAP feature. In comparison to TAG's structural representations based on adjoining, PG's promotion/linearization yields a more efficient analysis. Compared to HPSG, it can give rise to more declarative word ordering.

References

EARLEY, J. (1970). An Efficient Context-free Parsing Algorithm. *Comm. of the ACM, 13:2.*

HAEGEMAN, L. (1994). *Introduction to Government and Binding Theory (2nd ed.).* Oxford: Blackwell.

HEGNER, S. (1995). *Distributivity in Incompletely Specified Type Hierarchies: Theory and Computational Complexity.* University Tübingen, Linguistics Dept., Tech. Report 4.

JOSHI, A.K. (1987). The Relevance of Tree Adjoining Grammar to Generation. In: Kempen, G. (Ed.), *Natural language generation.* Dordrecht: Kluwer.

JOSHI, A.K., SCHABES, Y. (1997). Tree Adjoining Grammars. In: G. Rozenberg, A. Salomaa (Eds.), *Handbook of Formal Languages (Vol. 3).* Berlin: Springer.

KAHANE, S., NASR, A. & RAMBOW, O. (1998). Pseudo-Projectivity: A Polynomially Parsable Non-Projective Dependency Grammar. *Procs. of COLING-ACL*, Montreal.

KEMPEN, G. (1999). Human Grammatical Coding. Ms. Leiden University.

KEMPEN, G., HARBUSCH, K. (1998). A 'Tree Adjoining' Grammar without Adjoining. In: *Procs. of TAG+4*, Philadelphia PA.

RAMBOW, O. (1994). *Formal and Computational Aspects of Natural Language Syntax.* Ph.D. Thesis, University of Pennsylvania.

SAG, I.A., WASOW, T. (1999). *Syntactic Theory: A Formal Introduction.* Stanford: CSLI.

SHIEBER, S.M. (1984). Direct Parsing of ID/LP-Grammars. *Linguistics and Philosophy 7.*

SHIEBER, S.M., SCHABES, Y. (1990). Synchronous Tree-Adjoining Grammars. *Procs. of COLING-90*, Helsinki.

VIJAY-SHANKER, K. (1992). Using Descriptions of Trees in Tree Adjoining Grammars. *Computational Linguistics, 18:4.*

RELATIONSHIP BETWEEN STRONG AND WEAK GENERATIVE POWER OF FORMAL SYSTEMS*

Aravind K. Joshi

Department of Computer and Information Science and
Institute for Research in Cognitive Science
University of Pennsylvania
Philadelphia, PA 19104
USA
joshi@linc.cis.upenn.edu

Abstract

The relationship between strong and weak generative powers of formal systems is explored, in particular, from the point of view of squeezing more strong power out of a formal system without increasing its weak generative power. We examine a whole range of old and new results from this perspective. However, the main goal of this paper is to investigate the strong generative power of Lambek categorial grammars in the context of crossing dependencies, in view of the recent work of Tiede (1998).

Introduction

Strong generative power (SGP) relates to the set of structural descriptions (such as derivation trees, dags, proof trees, etc.) assigned by a formal system to the strings that it specifies. Weak generative power (WGP) refers to the set of strings characterized by the formal system. SGP is clearly the primary object of interest from the linguistic point of view. WGP is often used to locate a formal system within one or another hierarchy of formal grammars [1]. Clearly a study of the relationship between WGP and SGP is highly relevant, both formally and linguistically. Although there has been interest in the study of this relationship, almost from the beginning of the work in mathematical linguistics, the results are few, as this relationship is quite complex and not always easy to study mathematically (see Miller (1969) for a recent comprehensive discussion of SGP).

Our main goals in this paper are (1) to look at some old and recent results and try to put them in a general framework, a framework that can best be described by the slogan–How to squeeze more strong generative power out of a grammatical system?– and (2) to present a new result concerning Lambek categorial grammars. Our general discussion of the relationship of SGP and WGP will be in the context of context-free grammars, categorial grammars and lexicalized tree-adjoining grammars.

1. Context-Free Grammar (CFG)

McCawley(1967) was the first person to point out that the use of context-sensitive rules by linguists was really for checking structural descriptions (thus related to SGP) and not for generating strings (i.e., WGP), suggesting that this use of context-sensitive rules possibly does not

*This work was partially supported by NSF Grant SBR8920230

[1] SGP is also relevant in the context of annotated corpora. The annotations reflect aspects of SGP and not of the rules of a grammar and therefore WGP.

give more SGP than CFG's. Peters and Ritchie (1969) showed that this was indeed the case. These results are closely related to the notion of recognizable sets of trees (structural descriptions) as explained below.

In a CFG, G, the derivation trees of G correspond to the possible structural descriptions assignable by G. It is easily shown that there are tree sets whose yield language is context-free but the tree sets are not the tree sets of any CFG. That is, we are able to squeeze more strong power out of CFG's indirectly. Here is a simple example.

Let T be the set of trees defined by trees such as

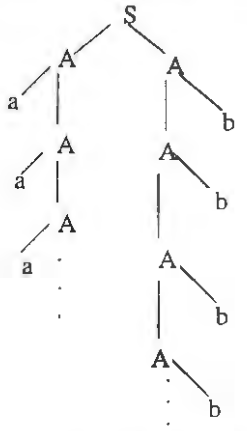


Figure 1: A Recognizable Set of Trees

T in Figure 1. is not a set of derivation trees for any CFG. Clearly in any CFG G, the rules for A will get mixed up and there will be no way we can make sure that all a's are on the left and all b's are on the right. The string language is, of course, $\{a^n b^m | m, n > 1\}$, which is a context-free language. What is the relationship between the trees of the CFG corresponding to this language and the set T? Thatcher (1967) showed that the relationship is very close. Sets such as T, called recognizable sets, are the same as the tree sets of CFG's (called local sets) except possibly for relabeling. It turns out that the tree sets 'analyzable' (i.e., checkable) by context-sensitive rules, as suggested by McCawley are indeed recognizable sets. All these systems have the property that they allow checking of 'local' constraints around a node in a tree. Thatcher's result shows that this notion of 'locality' can be captured by finite state tree automata. Later Joshi, Levy, and Yueh (1975) and Rogers (1997) showed that the notion of 'local context (local tree domains)' can be made substantially richer yet maintaining characterizability by finite state tree automata. All these results can be interpreted as attempts to squeeze more strong power out of a formal system, in this case, context-free grammars.

2. Lexicalized Tree-Adjoining Grammars (LTAG)

The earliest indication that more strong power can be obtained from LTAG by going to tree-local multicomponent TAG (still preserving the weak power of LTAG) is in Weir (1987) and Kroch and Joshi (1989). Shieber and Schabes (1992) introduced the notion of multiple adjoining at the same node in a derivation tree. This move can also be seen as an attempt to get more strong power out of LTAG without going beyond the weak power of LTAG. In fact, the whole range of recent works (Candito (1997), Joshi and Vijay-Shanker (1998), Kulick (2000), Kallmeyer

and Joshi (1999) can be seen as attempts to get more SGP from LTAG without going beyond the WGP of LTAG. This is achieved by providing flexibility in interpreting the derivation trees in LTAG. In particular, in Joshi and Vijay-Shanker (1998), Kulick (2000)[2], and Kallmeyer and Joshi (1999), all of which use tree-local multi-component LTAG, flexibility is introduced in the derivation in LTAG resulting in increased strong power without exceeding the weak power of LTAG. This notion of flexibility (flexibility in composition, i.e., in the directionality of the composition) can be briefly defined as follows, at least for the approaches in Joshi and Vijay-Shanker (1998) and Kallmeyer and Joshi (1999). Given a pair of trees, say, $\gamma(1)$ and $\gamma(2)$ the composition (i.e., attachments by substitution and adjoining[3]) can proceed from $\gamma(1)$ to $\gamma(2)$, i.e., $\gamma(1)$ composes with $\gamma(2)$ if $\gamma(2)$ is an elementary tree, otherwise $\gamma(2)$ composes with $\gamma(1)$ if $\gamma(1)$ is an elementary tree, assuming, of course, that $\gamma(1)$ and $\gamma(2)$ are semantically related, i.e., composition of arbitrary unrelated trees is not allowed. Such a notion of flexibility can be introduced in CFG's as well as in Categorial Grammars. However, as far as I know, such a move does not open the door for squeezing more SGP out of the formal system. This is due to the fact that CFG's and Categorial Grammars are essentially string rewriting systems, while systems such as LTAG are tree rewriting systems and the complex topology of the initial trees, when combined with the flexibility discussed above, allows the possibility of augmenting the SGP of the system.

3. Categorial Grammars (CG)

It is well known that the Ajdukiewicz and Bar-Hillel categorial grammars (CG(AB)) are weakly equivalent to CFG's. The derivation trees of CG(AB) are essentially the same as the derivation trees of CFG's (i.e., local sets and therefore recognizable sets (see Tiede (1998)). The relationship of recognizable tree sets to the derivation trees of CG(AB) is not discussed by Tiede. The relationship is the same as between the derivation trees of CFG's and recognizable sets, i.e., they are the same except for relabeling[4].

However, for Lambek Grammars (LG) the situation can be different. In LG, the assignment of categories to lexical items is similar to the assignments in CG(AB) but we have the inference rules associated with the calculus. Although LG's (Lambek, 1958) were long conjectured to be weakly equivalent to CFG's, the conjecture was only recently proved to be true by Pentus (1993). So now the question arises: Do LG's provide more strong generative power than CFG's, in other words, is it possible to characterize the proof trees of LG in terms of something like the recognizable sets or even beyond recognizable sets. This question was raised by Buszkowski and van Benthem[5]. However, only recently a serious attempt has been made by Tiede (1998) to try to answer this question. Tiede (1998) covers a number of aspects and, in particular, suggests that the proof trees of LG may be beyond recognizable sets, i.e., there is a Lambek grammar whose proof tree language is not regular. In fact, he suggests that it will be possible to characterize crossing dependencies. Our main point in this paper is to show that this claim is very limited and that the crossing dependencies that can be described are very degenerate (i.e., the dependencies are between a lexical item and a lexically empty element).

First, note that if indeed true (i.e., nondegenerate) crossing dependencies can be characterized by the proof trees of LG then this would be very surprising indeed. From all that we know

[2]The exact equivalence of the system in Kulick (2000) to tree-local TAG's has not been established yet.

[3]Adjoining at the root and substitution at the foot can be treated as attachments of the same kind.

[4]Another recent work concerning SGP and WGP of categorial grammars is by Jäger (1998) who has investigated the generative capacity of multimodal categorial grammars.

[5]Both these results are discussed in *Handbook of Logic and Language (eds. Johan van Benthem and Alice ter Meulen)*, MIT Press, Cambridge, 1998, pp. 683-736.

so far, any formal system that characterizes crossing dependencies (say, between the a's and b's in $a^n b^n$) is more powerful than CFG's, for example, TAG's, Combinatory Categorial grammars (CCG), Linear Indexed Grammars (LIG), etc. because they can all generate the language $\{a^n b^n c^n | n \geq 1\}$. Figure 2 shows the topologies needed to obtain the nondegenerate crossing dependencies[6]. Now once we have trees with this topology it is easy to see that the same topology can be used to generate the language $\{a^n b^n c^n | n \geq 1\}$. The relevant tree will be the same as the tree on the left in Figure 2 with c replacing t. Given that LG characterize only context-free languages, this would lead to a paradox.

Tree topologies needed for nondegenerate crossing dependencies

Figure 2: Nondegenerate Crossing Dependencies

We will show that the crossing dependencies claimed by Tiede are degenerate. In particular, they are dependencies between pairs, where the first element is a non empty lexical item and the other an empty element. We will illustrate this by the example in Figure 3.

L = { a, aa, aaa, . . .} a: S, S/(A/A), S/(S/(A/A))

Proof tree for aa (natural deduction style)

$$\dfrac{a(1) \quad \dfrac{a(2) \quad \dfrac{t(1)\ t(2)}{[A/A]\ [A]}\ [/E]}{A}\ [/I]}{S}\ [/I] \qquad \dfrac{S/(S/(A/A)) \quad \dfrac{S/(A/A) \quad A/A}{S}\ [/E]}{S}\ [/E]$$

Proof trees with empty elements (t's).
Indices are shown for convenience
a(1) a(2) t(1) t(2)

Figure 3: Degenerate Crossing Dependencies in Lambek Grammar

By suitably arranging the introduction and discharge of assumptions in the hypothetical reasoning in the LG we have crossing dependency relations between the a's and the t's, where the t's are the empty elements. In Figure 3 the two assumptions [A/A] and [A] (assumptions are enclosed in []) are introduced at the top level of the deduction. These two assumptions correspond to the empty elements t1 and t2 respectively. Each one of these assumptions is then

[6]The auxiliary tree on the right in Figure 2 is adjoined at the interior nodes of the two trees. We have left out the details about the constraints on the nodes to get precisely the language mentioned above as this is not relevant to the present discussion. Without the constraints the language is more complex, however, this does not affect the argument presented here.

withdrawn using the [/I] rule, the introduction rule (the [/E] rule is used for elimination). Both the assumptions are withdrawn in the deduction as is required in the natural deduction proof. The assumptions that are introduced and then withdrawn have to appear always at the periphery of the proof tree. In our example in Figure 3 they appear at the right periphery. The dependencies between the a's and t's (corresponding to the assumptions) can be seen as follows. In the second [/E] step in the deduction (second from the top) the category A/A is eliminated in combination with S/(A/A) corresponding to a(2). The category A/A in this step resulted from the withdrawal of the assumption [A] (corresponding to t(2)) at the top level. Thus a(2) corresponds to t(2). Similarly a(1) corresponds to t(1). It is easy to see that a natural deduction proof can be constructed for each string in L. Thus we have crossing dependencies between the a's and t's.

For 'true' crossing dependencies both the elements have to be non-empty. The way this is accomplished is by creating two sets of nested dependencies, say between a's and t's and between t's and b's, where the t's are empty elements. Then the resulting dependencies between the a's and b's become crossed as shown in Figure 2 above. The dependencies between a's and t's are nested and those between t's and b's are also nested, resulting in crossing dependencies between a's and b's. Note that the empty elements, t's, are not at the periphery. It is not possible to achieve this in LG because the empty elements have to be at the periphery in the Lambek deduction. So in a real sense the crossing dependencies which Tiede talks about are degenerate and LG is incapable of capturing true crossing dependencies.

Since the Tree-Insertion Grammars (TIG) of Schabes and Waters (1993) are weakly equivalent to CFG's but not strongly, we will explore the implications of TIG for Tiede's work. In fact, we will show that the degenerate case studied by Tiede can be characterized in a TIG. In a TIG, both substitution and adjoining are used. However, adjoining is limited in the following way. First, in each auxiliary tree the footnode is the leftmost (or rightmost) daughter of the root. Further, adjoining is only allowed on the right (or left) frontier. Schabes and Waters (1993) have shown that TIG's are weakly equivalent to CFG's, They do not explore the issue of strong power. Their motivation was to show that TIG's lexicalize CFG's without going beyond the weak power of CFG. We show that strong power is increased, although only to the extent of covering the case of degenerate crossing dependencies considered by Tiede. This suggests the tantalizing conjecture that TIG's are adequate to characterize the proof trees of LG. We have no complete proof of this conjecture at this time.

The proof of the claim that TIG can characterize the degenerate case of crossing dependencies follows from the fact that these dependencies can be implemented by using the TIG in Figure 4. In the tree b1 the footnode is not the rightmost nonterminal on the frontier, however, the frontier to the right of this footnode is lexically empty[7]. Surprisingly, this possibility is allowed in the definition of TIG as it is crucially needed by Schabes and Waters to prove their main result–TIG's are equivalent to CFG's.

Now it is easily seen how the degenerate crossing dependencies of Tiede can be described in this TIG (see Figure 5; b1 is adjoined to a1 at the indicated node in a1).

4. Summary

We have explored a number of old and new results in the study of strong and weak generative powers of formal systems from the point of view of squeezing more strong generative power

[7]What is the equivalent of this result to the case of regular form TAG's defined by Rogers (1994)? TIG's are defined with respect to the topology of the elementary trees and a restriction on adjoining. However, regular form TAG's are defined with respect to derivations. Hence, it is not obvious how the construction will proceed. However, I would conjecture that it should be possible to get a similar result for the regular form TAG's.

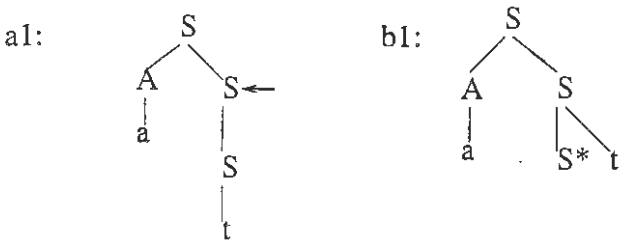
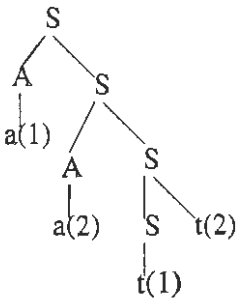


Figure 4: A TIG for Degenerate Crossing Dependencies



Indices are shown for convenience

a(1) a(2) t(1) t(2)

Figure 5: A Derivation in the TIG in Figure 4

out of a formal system without increasing its weak generative power. We have also presented some new results concerning the SGP of Lambek categorial grammars as they relate to crossing dependencies.

5. References

CANDITO M. & KAHANE S. (1998). Defining DTG derivations to get semantic graphs. In *Proc. of the TAG+4 Workshop*, p.25-28, University of Pennsylvania, Philadelphia, PA, USA.

JÄEGER G. (1998). On the generative capacity of multimodal categorial grammars. *Technical Report, 98-26*, Institute for Research in Cognitive Science, University of Pennsylvania, Philadelphia, PA, USA.

JOSHI A. K., LEVY L. & YUEH K. (1975). Local constraints on transformations. *SIAM Journal of Computing*, **8**(4).

KALLMEYER L. & JOSHI A. K. (1999). Factoring predicate argument and scope semantics: Underspecified semantics with LTAG. *Proc. of the Twelfth Amsterdam Colloquium*, p.169-174, University of Amsterdam, Amsterdam, The Netherlands.

KROCH A. & JOSHI A. K. (1985). Linguistic relevance of tree-adjoining grammars. *Technical Report*, University of Pennsylvania, Philadelphia, PA, USA.

KULICK S. (2000). Constrained non-locality: Long-distance dependencies in TAG. *Ph.D. Dissertation manuscript*, University of Pennsylvania, Philadelphia, PA, USA.

LAMBEK J. (1958). The mathematics of sentence structure. *American Mathematical Monthly*, **65**(5).

MCCAWLEY J. W. (1967). Concerning the base component of a transformational grammar. *Foundations of Language*, **4**(3).

MILLER P. H. (1999). *Strong Generative Capacity*, CSLI Publications, Stanford University, Stanford CA, USA.

PENTUS M. (1993). Lambek grammars are context-free. In *Proc. of the 8th Annual Symposium on Logic in Computer Science*, p.35-42, Utrecht, The Netherlands.

PETERS S. & RITCHIE R. (1969). Context sensitive immediate constituent analysis: Context-free languages revisited. In *Proc. ACM Symposium on Theory of Computing*, Marina del Rey, , USA.

ROGERS J. (1994) Capturing CFLs with tree adjoining grammars, In *Proc. of the 32nd Annual Meeting of the Association for Computational Linguistics*, Las Cruces, NM, USA.

ROGERS J. (1997). Grammarless phrase structure grammar. *Linguistics and Philosophy*, **20**(3).

SCHABES Y. & SHIEBER S. M. (1994). An alternative conception of tree-adjoining derivation. *Computational Linguistics* **20**(1).

SCHABES Y. & WATERS R. (1994). Tree insertion grammar. *Technical Report TR-94-13*, Mitsubishi Electric Research Laboratories, Cambridge, MA, USA.

THATCHER J. W. (1967). Characterizing derivation trees of context-free grammars through a generalization of finite automata theory. *Journal of Computer System Science*, **1**(4).

TIEDE H. (1998). Deductive Systems and Grammars. *Ph.D. Dissertation*, Indiana University, Bloomington, IA, USA.

WEIR D. (1988). Characterizing mildly context-sensitive grammar formalisms. *Ph. D. Dissertation*, University of Pennsylvania, Philadelphia. PA, USA.

An alternative description of extractions in TAG

KAHANE Sylvain, CANDITO Marie-Hélène & DE KERCADIO Yannick

LaTTiCe - TALaNa
sk@ccr.jussieu.fr

LexiQuest
marie-helene.candito@lexiquest.fr

LIMSI
kercadio@limsi.fr

Abstract

The aim of the paper is to propose a new description of extraction in plain TAG. Contrary to Kroch 1987's analysis, our description is based on the fact that the power of a relative clause to adjoin on a noun can be attached to the wh-word rather than to a verb. This analysis solves some problems of the previous analysis, notably by giving the right semantic dependency in case of pied-piping.

We are thankful to our two reviewers for many valuable comments.

Introduction

The only description of extractions in TAG we know has been developed by Kroch & Joshi (1986), Kroch (1987) and implemented in the developed grammars of English (XTAG 1995) and French (Abeillé 1991, Candito 1999). This implementation solves the unboundedness of extractions with predicative adjoining, but the pied-piping is solved using a special feature. We think that this solution of pied-piping is not absolutely convenient, because some edge of the derivation tree cannot be interpreted as semantic dependency (Candito & Kahane 1998). Our assumption is based on the fact that a TAG derivation tree can be interpreted as a semantic graph, that is a predicate-argument structure. Moreover this implementation fails to describe some cases of extraction, such as some French *dont*-relatives. We propose a new description of extraction in TAG which solve most of these problems. Nevertheless, our study must rather be appreciated as an investigation of the limits of the TAG formalism, because we think that TAG is not the most appropriate framework for the implementation of our description of extractions. The same analysis is more suitably implemented in GAG/DTG (Candito & Kahane 1998).

1. Semantic dependencies

The meaning of a sentence comes from the combination of the meaning of the lexical units of the sentence. A lexical meaning or **semanteme** can be considered as a semantic functor or predicate. For instance, consider:

(1) *Peter often saw black cats.*

In (1), the meaning 'see' is a binary functor whose argument are 'Peter' and 'cat', whereas 'often' and 'black' are unary functors with respectively 'see' and 'cat' as arguments. This predicate-argument structure can be represented by a graph (Fig. 1), called a **semantic graph** (Zolkovski & Mel'çuk 1967, Mel'çuk 1988). An edge of such a graph is called a **semantic dependency**. The two extremities of a semantic dependency are called the **semantic governor** and **the semantic argument**. A semantic graph can be converted into a logical formula by reification : for each semanteme a variable is introduced as first argument of the predicate; this variable is used by other predicates pointing on it in the semantic graph. The semantic graph of Fig. 1 is thus converted in the formula:

'Peter'(x) & 'cat'(y) & 'black'(p,y) & 'see'(e,x,y) & 'often'(q,e)

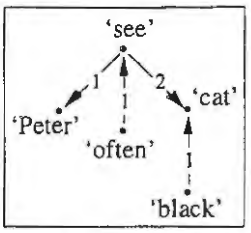


Fig. 1. The semantic graph of (1).

2. Principles for our TAG

We assume the following linguistic properties for elementary trees. The elementary trees correspond to exactly one semantic unit (Abeillé 1991), and respect the predicate-argument co-occurrence principle (PACP), with a semantic interpretation (Candito & Kahane 1998, Candito 1999): semantic predicates anchor trees with positions for the syntactic expression of *all and only* their semantic arguments.[1] It is important to note that the PACP concerns any position to extend, whether substitution or foot node.

Therefore, **the arcs of a TAG derivation tree can be interpreted as semantic dependencies**. In the following, substitution arcs will be represented by down arrows and adjoining arcs, by up arrows. The label on an arrow indicates the position of the semantic argument in the predication (first, second...). A last word about complementizers: as noted by Tesnière (1959), which called them *translatifs*, they are grammatical words that mark a link between two words. Contrary to Franck 1992, we think that complementizers must be attached to the SEMANTIC governor, that is the word that controls the link. For instance, in *Peter thinks that Mary likes beans*, *that* will be a co-anchor of the elementary tree anchored by *thinks*—the semantic governor of *likes*—, while in *the beans that Mary likes*, *that* will be a co-anchor of *likes*—the semantic governor of *beans*. See our solution of *qui/que* alternation of the complementizer in French for an illustration of this principle (Fig.14).

The plain TAG formalism constrains adjoining in the following manner: the root and foot nodes of an auxiliary tree β must be of same categories. It follows that, in a predicative adjunction, the anchor of β and the semantic argument on which β adjoins must be of same categories. In order to allow predicative adjunction on a semantic argument of a different category this constraint must be relaxed. Although it is well known that it does not modify the generative power (Vijay-Shanker 1987, 1992), we do not think that it was really used for linguistic descriptions in TAG.[2] The solution simply consists in considering categories as top and bottom features. In this case, all nodes will have a same transparent category X and real syntactic categories will only appear in top and bottom features. The following notation will be adopted: [A|B] := [X,t:A,b:B]. For the sake of simplicity, a node with same top and bottom categories A will be noted A: A := [A|A]. Note that a node that has different top and bottom categories has to receive an adjunction. This little change in the formalism (which does not change the generative power) allows new linguistic descriptions. Before going to the extraction, we will study the case of determiners, predicative adjectives and *tough*-movement.

---

[1] This counts for expressed semantic arguments only, so not for the agent in agentless passive constructions for instance. Moreover this principle cannot be respected to handle control cases, for which there is a cycle in the semantic graph, as in *Bill wants to sleep*. Nevertheless different formal devices can be developed to recover both semantic dependencies between *want* and *Bill* and between *sleep* and *Bill*.

[2] It can be noted that it was done in other formalisms of the TAG family such as DTG (Rambow *et al.* 1995).

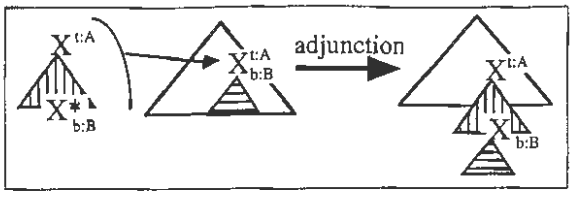


Fig. 2. Adjunction and top and bottom features

Determiner. In TAG, it is usual to consider that the determiner adjoins on the noun, which gives us the right semantic dependencies. Nevertheless, in usual TAG, this analysis needs to attribute the same categories to a phrase with and without a determiner and to distinguish them by a special feature (generally called [det]). It is now possible to use different categories (Fig. 3).

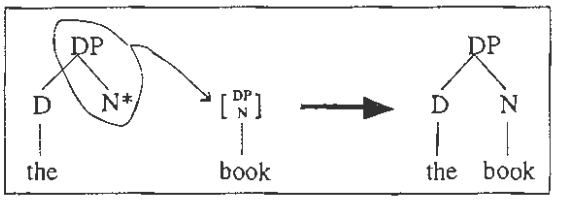


Fig. 3. Determiner's adjunction

Note that it does not change anything here if we use a NP label rather a DP label. In the following, determiners are no longer considered, and a N label will be used for noun phrases (as in Abeillé 1991).

Predicative adjective. Basic adjectives are considered as unary predicates, which adjoin on their semantic argument when they are attributive. Conversely, when they are predicative, their semantic argument substitutes. So in *Peter seems happy*, *Peter*, which is a semantic argument of *happy* and not of *seems*, will substitute in *happy* and *seems* will adjoin in *happy*. The tree α*happy* will thus contain a [VP|A] node on which β*seem* will adjoin. Note that such a category forces the adjunction of a verb. The verb *be* will be treated, in this case, as *seem*, although it is semantically empty.[3]

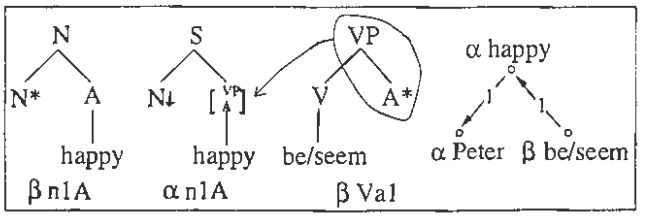


Fig. 4. Derivation tree for *Peter is/seems happy*

Tough-movement. Tough-movement is described in the same way as predicative adjective and the same trees are used for the copulative verb *be* and the raising verb *seem* (Fig. 5 and 6).[4] The

[3] The verbs *be* and *seem* differ not only semantically but syntactically: *Is Peter happy? / Does Peter seem happy?* Even if they share the tree of Fig. 4, they do not share the same family of trees.

[4] We have represented the complement of *easy* as a small clause labeled S. Phrase such *as easy for Mary to read* are described in the same way. The treatment of unbounded tough-movement *(This book is easy for me to believe that John would ever read*, adapted from Bresnan 1982: 255) can also be analyzed; it requires a tree β*for...to believe that* which will adjoin on a special tree α*read* (similar to the tree of Fig. 5, but with a finite S) and on which the tree β*easy* of Fig. 5 will adjoin. To avoid overgeneration, the tree *easy* must specify explicitly that its foot node is a S[(*for*)...*to*]

derivation tree can again be interpreted as a correct semantic graph. Note that *easy* needs different trees in the two constructions considered, which is avoided in GAG/DTG (Candito & Kahane 1998).

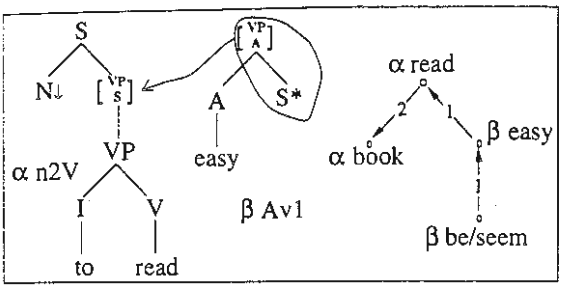


Fig. 5. The derivation of *the book is easy to read*

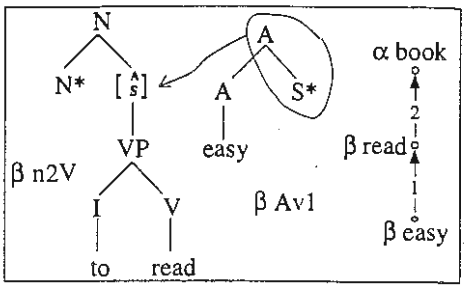


Fig. 6. The deriv. of *a book easy to read*

3. Extractions

We will consider a case of pied-piping in French:

(2) *Marie connaît la fille à la mère de qui Pierre parle.*
M. knows the girl to the mother of which P. talks.

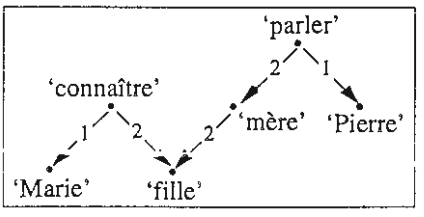


Fig. 7. The semantic graph of (2)

Three solutions will be considered. In the first one (Fig. 8), the verb *parle* 'talk' and the wh-word *qui* 'which' co-anchor a tree β*à qui-parle*, which adjoins on the antecedent *fille* 'daughter'. To obtain (2), β*mère* must adjoin on β*à qui-parle*. In this case, the derivation tree cannot be satisfactorily interpreted as a semantic graph, because *parle* 'talk' is not the semantic argument of *mère* 'mother'. Nevertheless, this is a good solution from a weak generative capacity viewpoint.

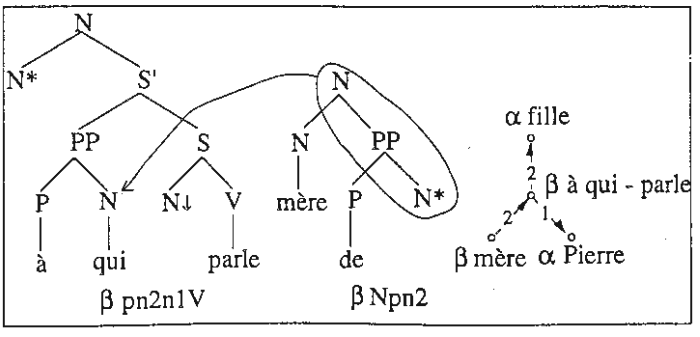


Fig. 8. A first (non suitable) derivation for (2)

The second solution (Fig. 9) is adapted from Kroch 1987 and is adopted by all the studies we know in TAG. The tree β*à qui-parle* of the first solution is broken in two trees: a tree β*parle*, which still adjoins on the antecedent, and a tree α*qui*, which substitutes in it.

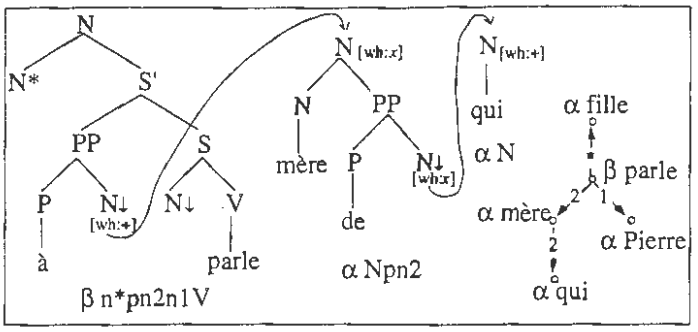


Fig. 9. A second possible derivation for (2)

In this solution, *mère* is the semantic argument of *parle*, but there is also an adjunction arc between β*parle* and the antecedent that cannot be interpreted as a semantic dependency. Moreover, a feature [wh] is necessary to ensure that the noun phrase that substitutes in the extracted position of β*parle* contains a wh-word. So a wh-word must be [wh:+] and a tree such as α*mère* must have two coreferent features [wh:*x*]. To avoid that a noun phrase without a wh-word substitute on a [wh:+] position, a noun must be [wh:-].

The idea of the third solution (Fig. 10) is to break the tree β*à qui-parle* of the first solution in another way. Following Tesnière 1959, we consider that the wh-word plays two roles: on one hand, it fills a position in the relative as pronoun and on the other hand it controls the distribution of the relative. If we follow this idea, it is more natural to attach the power to adjoin on a noun to the wh-word than to the verb of the relative. The adjoining arc between β*qui* and the antecedent (labeled =) can be interpreted as a link of coreference which can be collapsed to keep only the semantic dependencies.

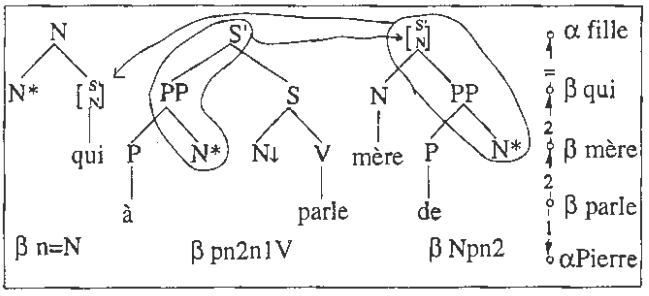


Fig. 10. A third (more suitable) derivation for (2)

As we see, β*parle*, which have a top node of top category S' and a foot node of bottom category N, can adjoin on the node of category [S'|N] of β*qui*. In addition to the fact that this analysis gives us the right semantic dependencies, there is another advantage: the same trees β*parle* and β*mère* can be used for other extractions, such as topicalization and direct or indirect interrogatives:

(3) **a.** *A la mère de Marie, Pierre parle.*
To the mother of Mary, Peter talks.
b. *Marie sait à la mère de qui Pierre parle.*
M. knows to the mother of which P. talks.

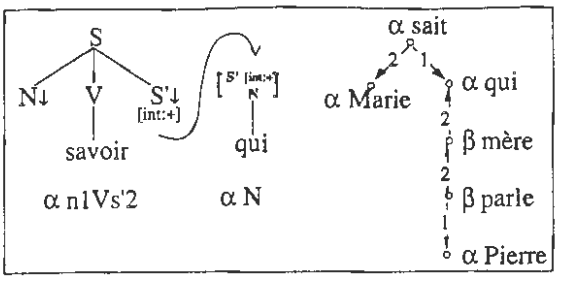


Fig. 11. Derivation of (3b)

This solution makes it possible to handle constructions that cannot be described in the Kroch 1987 analysis, without using multi-component TAG. That is the case of French *dont*-relative where a noun complement of a subject or a direct object is extracted:

(4) *le livre dont Pierre aime la fin*
the book of-which Peter likes the end
'The book whose end Peter likes'

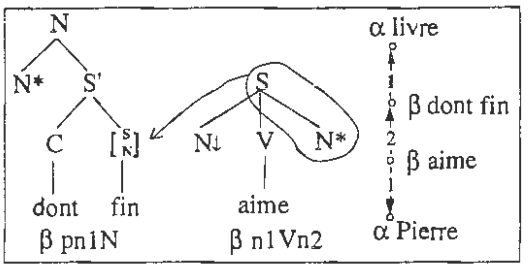


Fig. 12. Derivation of (4)

English sentences with extraction out of a noun complement can be analyzed in the same way:

(5) **a.** *the girl who Peter painted (a copy of) a picture of*
b. *Peter painted (a copy) of a picture of this girl*

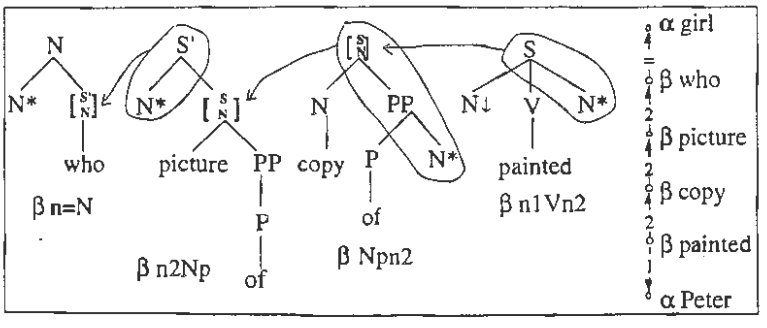


Fig. 13. Derivation of (5a)

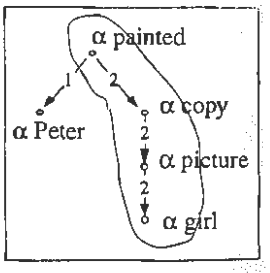


Fig. 14. Derivation of (5b)

We will now give an analysis of a well known and puzzling construction in French (Kayne 1975). As it can be seen in (6), the extraction of a subject phrase out of subordinate clause is possible, but only with a strange alternation of the complementizers:

(6) **a.** *le type qui dort*

the guy who is-sleeping

b. *Je pense que ce type dort*
I think that this guy is-sleeping

c. *le type que je pense qui dort*
the guy that I think [that] is-sleeping

d. * *le type qui je pense que dort*

Our analysis is based on the following assumptions:

1) *que* and *qui* are two forms of a same lexeme *qu-* : *qui* = $qu\text{-}_{[nom:+]}$ and *que* = $qu\text{-}_{[nom:-]}$.
2) A phrase of category S' must contain one and only one term in the nominative case: it is either the subject of the verb or, if the subject is extracted, the complementizer. For this reason, the two subconstituents of an S' must bear [nom] features with opposite values.

In other words, our analysis supposes that a subject can be extracted, but not the nominative case borne by it. In conformity with our assumption that a complementizer is attached to the semantic governor of the link that it marks, the wh-word *qu-* introducing the relative clause co-anchors the tree of a verb whose subject has been extracted (tree βn1qu-V, Fig. 14), which is the semantic governor of the antecedent noun. If no bridge verb is inserted, as in sentence (6a), *qu-* becomes [nom:+] and is realized by *qui*, else it becomes [nom:-] and is realized by *que*, as in sentence (6c). Conversely, the complementizer *qu-* that introduces the subordinate clause subcategorized by the bridge verb *pense* 'think' co-anchors the tree β*pense*. If the bridge verb adjoins on a verb with a subject, as in (6b), *qu-* becomes [nom:-] and is realized by *que*, while it becomes [nom:-] and is realized by *qui* if it adjoins on a verb whose subject has been extracted, as in (6c). Our solution differs from Franck 1992:173, where the complementizers are not attached to the semantic governors and it is not possible to use the same elementary trees to derive the sentences (6a-c).

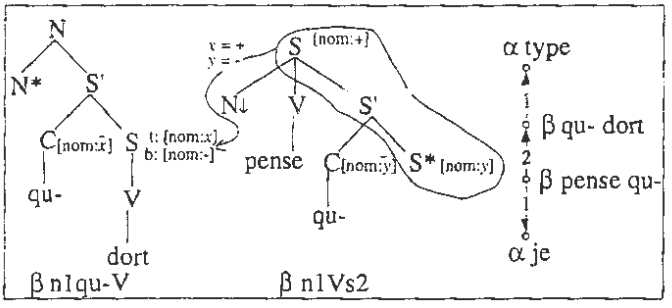


Fig. 14. Derivation of (6a) and (6c)

4. Conclusion

The main attraction of Kroch's analysis is its ability to derive a variety of constraints on extraction. Our analysis retains this particularity and even extends it to pied-piping cases. Extractions are a case of mismatch between syntactic and semantic dependencies: the syntactic head of a relative clause—the main verb of the clause—, which syntactically depends on the antecedent, is generally not semantically linked to the antecedent (e.g. *parle* in (2), *aime* in (4) or *pense* in (6c)). As proposed in Kahane & Mel'çuk 1999, the constraints on extraction can be expressed on the string of syntactic dependencies between the syntactic head of the clause following the extracted element and the gap. One particularity of the TAG description concerns this string: in case of extraction, the hierarchy induced by the derivation tree on this string is the converse of the hierarchy in the syntactic dependency tree, which is also the hierarchy generally adopted for a derivation without extraction (compare Fig 13 and 14). For this reason, all the string of nodes between the syntactic head and the gap is realized by predicative trees. Moreover, these trees have the following characteristics: the nodes that have been piped and are in COMP (*mère* in (2), Fig 10 will receive a

predicative tree rooted by S' without S node, while the node which is linked to COMP (e.g. *parle* in (2)/Fig. 10; *picture* in (5a)/Fig. 13) will receive a predicative tree rooted by S' with a S node. The nodes that are between the node linked to COMP and the syntactic head of the relative will receive a predicative tree rooted by S.[5] And the converse is true. In other words, a lexical unit can be in one of the three positions considered in the string between the syntactic head and the gap if it has a tree of one of three types proposed.

Although our analysis handles more extractions than Kroch 1987's analysis, some constructions still cannot be suitably described. For instance, problems arise when one of the dependencies between the syntactic head and the gap is a substitution arc: it is the case for extractions outside an interrogative clause (*le livre* ***que*** *je sais à qui* ***donner*** 'the book that I know to which to-give': αlivre <-2- βque donner **-3->** αqui <-2- βsais) or extractions where the wh-word is a modifier in the relative and might be both adjoined in the relative and on the antecedent (*the guy* ***whose*** *car I borrowed*: αguy <-1- βwhose **-2->** αcar <-1- βborrowed).[6] In both cases, the tree which substitutes (α*qui* or α*car*) is not in an adequate position for the tree that might adjoin on it. All these problems can be avoided in GAG/DTG where multiple adjoining and substitution of a same elementary tree are possible (Candito & Kahane, 1998). For instance, the wh-word *where* will receive an elementary structure which can adjoin simultaneously on the antecedent *bed* and on the verb *slept* it modifies. Similarly, the wh-word *qui* in (2) will receive an elementary structure that can simultaneously adjoin on its antecedent and substitute in the relative clause. But contrary to Kroch's analysis and our analysis, constraints on extraction are not directly assumed by the categorial features of nodes and special features must be added for not overgenerating.

References

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

CANDITO M.-H. (1999) *Organisation modulaire et paramétrable de grammaires électroniques lexicalisées*. Ph.D. Thesis. Univ. Paris 7.

CANDITO M.-H. & KAHANE S. (1998). "Defining DTG derivations to get semantic graphs". *TAG+4*, Philadelphia, 25-28.

FRANCK R. (1992). *Syntactic Locality and Tree Adjoining Grammar : Grammatical, Acquisition and Processing Perspectives*. Ph.D. Thesis. Univ. of Pennsylvania.

KAHANE S. & MEL'CUK I. (1999). "La synthèse sémantique ou la correspondance entre graphes sémantiques et arbres syntaxiques – Le cas des phrases à extraction", *T.A.L.*, **40**:2, 25-85.

KAYNE R. (1975). *French Syntax : The Transformational Cycle*, MIT Press.

KROCH A. (1987). "Subjacency in a Tree-Adjoining Grammar". In Manaster-Ramer A. (ed), *Mathematics of Language*, Benjamins, 143-71.

KROCH A. & JOSHI A. (1986). "Analysing Extrapositions in a TAG". In Huck G. & Ojeda A. (eds), *Discontinuous Constituents, Syntax and Semantic 20*, Academic Press, 107-49.

MEL'CUK I. (1988). *Dependency Syntax : Theory and Practice*. State Univ. of NY Press.

RAMBOW O., VIJAY-SHANKER K. & WEIR D. (1995). "D-tree Grammars". *ACL'95*.

TESNIERE L. (1959). *Eléments de syntaxe structurale*, Klincksieck, Paris.

VIJAY-SHANKER K. (1987). *A Study of TAG*. PhD Thesis, Univ. Pennsylvania, Philadelphia.

VIJAY-SHANKER K. (1992). "Using descriptions of trees in TAG". *Computational Linguistics*, **18**:4, 481-517.

XTAG Research Group (1995). "A Lexicalized TAG for English". Technical Report IRCS 95-03, Univ. of Pennsylvania. (On line updated version).

ZOLKOVSKI A. & MEL'CUK I. (1967). O semanticeskom sinteze [On semantic synthesis]. *Problemy kybernetiki*, **19**, 177-238.

[5] The only exception to this principle concerns the raising verbs, such as *seem*, which receive a tree rooted by VP; but as noted in Candito & Kahane 1998, it is not always possible to obtain the right dependencies with such trees.

[6] Note that the phrase *the bed* ***where*** *I slept* (αbed <-1- βwhere **-2->** αslept) can be analyzed, but with a rather strange tree for *where*, where the clause modified by *where* must substitute in the tree β*where*.

How to solve some failures of LTAG

Sylvain KAHANE
LaTTiCe – TALaNa (Université Paris 7)
sk@ccr.jussieu.fr

Abstract.

The paper presents a lexicalized dependency grammar which solves some failures of Lexicalized TAGs, such as the combinatorial explosion of the number of elementary trees and the non adequacy for the analysis of some constructions.

Introduction

Wide coverage grammars for natural languages have been developed in Lexicalized TAG (cf. Abeillé 1991, Candito 1999 for French and Paroubek *et al.* 1992, XTAG 1995 for English). These implementations have brought to the fore some failures of the formalism for natural language description which cannot be solved without adopting a descriptively more powerful formalism. These failures concern most of lexicalized grammars, including Categorial Grammars (CG). In this paper, we will present some of these failures and propose solutions in a lexicalized dependency grammar based on Nasr 1995, 1996.

1. Lexicalized grammars

An LTAG is a particular case of **lexicalized grammar** (LG). A LG is a formal grammar that has the form of a lexicon: each lexical unit is associated to a set of elementary structures. The grammar has an operation of combination[1] and each sentence (= a string of word) can be associated to set of structures obtained by combinations of elementary structures associated to the words of the sentence.

Formally, a LG is a 5-uple $G = < \mathcal{L}, S, S_F, \varphi, c >$ where:

- $\mathcal{L}$ is the lexicon;
- S is the set of structures; it is an infinite set but it must be finitely defined;
- S_F is the subset of S of final structures;
- φ is a many-to-many map from $\mathcal{L}$ to S;
- c is the operation of combination of structures; it is a many-to-many map from $S \times S$ to S.[2] Below $c(\alpha,\beta)$ will be noted $\alpha.\beta$.

The operation c induces an operation c^* from S^* to S which associates to a sequence of structures of S all the structures of S obtained by combination of these structures. For instance, $c^*(\alpha,\beta,\gamma)$ is all the structures obtained by the combinations $(\alpha.\beta).\gamma$ and $\alpha.(\beta.\gamma)$. The grammar G defines a correspondence (= many-to-many map) φ^* between $\mathcal{L}^*$ and S_F: a sentence $u = x_1 x_2 \ldots x_n$ in $\mathcal{L}^*$ and a structure S in S_F are in correspondence if for each word x_i there is a structure $S_i = \varphi(x_i)$ such that $c^*(S_1,S_2,\ldots,S_n) = S$.

---

[1] Most of formalisms consider several operations of combinations (e.g. substitution and adjoining in TAG), but we can suppose that there is only one, which is the union of all of them.
[2] We do not exclude that two structures can combine in several ways.

We will now present an LG adapted from Nasr 1995, 1996, which we call Lexicalized Dependency Grammar (LDG). The set of structures S of LDG is a set of dependency trees (Tesnière 1959, Mel'čuk 1988); nodes are labeled by a lexical unit, its part of speech and some grammatical features (not considered here), while branches are labeled by a syntactic relation and a weight (see below). Moreover each label contains a feature `type` with value 0 (= white) or 1 (= black) such that $0 \cup 0 = 0$, $0 \cup 1 = 1$ and $1 \cup 1 =$ `failure`. S_F is the subset of dependency trees in S whose all nodes and branches are black, that is have the value `type:1`. The feature `type` ensures that each element is build one and only one time: black elements can be considered as element which are build and white elements, as requests.

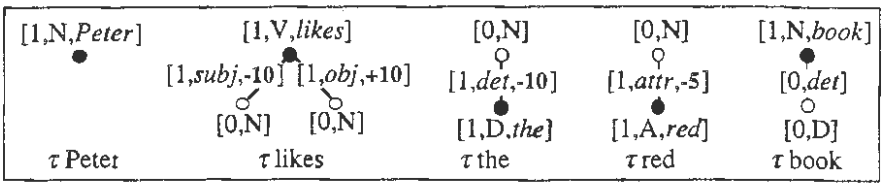


Figure 1. Elementary trees

In the plain case, elements of S combine by unification of one node. In some cases, several nodes and branches can unify (e.g., the combination of the tree of *book* with the tree of its determiner *the*, Fig.2). The feature `type` allows a black element to unify only with a white element.

A sentence u **corresponds** to a tree T of S_F if:

- the nodes of T are labeled by the words of u and **correspond one-to-one** to them;
- the product structure T×u, that is the tree T with the linear order on the nodes induced by u, is a **projective** ordered tree (no arcs cross each other and no arc covers the root);
- the **local order constraints** given by the weights on the branches are respected: the sign of the weight (- or +) indicates if the dependent is before or after the governor and the absolute value of the weight indicates the relative distance between the dependent and the governor.

Fig. 2 shows the dependency tree resulting from the combination of the elementary trees of Fig. 1 and the correspondence between this tree and the sentence *Peter likes the red book*.

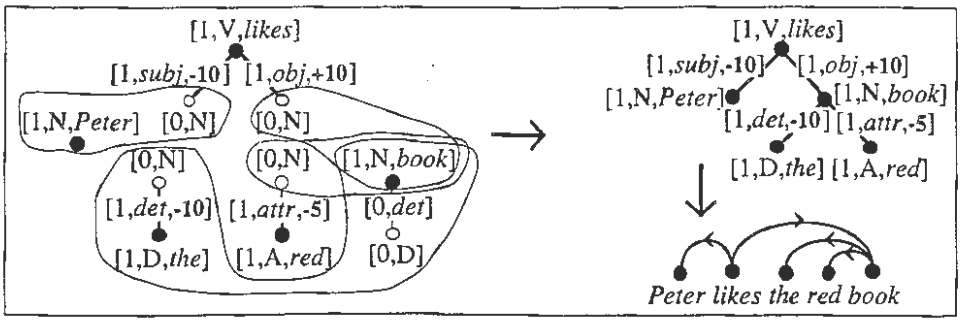


Figure 2. Combination

2. Avoid the combinatorial explosion of the number of elementary trees

The first failure of LTAGs is certainly the combinatorial explosion of the number of elementary trees associated to a given lexical unit. Due to the fact that for each non-canonical position of an

argument (deletion, topicalization, inversion, relativization, cliticization, raising, heavy shift...) a different tree is necessary and due to the crossing with the different arguments, several hundred of elementary trees can correspond to a same lexical unit. Tools have been proposed to write grammars in formalisms which avoid redundancies and allow to generate an LTAG (cf. Vijay-Shanker & Schabes 1992, Candito 1996, 1999). But such formalism—called a **metagrammar** in Candito 1999—cannot be used directly as a grammar and must be compiled into a LTAG before using. And due to the great number of elementary trees, LTAG parsers are not very efficient and consume a lot of space memory.[3] Our proposition consists to propose a lexicalized grammar which has more or less the property of a metagrammar, but which can be used directly as an LTAG.

We claim that the number of elementary trees associated to a lexical unit depends on two factors:

1) the repartition of the linguistic information;

2) the expressiveness of S and the powerfulness of c.

We will now study some examples and propose solutions with our LDG.

Attribute and predicative adjectives. In LTAG, adjectives receive two different elementary structures for their attribute and predicative uses. Compare *the red book* and *the book is red*. The LTAG's elementary tree of the attributive *red* has a nominal foot node in order to adjoin on a noun (here *book*), while the LTAG's elementary tree of the predicative *red* has a nominal substitution node (occupied here by *book*) and a verbal node where the copulative verb will adjoin. But the particular behavior of predicative adjective can be attributed to the copulative verbs rather than to the adjective and a same elementary tree should be attributed to attributive and predicative adjectives. But the TAG formalism is not powerful enough for that. Our LDG can be enriched to solve this problem. We consider a new type of branches, called **quasi-dependency**, with a feature `+quasi`. Quasi-dependencies do not intervene in the tree hierarchy nor in the linearization (they do not bear a weight), but they can unify with a true dependency (the result is still a quasi-dependency). The elementary tree τred (Fig.1), which is used for attributive constructions (Fig. 2), can also be used for predicative constructions. In this case, the *attr* dependency adjoins with a quasi-dependency of the elementary tree of the copula (Fig. 3). In other words, we have given the copula the power to absorb this dependency and to give another syntactic governor to *red* than the noun governing it in its elementary tree. The problem has been solved by adopting a different repartition of linguistic information (properties of predicative constructions are attached to the adjective to the copula, rather than to the adjective as in LTAG), which was made possible by an enrichment of the formalism.[4]

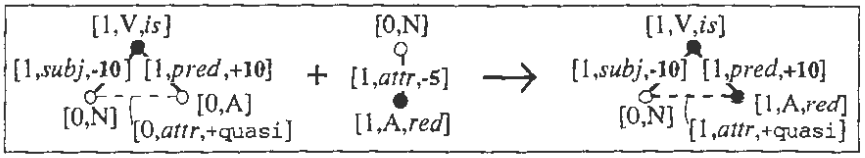


Figure 3. Derivation of [*The book*] *is red*

[3] Parsing algorithm for LTAG have time complexity in $C.|G|^2.n^7$ and space complexity in $C.|G|.n^3$, where $|G|$ is the size of G, that is the number of elementary structures.

[4] The problem can also be solved in CG: the noun *book* will receive the category N and the adjective *red* the category N/N, in order to adjoin on the noun. Then the copula receives the category N\S/(N/N). Nevertheless, CG presents some failures; in paticular, CG has not a convenient treatment of adjoining. For instance, if we want to specify that a noun must have a determiner we will give it the category D\N, but, in this case, *red* must receive the category (D\N)/(D\N). And if several categories are considered for nouns, several categories must be considered for *red*. Another point: at first view, CG is not exactly a lexicalized grammar in the sense considered here, because the combination of categories does not build. But a structure can be derived from the reduction process or categories can be enriched with lambda terms whose combination gives a semantic structure.

Non-canonical position. In LTAG, all the arguments are positioned in the elementary tree of their governor. But the particular behavior of some elements (wh-words, clitics...) might be attributed to them rather than to their governor. And again the TAG formalism is not sufficiently powerful for that.

Our LDG can be enriched to solve this problem. We consider a new type of feature value, called **priority value**: rather to unify with another value, a priority value replaces it. Fig. 4 proposes a solution for clitics in French. Object clitic *le* is positioned before the verb and the relative order of clitics is very constrained (roughly *se* < *le* < *lui* < *en* < *y*). Therefore, the clitic *le* will receive an elementary tree with a white *obj* governor dependency bearing a priority weight of –4; consequently, the clitic *le* can only combine with an *obj* request and its priority weight value will ensure its correct positioning. In our figures, priority values are underlined.

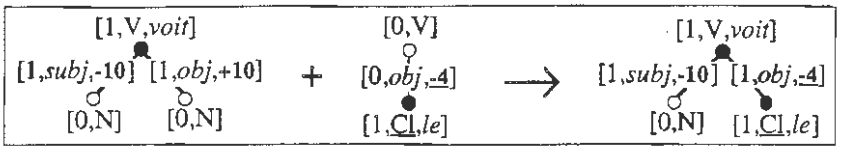


Figure 4. Derivation of Fr. [*Pierre*] ***le voit*** 'Peter sees it' (first proposal)

Non projective constructions. Our first proposal for clitics operates only for projective case, that is when the clitic is on the word that subcategorizes it. We will propose here a solution for **clitic climbing** in French:

(1) *Pierre l'a vu*, lit. Peter LE has seen 'Peter has seen it'

(2) *Pierre en aime la fin*, lit. Peter EN likes the end, 'Peter likes the end of it'

Case (1) is solved in LTAG by adjoining the auxiliary verb *a* 'has' on the past participle *vu* 'seen' (Abeillé 1991). It is not satisfactory because the auxiliary is the syntactic head of the clause; for instance, it receives the negation *ne...pas*: *Pierre ne l'a pas vu*, Peter NE LE has not seen, 'Peter has not seen it'. This last sentence cannot be satisfactorily derived in LTAG, because the clitic *ne*, which is borne by the auxiliary, cannot adjoin on it because of the clitic *le*, which is on the tree of the past participle. The case of (2) is even more problematic: the only way to solve it is to use set-local multi-component TAG (Bleam 1994).

Our solution is inspired from Hudson 2000 and can be compared to the Slash analysis: the clitic is lifted from its **syntactic governor** (the word which subcategorizes it) to its **linear governor** (the word on which it positions). As the dependencies are used for the linearization, the clitic must depends on its linear governor by a true dependency (with a weight), while the dependency with its syntactic governor (in the elementary tree of its syntactic governor) must become a quasi-dependency. For these reasons, the elementary structure of a clitic has a dependency labeled ***aff(ix)*** linked to its linear governor, which ensures its good linearization, and a quasi-dependency linked to its syntactic governor, which must unify with the request of its syntactic governor (Fig. 5).[5] The most difficult problem is to ensure that the clitic climbs on the good node. The lifting is controlled by a bubble, labeled β, containing both syntactic and linear governors of the clitic. We assume that a dependency on a node of a β bubble will be contained in the β bubble if and only if it is labeled iβ. Therefore, when the clitic's elementary tree τ*le* combines with the auxiliary verb's elementary tree τ*a*, the *aux* dependency of τ*a*, which is labeled iβ, must be contained in the bubble β. Moreover the

[5] Note the particular treatment of the past participle: it has a subject but this subject is linked by a quasi-dependency. This quasi-dependency unifies with the quasi-dependency of the auxiliary elementary tree. The "subject" of the past participle cannot be realized (and linearized) without being linked to the tree by a true dependency.

tree *tle* indicates that the linear governor of the clitic must be a finite or infinite verb. As the syntactic governor is not finite or infinite, the clitic climbing is needed.

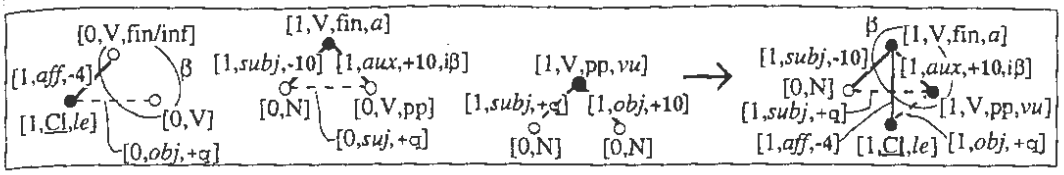


Figure 5. Clitic climbing (derivation of (1))

Even when there is no climbing, the same elementary tree can be used for the clitic: in this case, the two nodes of the β bubble unify and there is a dependency and a quasi-dependency between the clitic and its (syntactic and linear) governor (Fig. 6).

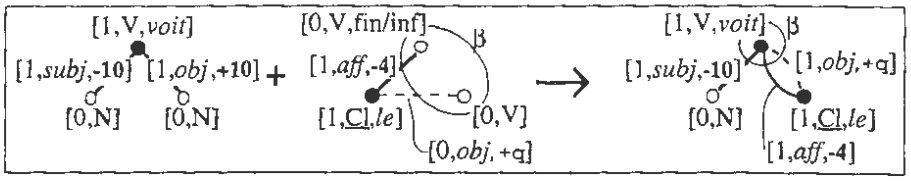


Figure 6. Derivation of Fr. [*Pierre*] ***le voit*** 'Peter sees it' (second proposal)

Let us come back to the problem of the negation *ne...pas*, which cannot be solved satisfactorily in TAG. The negation simply adjoins to the finite verb, *ne* with a weight -5 and *pas* with a weight +2.

Kahane 2000 proposes a similar solution for extractions.

3. Syntax and semantics

One of the main interest of LTAG is that the derivation tree can be interpreted as a semantic graph (= predicate-argument structures) (Candito & Kahane 1998a). To allow such an interpretation, some principles are required: the lexical nodes of an elementary tree must correspond to exactly one semantic unit (Abeillé 1991) and the non-lexical leafs of an elementary tree corresponds one-to-one to the arguments of this semantic unit (predicate-argument co-occurrence principle). But a strict application of this principle is too strong: for instance, it forbids that a syntactic element such as a copulative verb or a complementizer anchors its own tree. In the same way, it forbids that a lexical unit combines with a syntactic argument which is not a semantic argument such as the subject of a raising verb (such has *Peter* with *seems* in *Peter seems to be sleeping*). Such principles forbid also having a separate tree for the copulative verb, which is semantically empty.

Our solution consists in establishing the semantic connection, as in LTAG, while keeping the syntactic connections. In this case, it becomes necessary to indicate explicitly the semantic connection. For this reason, each node receives a `sem` feature, whose value is the semanteme corresponding to the word, and an `arg` feature, whose value is the list of the semantemes of its arguments. The elements of this list are equal to the `sem` values of the argument, which is indicated in the elementary tree by shared values.

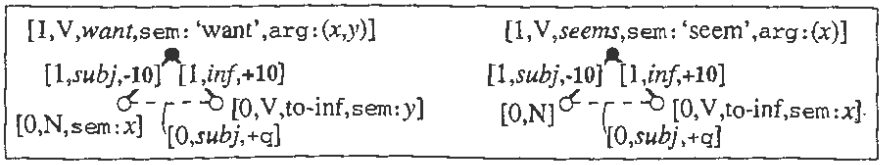


Figure 7. Control verb and raising verb

Fig. 7 gives us the elementary trees of the control verb *want* and of the raising verb *seem*; they have the same syntactic trees (in particular both have a syntactic subject) but they differ semantically: only the control verb has its syntactic subject as semantic argument. Moreover, our formalism allows recuperating directly the semantic dependencies even when there is a cycle (Fig. 8), which TAG cannot allow us.

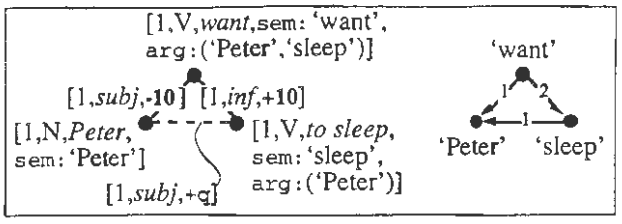


Figure 8. The structure and the corresponding semantic graph of *Peter wants to sleep*

4. Conclusion

Our conclusion is that the TAG formalism is not powerful enough to reach the objectives of computational and linguistic adequacies required to it. Nevertheless, it is possible to develop near formalisms which reach these goals, as well as they keep its advantages, such as lexicalization, simplicity of the operation of combination or readability of the elementary structures.

5. References

ABEILLÉ Anne (1991) : *Une grammaire lexicalisée d'Arbres Adjoints pour le français*, Thèse de Doctorat, Univ. Paris 7.

BLEAM Tonia, (1994) : "Clitic Climbing and the Power of Tree Adjoining Grammar", in *Symposium on TAG*, Paris. To appear in Abeillé A. & Rambow O., *Tree Adjoining Grammar*, CSLI.

CANDITO Marie-Hélène (1996) : "A Principled-based Hierarchical Representation of LTAG", *COLING'96*, Copenhague, pp. 194-99.

CANDITO Marie-Hélène (1999) : *Organisation modulaire et paramétrable de grammaires électroniques lexicalisées. Application au français et à l'italien*, Thèse de Doctorat, Univ. Paris 7.

CANDITO Marie-Hélène & KAHANE Sylvain (1998a) : "Can the TAG Derivation Tree represent a Semantic Graph ? An Answer in the Light of Meaning-Text Theory", *TAG+4*, Philadelphie, pp. 25-28.

CANDITO Marie-Hélène & KAHANE Sylvain (1998b): "Defining DTG Derivations to get Semantic Graphs", *TAG+4*, Philadelphie, pp. 25-28.

GERDES Kim (1998) : *Le cas allemand en TAG*, Mémoire de DEA, Univ. Paris 7.

HUDSON Richard (2000) : "Discontinuity", *Special Issue on Dependency Grammar, T.A.L, 41:1*, , Paris, 38p.

KAHANE Sylvain (1997) : "Bubble Trees and Syntactic Representations", in Becker & Krieger (eds), *Proc. MOL'5*, Saarbrücken : DFKI, 70-76.

KAHANE Sylvain (2000) : "Une grammaire de dépendance à bulles pour traiter l'extraction", *Special Issue on Dependency Grammar, T.A.L, 41:1*, Paris, 30p.

MEL'CUK Igor (1988) : *Dependency Syntax: Theory and Practice*, NY : State Univ. of NY Press.

NASR Alexis (1995) : "A Formalism and a Parser for Lexicalised Dependency Grammars", *4th Int. Workshop on Parsing Technologies*, State Univ. of NY Press.

NASR Alexis (1996) : *Un modèle de reformulation automatique fondé sur la Théorie Sens-Texte – Application aux langues contrôlées*, Thèse de Doctorat, Univ. Paris 7.

PAROUBEK Patrick, SCHABES Yves & JOSHI Aravind K. (1992) : "XTAG; a graphical Workbench for developing TAGs", *ANLP*, Trento, 223-27.

RAMBOW Owen (1994) : *Formal and Computational Aspects of Natural Language Syntax*, PhD Thesis, Univ. Of Pennsylvania, Philadelphia.

TESNIÈRE Lucien (1959) : *Eléments de syntaxe structurale*, Paris : Kliencksieck.

XTAG Research Group (1995) : *A Lexicalized TAG for English*, Technical Report IRCS 95-03, Univ. of Pennsylvania, (updated version on the web).

Scrambling in German and the non-locality of local TDGs

Laura Kallmeyer

SFB 441, University of Tübingen
lk@sfs.nphil.uni-tuebingen.de

Abstract

Existing analyses of German scrambling phenomena within TAG-related formalisms all use non-local variants of TAG. However, there are good reasons to prefer local grammars, in particular with respect to the use of the derivation structure for semantics. Therefore this paper proposes to use local TDGs, a TAG-variant generating tree descriptions that shows a local derivation structure. However the construction of minimal trees for the derived tree descriptions is not subject to any locality constraint. This provides just the amount of non-locality needed for an adequate analysis of scrambling. To illustrate this a local TDG for some German scrambling data is presented.

1. Introduction

Scrambling in German poses a problem for most grammar formalisms. Neither Tree Adjoining Grammar (TAG, Joshi *et al.*, 1975) nor even linear context-free rewriting systems (LCFRS, Weir, 1988) are powerful enough to deal with scrambling and the free word order in German (see Becker *et al.*, 1992). (Becker *et al.*, 1991) propose a scrambling analysis with non-local multicomponent TAG (MCTAG, Weir, 1988), and (Rambow & Lee, 1994; Rambow, 1994) propose the use of vector TAG (V-TAG). These formalisms are both non-local in the sense that when adding a new element of the grammar in a derivation step, this element is not attached to one single previously added element of the grammar.

There are however good reasons to prefer a local grammar. Firstly, locality often restricts the parsing complexity, and local grammars often generate only semilinear languages. (Though some non-local formalisms (lexicalized V-TAG for instance) also can be shown to be polynomially parsable.) Secondly, in a local grammar, the derivation structure might reflect a dependency structure based on which semantic representations can be built (as for TAGs in Joshi & Vijay-Shanker, 1999; Kallmeyer & Joshi, 1999). In a non-local grammar, the derivation structure does not directly determine a suitable dependency structure. In some formalisms, it is possible to identify parts of elementary structures that are relevant for the dependency structure (e.g. in D-Tree Grammars, Rambow *et al.*, 1995, the relevant part is the part of a d-tree that is substituted in a subsertion operation). But there is not one single structure that records the complete derivation and that is a suitable dependency structure.

As an alternative, I propose to use local Tree Description Grammars (local TDG, Kallmeyer, 1997; Kallmeyer, 1999). Local TDGs generate tree descriptions with a local derivation process. They have a context-free derivation structure and generate only semilinear languages. The descriptions generated by local TDGs allow an underspecification of the dominance relation, and the construction of so-called minimal trees for these descriptions is not subject to locality constraints. This limited amount of non-locality allows to deal with scrambling, as illustrated by a local TDG for some German scrambling and extraposition data.

2. Scrambling: The data

The paper accounts for data like word order variations of (1), taken from (Rambow, 1994).

(1) Weil niemand das Fahrrad zu reparieren zu versuchen verspricht
because nobody the$_{acc}$ bike$_{acc}$ to repair to try promises
because nobody promises to try to repair the bike

Assuming that each NP precedes its verb, we get 30 word orders when combining scrambling with extraposition. According to Rambow, 6 of them are clearly not acceptable. The other 24 also show differences with respect to the judgment, but in principle it should be possible to generate them all. The word orders without extraposition and their judgments are shown in (2). Word orders that are ruled out occur with extraposition of *reparieren* as in (3).

(2) a. ok Weil niemand das Fahrrad zu reparieren zu versuchen verspricht
b. ? Weil das Fahrrad niemand zu reparieren zu versuchen verspricht
c. ok Weil das Fahrrad zu reparieren niemand zu versuchen verspricht
d. ? Weil das Fahrrad zu reparieren zu versuchen niemand verspricht

(3) a. * Weil zu versuchen das Fahrrad niemand zu reparieren verspricht
b. * Weil das Fahrrad zu versuchen niemand zu reparieren verspricht
c. * Weil zu versuchen niemand das Fahrrad zu reparieren verspricht
d. * Weil niemand zu versuchen das Fahrrad verspricht zu reparieren
e. * Weil zu versuchen niemand das Fahrrad verspricht zu reparieren
f. * Weil zu versuchen das Fahrrad niemand verspricht zu reparieren

I will also consider more than two levels of embedding as in (4).

weil das Fahrrad niemand glaubt zu reparieren zu versuchen versprechen
because the$_{acc}$ bike$_{acc}$ nobody thinks to repair to try promise
(4) zu müssen
to need
because nobody thinks it necessary to promise to try to repair the bike

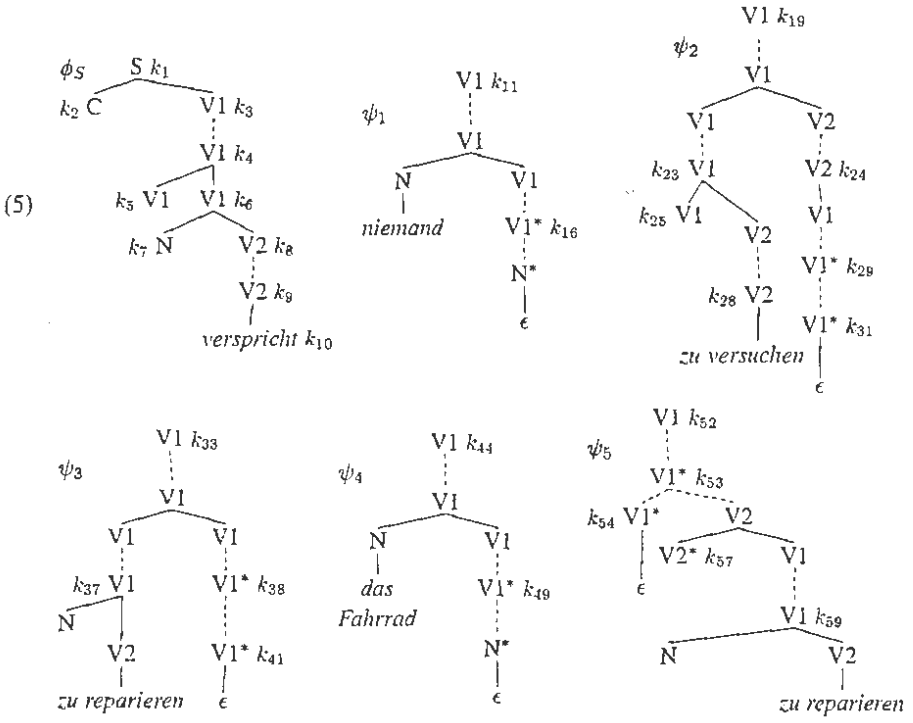
3. A local TDG for scrambling

Local TDGs consist of tree descriptions (*elementary descriptions*) and a *start description*. The tree descriptions are negation and disjunction free formulas in a quantifier-free first order logic. The logic allows to express relations between node names k_1, k_2 such as immediate dominance $k_1 \lhd k_2$, dominance (reflexive transitive closure of $\lhd$) $k_1 \lhd^* k_2$, 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. δ denotes the labeling function, $\delta(k) \approx t$ 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 $\langle a, v \rangle$. Roughly, tree descriptions in a local TDG are fully specified (sub)tree descriptions that are connected by dominance relations.[1] In elementary descriptions, some node names are *marked*; this is important for the derivation. In the graphical representations, marked names are equipped with an asterisk.

(5) shows a local TDG for some scrambling data with $\phi_S = k_1 \lhd k_2 \wedge k_1 \lhd k_3 \wedge k_2 \prec k_3 \wedge k_3 \lhd^* k_4 \wedge \ldots \wedge cat(\delta(k_1)) \approx S \wedge \ldots$ etc. (dotted edges represent dominance relations). Conjuncts as $k_3 \lhd^* k_4$ in ϕ_S not entailed by the rest of the formula are called *strong dominance*.

[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, 1999, Chapter 4).

(5)



The labels V1 and V2 distinguish between VPs not allowing extraposed material to attach (V1) and VPs that allow this (V2). ψ_5 is an elementary description used for an extraposed clause. In the following we will see how the descriptions in (5) combine with each other.

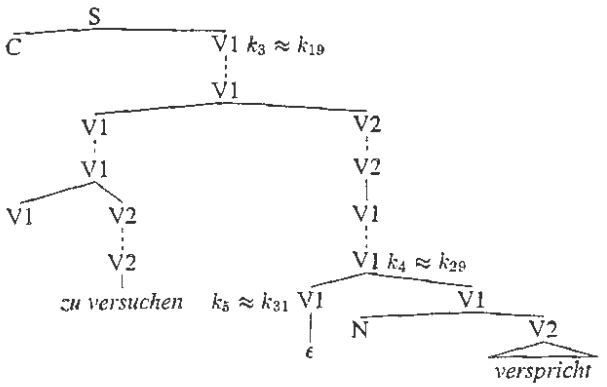
4. Local derivation and underspecification

Derivations start with ϕ_S. In each step, an old ϕ_1 and an elementary ψ are combined to obtain a new ϕ_2. ϕ_2 can be viewed as a conjunction of ϕ_1, ψ and new formulas $k \approx k'$ 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 k_ψ is either marked or minimal (dominated by no other name, e.g. k_1 in ϕ_S in (5)),
2. a marked or minimal k' in ψ that is not a leaf name (i.e. dominates other names) but does not dominate any other marked name becomes equivalent to a leaf name in ϕ_1
3. the names k from ϕ_1 used for new equivalences are part of one single elementary or start description, the *derivation description* of this step (first locality condition),
4. for each marked name k_ψ in ψ with a parent, there is 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 is dominated by k_1 (second locality condition),
5. and the result ϕ_2 is maximally underspecified.

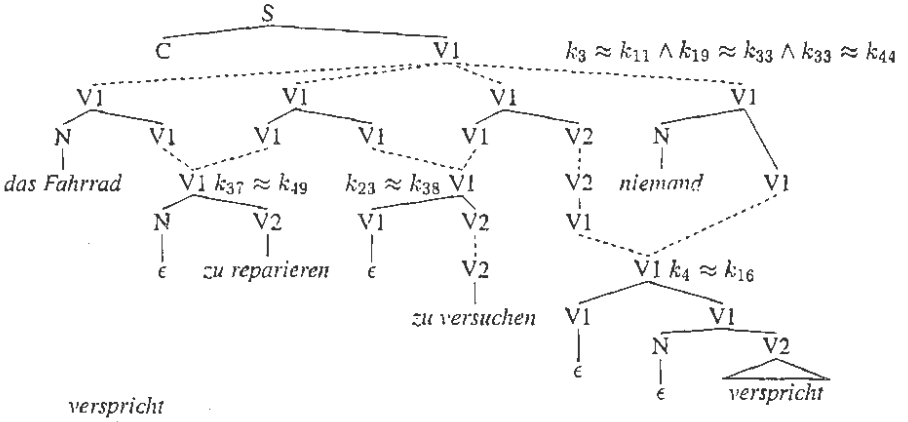
The 3. and 4. condition express the locality of the derivations. They are comparable to the locality constraint on derivations in set-local MCTAG. In fact, for each set-local MCTAG, an equivalent local TDG can be constructed in a straight-forward way (see Kallmeyer, 1999).
As a sample derivation step consider adding ψ_2 to ϕ_S in (5) which leads to $\phi_S \wedge \psi_1 \wedge k_3 \approx k_{19} \wedge k_4 \approx k_{29} \wedge k_5 \approx k_{31}$ in (6).

(6)

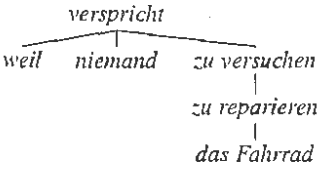


If a marked name has no parent, an underspecification of the dominance can occur. The fifth condition then ensures that the most general solution is generated. E.g., adding ψ_1, ψ_3 and ψ_4 to (6) with derivation descriptions ϕ_S, ψ_2 and ψ_2 respectively gives (7). The derivation structure of (7), shown in (8), is the correct dependency structure.

(7)



(8)



Descriptions generated by a local TDG G denote infinitely many trees. The *tree language* of G contains "minimal" trees of these descriptions. A *minimal* tree γ of a description ϕ satisfies ϕ in such a way that all subtrees of heigth 1 of γ are described exactly once in ϕ. The minimal trees of (7) yield the strings in (2).

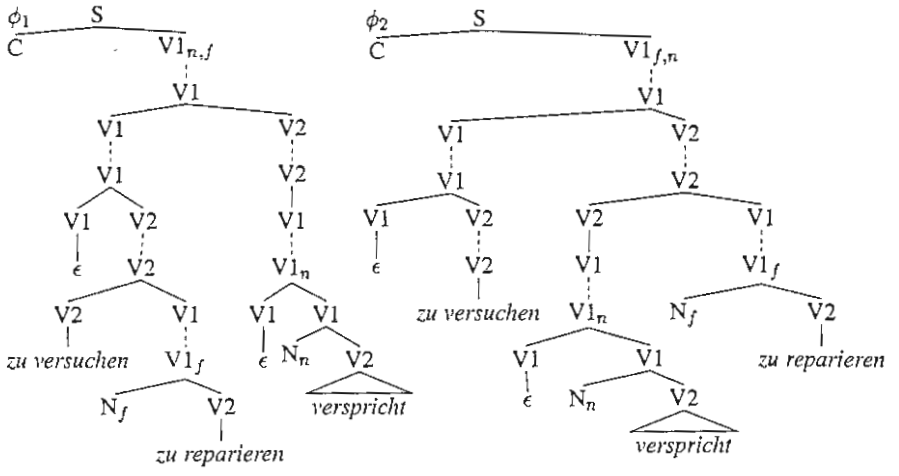
The possibility of underspecification increases the expressive power of local TDGs beyond LCFRS. However, despite this additional power, it is possible to find a context-free derivation grammar and thereby to show that the languages generated by local TDGs are semilinear.

5. Scrambling and extraposition

In ψ_2 there are two attachment sites (label V2) for extraposed clauses, k_{28} and k_{24}. This accounts for the different cases of extraposing *zu reparieren* (i) only past *zu versuchen* and (ii) past *zu versuchen* and *verspricht*. For extraposed VPs, elementary descriptions like ψ_5 for *zu reparieren* in (5) are needed. Adding ψ_5 to (6) with derivation description ψ_2 either leads to ϕ_1 or to ϕ_2 in (9). The subscripts $_n$ and $_f$ mark the names chosen for new equivalences when adding ψ_1 and

ψ_4 for *niemand* and *das Fahrrad* respectively. With ϕ_1, *niemand* is either left of all verbs or between *zu reparieren* and *verspricht*, which excludes (3)a., b. and c. With ϕ_2, *das Fahrrad* is either between *verspricht* and *zu reparieren* or left of all verbs. This excludes (3)d., e. and f.

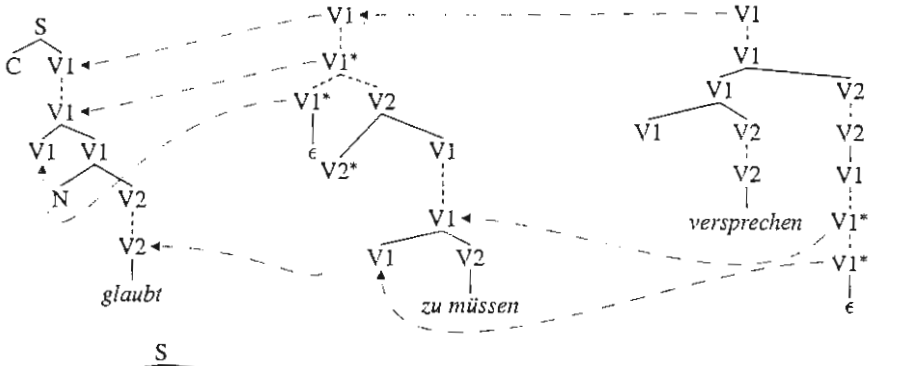
(9)



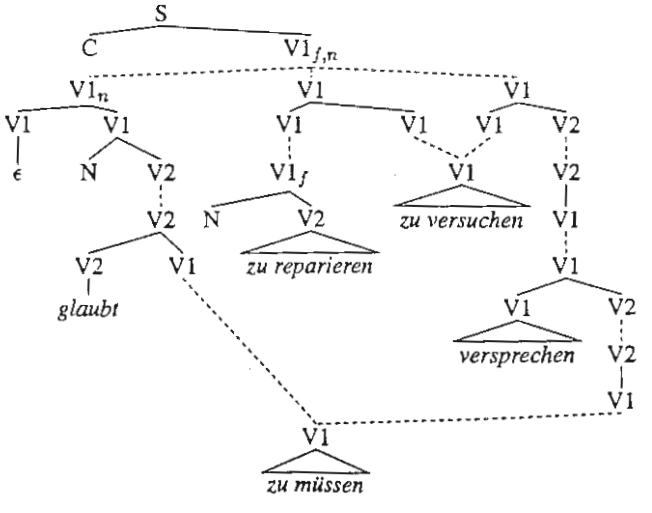
6. More than two levels of embedding

So far, we have considered only examples with up to two levels of embedding. Next, I will consider the analysis of (4), a sentence with four levels of embedding.

(10)



(11)



First, elementary descriptions for *glaubt*, *zu müssen* and *versprechen* are put together as sketched in (10). Then ψ_2 and ψ_3 from (5) for *zu versuchen* and *zu reparieren* are added which leads to (11). Further adding ψ_1 and ψ_4 gives a description that is such that in the minimal trees, *glaubt* is left of *zu müssen*, *zu reparieren* is left of *zu versuchen* which is left of *versprechen*, and *versprechen* is left of *zu müssen*. Furthermore, *niemand* is left of *glaubt* and *das Fahrrad* is left of *zu reparieren*. One of the minimal trees yields (4).

7. Conclusion

This paper addresses the problem that on the one hand, long-distance scrambling in German seems to be non-local in a limited way. On the other hand, there are good reasons to prefer a grammar with a local derivation process that leads to an appropriate dependency structure. I have proposed local TDGs as an alternative to other formalisms previously used to deal with scrambling. Local TDGs have the desired locality property but allow underspecification of the dominance relation. The construction of minimal trees is not subject of any locality constraint. Therefore, local TDGs show a very limited amount of "non-locality", which gives sufficient expressive power to account for scrambling phenomena. This was illustrated by a local TDG analysis of some German data.

Acknowledgments

I would like to thank two referees of this paper for their very valuable comments, which helped to improve the content of this paper.

References

BECKER T., JOSHI A. K. & RAMBOW O. (1991). Long-distance scrambling and tree adjoining grammars. In *Proceedings of ACL-Europe*.

BECKER T., RAMBOW O. & NIV M. (1992). *The Derivationel Generative Power of Formal Systems or Scrambling is Beyond LCFRS*. Technical Report IRCS-92-38, University of Pennsylvania.

JOSHI A. K., LEVY L. S. & TAKAHASHI M. (1975). Tree Adjunct Grammars. *Journal of Computer and System Science*, **10**, 136–163.

JOSHI A. K. & VIJAY-SHANKER K. (1999). Compositional Semantics with Lexicalized Tree-Adjoining Grammar (LTAG): How Much Underspecification is Necessary? In H. C. BLUNT & E. G. C. THIJSSE, Eds., *Proceedings of the Third International Workshop on Computational Semantics (IWCS-3)*.

KALLMEYER L. (1997). Local Tree Description Grammars. In *Proceedings of the Fifth Meeting on Mathematics of Language, DFKI Research Report*, p. 77–84.

KALLMEYER L. (1999). *Tree Description Grammars and Underspecified Representations*. PhD thesis, Universität Tübingen. Technical Report IRCS-99-08, University of Pennsylvania, Philadelphia.

KALLMEYER L. & JOSHI A. K. (1999). Factoring Predicate Argument and Scope Semantics: Underspecified Semantics with LTAG. In P. DEKKER, Ed., *12th Amsterdam Colloquium. Proceedings*.

RAMBOW O. (1994). *Formal and Computational Aspects of Natural Language Syntax*. PhD thesis, University of Pennsylvania.

RAMBOW O. & LEE Y.-S. (1994). Word order variation and Tree-Adjoining Grammars. *Computational Intelligence*, **10** (4 p.), 386–400.

RAMBOW O., VIJAY-SHANKER K. & WEIR D. (1995). D-Tree Grammars. In *Proceedings of ACL*.

WEIR D. J. (1988). *Characterizing mildly context-sensitive grammar formalisms*. PhD thesis, University of Pennsylvania.

Contextual Tree Adjoining Grammars

Martin Kappes

Fachbereich Informatik, Johann Wolfgang Goethe-Universität, D - 60054 Frankfurt am Main, Germany, E-Mail: kappes@psc.informatik.uni-frankfurt.de

Abstract

In this paper, we introduce a formalism called contextual tree adjoining grammar (CTAG). CTAGs are a generalization of multi bracketed contextual rewriting grammars (MBICR) which combine tree adjoining grammars (TAGs) and contextual grammars. The generalization is to add a mechanism similar to obligatory adjoining in TAGs. Here, we present the definition of the model and some results concerning the generative capacity and closure properties of the classes of languages generated by CTAGs.

Introduction

Contextual grammars are a formalization of the linguistic idea that more complex, well formed strings are obtained by inserting contexts into already well formed strings. They were first introduced by Marcus in 1969; all models presented here are based on so-called internal contextual grammars which were introduced by Păun and Nguyen. References and further details about contextual grammars can be found in the monograph (Păun, 1997); a survey is given in (Ehrenfeucht *et al.*, 1997).

Tree adjoining grammars (TAGs) and contextual grammars are linguistically well motivated and have been considered as a good model for the description of natural languages (c.f. (Marcus, 1997)). Although contextual grammars and tree adjoining grammars seem very different at first sight, a closer look reveals many similarities between both formalisms. Therefore, it seems natural to combine those formalisms in order to obtain a generalized class of grammars for the description of natural languages, which combines the mechanisms of various classes. A first step were so-called multi-bracketed contextual grammars (MBIC) and multi-bracketed contextual rewriting grammars (MBICR), c.f. (Kappes, 1999). These grammars operate on a tree structure induced by the grammar (the first approach aiming in this direction was introduced in (Martin-Vide & Păun, 1998)).

However, the families of languages generated by MBIC and MBICR-grammars are either strictly included in or incomparable to the family of languages generated by TAGs. This is the case since, in MBIC and MBICR-grammars, each yield of a derived tree is immediately a word in the language generated by the grammar. In other words, there is no mechanism to distinguish between "finished" and "unfinished" trees like obligatory adjoining allows in TAGs. Here, by adding obligatory adjoining to MBICR-grammars, we obtain a generalized class which is also a proper extension of TAGs.

Definition and Example

Let Σ^* denote the set of all words over the finite alphabet Σ and $\Sigma^+ = \Sigma^* - \{\lambda\}$, where λ denotes the empty word. We denote the length of a string x by $|x|$. In this paper, we use the term derived tree for a tree where the internal nodes are labelled by symbols from a nonterminal alphabet Δ and the leaves are labelled by symbols from a terminal alphabet Σ. We use

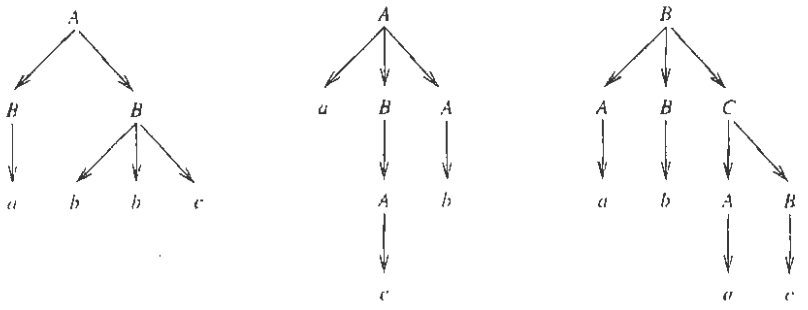


Figure 1: Derived trees corresponding to the Dyck-covered words (from left to right) $[_A[_B a]_B[_B bbc]_B]_A$, $[_A a[_B[_A c]_A]_B[_A b]_A]_A$ and $[_B[_A a]_A[_B b]_B[_C[_A a]_A[_B c]_B]_C]_B$.

a linear representation of derived trees called Dyck-covered words. A Dyck-covered word is a string consisting of terminal symbols and opening and closing brackets indexed with nonterminal symbols. Formally, for the nonterminal alphabet Δ we define the bracket alphabet $B_\Delta = \{[_A,]_A \mid A \in \Delta\}$. Throughout the paper we always assume $\Sigma \cap B_\Delta = \emptyset$. The set of all Dyck-covered words $\mathrm{DC}_\Delta(\Sigma)$ over Σ with respect to the index alphabet Δ is inductively defined by

- For all $w \in \Sigma^+$ and $A \in \Delta$, $[_A w]_A$ is in $\mathrm{DC}_\Delta(\Sigma)$.
- Let $n \geq 1$ be a positive integer. If $A \in \Delta$ and $\alpha_1, \alpha_2, \ldots, \alpha_n$ are in $\mathrm{DC}_\Delta(\Sigma) \cup \Sigma$, then $[_A \alpha_1 \alpha_2 \ldots \alpha_n]_A$ is in $\mathrm{DC}_\Delta(\Sigma)$.

It is not difficult to see that each $\alpha \in \mathrm{DC}_\Delta(\Sigma)$ can be interpreted as unique encoding for a derived tree, where Δ is the label alphabet for the internal nodes and Σ is the label alphabet for the leaf nodes in the following way: A string $[_A \alpha]_A \in \mathrm{DC}_\Delta(\Sigma)$ is identified with a tree where the root is labelled by A, and the subtrees of the root are determined by the unique decomposition of $\alpha = \alpha_1 \alpha_2 \ldots \alpha_n$ such that $\alpha_i \in \mathrm{DC}_\Delta(\Sigma) \cup \Sigma$, $1 \leq i \leq n$. For examples see Figure 1. By $\mathrm{DC}^A_\Delta(\Sigma)$ we denote the set of all elements in $\mathrm{DC}_\Delta(\Sigma)$ where the root node is labelled by A.

A contextual tree adjoining grammar (CTAG) is a tuple $G = (\Sigma, \Delta, \Upsilon, \Omega, P)$, where Σ is a finite set of terminals, Δ is a finite set of indices, $\Upsilon \subseteq \Delta$ is a set of permitted indices, $\Omega \subseteq \mathrm{DC}_\Delta(\Sigma) \cup \{\lambda\}$ is a finite set of axioms and P is a finite set of productions. Each production is of the form (S, C, K, H), where $S \subseteq \Sigma^+$ is the selector language, $K, H \subseteq \Delta$ are sets of nonterminals and C is a finite subset of contexts where each context is of the form (μ, ν) such that $\mu\nu \in \mathrm{DC}_\Delta(\Sigma)$.

The derivation process in a CTAG is illustrated in Figure 2: A context (μ, ν) may be adjoined to an $\alpha = \alpha_1[_B \alpha_2]_B \alpha_3$ yielding a tree $\alpha_1 \mu[_E \alpha_2]_E \nu \alpha_3$ if and only if there is an $(S, C, K, H) \in P$ such that the yield of α_2 is in S, $(\mu, \nu) \in C$, $[_B \alpha_2]_B \in \mathrm{DC}^B_\Delta(\Sigma)$, $B \in K$ and $E \in H$. The string $[_A \alpha_2]_A$ is called selector. In the above figure, we have $\alpha \in \mathrm{DC}^A_\Delta(\Sigma)$, $\mu\nu \in \mathrm{DC}^D_\Delta(\Sigma)$ and the yield of α_1, α_2, α_3, μ, ν is w_1, w_2, w_3, u, v respectively. The set of all sentential forms of G, $S(G)$, consists of all trees which can be derived in the above way starting from an axiom in Ω. The set of all trees derived by a CTAG G, $T(G)$, consists of all trees in $S(G)$ where the internal nodes are only labelled by nonterminals in Υ. The weak generative capacity $L(G)$ is the yield of all trees in $T(G)$. Hence, internal nodes labelled by symbols from $\Delta - \Upsilon$ have to be relabelled during the derivation process in order to obtain a tree in $T(G)$.

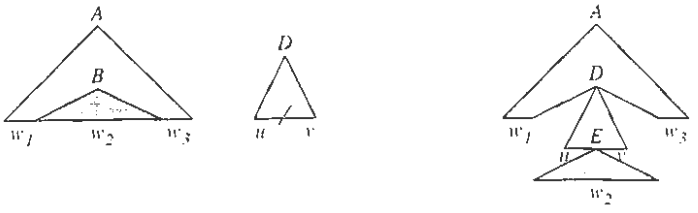


Figure 2: The derivation process in a CTAG

Up to some technical modifications necessary to keep our formalism consistent to the usual model of contextual grammars, we only added selector languages to the productions of a TAG. These selector languages are used to control the derivation process as they do in contextual grammars, the adjunction of an auxilliary tree is only possible if the yield of the node where the adjunction takes place is in the selector language.

We can classify CTAGs by their selector languages: A CTAG $G = (\Sigma, \Delta, \Upsilon, \Omega, P)$ is called with F-choice for a family of languages F, if $S \in F$ for all $(S, C, K, H) \in P$.

Consider for example the CTAG with Σ^+-selection

$$\begin{aligned} G &= (\{a,b,c,d,e\}, \{A,B\}, \{A\}, \{[_A a[_B bc]_B d]_A\}, \{\pi_1, \pi_2\}) \text{ where} \\ \pi_1 &= (\Sigma^+, \{([_A a[_B b, c]_B d]_A)\}, \{B\}, \{A\}) \text{ and} \\ \pi_2 &= (\Sigma^+, \{([_A c, e]_A)\}, \{B\}, \{A\}). \end{aligned}$$

It is not difficult to see that using π_1 i times yields a derivation

$$[_A a[_B bc]_B d]_A \overset{i}{\Longrightarrow}_G ([_A a)^{i+1}[_B b([_A b)^i(c]_A)^i c]_B(d]_A)^{i+1}.$$

In order to obtain a string in $T(G)$ we have to use production π_2 exactly once to remove the pair of brackets indexed by B from the sentential form. After applying π_2 once, no further derivation steps are possible, hence $L(G) = \{a^n c b^n c^n c d^n \mid n \geq 1\}$.

Generative Capacity

CTAGs are a generalization of MBICR-grammars. For $\Delta = \Upsilon$ these models are equivalent (CTAGs could thus also be called multi-bracketed contextual grammars with obligatory rewriting (MBICRO)). The obligatory adjoining feature increases the generative capacity. For instance, the language in the above example cannot be generated by any MBICR-grammar. This is due to the fact that each language L generated by an MBICR-grammar fulfills the so-called internal bounded step property (c.f. (Păun, 1997)): There is a constant p such that for each string $x \in L$, $|x| > p$ there is a $y \in L$ such that $x = x_1 u x_2 v x_3$, $y = x_1 x_2 x_3$ and $0 < |uv| \leq p$.

CTAGs using only the selector language Σ^+, i.e., in effect ignoring the selector language mechanism, and TAGs are, up to some details, descriptions of the same model. It is possible to construct a TAG equivalent to a given CTAG with Σ^+-choice and vice versa. The technical detail is that all elementary trees of a TAG must be elements of $DC_\Delta(\Sigma)$ if the foot nodes of the auxilliary trees are not taken into account. Formally, the equivalence holds if the initial trees in a TAG are elements of $DC_\Delta(\Sigma)$ and each auxilliary tree i of G is of the form $\alpha_i = \mu_i[_{A_i}]_{A_i}\nu_i$ such that $\mu_i\nu_i \in DC_\Delta^{A_i}(\Sigma)$. Notice that the pair $[_{A_i}]_{A_i}$ represents the foot node of α_i. The construction of an equivalent TAG for a given CTAG with Σ^+-choice is a straightforward generalization of

a similar construction for MBICR-grammars which can be found in (Kappes, 1999).

For the other direction, consider a TAG G of the above form. Let X denote the selective (or $\overline{X}$ in case of an obligatory) adjoining constraint of an internal node in an elementary tree. X (or $\overline{X}$) thus dereferences the subset of auxilliary trees which may be adjoined at this node. We can construct an equivalent CTAG $G' = (\Sigma, \Delta', \Upsilon', \Omega', P')$ with Σ^+-choice as follows: The set of indices Δ' and the set of permitted indices Υ' of G' is given by

$$\begin{aligned}\Delta' &= \{(A, \tilde{X}) \mid A \in \Delta \text{ and } \tilde{X} \text{ is a (selective or obligatory) adjoining constraint}\}\\ \Upsilon' &= \{(A, X) \mid A \in \Delta \text{ and } X \text{ is a selective adjoining constraint}\}.\end{aligned}$$

For each initial tree α of G we insert a tree α' into Ω', where each node labelled by $A \in \Delta$ with (selective or obligatory) adjoining constraint $\tilde{X}$ is replaced by the index $(A, \tilde{X})$. We thus consider the adjoining constraint of a node as part of its index. For each auxilliary tree $i : \alpha_i = \mu_i[_A]_A\nu_i$ we insert a production $\pi_i = (\Sigma^+, \{(\mu_i', \nu_i')\}, \{(A_i, \tilde{X}) \mid i \in \tilde{X}\}, \{(A_i, \tilde{Z})\})$ into P' where $\mu_i'\nu_i'$ is obtained from $\mu_i\nu_i$ by the same procedure as above and $\tilde{Z}$ is the (selective or obligatory) adjoining constaint of the foot node of α_i. It is possible to prove that both grammars are equivalent.

It can be shown that each CTAG with finite selection generates a context-free language. This is the case since the length of each string which may be used as selector in a derivation step can be bounded by some constant. Due to the bracket structure it is impossible to shift information through the sentential form of a CTAG if the length of the selectors is finite. Therefore it is possible to construct a context-free grammar generating the same language. Also, for each context-free language there is a CTAG with finite selection generating that language. So, CTAGs with finite selectors generate exactly the context-free languages.

CTAGs with regular selectors can generate languages which cannot be generated by TAGs even if we do not take advantage of the obligatory adjoining feature. The language $L(G) = \{a^n b^n c^m d^n c^n \mid m \geq n \geq 1\}$ can be generated by an MBICR-grammar and hence by a CTAG with regular selector languages (c.f. (Kappes, 1999)) but not by any TAG because of the pumping-lemma for TAGs (cf. (Vijay-Shanker, 1988)).

With context-sensitve selector languages, CTAGs generate exactly the context-sensitive languages: Let $L \subseteq \Sigma^+$ be a context-sensitive language. We construct the CTAG

$$\begin{aligned}G &= (\Sigma, \{A, B\}, \{A\}, \Omega, \{\pi\} \cup \{\pi_\sigma \mid \sigma \in \Sigma\}), \text{ where}\\ \Omega &= \{[_A x]_A \mid x \in L, |x| = 1\} \cup \{[_B \sigma]_B \mid \sigma \in \Sigma\}\\ \pi &= (\Sigma^+, \{([_B \sigma,]_B) \mid \sigma \in \Sigma\}, \{B\}, \{A\}) \text{ and}\\ \pi_\sigma &= (\{x \in \Sigma^+ \mid \sigma x \in L\}, \{([_A \sigma,]_A)\}, \{B\}, \{A\}).\end{aligned}$$

Since the family of context-sensitive language is closed under quotient with singleton sets, all selector languages are context-sensitive, and it is not difficult to prove $L(G) = L$.

This result shows that the combined use of selector languages and obligatory adjoining leads to a very powerful formalism. Whereas there are context-sensitive languages (such as $L = \{a^n c b^n c^n c d^n \mid n \geq 1\}$) which cannot be generated by any MBICR-grammar regardlessly of the used selector languages, the above construction shows that for each family of languages F closed under quotient with singleton sets and containing all finite languages each $L \in F$ can also be generated by a CTAG with F-choice.

Closure Properties

The class of languages generated by CTAGs with F-choice is closed under union, concatenation and Kleene-star for all families of languages F with $\Sigma^+ \in F$. Let $G_1 = (\Sigma_1, \Delta_1, \Upsilon_1, \Omega_1, P_1)$

and $G_2 = (\Sigma_2, \Delta_2, \Upsilon_2, \Omega_2, P_2)$ be two CTAGs with F-choice for a family of languages F with $\Sigma^* \in F$. Without loss of generality we may assume that $\Delta_1 \cap \Delta_2 = \emptyset$. Therefore it is easy to see that for $G = (\Sigma_1 \cup \Sigma_2, \Delta_1 \cup \Delta_2, \Upsilon_1 \cup \Upsilon_2, \Omega_1 \cup \Omega_2, P_1 \cup P_2)$ we have $L(G) = L(G_1) \cup L(G_2)$. For concatenation we take a new index $S \notin \Delta_1 \cup \Delta_2$ and construct $G' = (\Sigma_1 \cup \Sigma_2, \Delta_1 \cup \Delta_2 \cup \{S\}, \Upsilon_1 \cup \Upsilon_2 \cup \{S\}, \{[_S \alpha \beta]_S \mid \alpha \in \Omega_1 - \{\lambda\}, \beta \in \Omega_2 - \{\lambda\}\} \cup \{\alpha \in \Omega_1 \mid \lambda \in \Omega_2\} \cup \{\alpha \in \Omega_2 \mid \lambda \in \Omega_1\}, P_1 \cup P_2)$. Clearly $L(G') = L(G_1) \cdot L(G_2)$. For Kleene-star we construct $G'' = (\Sigma_1, \Delta_1 \cup \{S\}, \Upsilon_1 \cup \{S\}, \{[_S \alpha]_S \mid \alpha \in \Omega_1 - \{\lambda\}\} \cup \{\lambda\}, P \cup \{\pi\})$, where $\pi = (\Sigma^*, \{([_S, \alpha]_S) \mid \alpha \in \Omega_1 - \{\lambda\}\}, \{S\}, \{S\})$ It is a technical exercise to prove $L(G'') = L(G_1)^*$.

For each CTAG G and regular language R we can construct a CTAG G' such that $L(G') = L(G) \cap R$. Furthermore, G' uses the same selector languages as G. Hence, this construction directly proves that the class of languages generated by CTAGs with F-choice is closed under intersection with regular languages for any family of languages F. For the relevance of closure under intersection with regular sets we refer the reader to (Lang, 1994).

In the following, we will present a sketch of the proof. Let $G = (\Sigma, \Delta, \Upsilon, \Omega, P)$ be an arbitrary CTAG and R a regular language. Without loss of generality we assume that G is in a normal form such that each internal node either has exactly one leaf or only internal nodes as immediate successors: formally for each $\alpha_1\alpha_2\alpha_3 \in T(G)$ such that $\alpha_2 \in \mathrm{DC}_\Delta(\Sigma)$ we either have $\alpha_2 = [_A a]_A$ for some $a \in \Sigma$ and $A \in \Delta$ or $\alpha_2 = [_A \beta_1 \ldots \beta_n]_A$ for an $A \in \Delta$ and $\beta_i \in \mathrm{DC}_\Delta(\Sigma)$, $1 \leq i \leq n$. Since R is regular, there exists a deterministic finite automaton $M = (Q, \Sigma, \delta, q_0, F)$ with $L(M) = R$ (c.f. (Hopcroft & Ullman, 1979) for notational details). We construct a grammar G' where the label of each internal node additionally carries two pairs of states of M, formally the set of indices of G' is given by $\Phi = \{(A, [p, q], [r, s]) \mid A \in \Delta, p, q, r, s \in Q\}$.

Intuitively, in the tree interpretation, if an internal node is labelled by $(A, [p, q], [r, s])$ then $[p, q]$ is a value propagated from the immediate predecessor of the node stating that this node is supposed to generate a yield w such that $\delta(p, w) = q$. The pair $[r, s]$ denotes that the immediate successors of the node are supposed to generate a yield w such that $\delta(r, w) = s$.

G' generates as sentential forms exactly the sentential forms of G where a label of an internal node A in $S(G)$ is replaced by all labels $(A, [p, q], [r, s])$, $p, q, r, s \in Q$, in $S(G')$ such that for the resulting strings $\alpha \in S(G')$ the following properties hold:

(1) For each partition $\alpha = \alpha_1\alpha_2\alpha_3$ such that $\alpha_2 \in \mathrm{DC}_\Delta(\Sigma)$ and $\alpha_2 = [_X \gamma_1 \ldots \gamma_n]_X$ we have $X = (A, [p, q], [p_0, p_n])$, $\gamma_i \in \mathrm{DC}_\Delta(\Sigma)$ and $\gamma_i = [_{Y_i} \gamma_i']_{Y_i}$, $Y_i = (B_i, [p_{i-1}, p_i], [r_i, s_i])$, $1 \leq i \leq n$. In other words, for each internal node with other internal nodes as immediate successors, the second pair of states of the node is consistent with the first pairs of states of its immediate descendants (in the sense of the usual triple costruction). See Figure 3 for an illustration.

(2) For each partition $\alpha = \alpha_1\alpha_2\alpha_3$ such that $\alpha_2 \in \mathrm{DC}_\Delta(\Sigma)$ and $\alpha_2 = [_X \sigma]_X$ where $X = (A, [p, q], [r, s])$ and $\sigma \in \Sigma$ we have $\delta(r, \sigma) = s$. In other words, for all internal nodes having a leaf labelled by σ as immediate successor we have $\delta(r, \sigma) = s$ for the second pair of states $[r, s]$.

(3) For each $\alpha = [_X \alpha']_X$ we have $X = (A, [q_0, f], [r, s])$ where q_0 is the initial state of M and f is a final state of M, $f \in F$. In other words, the first pair of states of the root node of each tree consists of M's initial state and a final state of M.

The details of converting the axioms and contexts of G into axioms and contexts of G' are omitted due to the limited space. The conversion leaves the selector languages untouched, so

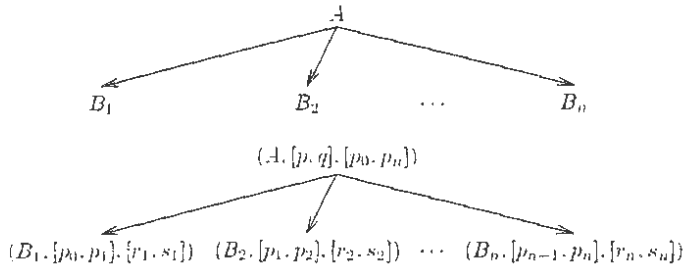


Figure 3: Example for a part of a tree in $S(G')$ corresponding to a part of a tree in $S(G)$. The above part of a tree with root labelled by A and immediate nonterminal successors $B_1, \ldots, B_n$ is converted into all parts of the above form for arbitrary $p, q, p_i, r_i, s_i \in Q$, $0 \leq i \leq n$ (not considering further restrictions due to the immediate predecessor or the immediate descendants of this part of the tree).

G' uses the same selector languages as G. If we define the set of permitted indices of G' by $\Phi' = \{(A,[p,q],[p,q]) \mid A \in \Upsilon, p, q \in Q\}$ we obtain $L(G') = L(G) \cap R$.

The same construction can also be used to show the closure of TAL under intersection with regular sets without involving a corresponding automata model like EPDAs.

Conclusion and Further Work

In this paper, we introduced CTAGs and discussed their generative capacity and some closure properties. CTAGs seem a significant progress compared to MBICR-grammars. As allowing both obligatory adjoining and selector languages leads to a very powerful model, our future work will focus on CTAGs with "weak" selector languages. Open problems which we would like to tackle in the future are whether the classes of languages generated by such grammars are closed under homomorphism and inverse homomorphism or not and the relationship to other formalisms such as range concatenation grammars and recursive matrix systems.

References

EHRENFEUCHT A., PĂUN G. & ROZENBERG G. (1997). Contextual grammars and formal languages. In G. ROZENBERG & A. SALOMAA, Eds., *Handbook of Formal Languages Vol.2: Linear Modeling*, p. 237–294. Berlin: Springer.

HOPCROFT J. E. & ULLMAN J. D. (1979). *Introduction to Automata Theory, Languages and Computation*. Reading, MA, USA: Addison-Wesley.

KAPPES M. (1999). Combining contextual grammars and tree adjoining grammars. In *Proceedings of the 6. Meeting on Mathematics of Language*, p. 337–345, Orlando.

LANG B. (1994). Recognition can be harder than parsing. *Computational Intelligence*.

MARCUS S. (1997). Contextual grammars and natural languages. In G. ROZENBERG & A. SALOMAA, Eds., *Handbook of Formal Languages Vol.2: Linear Modeling*, p. 215–235. Berlin: Springer.

MARTIN-VIDE C. & PĂUN G. (1998). Structured contextual grammars. *Grammars*, 1 (1 p.), 33–55.

PĂUN G. (1997). *Marcus Contextual Grammars*. Dordrecht, Boston, London: Kluwer Academic Publishers.

VIJAY-SHANKER K. (1988). *A Study of Tree Adjoining Grammars*. PhD thesis, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA.

Even better than Supertags : Introducing Hypertags !

Alexandra Kinyon

TALaNa – LaTTice
UFRL, University Paris 7, case 7003
2, pl. Jussieu
F-75251 Paris Cedex 05
Email : Alexandra.Kinyon@linguist.jussieu.fr

Abstract

In this paper, we introduce the notion of Hypertag, which allows to factor the information contained in several Supertags into a single structure. We also discuss why this approach is useful within frameworks other than LTAGs, and how it can be used for annotating and searching corpora.

Introduction

Traditional part of speech tagging assigns very limited information (i.e. morphological and local) to lexical items, thus providing only limited help for parsing. To solve this problem, (Joshi & Srinivas 94, Srinivas 97) extend the notion of POS by introducing Supertags, within the framework of Lexicalized Tree Adjoining Grammars (LTAGs). Unfortunately, words are assigned on average a much higher number of Supertags than traditional POS : On average for English a word is associated with 1.5 POS and with 9 supertags (Joshi 99). One common solution to the problem is to only retain the "best" supertag for each word, or eventually the 3 best supertags for each word, which is what (Srinivas 97) does in a probabilistic manner. But then, early decision has an adverse effect on the quality of parsing if the wrong supertag(s) have been kept : one typically obtains between 75% and 92% accuracy when keeping only one supertag / item (depending on the type of text being supertagged and on the technique used) (cf. Srinivas 97, Chen & al. 99) which means that it may be the case that every word in 4 will have a wrong supertag, whereas typical POS taggers usually achieve an accuracy above 95%.

Solutions for packing several supertags into a single structure have been proposed in the past, for example by resorting to logical formulae (Kallmeyer 99) or linear types of trees (Halber 99). But as argued in (Kinyon 00a), these solutions are unsatisfactory because they rely only on mathematical properties of trees, and lack a linguistic dimension.

In this paper, we introduce the notion of Hypertag, which allows to factor the information contained in several Supertags, so that a single structure can be assigned to each word. In addition of being well-defined computational objects, hypertags should also be "readable" and also motivated from a linguistic point of view . In a first part, we explain the solution we have adopted, building up on the notion of MetaGrammar introduced by (Candito 96) & (Candito, 99). Finally, we discuss how this approach can be used in practice, and why it is interesting for frameworks other than LTAGs. We assume the reader is familiar with LTAGs and Supertags and refer respectively to (Joshi 87) & to (Srinivas 97) for an introduction.

1. Exploiting a MetaGrammar

(Candito 96,99) has developed a tool to generate semi-automatically elementary trees She uses an additional layer of linguistic description, called the metagrammar (MG), which imposes a general organization for syntactic information in a 3 dimensional hierarchy :

- **Dimension 1:** initial subcategorization

- **Dimension 2:** redistribution of functions and transitivity alternations
- **Dimension 3:** surface realization of arguments, clause type and word order

Each terminal class in dimension 1 describes a possible initial subcategorization (i.e. a tree family). Each terminal class in dimension 2 describes a list of ordered redistributions of functions (e.g. it allows to add an argument for causatives). Finally, each terminal class in dimension 3 represents the surface realization of a (final) function (e.g. cliticized, extracted ...).

Each class in the hierarchy corresponds to the partial description of a tree (cf. Rogers & Vijay-Shanker 94). An elementary tree is generated by inheriting from one terminal class in dimension 1, from one terminal class in dimension 2 and from n terminal classes in dimension 3 (were n is the number of arguments of the elementary tree). [1] The hierarchy is partially handwritten. Then crossing of linguistic phenomena (e.g. passive + extraction), terminal classes and from there elementary trees are generated automatically off line[2]. This allows to obtain a grammar which can then be used to parse in real time. When the grammar is generated, it is straight forward to keep track of the terminal classes each elementary tree inherited from : Figure 1 shows seven elementary trees which can supertag "donne" (gives), as well as the inheritance patterns[3] associated to each of these supertags. All the examples below will refer to this figure.

The key idea then is to represent a set of elementary trees by a disjunction for each dimension of the hierarchy. Therefore, a hypertag consists in three disjunctions (one for dimension 1, one for dimension 2 and one for dimension 3). The cross-product of the three disjunctions can then be done automatically and from there, the set of elementary trees referred to by the hypertag will be automatically retrieved We will now illustrate this, first by showing how hypertags are built, and then by explaining how a set of trees (and thus of supertags) is retrieved from the information contained in a hypertag.

1.1 Building hypertags : a detailed example

Let us start with a simple exemple were we want "donner" to be assigned the supertags $\alpha 1$ (*J. donne une pomme à M.*) and $\alpha 2$ (*J donne à M. une pomme*). On figure 1, one notices that these two trees inherited exactly from the same classes : the relative order of the two complements is left unspecified in the hierarchy, thus one same description will yield both trees. In this case, the hypertag will thus simply be identical to the inheritance pattern of these two trees :

Dimension 1:	n0vn1(àn2)
Dimension 2 :	no redistribution
Dimension 3	subj :nominal-canonical obj : nominal-canonical a-obj: nominal-canonical

Let's now add tree $\alpha 3$ (*J. donne une pomme*) to this hypertag.This tree had its second object declared empty in dimension 2 (thus it inherits only two terminal classes from dimension 3, since it has only two arguments realized). The hypertag now becomes[4] :

---

[1] The idea to use the MG to obtain a compact representation of a set of SuperTags was briefly sketched in (Candito 99) and (Abeillé & al. 99), by resorting to MetaFeatures, but the approach here is slightly different since only information about the classes in the hierarchy is used (and not explicit information about the function of arguments)

[2] This point has been misunderstood by (Xia & al. 98, p.183) : terminal classes and classes for crossings of phenomena ARE NOT manually created

[3] We call inheritance patterns the structure used to store all the terminal classes a tree has inherited from.

[4] What has been added to a supertag is shown in bold characters.

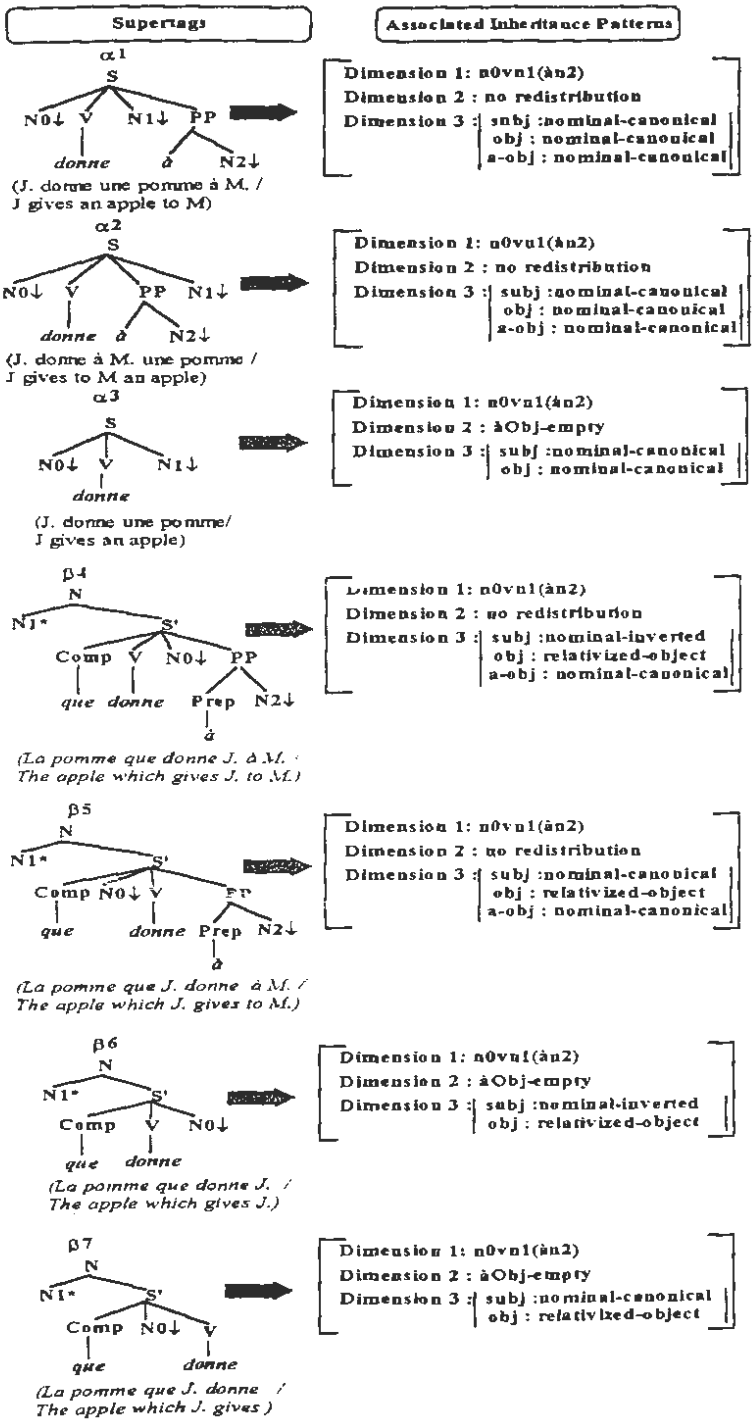


FIGURE 1 : SuperTags and associated inheritance patterns

Dim. 1:	n0vn1(àn2)
Dim. 2 :	no redistribution **OR àObj- empty**
Dim. 3	subj :nominal-canonical obj : nominal-canonical a-obj: nominal-canonical

Let's now add the tree β4 for the object relative to this hypertag. This tree has been generated by inheriting in dimension 3 from the terminal class "nominal inverted" for its subject and from the class "relativized object" for its object. This information is simply added in the hypertag, which now becomes :

Dim. :	n0vn1(àn2)
Dim. 2 :	no redistribution OR àObj- empty
Dim. 3	subj :nominal-canonical **OR nominal-inverted** obj : nominal-canonical **OR relativized-object** a-obj: nominal-canonical

Also note that for this last example the structural properties of β4 were quite different than those of α1, α2 and α3 (for instance, it has a root of category N and not S). But this has little importance since a generalization is made in linguistic terms without explicitly relying on the shape of trees.

It is also clear that hypertags are built in a monotonic fashion : each supertag added to a hypertag just adds information. Also, the process of building hypertags is rather simple. We observe that hypertags allow to label each word with a unique structure[5]. Moreover, hypertags contain rich syntactic information about lexical items (For our example, the word "*donne*"), and also contain functional information (not explicitly available in supertags). They are linguistically motivated, but also yield a readable output. They can be enriched or modified by human annotators or easily fed to a parser or shallow parser.

1.2 Retrieving information from hypertags

Retrieving information from hypertags is pretty straightforward. For example, to recover the set of supertags contained in a hypertag[6], one just needs to perform the crossing between the 3 dimensions of the hypertag, as shown on Figure 2, in order to obtain all inheritance patterns. These inheritance patterns are then matched with the inheritance patterns contained in the grammar (i.e. the right column in Figure 1) to recover all the appropriate supertags. Inheritance patterns which are generated but don't match any existing trees in the grammar are simply discarded[7].

We observe that the 4 supertags α1, α2 and α3 and β4 which we had explicitly added to the hypertag in 2.1 are correctly retrieved. But also, the supertags β5, β6 and β7 are retrieved, which we did not explicitly intend since we never added them to the hypertag. But this is not a problem, since if a word can anchor the 4 first trees, then it will also necessarily anchor the three last ones. In fact, the automatic crossing of disjunctions in the hypertag insures consistency[8].

---

[5] We presented a simple example for sake of clarity, but traditional POS ambiguity is handled in the same way, except that disjunctions are then added in dimension 1 as well.

[6] This is to show that supertags can be retrieved from a hypertag. But it is not indispensable to do so : using hypertags directly is more appealing and will be addressed in future work.

[7] When the full 5000 trees grammar is generated with the MetaGrammar, these same trees are discarded by general linguistic principles such as "canonical nominal objects prevent subject inversion" (cf. Abeillé & al. 00). So Hypertags do not "overgenerate".

[8] Again, for the same reasons the MetaGrammar insures consistency.

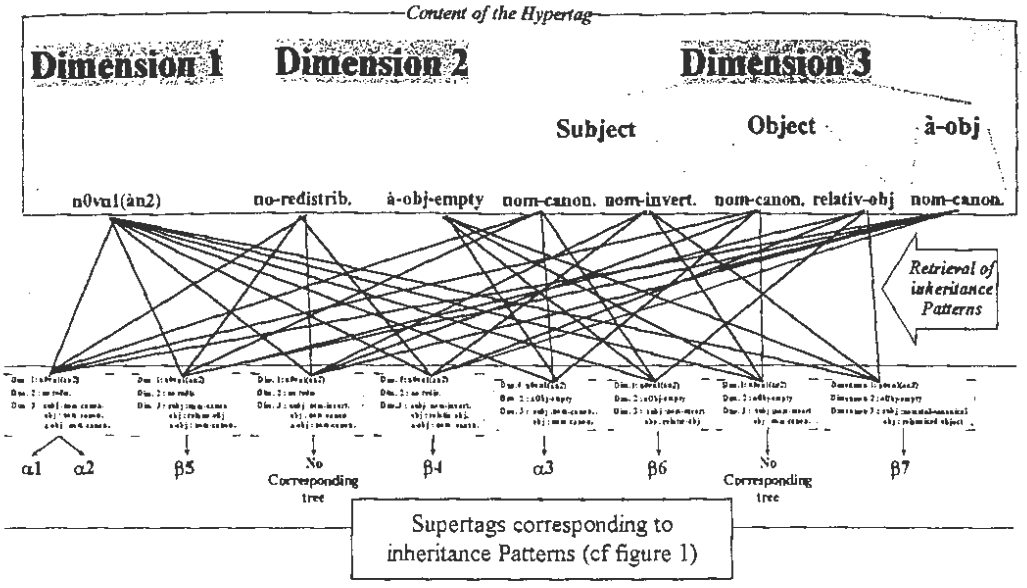


FIGURE 2 :Retrieving inheritance patterns and Supertags from a Hypertag

Also note that no particular mechanism is needed for dimension 3 to handle arguments which are not realized : if àObj-empty is inherited from dimension 2, then only subject and object will inherit from dimension three (since only arguments that are realized inherit from that dimension when the grammar is generated).

Information can also be modified at runtime in a hypertag, depending on the context of lexical items. For example *"relativized_object"* can be supressed in dimension 3 from the hypertag shown on Figure 2, in case no Wh element is encountered in a sentence. Then, the correct set of supertags will still be retrieved from the hypertag by automatic crossing (that is, trees $\alpha 1$,$\alpha 2$ and $\alpha 3$), since the other inheritance patterns generated won't refer to any tree in the grammar (here, no existing tree inherits in dimension 3 *"subject:inverted-nominal"*, without inheriting also *"object: relativized-object"*)

2. Practical use

An LTAG can be seen as a dictionary, in which each lexical entry is associated to a set of elementary trees. But with hypertags, each lexical entry is now paired with one unique structure. Therefore, automatically hypertagging a text is easy (i.e. simple dictionary lookup). The equivalent of finding the "right" supertag for each lexical item in a text (i.e. reducing ambiguity) then consists in dynamically removing information from hypertags (i.e. suppressing elements in disjunctions). We hope this can be achieved by specific rules, which we are currently working on. It is important to note though that the resulting output can easily be manually annotated in order to build a gold-standard corpus : manually removing linguistically relevant pieces from information in a disjunction from a single structure is simpler than dealing with a set of trees. In addition of obvious advantages in terms of display (tree structures, especially when presented in a non graphical way, are unreadable), the task itself becomes easier because topological problems are solved automatically: annotators need just answer questions such as *"does this verb have an extracted object ?"*, *"is the subject of this verb*

inverted ?" to decide which terminal classe(s) must be kept[9] .We believe that these questions are easier to answer than *"Which of these trees have a node N1 marked wh+ at address 1.1 ?"* (for an extracted object).

Also, supertagged text are difficult to use outside of an LTAG framework, contrary to hypertagged texts, which contain general linguistic information. An example would be searching and extracting syntactic data on a large scale : suppose one wants to extract all the occurrences where a given verb V has a relativized object. To do so on a hypertagged text simply involves performing a "grep" on all lines containing a V whose hypertag contains *"dimension 3 : objet:relativized"*, without knowing anything about the LTAG framework. Performing the same task with a supertagged text involves knowing how LTAGs encode relativized objects in elementary trees, scanning potential trees associated with V... Another example would be using a hypertagged text as an input to a parser based on a framework other than LTAGs : for instance, hypertags could be used by an LFG parser to constrain the construction of an F-structure, whereas it's unclear how this could be achieved with supertags.

3. Conclusion

We have introduced the notion of hypertag. Hypertags allow to assign one unique structure to lexical items. Moreover this structure is readable, linguistically and computationally motivated, and contains much richer syntactic information than traditional POS, thus a hypertagger would be a good candidate as the front end of a parser. It allows in practice to build large annotated resources which are useful for extracting syntactic information on a large scale, without being dependant on a given grammatical formalism. Also, hypertags are being used to develop a psycholinguistically motivated processing model for LTAGs (Kinyon 00b).

We have shown how hypertags are built, how information can be retrieved from them. Further work will investigate how hypertags can be combined directly.

References

Abeillé A., Candito M., Kinyon A. 1999. FTAG: current status and parsing scheme. Proc. Vextal '99. Venice.

Abeillé A. Candito M.H., Kinyon A. 2000. The current status of FTAG. Proc. TAG+5. Paris.

Candito M-H. 1996. A principle-based hierarchical representation of LTAGs. Proc. COLING'96 Kopenhagen.

Candito M.-H. 1999. Représentation modulaire et paramétrable de grammaires électroniques lexicalisées. Application au français et à l'italien. PhD dissertation. University Paris 7.

Chen J., Srinivas B., Vijay-Shanker K. 1999 New Models for Improving Supertag Disambiguation. Proc. EACL'99 pp. 188-195. Bergen.

Halber A. 1999. Stratégie d'analyse pour la compréhension de la parole : vers une approche à base de Grammaires d'Arbres Adjoints Lexicalisées. PhD thesis. ENST. Paris

Joshi A. : 1987. *An introduction to Tree Adjoining Grammars.* In Mathematics of Language. A. Manaster-Ramer (eds). John Benjamins Publishing Company. Amsterdam.Philadelphia. pp. 87-114.

Joshi A. 1999. Explorations of a domain of locality. CLIN'99. Utrecht.

Joshi A. Srinivas B. 1994. Disambiguation of Super Parts of Speech (or Supertags) : Almost parsing. Proceeding COLING'94. Kyoto.

Kallmeyer L. 1999, Tree Description Grammars and Underspecified Representations. PhD thesis. Universität Tübingen.

Kinyon A. 2000a. Hypertags. Proceedings Coling'00. Sarrebrucken.

Kinyon A. 2000b. Towards a psycholinguistically motivated processing model for LTAGs. Proc. Cogsci' 2000. Philadelphia.

Srinivas B., 1997. Complexity of lexical descriptions and its relevance for partial parsing, PhD thesis. Univ. of Pennsylvania.

Rogers J., Vijay-Shanker K. 1994. Obtaining trees from their descriptions : an application to TAGs. Computational Intelligence. 10:4 pp 401-421.

Xia F. & Palmer M., Vijay-shanker K., Rosenzweig J. 1998. Consistent Grammar development using partial tree description for LTAGs. Proc. TAG+4. Philadelphia.

[9] This of course implies that one must be very careful in choosing evocative names for terminal classes.

Building a class-based verb lexicon using TAGs

Karin Kipper, Hoa Trang Dang, William Schuler, and Martha Palmer

Department of Computer and Information Sciences
University of Pennsylvania
200 South 33rd Street
Philadelphia, PA 19104, USA
{kipper,htd,schuler,mpalmer}@linc.cis.upenn.edu

Abstract

We present a class-based approach to building a verb lexicon that makes explicit the close relation between syntax and semantics for Levin classes. We have used a Lexicalized Tree Adjoining Grammar to capture the syntax associated with each verb class and have added semantic predicates to each tree, which allow for a compositional interpretation.

1. Introduction

We describe a computational verb lexicon called VerbNet which utilizes Levin verb classes (Levin, 1993) to systematically construct lexical entries. We have used Lexicalized Tree Adjoining Grammar (LTAG) (Joshi, 1985; Schabes, 1990) to capture the syntax associated with each verb class, and have added semantic predicates. We also show how regular extensions of verb meaning can be achieved through the adjunction of particular syntactic phrases. We base these regular extensions on intersective Levin classes, a fine-grained variation on Levin classes, as a source of semantic components associated with specific adjuncts (Dang *et al.*, 1998). Whereas previous research on tying semantics to Levin classes (Dorr, 1997) has not explicitly implemented the close relation between syntax and semantics hypothesized by Levin, our lexical resource combines traditional lexical semantic information, such as thematic roles and semantic predicates, with syntactic frames and selectional restrictions. In order to increase the utility of VerbNet, we also include links to entries in WordNet, which is one of the most widely used online lexical databases in Natural Language Processing applications.

2. Levin Classes and WordNet

Two current approaches to English verb classifications are WordNet and Levin classes. WordNet is an on-line lexical database of English that currently contains approximately 120,000 sets of noun, verb, adjective, and adverb synonyms, each representing a lexicalized concept. A synset (synonym set) contains, besides all the word forms that can refer to a given concept, a definitional gloss and – in most cases – an example sentence. Words and synsets are interrelated by means of lexical and semantic-conceptual links, respectively. Antonymy or semantic opposition links individual words, while the super-/subordinate relation links entire synsets. WordNet was designed principally as a semantic network, and contains little syntactic information. Even as a semantic resource, however, it is missing some of the information that has traditionally been required by NLP applications, including explicit predicate-argument structures. WordNet senses are often too fine-grained as well, lacking an underlying notion of semantic components and a systematic extension of basic senses to produce these fine-grained senses.

The Levin verb classification, on the other hand, does explicitly state the syntax for each class, but still falls short of assigning semantic components to each class. The classes are based on the ability or inability of a verb to occur in pairs of syntactic frames that are in some sense meaning preserving (diathesis alternations) (Levin, 1993). The sets of syntactic frames associated with a particular Levin class are supposed to reflect underlying semantic components that constrain allowable arguments and adjuncts. For example, *break* verbs and *cut* verbs are similar in that they can all participate in the transitive and middle constructions. However, only *break* verbs can also occur in the simple intransitive, and *cut* verbs can occur in the conative, where *break* verbs cannot. The explanation given is that *cut* describes a series of actions directed at achieving the goal of separating some object into pieces. It is possible for these actions to be performed without the end result being achieved, but where the *cutting* manner can still be recognized (i.e., "John cut at the loaf"). For *break*, the only thing specified is the resulting change of state where the object becomes separated into pieces. If the result is not achieved, no attempted *breaking* action can be recognized.

1. Transitive construction
   (a) John broke the window.
   (b) John cut the bread.
2. Middle construction
   (a) Glass breaks easily.
   (b) This loaf cuts easily.
3. Intransitive construction
   (a) The window broke.
   (b) *The bread cut.
4. Conative construction
   (a) *John broke at the window.
   (b) John valiantly cut/hacked at the frozen loaf, but his knife was too dull to make a dent in it.

The fundamental assumption is that the syntactic frames are a direct reflection of the underlying semantics. However, Levin classes exhibit inconsistencies that have hampered researchers' ability to reference them directly in applications. Many verbs are listed in multiple classes, some of which have conflicting sets of syntactic frames. For instance, *carry* verbs are described as not taking the conative (*"The mother carried at the baby"), and yet many of the verbs in the *carry* class *(push, pull, tug, shove, kick)* are also listed in the *push/pull* class, which does take the conative. Dang et al. (1998) showed that multiple listings could in some cases be interpreted as regular sense extensions, and defined intersective Levin classes, which are a more syntactically and semantically coherent refinement of basic Levin classes. We implement these verb classes and their regular sense extensions in the Lexicalized Tree Adjoining Grammar formalism.

3. Verb lexicon

VerbNet can be viewed in both a static and a dynamic way. The static aspect refers to the verb entries and how they are organized, providing the characteristic descriptions of a verb sense or a verb class. The dynamic aspect of the lexicon constrains the entries to allow a compositional interpretation in LTAG derivation trees, capturing extended verb meanings by incorporating adjuncts.

3.1. Static description

Each verb entry refers to a set of classes, corresponding to the different senses of the verb. For example, the manner of motion sense of "run" is a member of the *Manner of Motion* class, whereas "run" as in "the street runs through the district" is a member of the *Meander* class. For each verb sense there is a verb class as well as specific selectional restrictions (e.g., an instrument of "kick" must be of type $foot$) and semantic characteristics (e.g., a particular manner of directed motion) that may not be captured by the class membership. In order to provide a mapping to other dictionaries, we also include links to WordNet synsets. Because WordNet has more fine-grained sense distinctions than Levin, each verb sense in VerbNet references the set of WordNet synsets (if any) that captures the meaning appropriate to the class.

Verb classes allow us to capture generalizations about verb behavior. This reduces not only the effort needed to construct the lexicon, but also the likelihood that errors are introduced when adding a new verb entry. Each verb class lists the thematic roles that the predicate-argument structure of its members allows, and provides descriptions of the syntactic frames corresponding to licensed constructions, with selectional restrictions defined for each argument in each frame. Each frame also includes semantic predicates describing the participants at various stages of the event described by the frame.

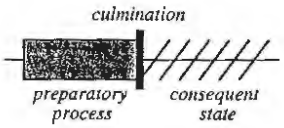


Figure 1: Moens and Steedman's tripartite structure of events

We decompose each event E into a tripartite structure in a manner similar to Moens and Steedman (1988), introducing a time function for each predicate to specify whether the predicate is true in the preparatory ($during(E)$), culmination ($end(E)$), or consequent ($result(E)$) stage of an event. The tripartite event structure (Figure 1) allows us to express the semantics of classes of verbs like change of state verbs whose adequate description requires reference to a complex event structure. In the case of a verb such as "break", it is important to make a distinction between the state of the object before the end of the action ($during(E)$), and the new state that results afterwards (*result(E)*).

Verb classes are hierarchically organized, ensuring that each class is coherent – that is, all its members have common semantic elements and share a common set of thematic roles and basic syntactic frames. This requires some manual restructuring of the original Levin classes, which is facilitated by using intersective Levin classes. In addition, a particular verb may add more semantic information to the basic semantics of its class.

Figure 2 shows a partial entry for the *Hit* class. This class allows for three thematic roles: Agent, Patient and Instrument, with constraints that the Agent is generally animate; the Patient concrete; and the Instrument concrete and inanimate.[1] These selectional restrictions refer to

[1]These constraints are more like preferences that generate a preferred reading of a sentence. They may be

HIT class

⟨⟨MEMBERS ⟩⟩	[⟨*hit*, 1⟩, ⟨*kick*, 1⟩, ⟨*slap*, 1⟩, ⟨*tap*, 1⟩, . . .]
⟨⟨THEMATIC ROLES ⟩⟩	Agent(A), Patient(P), Instrument(I)
⟨⟨SELECT RESTRICTIONS ⟩⟩	Agent[+animate], Patient[+concrete], Instrument[+concrete,-animate]

⟨⟨FRAMES and PREDICATES ⟩⟩

Basic Transitive	A V P	manner(during(E),directedmotion,A) ∧ manner(end(E),forceful,A) ∧ contact(end(E),A,P)
Transitive with Instrument	A V P with I	manner(during(E),directedmotion,I) ∧ manner(end(E),forceful,I) ∧ contact(end(E),I,P)
Conative	A V at P	manner(during(E),directedmotion,A)
With/against alternation	A V I against/on P	manner(during(E),directedmotion,I) ∧ manner(end(E),forceful,I) ∧ contact(end(E),I,P)
Transitive	I V P	manner(during(E),directedmotion,I) ∧ manner(end(E),forceful,I) ∧ contact(end(E),I,P)

Figure 2: Partial entry for the *Hit* class

a feature hierarchy where *animate* subsumes *animal* and *human*, *concrete* subsumes both *animate* and *inanimate*, and so forth. This representation does not suffer from some drawbacks of theta role analysis because our roles are not global primitives, but are only used to describe relationships within a class.

The strength of our representation comes from the explicit relationship between syntax and semantics captured in each entry. Figure 2 shows some of the syntactic frames allowed for the *Hit* class and the semantic predicates for each frame. Thematic roles are used as descriptors which are mapped into arguments of semantic predicates as well as the argument positions in a TAG elementary tree.

The tripartite event structure also handles the conative construction, in which there is an intention of a goal during the event which is not achieved at the end of the event. The example shown in Figure 2 for the conative construction has the predicate

manner(during(E),directedmotion,A)

but because the intended contact by sudden impact is not satisfied, the semantics does not include the predicates

manner(end(E),forceful,A) ∧ *contact(end(E),A,P)*.

3.2. Compositional Semantics

We use TAG elementary trees to describe syntactic frames and associate semantic predicates and selectional restrictions with each tree. Elementary trees capture the basic semantics of the verbs in each class. Each frame in the static aspect of the lexicon maps onto a TAG elementary tree, in which the thematic roles correspond to substitution sites. Some auxiliary trees are class-based because they interact with the verbs in the class in peculiar ways and add seman-

relaxed depending on the domain of a particular application.

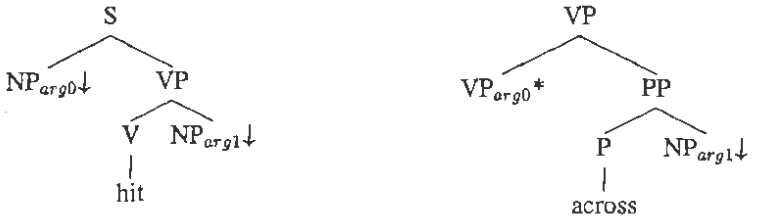


Figure 3: Initial transitive tree for "hit" and auxiliary tree for "across"

tic content specific to the class. Others, such as temporal adjuncts, bring the same semantic predicate independent of the verb. We use a flat semantic representation like that of Joshi and Vijay-Shanker (1999) in which the semantics of a sentence is the conjunction of the semantic predicates of the trees used to derive the sentence.

We ensure that all the semantic arguments of basic predicates are local to the syntactic initial tree. For example, the basic transitive frame in Figure 2 shows that the Agent is in direct motion and contacts the Patient in a forceful manner. If an instrument is specified, it replaces the Agent in these predicates. Since the instrument can be an argument in the basic predicates of the *Hit* class, it must appear in the elementary trees whenever it is specified, even if it is in a prepositional phrase.

The ability of certain verbs to take on extended senses based on their adjuncts is captured in a natural way by the TAG operation of adjunction and our conjunction of semantic predicates. Figure 3 shows an initial transitive tree anchored by "hit" and the semantic predicates associated with this syntactic frame. The original *Hit* verb class does not include movement of the direct object as part of the meaning of "hit" – only one event of contact by sudden impact is described. This event is subdivided into three predicates: the first,

manner(during(E),directedmotion,X_{arg0}*)*

specifies that during the event E, X_{arg0} is in directed motion; the second,

manner(end(E),forceful,X_{arg0}*)*

refers to the forceful contact of X_{arg0} at the end of E; and the third,

contact(end(E),X_{arg0}*,*X_{arg1}*)*

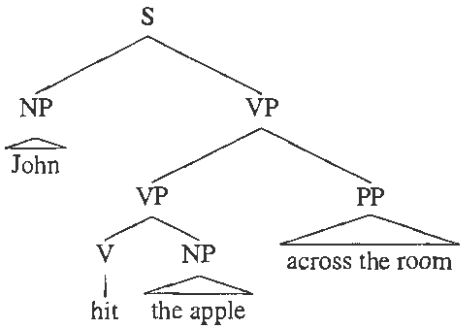
establishes that at the end of event E, contact between X_{arg0} and X_{arg1} has been achieved.

By adjoining a path PP such as "across NP", we get an extended meaning, and a change in Levin class membership to the *Throw* class. Figure 3 shows the auxiliary tree anchored by the preposition "across" together with its semantic predicates. The class-specific path PP adds the predicates

meets(E_{arg0}*,E)* $\wedge$ *motion(during(E),*$X_{arg0.arg1}$*)* $\wedge$ *via(during(E),*$X_{arg0.arg1}$*,*X_{arg1}*),*

introducing a motion event that immediately follows (*meets*) the contact event, which is the basic sense of the *Hit* class.

In Figure 4, we show the derived tree for the sentence "John hit the apple across the room" with all the predicates instantiated. The arguments are recovered from the derivation tree, following Candito and Kahane (1998). When an initial tree, such as α_{John}, is substituted into another



$$manner(during(e1), directedmotion, john) \wedge manner(end(e1), forceful, john) \wedge$$
$$contact(end(e1), john, apple) \wedge meets(e1, e2) \wedge motion(during(e2), apple) \wedge$$
$$via(during(e2), apple, room)$$

Figure 4: Sense extension of "hit" through adjunction of a path PP

tree α_{hit}, the dependency mirrors the derivation structure, so the variables associated with the substituting tree can be referenced as arguments in the host tree's predicates (see Figure 5). When an auxiliary tree β_{across} is adjoined, the dependency for the adjunction is reversed, so that variables associated with the host tree can be referenced as arguments in the adjoining tree's predicates. With this dependency from β_{across} to α_{hit} (labeled $arg0$), it is now possible for the semantic predicates associated with β_{across} to predicate over variables in the dependent tree α_{hit}, including the variable $X_{arg0.arg1}$ instantiated as *apple*, resulting in the predicates *motion(during(e2),apple)* $\wedge$ *via(during(e2),apple,room)*.

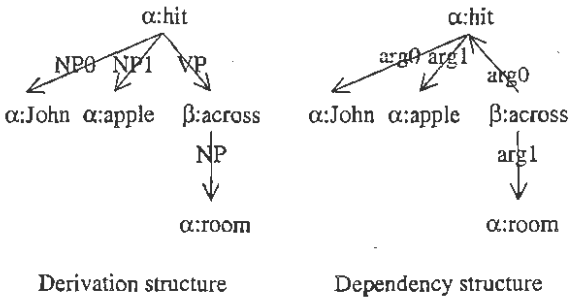


Figure 5: Derivation and dependencies

Verbs in the intersective class formed by the *Push/Pull* verbs and the *Carry* verbs behave in a similar manner. The core meaning of this verb class is exertion of force. Adjunction of a path PP implying motion modifies membership of these verbs to the *Carry* class. *Push/Pull* verbs can appear in the conative construction, which emphasizes their forceful semantic component and ability to express an *attempted* action where any result that might be associated with the verb is not necessarily achieved; *Carry* verbs (used with a goal or directional phrase) cannot take the conative alternation because this would conflict with the causation of motion which is the intrinsic meaning of the class (Dang *et al.*, 1998).

Palmer et al. (1999) and Bleam et al. (1998) also defined compositional semantics for classes of verbs implemented in FB-LTAG, but they represented general semantic components (e.g., motion, manner) as features on the nodes of the trees. Our use of separate logical forms gives a more detailed semantics for the sentence, so that for an event involving motion, it is possible to know not only that the event has a *motion* semantic component, but also which entity is actually in motion.

4. Conclusion

We have presented a class-based approach to building a verb lexicon that makes explicit the close association between syntax and semantics, as postulated by Levin. By using verb classes we capture generalizations about verb behavior and reduce not only the effort needed to construct the lexicon, but also the likelihood that errors are introduced when adding new verbs. Another important contribution of this work is that by dividing each event into a tripartite structure, we permit a more precise definition of the associated semantics, which is necessary for applications such as animation of natural language instructions (Bindiganavale *et al.*, 2000). The power of the lexicon comes from its dynamic aspect which is based on the LTAG formalism. The operation of adjunction in TAGs provides a principled approach to representing the type of regular polysemy that has been a major obstacle in building verb lexicons.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments. This research was partially supported by NSF grants IIS-9800658 and IIS-9900297, Office of Naval Research AASERTs N00014-97-1-0603, and CAPES grant 0914-95.

References

BINDIGANAVALE R., SCHULER W., ALLBECK J. M., BADLER N. I., JOSHI A. K. & PALMER M. (2000). Dynamically Altering Agent Behaviors Using Natural Language Instructions. *Fourth International Conference on Autonomous Agents.*

BLEAM T., PALMER M. & VIJAY-SHANKER K. (1998). Motion Verbs and Semantic Features in TAG. In *Proceedings of the Fourth TAG+ Workshop*, Philadelphia, PA.

CANDITO M.-H. & KAHANE S. (1998). Can the TAG Derivation Tree Represent a Semantic Graph? An Answer in the Light of Meaning-Text Theory. In *Proceedings of the Fourth TAG+ Workshop*, p. 21–24, Philadelphia, PA.

DANG H. T., KIPPER K., PALMER M. & ROSENZWEIG J. (1998). Investigating Regular Sense Extensions Based on Intersective Levin classes. In *Proceedings of COLING-ACL98*, Montreal, CA.

DORR B. J. (1997). Large-Scale Dictionary Construction for Foreign Language Tutoring and Interlingual Machine Translation. *Machine Translotion*, **12**, 1–55.

JOSHI A. K. (1985). How Much Context Sensitivity Is Necessary for Characterizing Structural Descriptions: Tree Adjoining Grammars. In L. K. D. DOWTY & A. ZWICKY, Eds., *Natural language parsing: Psychological, computational and theoretical perspectives*, p. 206–250. Cambridge, U.K.: Cambridge University Press.

JOSHI A. K. & VIJAY-SHANKER K. (1999). Compositional Semantics with Lexicalized Tree-Adjoining Grammar: How Much Under-Specification Is Necessary? In *Proceedings of the Third International Workshop on Computational Semantics (IWCS-3)*, p. 131–145, Tilburg, The Netherlands.

LEVIN B. (1993). *English Verb Classes and Alternation: A Preliminary Investigation*. The University of Chicago Press.

MOENS M. & STEEDMAN M. (1988). Temporal Ontology and Temporal Reference. *Computational Linguistics*, 14, 15–38.

PALMER M., ROSENZWEIG J. & SCHULER W. (1999). Capturing Motion Verb Generalizations in Synchronous Tree-Adjoining Grammar. In P. SAINT-DIZIER, Ed., *Predicative Forms in Natural Language and in Lexical Knowledge Bases*. Kluwer Press.

SCHABES Y. (1990). *Mathematical and Computational Aspects of Lexicalized Grammars*. PhD thesis, Computer Science Department, University of Pennsylvania.

LTAG Workbench: A General Framework for LTAG

Patrice Lopez

DFKI GmbH
Stuhlsatzenhausweg 3
D-66123 Saarbrcken, Germany
lopez@dfki.de

Abstract

This paper presents the LTAG Workbench, a set of graphical tools and parsers freely available for LTAG. The system can be view as a modern alternative to the XTAG system. We present first the outlines of the workbench including different graphical editors and two chart parsers. The encoding of resources and results is based on an XML application called TagML. We present then future works dedicated to speed efficiency: Optimization based on sharing techniques and preprocessing of features. The whole system has been developed in Java which allows a strong portability and interesting reusability properties.

1. Introduction

The success of a linguistic formalism can largely depend on the availability of dedicated tools. They are needed first for maintaining the consistency of a grammar and for checking its correctness, but also for proving the adequacy of a formalism for computational applications. Such tools raise several engineering problems that should not be neglected as portability, reusability, user-friendly graphical interface, easy installation procedure and recycling of existing grammars, see for instance (Erbach & Uszkoreit, 1990) for a overview of these problems. Focusing on these features, we present a set of freely available tools dedicated to the LTAG formalism (Joshi *et al.*, 1975) which aims to be an alternative to the XTAG system (XTAG research group, 1998). The *LTAG workbench* is still an on-going work and we hope that it will appear enough promising to give rise to interests and possible contributions from the LTAG community.

We present first the outlines of the current workbench including different graphical editors and two chart parsers. We introduce then our solution for resource management which is based on a XML application called TagML. The section 4 is dedicated to future optimizations for speed efficiency that emphasize precompilation techniques and sharing of computation on the basis of grammar redundancies.

2. The LTAG Workbench

2.1. Editors

The workbench proposes general editors for the set of elementary trees schema and for the morphologic and syntactic lexicon. The graphical editors are based on a general tree editor developed at Thomson-CSF (France). It includes a *lexicalizer* function, similar to the one of the XTAG system, that allows to visualize an instancied elementary tree given a schema and lexical entry. These editors covers the functionality of the XTAG system and include browsers for lexicons.

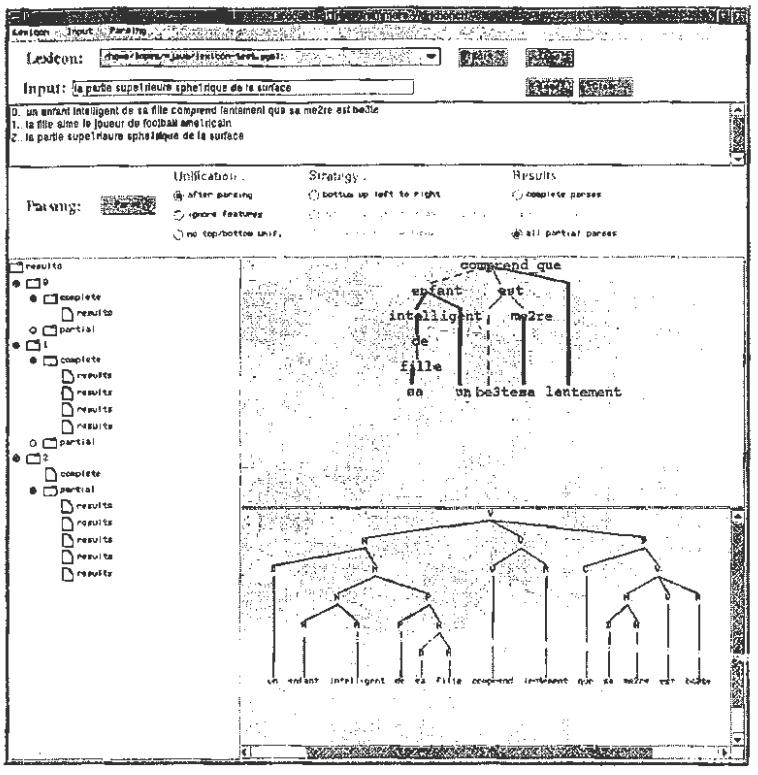


Figure 1: Screen shot of the LTAG parsing workbench.

2.2. Parsers

The workbench includes currently two parsers in a parsing test workbench (see screen shot on figure 1):

- A bottom-up *connection driven* parser which can deliver extended partial results compared with other classical bottom-up algorithms and without time penalty (Lopez, 2000).
- An implementation of the top-down Earley-like parser proposed in (Schabes, 1994).

Note that both of them are complete chart parsers, including extraction of results from the shared parse forest and two-step feature based processing. The bottom-up parser gives complete and partial parses considering several parsing heuristics with or without unification of the feature structures used in Feature Based LTAG. It is also possible to test and compare various parsing heuristics and strategies in term of speed efficiency.

2.3. Results

The system can deliver and edit different kinds of results: complete parses (derivation and derived trees) or partial parses, with complete unification, with only the first unification steps or without any unification. These different kind of results aims:

- To test a grammar by identifying the step involved in the failure of a parse during grammar debugging.

- To study out of grammar phenomena.

The workbench is implemented in Java for portability and reusability reasons. The Java sources, classes and documentation of the editors and parsing test workbench will be freely available by the end of May 2000. We present now another facet of the technical choices concerning the workbench: all the involved data are encoded with the highly portable formalism XML.

3. TagML

3.1. *Motivations*

A significant number of works are based on the TAG formalism. Still, for the moment, none has led to a common representation format of the grammars which would facilitate the exchange of TAG grammars and associated data or the development of normalised parsers and generic tools. A working group gathering people, mainly from TALaNa (University of Paris 7, France), ENST (Paris, France), INRIA (Rocquencourt, France), LORIA (Nancy, France) and DFKI (Saarbrücken, Germany) who are currently working on this formalism, made it necessary to define a shared and common representation with the aim of exchanging grammars and associated data, developing normalized parsers and specifying generic tools. Our proposal, TagML (Tree Adjoining Grammars Markup Language) is a general recommendation for the encoding and the exchange of the resources involved in LTAG. Anyone implementing a tool on the basis of this encoding can guarantee its interoperability with existing ones.

The XTAG system (XTAG research group, 1998), developed in the early nineties, offers the first workbench dedicated to LTAG grammar design and an Earley-like parser. However, this integrated parser provides only a binary answer (accepted or rejected sentence) hardly compatible with the test of a large grammar. Partial results and diagnostics about errors are necessary to test a grammar and to identify the step involved in the failure of a parse during grammar debugging. Thus, designing a new parser is justified but integrating new components to the XTAG system is technically very difficult for someone that has not been involved in the initial development of the system. More generally, this system has not been developed technically to be distributed since it is based on proper and non specified formats. It requires a narrowly-specialised skill for its installation, its usage and its maintenance. TagML can be viewed as a standardization and an extension of the XTAG formats and more generally as an answer to these technical problems. We present in the following sections the broad outlines of TagML, for more details see (Bonhomme & Lopez, 2000).

3.2. *Principles*

The definition of a generic tool for parsing and managing LTAG grammars supposes a common language specification, shared by the concerned community. The first step toward more generic and flexible tools undergoes the definition of an appropriate encoding for the management of large-size linguistic resources. This encoding should be able to structure possibly heterogeneous data and to give the possibility to represent the inevitable redundancies between lexical data. Given these expectations, we decided to define TagML as an application of the XML recommendation.

A LTAG grammar is defined by a morphological lexicon, a syntactic lexicon and a set of elementary tree schemas. The schema are ordered in tree families in order to capture the general aspects of the lexicalization process. This lexicalization is obtained on the basis of information given in the syntactic lexicon. For the moment, a complete Document Type Definition (DTD) has been proposed for the schema.

In an elementary tree schema, we can distinguish:

- The structural part, i.e. a partial phrase structure or a partial parsing tree.
- The set of feature equations constraining top and bottom feature structures.

We keep from (Issac, 1998) most of the elements involved in the encoding of schema structures:

- $< t >$: elementary tree, document that we specify in this part.
- $< n >$: general node, the attribute *cat* gives the category of this node and the attribute *type* distinguishes foot node, substitution node and anchor.
- $< fs >$: feature structure, of type *bottom* or *top*
- $< f >$: typed feature (attribute-value) similarly to the TEI. For typed feature equation and their re-usability, we introduce the element *linkGrp* as specified in the TEI to group internal or external links (element *link*) (Sperberg-McQueen & Burnard, 1994).

3.3. Structural component of schema

Similarly to (Issac, 1998) proposal, we represent straightforwardly the tree structure of a schema by an isomorphy with the XML tree structure (see figure 2).

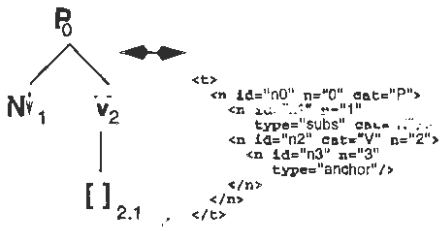


Figure 2: Isomorphy between the elementary tree schema and the XML tree structure

In practice in a broad-covering lexicalized grammar, the redundancy of common substructures is very important. For instance, the subtree dominated by a V category with a depth of 1 (the anchor and the pre-terminal category) is shared by most of the trees describing a verbal syntactical context (several hundred of trees for the English XTAG grammar, several thousand for the French LTAG grammar). This redundancy can be very useful to encode for linguistic or efficiency issues. In order to represent these redundancies, we propose to use the XML Link mecanism (DeRose *et al.*, 1999) and to identify systematically every nodes. We use the principle of virtual resources systematically to obtain only one representation of the different nodes within the whole grammar.

3.4. Feature equations

The TEI (Sperberg-McQueen & Burnard, 1994) proposes a recommendation for the encoding of feature structures that we propose to integrate to TagML. This standardization allows to type the features and to represent explicitly feature percolation. Note that the features used in the LTAG formalism have atomic values thanks to the extended domain of locality principle. The feature equations of an elementary tree schema can be view as a global term for a complete elementary tree, or as several terms distributed in the various nodes of an elementary tree sharing common variables. We propose to link directly the shared features in order to avoid the necessity to manage shared labels during the parsing of the features structures. These links are specified in *linkGrp*.

We have the possibility to give a type to a *linkGrp*, i.e. for a feature equation, for instance *subject-verb agreement*, then by identifying this *linkGrp* to share the corresponding feature equation to several elementary tree schemas. If we still consider the example of *subject-verb agreement* feature equation, the corresponding *linkGrp* will be shared by all elementary tree schemas that include this kind of agreement. The nodes corresponding to the features linked by percolation can be identified hy a special attribute which gives the function of each terminal node. The access to these specific nodes are obtained with the selection language proposed both for XSL Tranformation Language (Clark, 1999) and for the XML pointers called XML Paths (Clark & DeRose, 1999).
As we can see in figure 3, the percolated feature is linked to the *linkGrp* corresponding to the feature equation, so it is straightforward to access with this link all the other features which share the same value, without dealing with any labels and tables of labels.

```
<n cat="P" id="n0">
  <fs type="top" id="fs0">
    <f name="num" id="f0">
      <link xlink:type="simple"
            xlink:href="doc#id(I0)"/>
    </f>
    <f name="det" id="f1"><minus/></f>
  </fs>
  <fs type="bottom" id="fs1">
    /* ... */
  </fs>
</n>

/* External document */

<linkGrp type="accord">
  <link targets="
    id(n0)/fs[1][@type,top]/f[1][@name,num]
    id(n2)/fs[1][@type,bottom]/f[1][@name,num]"
        id="I0"/>
</linkGrp>
/* ... */
```

Figure 3: Shared features and factorisation of common feature equation

3.5. Tree family

In order to manage efficiently a set of elementary trees that could be quite large, TagML provides a mechanism allowing to gather elementary trees sharing the same sub-categorisation frame. A tree family is described (indicated by the tag $< tfamily >$) by defining a set of links to a subset of elementary tree schemas. The figure 4 presents an example of tree family definition (in this example *I1_VTA_0* and *I2_VTD_1B* refers to two elementary tree schemas for transitive verbs and *I2_adjectif6* and *I1_adjectif1* to two elementary tree schemas for adjective).
The encoding of the syntactic lexicon which is much more complex will be the subject of further research. The current system works with a very basic XML encoding of lexicon closed to the XTAG system flat representation.

3.6. Existing tools

Our implementations are based on the Silfide XML toolkit[1]. The following tools are currently available:

- A XSL style sheet allowing the automatic generation of Latex documentation from the TagML data.
- A conversion tool for the XTAG format.

[1] http://www.loria.fr/projets/XSilfide/EN/sxp/

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE tag SYSTEM "tagml.dtd">
<tag xmlns:xlink="http://www.w3.org/XML/XLink/0.9">
  <desc>Our tree families</desc>
  <tfamily name="transitive verb">
    <desc>Tree family for transitive verbs</desc>
    <t xlink:type="simple"
       xlink:href="l1_VTA_0.xml"
       xlink:show="replace"
       xlink:actuate="auto"/>
    /* ... */
    <t xlink:type="simple"
       xlink:href="l2_VTD_1B.xml"
       xlink:show="replace"
       xlink:actuate="auto"/>
  </tfamily>

  <tfamily name="adjective">
    <desc>Tree family for adjectives</desc>
    <t xlink:type="simple"
       xlink:href="A1_adjectif1.xml"
       xlink:show="replace"
       xlink:actuate="auto"/>
    /* ... */
    <t xlink:type="simple"
       xlink:href="l2_adjectif6.xml"
       xlink:show="replace"
       xlink:actuate="auto"/>
  </tfamily>

  /* ... */
</tag>
```

Figure 4: Sample of a TagML document and two tree families

Every parser which respects the XML encoding of LTAG resources and a specific Java API can be directly integrated to the LTAG Workbench.
We plan to improve the conversion tool by performing a grammar simplification and compaction at various levels. We will see in the next section that our main goal here is to exploit redundancies of data to to reduce the processing cost.

4. Sharing computation and feature processing optimization

Consequence of the important size of existing large-coverage Lexicalized TAG grammars, the current parsers suffer from a lack of speed performance. Speed is an important factor for real-world application but also because the tools are constantly used during grammar development. We argue the improvement of LTAG parsers and tools depends on how the huge amount of data consequence of the lexicalization can be put into factor in order to share the computation. The initial idea is structure sharing by the way of Finite State Techniques for the elementary tree skeletons (Evans & Weir, 1997). Still we will see that similar sharing for feature equations and derivation extraction is also possible.

4.1. Structural Sharing

Lexicalization raises the problem of multiplication of the same substructures which can be serious. In Context Free Grammars the same rule can be used for all possible parsing trees which contain the corresponding substructure, but in Lexicalized Tree Grammars these substructures are duplicated. Considering classical linguistics choices for LTAG grammar design, polystructures (example : *to speak to..., speak about..., to speak to ... about...*) are very common, the corresponding elementary trees must share common substructures and therefore do not cost as much as an independant elementary tree for each.
(Evans & Weir, 1997) shares different substructures of elementary trees using Finite State Automata (FSA) and classical minimizing techniques. As presented in (Evans & Weir, 1997),

the authors use automata corresponding to one particular traversal of the trees. We use similar techniques to share here linearized structures between different elementary trees and obtain automata similar to the ones presented in (Roche, 1996). The main difference with (Evans & Weir, 1997) is that, since it represents elementary trees for any kind of tree walk, this FSA does not impose a specific strategy during the parsing.

When FSA are shared in a single one, each state contains identifiers of the elementary trees which pass through it and each item the list of elementary tree identifiers valid for the item's positions. To test conditions of a rule we must consider every possible transitions paths according to the shared FSA. The resulting item of a rule is valid for a subset of identifiers of the elementary trees passing through the both position states. The "uncompaction" can be done when we enumerate the derivations.

4.2. Preprocessing of center features

In Feature-Based LTAG, two sets of features, top and bottom, are associated to each nodes. This separation is necessary because of potential adjunctions which can change the value of a possible feature at a given node. Categories used as node labels are never changed after an adjunction which only capture recursive structures (i.e. root and foot node of auxiliary trees must have the same label). We say that the category value at a given node is monotonic according to the adjunction operation. We propose to define an additional set of features, called *center features*, in order to gather features which are also monotonic according to the adjunction operation. The main interest of this new set of features is computational efficiency by the improvement of the predictive power of the grammar. The set of possible trees for an attachment at a given node N is not only trees with a matching category but also trees that present unifiable center features.

The center features can be computed easily simply by identifying which features are never changed by any existing auxiliary trees. Unfortunately, considering the whole XTAG grammar for instance, we can always find an auxiliary tree modifying the value of a given feature. Still, it is possible to compute significant center features considering only the subset of the grammar which is valid after the lexicalization process.

Features as *aux* or *det* of the French LTAG grammar should still need a separation in top and bottom features, but many others, in particular morphological features as *num* or *gender*, will be in general monotonic for adjunction after the lexicalization process. With this simple preprocessing, we expect a significant speed-up factor during parsing.

4.3. Sharing of feature equations

Similarly to the problem of redundancy of common substructures between different elementary trees, the same *feature equations* (i.e. the same kind of percolation of feature values) are duplicated in many trees. For instance the subject-verb agreement could be shared between hundred of trees (Candito, 1996). Our idea is to associate a unique feature term to the set of derivations and to improve the sharing of the corresponding DAG. Given a feature equation, this improvement supposes to identify the common nodes which are linked by the feature percolation. This identification can be done not on the basis of similar Gorn Adress of nodes but by identifying the functions (subject, object1, object2, syntactic verbal head, ...) associated to each nodes. For instance the subject-verb agreement percolates feature values linked to the subject and the syntactic verbal head (main verb, modal or auxiliary). This feature equation could be evaluated only one time for all elementary trees (i) containing this feature equation and (ii) combined with the same elementary trees at nodes with the same functions.

5. Conclusion

We have presented a general framework dedicated to LTAG grammars with a special regard to portability and reusability. A lot of efforts are still necessary to achieve efficiency and practical real-world application but we have proposed some possible optimizations and, more generally, an ambitious basis which can be freely exploited and enriched.

Aknowledgement

We would like to thank all the participants of the TagML working group: Anne Abeillé, Nicolas and Sébastien Barrier, Marie-Hélene Candito, Lionel Clement, Alexandra Kinyon and Kim Gerdes (TALaNa), Patrice Bonhomme, Laurence Danlos and Laurent Romary (LORIA), Pierre Boullier, Eric de la Clergerie and Philippe Deschamp (INRIA Rocquencourt), Ariane Halber (ENST), Fabrice Issac (UT Compiegne), Yannick de Kerkadio (LIMSI), Rodrigo Reyes (LCR Thomson) and David Roussel (Aérospaciale).

References

BONHOMME P. & LOPEZ P. (2000). TagML: XML Encoding of Resources for Lexicalized Tree Adjoining Grammars. In *Second International Conference on Linguistic Ressources and Evaluation (LREC'2000)*, Athens.

CANDITO M.-H. (1996). A principle-based hierarchical representation of LTAGs. In *COLING'96*, Copenhagen, Denmark.

CLARK J. (1999). *XSL Transformations (XSLT) Version 1.0.* W3C, `http://www.w3.org/TR/-xslt`. W3C Recommendation 16 November 1999.

CLARK J. & DEROSE S. (1999). *XML path language (XPath) version 1.0. W3C,* `http://-www.w3.org/TR/xpath`, *8 October 1999.*

DEROSE S., ORCHARD D. & TRAFFORD B. *(1999).* XML linking language (XLink). W3C, `http://www.w3.org/TR/WD-xlink`, 26 July 1999.

ERBACH G. & USZKOREIT H. (1990). *Grammar Engineering: Problems and Prospects - Report on the Saarbrücken Grammar Engineering Workshop.* Technical report, CLAUS-Report Nr.1.

EVANS R. & WEIR D. (1997). Automaton-based Parsing for Lexicalized Grammars. In *Fifth International Workshop on Parsing Technologies*, Cambridge, Mass.

ISSAC F. (1998). A Standard Representation Framework for TAG. In *Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+4).*

JOSHI A., LEVI L. & TAKAHASHI M. (1975). Tree adjunct grammars. *Journal of the Computer and System Sciences.*

LOPEZ P. (2000). Extended Partial Parsing for Lexicalized Tree Grammars. In *International Workshop on Parsing Technology (IWPT2000)*, Trento, Italy.

ROCHE E. (1996). *Parsing with Finite-State Transducers.* Technical report, TR-96-30, Mitsubishi Electric Research Laboratories (MERL).

SCHABES Y. (1994). Left to Right Parsing of Lexicalized Tree Adjoining Grammars. *Computational Intelligence*, **10**, 506–524.

SPERBERG-MCQUEEN C. & BURNARD L. (1994). *TEI guidelines for electronic text encoding and interchange (P3). Chicago and Oxford,* `http://etext.virginia.edu/TEI.html`.

XTAG RESEARCH GROUP *(1998).* A Lexicalized Tree Adjoining Grammar for English. *Technical report, IRCS 98-03, University of Pennsylvania.*

Derivational Minimalism in Two Regular and Logical Steps

Jens Michaelis and Uwe Mönnich and Frank Morawietz

Universität Tübingen, Wilhelmstr. 113, 72074 Tübingen, Germany
{michael,um,frank}@sfs.nphil.uni-tuebingen.de

Abstract

In this paper we extend the work by Michaelis (1999) which shows how to encode an arbitrary Minimalist Grammar in the sense of Stabler (1997) into a weakly equivalent multiple context-free grammar (MCFG). By viewing MCFG rules as terms in a free Lawvere theory we can translate a given MCFG into a regular tree grammar. The latter is characterizable by both a tree automaton and a corresponding formula in monadic second-order (MSO) logic. The trees of the resulting regular tree language are then unpacked into the intended "linguistic" trees with an MSO transduction based upon tree-walking automata. This two-step approach gives an operational as well as a logical description of the tree sets involved.

1. Introduction

Over the last couple of years, a rich class of mildly context-sensitive grammar formalisms has been proven to be weakly equivalent. Among others, the following families of (string) languages are equivalent: $STR(HR)$ [languages generated by string generating hyperedge replacement grammars], $OUT(DTWT)$ [output languages of deterministic tree-walking tree-to-string transducers], $yDT_{fc}(REGT)$ [yields of images of regular tree languages under deterministic finite-copying top-down tree transductions], $MCFL$ [languages generated by multiple context-free grammars], $MCTAL$ [languages generated by multi-component tree adjoining grammars], $LCFRL$ [languages generated by linear context-free rewriting systems], $LUSCL$ [languages generated by local unordered scattered context grammars] (more on these equivalences can be found, e.g., in Engelfriet 1997, Rambow & Satta 1999, Weir 1992).

The work by Michaelis (1999) shows how to encode an arbitrary minimalist grammar (MG) in the sense of Stabler (1997) into a weakly equivalent linear context-free rewriting system (LCFRS). The core idea is that for the set of trees appearing as intermediate steps in converging derivations corresponding to a given MG one can define a finite partition. The equivalence classes of this partition are formed by sets of trees where the features triggering movement appear in identical structural positions. Each nonterminal in a corresponding LCFRS represents such an equivalence class, i.e., an infinite set of trees. We take the resulting LCFRSs as our starting point and present in this paper a translation from multiple context-free grammars (MCFGs)—which are a weakly equivalent extension of LCFRSs—into regular tree grammars (RTGs)/monadic second-order (MSO) logic/tree automata. This is done via lifting by viewing MCFG rules as terms in a free Lawvere theory. Since this coding makes projection, tupling and composition explicit, the resulting trees contain these operations as labeled nodes. Therefore we use an MSO transduction—where the regular tree language constitutes the domain—to transform the lifted trees into the intended ones.

We think that our approach has decisive advantages. First, the operations of the relevant signature appear explicitly in the lifted trees and are not hidden in node labels coding instances of rule application. Second, our path component is not dependent on the particular regular tree

family or the domain defined via the MSO formula. The instruction set of the tree-walking automaton and the corresponding definition of the MSO transduction are universal and only serve to reverse the lifting process. In that sense the instructions are nothing else but a restatement of the unique homomorphism which exists between the free algebra and any other algebra of the same signature. Thus, the translation from MCFGs to RTGs constitutes a considerable simplification in comparison with other characterizations since it is not built upon derivation trees using productions of the original MCFG as node labels, but rather on the operations of projection, tuple-formation and composition alone.

In the following sections we limit ourselves to the special case of MCFG rules with only one nonterminal on the right hand side (RHS). This allows a significant simplification in the presentation since it requires only one level of tupling. The extension to the general case of using tuples of tuples is considerably more involved and, for lack of space, cannot be described here.

2. Background and Basic Definitions

We first present some basic definitions before we proceed with the actual translation. Let S be a set of sorts. A *many-sorted signature Σ (over S)* is an indexed family $\langle \Sigma_{w,s} \mid w \in S^*, s \in S \rangle$ 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 $|w|$, where $|w|$ denotes the length of w. Let $X = \{x_1, x_2, x_3, \ldots\}$ be a countable set of variables, and for $k \in \mathbb{N}$ define X_k as $\{x_1, \ldots, x_k\}$. Then, the set of k-ary trees $T(\Sigma, X_k)$ over Σ is built up from X_k using the operators in the usual way: If $\sigma \in \Sigma_{\varepsilon,s} \cup X_k$ for some $s \in S$ and $\varepsilon \in S^*$ with $|\varepsilon| = 0$ then σ is a (trivial) k-ary tree of sort s. If, for some $s \in S$ and $w = s_1 \cdots s_n$ with $s_i \in S$, $\sigma \in \Sigma_{w,s}$ and $t_1, \ldots, t_n$ are k-ary trees with t_i of sort s_i then $\sigma(t_1, \ldots, t_n)$ is a k-ary tree of sort s. Note that $T(\Sigma, X_k) \subseteq T(\Sigma, X_l)$ for $k \leq l$. Let $T(\Sigma, X) = \bigcup_{k \in \mathbb{N}} T(\Sigma, X_k)$.

The operator symbols induce operations on an algebra with the appropriate structure. A *Σ-algebra* $\mathbb{A}$ consists of an S-indexed family of sets $A = (A^s)_{s \in S}$ and for each operator $\sigma \in \Sigma_{w,s}$, $\sigma_A : A^w \to A^s$ is a function, where $A^w = A^{s_1} \times \cdots \times A^{s_n}$ and $w = s_1 \cdots s_n$ with $s_i \in S$. The set $T(\Sigma, X)$ can be made into a Σ-algebra $\mathbb{T}$ by specifying the operations as follows. For every $\sigma \in \Sigma_{w,s}$, where $s \in S$ and $w = s_1 \cdots s_n$ with $s_i \in S$, and every $t_1, \ldots, t_n \in T(\Sigma, X)$ with t_i of sort s_i we identify $\sigma_{\mathbb{T}}(t_1, \ldots, t_n)$ with $\sigma(t_1, \ldots, t_n)$.

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 $\mathbb{T}$, whose carriers $\langle T(u, v) \mid u, v \in S^* \rangle$ consist of the morphisms of the theory and whose operations are of the following types, where $n \in \mathbb{N}$, $u = u_1 \cdots u_n$ with $u_i \in S$ for $1 \leq i \leq n$ and $v, w \in S^*$,

$$\begin{array}{ll} \text{projection:} & \pi_i^u \in T(u, u_i) \\ \text{composition:} & c_{(u,v,w)} \in T(u, v) \times T(v, w) \to T(u, w) \\ \text{target tupling:} & (\ )_{(v,u)} \in T(v, u_1) \times \cdots \times T(v, u_n) \to T(v, u) \end{array}$$

The projections and the operations of target tupling are required to satisfy the obvious identities for products. The composition operations must satisfy associativity.

For S being a singleton and Σ a (many-sorted) signature over $S^* \times S^*$, the power set $\wp(T(\Sigma, X))$ of $T(\Sigma, X)$ constitutes the central example of interest for formal language theory. The carriers $\langle \wp(T(k, m)) \mid k, m \in \mathbb{N} \rangle$ of the corresponding $S^* \times S^*$-Lawvere algebra are constituted by the power sets of the sets $T(k, m)$, where each $T(k, m)$ is the set of all m-tuples of k-ary trees, i.e. $T(k, m) = \{(t_1, \ldots, t_m) \mid t_i \in T(\Sigma, X_k)\}$.[1] Composition is defined as substitution of the projection constants and target tupling is just tupling. For reasons of space, we cannot go into more details here. More on Lawvere theories in this context and their connection to linguistics can be found in Mönnich (1998).

[1] Since S is a singleton, S^* can be identified with $\mathbb{N}$, because up to length each $w \in S^*$ is uniquely specified.

A *multiple context-free grammar* (MCFG) is defined as a five-tuple $\mathcal{G} = \langle N, T, F, P, S\rangle$ with N, T, F and P being a finite set of ranked nonterminals, terminals, linear basic morphisms and productions, respectively. $S \in N$ is the start symbol. Each $p \in P$ has the form $A \longrightarrow f(A_0, \ldots, A_{n-1})$ for $A, A_0, \ldots, A_{n-1} \in N$ and $f \in F$ a function from $(T^*)^k$ to $(T^*)^l$ with arity $k = \Sigma_{i=0}^{n-1} k_i$ (k_i the rank of A_i) and l the rank of A (cf. Seki *et al.* 1991). Recall that the basic morphisms are those which use only variables, constants, concatenation, composition and tupling.

A *regular tree grammar* (RTG) is a 4-tuple $\mathcal{G} = \langle \Sigma, \mathsf{F}_0, S, \mathsf{P}\rangle$, where Σ is a many-sorted signature of *inoperatives* and F_0 a set of *operatives* of rank 0. $S \in \mathsf{F}_0$ is the starting symbol and P is a set of productions. Each $p \in \mathsf{P}$ has the form $F \longrightarrow t$, with $F \in \mathsf{F}_0$, and t a term (tree) over $\Sigma \cup \mathsf{F}_0$. An application of a rule $F \longrightarrow t$ "rewrites" F as the tree t. Since RTG rules always just substitute some tree for a leaf-node, it is easy to see that they generate recognizable sets of trees, i.e., context-free string languages (Mezei & Wright 1967).[2]

After these algebraic notions, we briefly present those related to monadic second-order (MSO) logic. MSO logic is the extension of first-order predicate logic with monadic second-order variables and quantification over them. In particular, we are using MSO logic on trees such that individual variables $x, y, \ldots$ stand for nodes in trees and monadic second-order ones $X, Y, \ldots$ for sets of nodes (for more details see, e.g., Rogers 1998).

Before we turn to purely logical notions, we introduce a concept which combines both automata theory and logic. We need a particular type of finite-state automaton: *tree-walking automata with MSO tests* (Bloem & Engelfriet 1997). Intuitively, those automata make transitions from nodes in a tree to other nodes along its branches.

A *tree-walking automaton (with tests)* over some ranked alphabet Σ is a finite automaton $\mathfrak{A} = (Q, \Delta, \delta, I, F)$ with states Q, directives Δ, transitions $\delta : Q \times \Delta \rightarrow Q$ and the initial and final states $I \subseteq Q$ and $F \subseteq Q$ which traverses a tree along connected edges using three kinds of directives: $\uparrow_i$—"move up to the mother of the current node (if it has one and it is its i-th daughter)", $\downarrow_i$—"move to the i-th daughter of the current node (if it exists)", and $\varphi(x)$—"verify that φ holds at the current node". For any tree $t \in T(\Sigma)$, such a tree-walking automaton $\mathfrak{A}$ computes a node relation $R_t(\mathfrak{A}) = \{(x, y) | (x, q_i) \stackrel{*}{\Rightarrow} (y, q_f)$ for some $q_i \in I$ and some $q_f \in F\}$, where for all states $q_k, q_l \in Q$ and nodes x, y in t $(x, q_k) \Longrightarrow (y, q_l)$ iff $\exists d \in \Delta : (q_k, d, q_l) \in \delta$ and y is reachable from x in t via d. Note that x is reachable from itself if the directive was a (successful) test. It is important not to confuse this relation with the *walking language* recognized by the automaton, i.e., the string of directives needed to move from the initial to the final node in a walk. Bloem and Engelfriet show that these automata characterize the MSO definable node relations, i.e., every tree-walking automaton we specify can be inductively transformed into an equivalent MSO formula and vice versa.

The following paragraphs go directly back to Courcelle (1997). Recall that the representation of objects within relational structures makes them available for the use of logical description languages. Let R be a finite set of relation symbols with the corresponding arity for each $r \in R$ given by $\rho(r)$. A relational structure $\mathcal{R} = \langle D_{\mathcal{R}}, (r_{\mathcal{R}})_{r \in R}\rangle$ consists of the domain $D_{\mathcal{R}}$ and the $\rho(r)$-ary relations $r_{\mathcal{R}} \subseteq D_{\mathcal{R}}^{\rho(r)}$. There does not seem to be a convenient machine model for tree transformations. Fortunately, one can use logic directly to define the desired transduction. The classical technique of interpreting a relational structures within another one forms the basis for MSO transductions. Intuitively, the output tree is interpreted on the input tree. E.g., suppose that we want to transduce the input tree t_1 into the output tree t_2. The nodes of the output tree t_2 will be a subset of the nodes from t_1 specified with a unary MSO relation ranging over the nodes of t_1. The daughter relation will be specified with a binary MSO relation with free variables x

[2] Appropriate definitions for derivations and the tree languages generated can be found in Kolb *et al.* (2000).

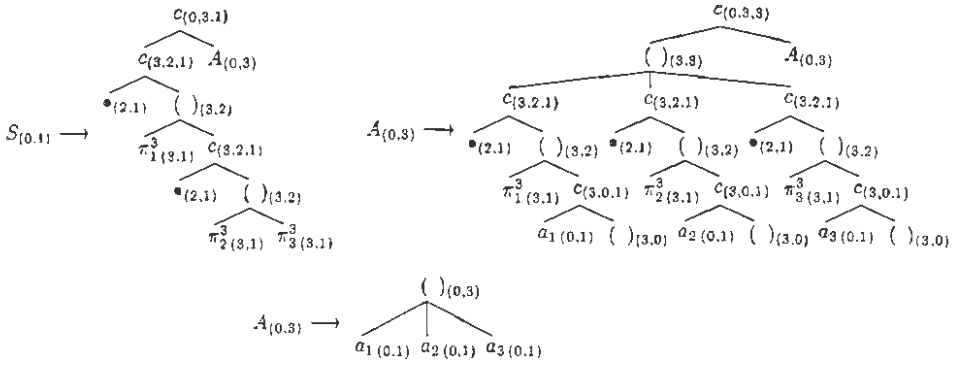


Figure 1: The translated example grammar $\mathcal{G}'$

and y ranging over the nodes from t_1. We will use this concept to transform the lifted trees into the intended ones.

A (non-copying) MSO transduction of a relational structure $\mathcal{R}$ (with set of relation symbols R) into another one $\mathcal{Q}$ (with set of relation symbols Q) is defined to be a tuple $(\varphi, \psi, (\theta_q)_{q \in Q})$. It consists of the formulas φ defining the domain of the transduction in $\mathcal{R}$ and ψ defining the resulting domain of $\mathcal{Q}$ and a family of formulas θ_q defining the new relations Q (using only definable formulas from the "old" structure $\mathcal{R}$).

In this sense, our description of non-contextfree phenomena with two devices with only regular power is an instance of the theorem that the image of an MSO-definable class of structures under a definable transduction is not MSO definable in general (Courcelle 1997).

3. Translating MCFGs to RTGs

Each rule of a given MCFG is recursively transformed into a RTG rule by coding the implicit operations of projection, tupling and composition as nonterminals or terminals. This becomes possible simply by viewing the terms appearing in the rules of the MCFG as elements of a free $\mathbb{N} \times \mathbb{N}$-sorted Lawvere algebra. The resulting RTG then "operates on" this Lawvere algebra.

As an example we consider the following MCFG $\mathcal{G} = \langle N, T, F, P, S \rangle$ with $N = \{S, A\}$, $T = \{a_1, a_2, a_3\}$, $F = \{g, h, l\}$ and $P = \{S \to g(A), A \to h(A), A \to l()\}$, where the functions $g : (T^*)^3 \to T^*$, $h : (T^*)^3 \to (T^*)^3$ and $l : (T^*)^0 \to (T^*)^3$ are given by

$$g(x_1, x_2, x_3) = x_1 x_2 x_3 \qquad h(x_1, x_2, x_3) = (x_1 a_1, x_2 a_2, x_3 a_3) \qquad l() = (a_1, a_2, a_3)$$

The language generated by $\mathcal{G}$ is $\{a_1^n a_2^n a_3^n \mid n > 0\}$.

Now, for $1 \leq i \leq 3$ let π_i^3 denote the i-th projection which maps a 3-tuple of strings from T^* to its i-th component, i.e. a 1-tuple, and let $\bullet$ denote the usual binary operation of concatenation defined for strings from T^*, i.e., $\bullet$ maps a 2-tuple to a 1-tuple. The corresponding (Lawvere) arity of S, a_1, a_2 and a_3 is $(0,1)$, of A $(0,3)$, of $\bullet$ $(2,1)$, and the one of π_1^3, π_2^3 and π_3^3 is $(3,1)$.

Applying the translation T given below to the MCFG $\mathcal{G}$ results in the RTG $\mathcal{G}' = \langle \Sigma, \mathsf{F}_0, S_{(0,1)}, \mathsf{P} \rangle$ with inoperatives $\Sigma = \langle \Sigma_{w,s} \mid w \in (\mathbb{N} \times \mathbb{N})^*, s \in \mathbb{N} \times \mathbb{N} \rangle$, operatives F_0 of rank 0, and productions P which (in tree notation) look as given in Fig. 1. We have $\Sigma_{\varepsilon,(3,0)} = \{(\ )_{(3,0)}\}$, $\Sigma_{\varepsilon,(2,1)} = \{\bullet_{(2,1)}\}$, $\Sigma_{\varepsilon,(0,1)} = \{a_{1\,(0,1)}, a_{2\,(0,1)}, a_{3\,(0,1)}\}$, $\Sigma_{\varepsilon,(3,1)} = \{\pi^3_{1\,(3,1)}, \pi^3_{2\,(3,1)}, \pi^3_{3\,(3,1)}\}$,

$$\begin{aligned}
\Sigma_{(0,3)(3,3),(0,3)} &= \{c_{(0,3,3)}\} & \Sigma_{(3,1)(3,1),(3,2)} &= \{(\ )_{(3,2)}\} \\
\Sigma_{(0,3)(3,1),(0,1)} &= \{c_{(0,3,1)}\} & \Sigma_{(0,1)(0,1)(0,1),(0,3)} &= \{(\ )_{(0,3)}\} \\
\Sigma_{(3,2)(2,1),(3,1)} &= \{c_{(3,2,1)}\} & \Sigma_{(3,1)(3,1)(3,1),(3,3)} &= \{(\ )_{(3,3)}\}
\end{aligned}$$

and $\mathsf{F}_0 = \{S_{(0,1)}, A_{(0,3)}\}$.[3]

As one can see in Fig. 1, the basic functions have been realized as terms with their respective implicit operations as nonterminal (composition and tupling) or terminal (projection and empty tupling) nodes. In the following paragraphs, we sketch the translation T from nonterminal rules of the example MCFG to RTG rules. T takes each rule $X \longrightarrow f(Y)$, where $X, Y \in N$ and $f \in F$, of the MCFG including the corresponding definition of the mapping $f(x_1, \ldots x_k)$ with $k \geq 0$ and transforms it into a RTG rule as follows. We create a mother node labeled with the appropriate binary composition $c_{(j,k,l)}$ such that the left daughter contains the "lifted" version of $f(x_1, \ldots, x_k)$ under T and the right daughter the translation of the nonterminal Y. Both nonterminals X and Y are used "unchanged", but annotated with the corresponding Lawvere arity resulting in the following schematic presentation of the translation: $X_{(j,l)} \longrightarrow c_{(j,k,l)}(\mathsf{T}(f(x_1, \ldots, x_k)), Y_{(j,k)})$, where f is a mapping from k-tuples to l-tuples of terminal strings.

The easiest case of translating a mapping $f \in F$ from our example via T is the terminal A-rule. We simply view the mapping as a Lawvere term. The function l just returns a triple of a_1, a_2 and a_3. The corresponding tree has a mother node labeled with a ternary tupling symbol and the three unary arguments of the mapping as daughters.[4] The S-rule is more complicated with the function g concatenating three (input) strings. The definition of the function can be written explicitly as the Lawvere term $c_{(3,2,1)}(\bullet, (\ )_{(3,2)}(\pi^3_1, c_{(3,2,1)}(\bullet\ (\ )_{(3,2)}(\pi^3_2, \pi^3_3))))$. Note that the implicit binary concatenation $\bullet$ in g now becomes the constant $\bullet_{(2,1)}$. The variables are simply replaced by the projections and concatenated. The resulting term is then applied to the operative $A_{(0,3)}$ such that we get the RHS displayed in the $S_{(0,1)}$-rule in Fig. 1. The recursive case of the A-rule is the most complicated. The mapping returns a triple, so we need a tupling "operator" of appropriate arity $(3,3)$ as the mother node with 3 daughters. The i-th of its daughters (labeled with $c_{(3,2,1)}$) is built by composing the concatenation constant $\bullet_{(2,1)}$ with the "tupling"-result $(\ )_{(3,2)}$ of the corresponding projection constant $\pi^3_{i(3,1)}$ (which is substituted for the variable x_i) and a particular constant tree. Namely the one which (in terms of the underlying Lawvere algebra) simply "lifts" the constant a_i to the Lawvere-arity of π^3_i just in order to allow for an appropriate tupling. So, the term (x_1a_1, x_2a_2, x_3a_3) is interpreted as the Lawvere term $(\ )_{(3,3)}(c(\bullet, (\ )_{(3,2)}(\pi^3_1, c(a_1, (\ )_{(3,0)}))), c(\ldots), c(\ldots))$ which appears as the RHS of the corresponding tree grammar rule.

Since RTGs can only generate recognizable (tree) languages, we can characterize them with both MSO logic on trees and tree automata.[5] The tree automaton $\mathfrak{A}_{G'}$ is constructed by transforming the grammar into a normal form such that each RHS is of depth one by introducing auxiliary operatives. Then we can easily construct appropriate transitions by basically reversing the arrow: the nonterminals become state names and the mother node will be read as alphabet symbol. It is know from Thomas (1990) how to transform this tree automaton into an MSO formula $\varphi_{\mathfrak{A}_{G'}}$ by encoding its behaviour. Details for our special case can be found in Kolb *et al.* (2000).

4. Reconstructing the Intended Trees

Rogers (1998) has shown the suitability of an MSO description language for linguistics which is based upon the primitive relations of immediate ($\lhd$), proper ($\lhd^+$) and reflexive ($\lhd^*$) dominance

[3] For simplicity and readability we will sometimes drop the subscript notion (k, m) from the inoperatives and operatives of rank 0, and sometimes even from the composition symbol $c_{(k,l,m)}$.

[4] Note that we do not need to use a further composition symbol dominating $\mathsf{T}(f)$ in case there is no nonterminal on the RHS of the rule of the MCFG.

[5] An introduction to tree automata can be found in Gécseg & Steinby (1984).

and proper precedence ($\prec$). We will show how to define these relations with an MSO transduction thereby implementing the unique homomorphism mapping the terms into elements of the corresponding regular tree language.. At the core of the transduction is a tree-walking automaton defining the binary relation of immediate dominance ($\lhd$) on the nodes belonging to the intended structures. It is based on some simple observations.[6]

1. Our trees feature three families of labels: the "linguistic" symbols L, i.e., the lifted symbols of the underlying MCFG; the "composition" symbols $\mathrm{C} = \{c_{(u,v,w)}\}$; the "tupling" symbols $(\quad)_{(v,u)}$ and the "projection" symbols $\Pi = \{\pi_i^k\}$.
2. All *n*onterminal nodes in T' are labeled by some $c \in \mathrm{C}$ or a "tupling" symbol. Note that no terminal node is labeled by some c.
3. The terminal nodes in T' are either labeled by some "linguistic" symbol, a "tupling" symbol of the form $(\quad)_{(k,0)}$, i.e. the "empty" tuple, or by some "projection" symbol π_i^k.
4. Any "linguistic" node dominating anything in the intended tree is on some left branch in T', i.e., it is the left daughter of some $c \in \mathrm{C}$ and the sister of a tupling symbol whose daughters evaluate to the intended daughters.
5. For any node ν labeled with some "projection" symbol $\pi_i^u \in \Pi$ in T' there is a unique node μ (labeled with some $c \in \mathrm{C}$) which properly dominates ν and which immediately dominates a node labeled with a "tupling" symbol whose i-th daughter will eventually evaluate to the value of π_i^k. Moreover, μ will be the first node properly dominating ν which is on a left branch and bears a composition symbol. This crucial fact is arrived at by induction on the construction of $\mathcal{G}'$ from $\mathcal{G}$.

By 4. it is not hard to find possible dominees in any T'. It is the problem of determining the actual "filler" of a candidate-dominee which makes up the complexity of the definition of $\lhd$. There are three cases to account for:

6. If the node considered carries a "linguistic" label, it evaluates to itself;
7. if it has a "composition" label c, it evaluates to whatever its leftmost daughter evaluates to;
8. if it carries a "projection" label π_i^k, it evaluates to whatever the node it "points to"—by (5.) the i^{th} daughter of a "tupling" node which is dominated by the first C-node on a left branch dominating it—evaluates to.

According to the observations made above, the automaton given in Fig. 2 starts on any node with a "linguistic" label (denoted here by L) which means for the given example $\bullet, a_1, a_2, a_3$. Then it has to go up the first branch, read a composition symbol and descend to its sister. If it reads a "linguistic" node, the automaton stops. If it reads a composition symbol, the automaton goes to the left daughter and tries again. If it reads a tupling symbol, the automaton proceeds with its daughters. On finding a projection symbol, it searches for the appropriate "filler" by going upwards until it is on a leftmost branch which is labeled with a composition symbol. Then it walks to the second sister or further down the leftmost branch until it hits a tupling node to whose appropriate daughter it descends to find the filler.
However, there is another interpretation of such an automaton. Viewed as an ordinary finite-state automaton over the alphabet Δ, $\mathfrak{A}_\lhd$ recognizes a regular (string-) language, the *walking language* W which can be translated recursively into an MSO formula $\mathrm{trans}_{W_\lhd}$ defining the relation $\lhd$ (see Bloem & Engelfriet 1997). We leave the rather tedious process of converting the walking language for the automaton given in Fig. 2 to the reader (a full example of such a conversion can be found in Kolb *et al.* (2000)).

[6]The reader is encouraged to check them against trees T' generated by $\mathcal{G}'$ given in Fig. 1.

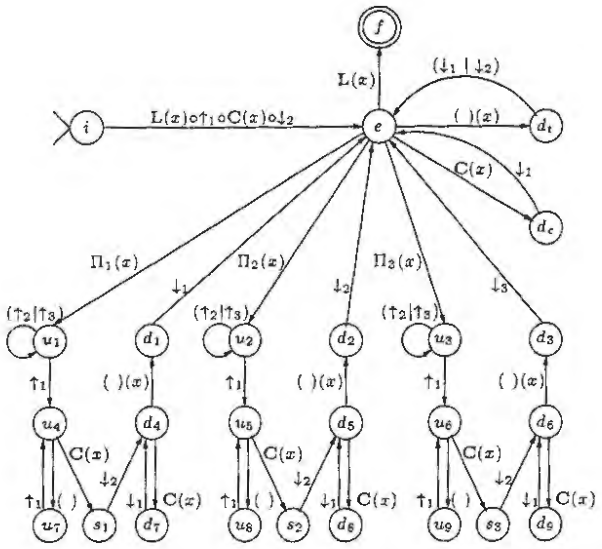


Figure 2: The tree-walking automaton for immediate dominance: $\mathfrak{A}_{\lhd}$

To present the actual MSO transduction, we need one further auxiliary definition. It is a well-known fact (e.g. Bloem & Engelfriet 1997) that the reflexive transitive closure R^* of a binary relation R on nodes is (weakly) MSO-definable, if R itself is. This is done via a second-order property which holds of the sets of nodes which are closed under R: $R\text{–closed}(X) \iff_{def} (\forall x, y)[x \in X \wedge R(x,y) \rightarrow y \in X]$.
Finally, the MSO transduction $(\varphi, \psi, (\theta_q)_{q \in Q})$ with $Q = \{\lhd, \lhd^*, \lhd^+, \prec, \dots\}$ we need to transform the lifted structures into the intended ones is given as follows:

$$\begin{aligned}
\varphi &\equiv \varphi_{\mathfrak{A}_{G'}} \\
\psi &\equiv (\exists y)[\mathsf{trans}_{W_\lhd}(x,y) \vee \mathsf{trans}_{W_\lhd}(y,x)] \\
\theta_\lhd(x,y) &\equiv \mathsf{trans}_{W_\lhd}(x,y) \\
\theta_{\lhd^*}(x,y) &\equiv (\forall X)[\lhd\text{–closed}(X) \wedge x \in X \rightarrow y \in X] \\
\theta_{\lhd^+}(x,y) &\equiv x \lhd^* y \vee x \not\approx y \\
\theta_\prec(x,y) &\equiv \text{another tree-walking automaton} \\
\theta_{\text{labels}} &\equiv \text{taken over from } R
\end{aligned}$$

As desired, the domain of the transduction is characterized by the MSO formula $\varphi_{\mathfrak{A}_{G'}}$ for the lifted trees. The domain, i.e., the set of nodes, of the intended tree is characterized by the formula ψ which identifies the nodes with a "linguistic" label which stand indeed in the new dominance relation to some other node. Building on it, we define the other primitives of a tree description language suited to linguistic needs. For reasons of space, we have to leave the specification of the precedence relation open. It is more complicated than dominance, but can be achieved with another tree-walking automaton.

5. Conclusion

Taking the result of Michaelis' translation of MGs as the input we have shown how to define a RTG by lifting the corresponding MCFG-rules by viewing them as terms of a free Lawvere

theory. This gives us both a regular (via tree and tree-walking automata) and a logical characterization (via MSO logic and an MSO definable transduction) of the intended syntactic trees. Equivalently, we provide both an operational and a denotational account of Stabler's version of Minimalism without having to go via derivation trees.

It remains to be seen whether one can find a machine model for the entire MSO transduction. A likely candidate are the macro tree transducers (MTT) introduced in Engelfriet & Maneth (1999). Since they characterize the class of MSO definable tree translations if extended with regular look-ahead and restricted to finite-copying, we are quite optimistic that we will be able to use them to efficiently implement the transduction. This would also characterize the class of languages we can handle. Engelfriet and Maneth show that the result of applying an MTT to a regular tree family yields the tree languages generated by context-free graph grammars.

References

BLOEM R. & ENGELFRIET J. (1997). *Characterization of Properties and Relations defined in Monadic Second Order Logic on the Nodes of Trees.* Tech. Rep. 97-03, Leiden University.

COURCELLE B. (1997). The expression of graph properties and graph transformations in monadic second-order logic. In G. ROZENBERG, Ed., *Handbook of Graph Grammars and Computing by Graph Transformation. Vol. I: Foundations*, p. 313–400. World Scientific.

ENGELFRIET J. (1997). Context-free graph grammars. In G. ROZENBERG & A. SALOMAA, Eds., *Handbook of Formal Languages. Vol. III: Beyond Words*, p. 125–213. Springer.

ENGELFRIET J. & MANETH S. (1999). Macro tree transducers, attribute grammars, and MSO definable tree translations. *Information and Computation*, **154**, 34–91.

GÉCSEG F. & STEINBY M. (1984). *Tree Automata.* Budapest: Akadémiai Kiadó.

KOLB H.-P., MÖNNICH U. & MORAWIETZ F. (2000). Descriptions of cross-serial dependencies. To appear in a special issue of *Grammars*. Draft available under `http://tcl.sfs.nphil.uni-tuebingen.de/~frank/`.

MEZEI J. & WRIGHT J. (1967). Algebraic automata and contextfree sets. *Information and Control*, **11**, 3–29.

MICHAELIS J. (1999). Derivational minimalism is mildly context-sensitive. In M. MOORTGAT, Ed., *LACL '98*, LNAI. Springer. To appear.

MÖNNICH U. (1998). TAGs M-constructed. In *TAG+ 4th Workshop, Philadelphia.*

RAMBOW O. & SATTA G. (1999). Independent parallelism in finite copying parallel rewriting systems. *Theoretical Computer Science*, **223**, 87–120.

ROGERS J. (1998). *A Descriptive Approach to Language-Theoretic Complexity.* Studies in Logic, Language, and Information. CSLI Publications and FoLLI.

SEKI H., MATSUMURA T., FUJII M. & KASAMI T. (1991). On multiple context-free grammars. *Theoretical Computer Science*, **88**, 191–229.

STABLER E. (1997). Derivational minimalism. In C. RETORÉ, Ed., *Logical Aspects of Computational Linguistics*, p. 68–95, Berlin: Springer. LNAI 1328.

THOMAS W. (1990). Automata on infinite objects. In J. VAN LEEUWEN, Ed., *Handbook of Theoretical Computer Science*, chapter 4, p. 133–191. Elsevier Science Publishers B. V.

WEIR D. J. (1992). Linear context-free rewriting systems and deterministic tree-walk transducers. In *30th Meeting of the Association for Computational Linguistics (ACL'92).*

Workshop TAG+5, Paris, 25-27 May 2000

A Logical Approach to Structure Sharing in TAGs

Adi Palm

Department of General Linguistics
University of Passau
D-94030 Passau

Abstract

Tree adjoining grammars (TAG) represent a derivational formalism to construct trees from a given set of initial and auxiliary trees. We present a logical language that simultaneously describes the generated TAG-tree and the corresponding derivation tree. Based on this language we formulate constraints indicating whether a tree and a derivation tree mean a valid TAG-generated tree. A method is presented that extracts the underlying TAG from an (underspecified) TAG-tree and its derivation. This leads to an alternative approach of representing shared structures by means of TAGs. The result is a more general representation of movement which requires no indices since it basically makes use of the properties of the adjunction operation.

1. Introduction

Recently, we find several approaches establishing a logical description of finite trees, e.g., first-order logic (Backofen *et al.*, 1995), dynamic logic (Kracht, 1995), temporal logic (Palm, 1999), monadic second-order logic (Rogers, 1998). However, most of them lead to the class of recognizable sets of trees (Thatcher & Wright, 1968). Provided a finite label domain this applies to all logical formalisms that are equal or weaker than the (weak) monadic second-order logic (Rabin, 1969). However, TAGs do not belong to this class, since TAGs are properly stronger than context-free grammars. But a set of trees is recognizable if and only if it can be recognized by tree automaton, which can be also encoded as a context-free grammar. Nevertheless, there are logical formalisms to specify structures beyond context-free derivations. For instance, Rogers proposes in (1999) and previous works a logical description of TAGs that is based on a 3-dimensional view of trees. The important issue of his approach is to combine the derived TAG-tree and its derivation tree to a single 3-dimensional structure.

Similarly, we propose a formal method to establish tree constraints outside the context-free paradigm that employs an additional tree structure that is linked with the tree in a particular manner. For TAGs we consider the corresponding TAG-derivation tree where each node of the derived TAG-tree is linked with the corresponding derivation node, e.g., if we adjoin the auxiliary tree β to the auxiliary tree α then we reach a derivation tree with the root m_α that has a single child m_β. Correspondingly, we link each node of the underlying initial tree α with the m_α node in the derivation tree and each node of the adjoined auxiliary tree β with the m_β node. Instead of labeling the nodes of the derivation tree with the name of the corresponding elementary tree and the tree address of the corresponding adjunction node, the former is sufficient due to these links. After adjoining a further β tree to the former β tree, the derivation tree includes a second m_β node below the first one. In addition, the nodes of the second β tree are linked with the second m_β node of the derivation tree. Obviously, the dominance relation in the derivation tree expresses nested auxiliary trees in the derived TAG-tree.

In contrast to Rogers' 3-dimensional trees, we keep the derived TAG-tree as a unit in order to

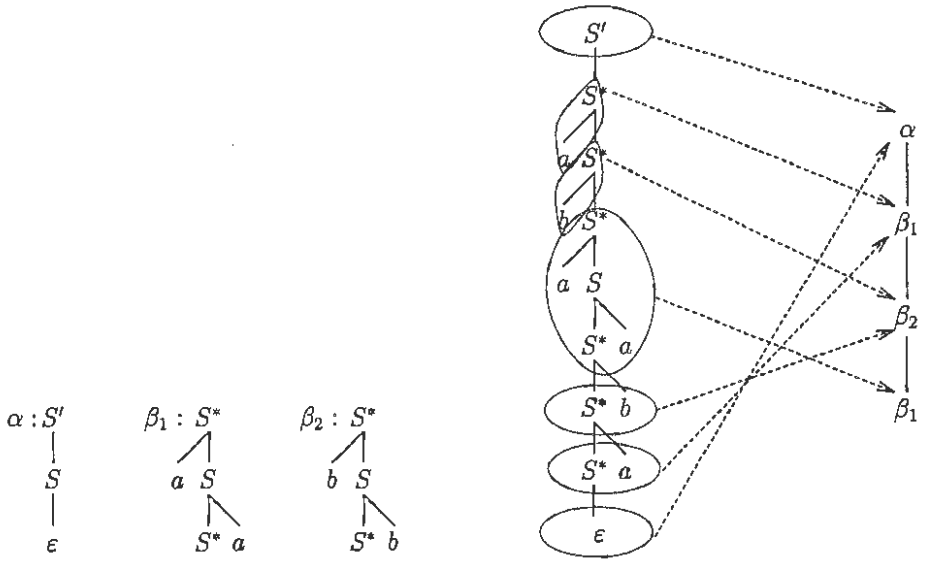


Figure 1: TAG generating the copy language and the derivation for *abaaba*

be able to access the TAG-tree directly without applying a particular projection (or a similar function) to the overall structure. Therefore we can still use one of the logical formalisms describing trees mentioned earlier to partially specify a set of TAG-trees. But if we want to make use of the special, non-context-free properties of TAGs, we must consider the links to refer to the corresponding nodes of the derivation tree. This linking function enables to specify sets of nodes in the TAG-tree, which we cannot describe in a formalism only capturing recognizable set of trees. As an illustrating example we consider a simple TAG generating the copy language $\{ww \mid w \in \{a,b\}^*\}$ (see Figure 1). Obviously, each occurrence of a letter in the first word shares the same auxiliary tree with the corresponding occurrence in the second word. In Figure 1 we find the corresponding TAG-tree and its derivation tree for the word *abaaba*.

In the approach presented here we make some important assumptions concerning TAGs. We employ a special node predicate *Adj* to indicate the adjunction nodes. Moreover, we take for granted that the root and the foot fails *Adj*, and the adjunction nodes do not immediately dominate each other. Therefore every adjoined tree is only bounded by nodes of the tree it was adjoined to. Instead of simple node labels we use a finite set Σ of unary node predicates. Hence, each node is labeled with the (finite) set of predicates that are valid for it. We may only adjoin an auxiliary tree at a node if this node, the root and the foot of this auxiliary tree share at least one common predicate. In the resulting tree the labels of the former root and the foot are the intersection of the labels of the adjunction node with the former labels of the root and foot, respectively. Finally, we consider the substitution, i.e., replacing a leaf with an elementary tree as a particular version of adjunction, where the foot of the adjoined tree remains a leaf.

2. A Logical Specification for TAGs and Their Derivations

Our specification language considers two structures, i.e., the resulting TAG-tree t and its derivation tree d, and the (total) function τ mapping the nodes of t to the corresponding nodes of d. We call the combined structure consisting of these components a *t/d-tree*, where the finite sets Σ and Σ_D denote the label domain for the TAG-tree and the derivation tree, respectively. In

detail, a t/d-structure $\langle T, D, \tau \rangle$ includes a Σ-labeled tree domain $T = \langle t, P_t \rangle$ for the TAG-tree, a Σ_D-labeled tree domain $D = \langle d, P_d \rangle$ and the linking function $\tau: t \rightarrow d$.
In order to specify a particular set of t/d-trees we employ a first-order style formalism which is similar to the one used in (Backofen *et al.*, 1995).[1] The resulting first-order language $L_{t/d}(\Sigma, \Sigma_D)$ includes the binary operators $\lhd$, $\prec$, $\lhd^*$, $\prec^*$, $\lhd_D$, $\lhd_D^*$ representing the immediate dominance relation, the sibling precedence relation and their transitive-reflexive closure for the TAG- and the derivation tree, respectively. The function τ maps each TAG-tree node to the corresponding derivation tree node. In addition, we introduce auxiliary predicates *root* and *foot* to mark the root and the foot of an elementary tree. Further, the predicate *leaf* indicates the leaves of an elementary tree that must keep this property during the whole derivation which is especially true for the foot of an initial tree.
Based on the first-order language $L_{t/d}(\Sigma, \Sigma_D)$ we can specify the formal properties of a well-formed t/d-tree. We consider the intended distribution of the linking function τ and the predicates *root* and *foot* when adjoining the auxiliary tree β to the tree α. Hence, the corresponding derivation tree includes two nodes m_α and m_β where $m_\alpha \lhd_D m_\beta$. Basically, for every derivation node m there is a unique root dominating a unique foot, either one referring to m $(T1)$. In addition, the root dominates all other nodes referring to m $(T2)$, and the foot dominates no other node referring to m $(T3)$. Finally, each node wearing the predicate *leaf* must be a leaf $(T4)$.

$$
\begin{aligned}
&(T1)\ \forall m \exists_1 n, n' \colon n \lhd^+ n' \wedge \tau(n) = m \wedge \tau(n') = m \wedge \mathit{root}(n) \wedge \mathit{foot}(n') \\
&(T2)\ \forall n, n' \colon \tau(n) = \tau(n') \wedge \mathit{root}(n) \Rightarrow n \lhd^+ n' \\
&(T3)\ \forall n, n' \colon \tau(n) = \tau(n') \wedge \mathit{foot}(n) \Rightarrow \neg\, n \lhd^+ n' \\
&(T4)\ \forall n, n' \colon \mathit{leaf}(n) \Rightarrow \neg\, n \lhd n' \\
&(T5)\ \forall n \lhd n' \colon (\tau(n) = \tau(n') \wedge \neg \mathit{root}(n) \wedge \neg \mathit{foot}(n')) \\
&\qquad \vee (\tau(n) \lhd_D \tau(n') \wedge \mathit{root}(n')) \vee (\tau(n') \lhd_D \tau(n) \wedge \mathit{foot}(n))
\end{aligned}
$$

where the quantifier $\exists_1$ denotes the unique existence. In $(T5)$ we consider the properties of pairs of immediately dominating nodes. Initially, an elementary tree is coherent, i.e., each node and each of its existing immediate neighbors are parts of the same elementary tree. But after adjoining a tree β at an adjunction node of α, this relationship is interrupted for α, namely between the root and the foot of β. Consequently, each pair of nodes n and n' with $n \lhd n'$ refer to the same elementary tree if neither $\mathit{foot}(n)$ nor $\mathit{root}(n')$ obtains. Otherwise, either n' is the root of β or n is the foot β, where α is the parent of β in the derivation tree, since according to the previous assumptions every adjoined tree is bounded by nodes of the tree we are adjoining to.
The constraints $(T1)$ to $(T5)$ sufficiently specify a valid TAG-tree and its derivation tree provided that either structure is a valid (ordered) finite tree. Hence, it must be possible to separate an arbitrary t/d-tree satisfying the above constraints into a corresponding set of elementary trees. We start this backward derivation at an arbitrary leaf m of the derivation tree. Following $(T1)$ there is a unique root n_r and a unique foot n_f in the TAG-tree marking the boundaries of the corresponding elementary tree. Due to $(T2)$ n_r dominates all nodes n with $\tau(n) = m$, and due to $(T3)$ n_f dominates none of them. Since m is a leaf, $(T5)$ asserts that all nodes dominated by n_r and not dominated by n_f refer to the same elementary tree m. Therefore we can undo the adjunction of the m-tree leading to an m-labeled auxiliary tree. We remove m in the derivation tree, and in the TAG-tree we replace the m-tree with a new adjunction node n_m referring to the parent of m and whose label is the union of the labels of n_r and n_f except *root* and *foot*. In the same manner we handle the remaining t/d-tree until a single derivation node

[1] Selecting first-order logic as the specifying formalism for both kind of structures should be considered as a working example rather than restricting our approach to this kind of logic. Nevertheless, one can employ all kinds of formalisms, e.g., monadic second order logic, that describe recognizable sets of (finite) trees.

remains which denotes the initial tree of the derivation. Finally we should note that this method does not ensure for elementary trees to be uniquely associated with their labels. However, an appropriate modification of the label domain Σ_D could assert this.
Consequently, every t/d-tree satisfying the constraints $(T1)$ to $(T5)$[2] is generated by a certain TAG whose necessary elementary trees result from the backward derivation described above. Since the backward derivation does not consider the inner structure of an elementary tree, i.e., the nodes satisfying neither *root* nor *foot*, this part of the considered TAG-tree can be underspecified. In that case the result of the backward derivation is underspecified, too. By a minor modified backward derivation which manages alternative results we could handle arbitrary underspecification as well.
Obviously, the TAG resulting from a backward derivation of an (underspecified) t/d-tree also generates other t/d-trees than the given one. More generally, one or more given (underspecified) t/d-trees may be considered as a system generating a TAG that recognizes at least these t/d-trees and its predecessors and successors in the TAG derivation. Thus, as a basic application the backward derivation can be employed to describe a particular property of TAG-trees by means of an underspecified t/d tree and to construct a corresponding set of elementary trees. As a linguistic application, we consider an underspecified t/d-tree describing a particular grammatical phenomenon, and hereafter, we achieve a corresponding TAG. Hence, we are able to obtain information on modeling syntactic properties by means of TAGs.

3. Representing Structure Sharing

Structure sharing is an important issue in natural language syntax. In general, it is necessary if a constituent occurs in a position that is different from the one licensing it (or at least a significant part of it). For instance, in the question "Which girl did we meet yesterday?", the object phrase "which girl" occurs in the sentence initial position rather than in the object position immediately after the verb, where it receives its case and θ-role. Typically, we represent structure sharing as a derivational process called movement, i.e., a moved phrase XP_i leaves a trace t_i at its former position; hence we write "[Which girl]$_i$ did we meet t_i yesterday?". Similarly, we handle topicalized objects, e.g., "[This nice girl]$_i$ we met t_i yesterday". However, the indices we use to indicate structure sharing give rise to a problem concerning the finiteness of the label domain Σ. In general, an arbitrary number of such indices may occur. This leads to an infinite number of necessary labels which we cannot handle in our $L_{t/d}$-formalism. However, we will illustrate how to handle structure sharing in TAGs without employing such indices.
Most TAG approaches to wh-movement and topicalization, e.g., XTAG (1999), assume an initial tree that describes the whole sentence structure including the moved phrase and its trace. Consequently, we require similar trees for all kinds of movement and sentence structures. Moreover the (structural) distance between the co-indexed nodes is bounded according to the specification in its initial tree. However, this method fails to represent long distance movement as in

Who$_i$ do we think that Bill knows that Rachel saw that John kissed t_i.

where the distance (within the tree) between the moved node and its trace is arbitrary since such a structure requires a series of adjunctions between the co-indexed nodes. In order to reach a more general approach to movement in TAGs, we apply the backward derivation to such crucial tree structures. In detail, we extend a given tree to a corresponding t/d-structure satisfying $(T1)$ to $(T5)$ by assigning appropriate values for τ, *root* and *foot*. As a basic property of t/d-trees we proposed that we store shared information in the derivation tree rather than using indices

[2]Obviously, (T4) can be ignored since the *leaf* predicate only prevents adjunctions beyond the tree considered. Nevertheless, to achieve a more restricted TAG we may assume that initially all leaves must satisfy *leaf*.

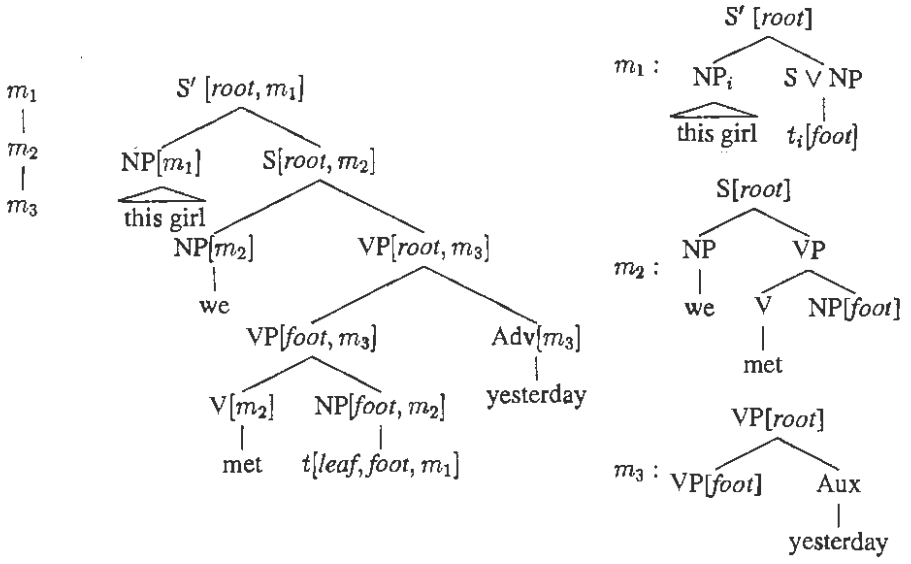


Figure 2: t/d-tree and TAG for "this girl we met yesterday"

in the derived TAG tree. Then we can express an arbitrary number of shared properties while keeping a finite label domain for the TAG tree. We presume therefore that fundamentally shared structures belong to the same elementary tree. Accordingly, we assign the moved noun phrase and its trace to the same elementary tree m_1 which must also include the S' node as its root. Note that this does not mean that S' must share any properties with the NP, actually m_1 only serves to store the common features of the NP and its trace. For the foot of m_1 we select the trace. Similarly, we obtain that S is the root of m_2 and the object NP its foot. Finally, the adverb is assigned to the auxiliary tree m_3 with both VP nodes as its root and foot. The resulting t/d-tree and the corresponding elementary trees is shown in (Figure 2) where we do not explicitly write the *leaf* predicate that is assigned to the trace and the lexical entries.

Generalizing the result obtained above, the starting and the landing position of a movement are part of the same elementary tree to which we must adjoin the structure occurring between. The distance between a moved phrase and its trace depends on the number and complexity of the elementary trees adjoined to the movement tree where additional constraints on the derivation tree can restrict this distance. However, adjoining the inner structure seems to be inconsistent to most other current TAG approaches to natural languages. Nevertheless, this conflict turns out to be only superficial if we assume initial trees where the position of the foot and the substitution nodes are underspecified. For instance, we consider the argument structure of a verb where the nodes for the arguments are marked for substitution or to be the foot. So we can move at most argument and the remaining ones must be substituted; for the moved argument we assume a corresponding substitution node at the landing position, too.

Since movement is not restricted to NPs we assume a more general elementary tree for movement where the category of the moved phrases is underspecified and the moved phrase must be inserted via substitution. Moreover if we select appropriate predicates for the adjunction node, we can specify the auxiliary trees that can be adjoined. Through the resulting elementary tree for movement we can express movement as a particular version of adjunction rather than as a lexical process. Since the moved phrase and its trace are linked by a corresponding elementary

tree no further co-indexing is necessary. As a result we obtain TAGs that require only a finite label domain. Thus, a corresponding the $L_{t/d}$-formula can specify such TAGs appropriately.

4. Conclusion

We have introduced a logical description of TAGs that simultaneously considers the derived and derivation tree, both of which are linked together via a special function. By the constraints $(T1)$ to $(T5)$ we have obtained a notion of TAG-validity that is applicable to arbitrary tree structures. In detail, we have established a backward derivation to verify whether an (underspecified) tree can be generated by a TAG. Using this method we have obtained an alternative approach to represent structure sharing without employing indices in TAG. This way we can describe structure sharing within the $L_{t/d}$ formalism, too. Formally seen, we focus the properties of the adjunction operation rather than putting together complex initial trees. As a further application, it should be possible to extend the backward derivation to a learning algorithm that extracts a TAG from a given tree corpus. Another obvious extension of our $L_{t/d}$ formalism emerges when we consider the derivation tree as a derived tree which is linked with a further derivation tree. This leads to Weir's hierarchy of control languages (1992).

An alternative approach to express structure sharing in TAGs is provided by several variants of *multi-component TAGs* (Weir, 1988; Rambow, 1994) where a set of elementary trees is simultaneously adjoined (or substituted). Obviously, such a set identifies its members. However, there may be an arbitrary number of such sets in a derived tree which means an arbitrary number of indices and, hence, an infinite label domain. Nevertheless our approach can be extended to such formalisms as long as the label domains are finite and the derivation trees are recognizable.

References

BACKOFEN R., ROGERS J. & VIJAY-SHANKER K. (1995). A first-order axiomatization of the theory of finite trees. *Journal of Logic, Language and Information*, 4, 5–39.

KRACHT M. (1995). Syntactic codes and grammar refinement. *Journal of Logic, Language and Information*, 4, 41–60.

PALM A. (1999). Propositional tense logic for finite trees. In *Proceedings of 6th Meeting on Mathematics of Language (MOL6)*.

RABIN M. (1969). Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, **141**, 1–35.

RAMBOW O. (1994). *Formal and Computational Aspects of Natural Language Syntax*. PhD thesis, Department of Computer and Information Science, University of Pennsylvania.

ROGERS J. (1998). *A Descriptive Approach to Language-Theoretic Complexity*. Stanford, California: CSLI.

ROGERS J. (1999). Generalized tree-adjoining grammars. In *Proceedings of 6th Meeting on Mathematics of Language (MOL6)*.

THATCHER J. & WRIGHT J. (1968). Generalized finite automata theory with an application to decision porblems of second-order logic. *Mathematical System Theory*, **2**, 57–81.

WEIR D. (1988). *Characterizing Mildly Context-Sensitive Grammar Formalisms*. PhD thesis, Department of Computer and Information Science, University of Pennsylvania, Phildelphia.

WEIR D. (1992). A geometric hierarchy beyond context-free grammars. *Theoretical Computer Science*, **104**, 235–261.

XTAG RESEARCH GROUP (1999). *A lexicalized tree-adjoining grammar for English*. technical report, Institut for Research in Cognitve Science, University of Pennsylvania, Philadelphia.

From Intuitionistic Proof Nets to Interaction Grammars

Guy Perrier

LORIA - Université Nancy2
Campus Scientifique - BP 239
54506 Vandœuvre-lès-Nancy Cedex -France
e-mail: perrier@loria.fr

Abstract

We show that the construction of proof nets in the implicative fragment of intuitionistic linear logic reduces to the generation of models in the shape of completely specified and neutral trees from polarised tree descriptions. This provides us with a new framework for revisiting grammatical formalisms and leads us to introduce Interaction Grammars which aim to take advantage of two main characteristics of this framework: under-specification and polarities.

Introduction

Apparently, Categorial Grammars (CG) and Tree Adjoining Grammars (TAG) are two very different approaches to the syntax of natural languages. CG are characterised as calculi of polarised syntactic types based on the idea that grammatical categories are consumable resources: some constituents behave as resource consumers whereas others behave as resource providers so that syntactic composition is viewed as a process in which consumers and providers try to cancel each other out; most often, CG are expressed in a logical framework that takes the Lambek Calculus as its nucleus, which combines resource sensitivity with order sensitivity. This intimate combination, which explains the central role of this logic as a framework for CG, is at the same time a cause of rigidity which limits its expressive power greatly. The search for an appropriate way of relaxing this framework constitutes an important research area of CG (Moo96).

TAG do not manipulate syntactic types but syntactic trees with the adjunction operation as their cornerstone. In this way, their expressivity goes beyond that of CG but their rigidity is also their weak point: like CG, they are lexicalised and all syntactic configurations in which a word is used are stored in the lexicon in the form of elementary trees. As soon as a word is used in a new syntactic configuration, a new elementary tree must be added to the lexicon directly or via a lexical rule. In this way, lexicons quickly become colossal and very awkward to manage.

Recent works have contributed to establish links between CG and TAG with the common aim to embed TAG in a logical setting (AFV97; JK97). Our proposal aims to provide a common framework for comparing CG and TAG and for overcoming some of their specific limitations in a new formalism which we call *Interaction Grammars* (IG). The common framework that we choose is that of *tree descriptions*. This notion is not new in the TAG community since it was introduced by (VS92) for making adjunction monotone and embedding TAG in a unification framework. The key idea behind this notion is to replace reasoning about syntactic trees as completely defined objects with reasoning about properties which are used for characterising

these trees; in this way, syntactic trees are viewed as models of descriptions. This allows one to use the notion of *under-specification* in a fruitful manner for structuring TAG lexicons (Can99) or for dealing with semantic ambiguity (MK; ENRX98) for instance. This also allows a new and promising constraint-based style of computing within linguistics (?; DT99; Bla99).
We propose to show that CG can be revisited in this framework with new developments which lead us to IG. The starting point of this proposal is purely theoretic since it concerns proof theory in Intuitionistic Linear Logic (ILL).

1. Intuitionistic proof nets as polarised tree descriptions

Resource sensitivity of linear logic entails a specific form of proof: *proof nets* (Gir87). In the general framework of classical linear logic, these proof nets are not directed so that each extremity of a proof net can be viewed as either an input or an output; in other words, each formula that is attached to an extremity of a proof net can be considered either as an assumption (input) or as a conclusion (output) of the proof.
In ILL, this symmetry is broken and things freeze in a configuration where all formulas become *polarised*, one as the output (denoted +) and the others as the inputs (denoted –). F. Lamarche has devised a correctness criterion for these proof nets which takes their specificity into account (Lam96). Hence, he has sketched a more abstract representation of proof nets which is inspired by the games semantics for PCF introduced by (HO93) and which only takes the induced order between atomic formulas into account.
By using the notion of tree description, we propose to perfect this representation for Implicative Intuitionistic Linear Logic (IILL), which is the implicative fragment of ILL, built only from the linear implication ($-\circ$); we choose this fragment because of its linguistic interest but our proposal can be easily extended to the whole multiplicative fragment.

1.1. Syntactic descriptions of IILL formulas

Let $\mathcal{P}$ be a set of propositions. The set of IILL formulas built from $\mathcal{P}$ is defined by the grammar $\mathcal{F} ::= \mathcal{P} \mid \mathcal{F} -\circ \mathcal{F}$. By adding a polarity + or – to every IILL formula, we obtain the set $\mathcal{F}(\mathcal{P})$ of polarised IILL formulas. From the syntax of these formulas, we abstract particular tree descriptions, called *IILL syntactic descriptions*.

Definition 1.1 *An IILL syntactic description D is a set of polarised atomic formulas taken from $\mathcal{F}(\mathcal{P})$ that is equipped with two binary relations:* dominance *(denoted $>^*$) and* immediate dominance *(denoted $>$).*

For every polarised IILL formula F^p (p represents the polarity + or – and $-p$ its opposite), we build its syntactic description, denoted $D(F^p)$ from the root, denoted $Root(D(F^p))$, to the leaves recursively according to the following definition.

Definition 1.2 *$D(F^p)$ is an IILL syntactic description such that:*

- *if F^p is atomic, then $D(F^p)$ is reduced to the unique element F^p, the two relations $>^*$ and $>$ are empty and $Root(D(F^p)) = F^p$;*
- *if $F^p = (F_1 -\circ F_2)^p$, then $D(F^p)$ is the disjoint union of $D(F_1^{-p})$ and $D(F_2^p)$ where the relations $>^*$ and $>$ are completed with a relation between $Root(D(F_1^{-p}))$ and $Root(D(F_2^p))$ according to the following rule:*
  - *if p=+, then $Root(D(F_2^+)) >^* Root(D(F_1^-))$ and $Root(D(F^p)) = Root(D(F_2^+))$;*

– *if* p=–, *then* $Root(D(F_2^-)) > Root(D(F_1^+))$ *and* $Root(D(F^p)) = Root(D(F_2^-))$.

According to the previous definition, an IILL syntactic description has a very particular shape: it appears as a hierarchy of levels which alternate positive and negative formulas and, at the same time, dominance links and immediate dominance links between them.

1.2. Provability in IILL as validity of syntactic descriptions

Syntactic descriptions are interpreted on trees according to the following definition:

Definition 1.3 *A tree T is a model of a syntactic description D if there is an interpretation I from D to T such that:*

- *For every node N of T, $I^{-1}(N)$ is composed of exactly two elements of D: F^+ and F^-.*
- *For every pair $(F_1^{p_1}, F_2^{p_2})$ of D, $F_1^{p_1} > F_2^{p_2}$ ($F_1^{p_1} >^* F_2^{p_2}$) entails that $I(F_1^{p_1})$ is the parent (an ancestor) of $I(F_2^{p_2})$ in T.*

If a description D accepts a model, D is said to be valid.

In others terms, a syntactic description is valid if one can merge its nodes by dual pairs while respecting its dominance constraints. Equivalence between provability of IILL sequents in linear logic and validity of the corresponding syntactic descriptions is established by the following theorem.

Theorem 1.1 *An IILL sequent $F_1, \ldots, F_n \vdash G$ is provable in linear logic if and only if the syntactic description $D((F_1 \multimap \cdots \multimap F_n \multimap G)^+)$ is valid.*

Sketch of proof 1.1 *To show that provability entails validity, we proceed by induction on proofs of IILL sequents in the linear sequent calculus. We consider the last inference I of any proof of such a sequent. By induction hypothesis, we get models of the syntactic descriptions of the I-premises and it is not very difficult to combine these models to build a model of the syntactic description of the I-conclusion.*
To show that validity entails provability, we proceed by induction on the number of nodes of syntactic descriptions. We consider any valid description of an IILL formula F. We drop the root R^+ of the description and its dual node R^- which match in a model T; all partial descriptions $D(F_i^-)$ which become unconnected in this way are linked to the children of R^- that dominate them in the model T. In this way, we obtain a set of valid syntactic descriptions to which we can apply the induction hypothesis; as a consequence, we obtain a set of provable sequents from which we deduce $\vdash F$.

Example 1.1 *The transitivity of linear implication is expressed by the provability of the IILL sequent $a \multimap b$, $b \multimap c \vdash a \multimap c$, which amounts to the provability of the one-sided sequent $\vdash (a \multimap b) \multimap (b \multimap c) \multimap (a \multimap c)$. From the left to the right, Figure 1 successively presents the proof net which establishes this provability, the corresponding syntactic description and the model [1] which guarantees the validity of this description. In the proof net, positive formulas are represented by down arrays and negative formulas by up arrays; axioms links are represented by dotted edges.*

Proof search, which, in IILL, takes the form of proof net construction, now reduces to the generation of models from syntactic descriptions; some details are forgotten while essentials

[1] The model is unique up to an isomorphism.

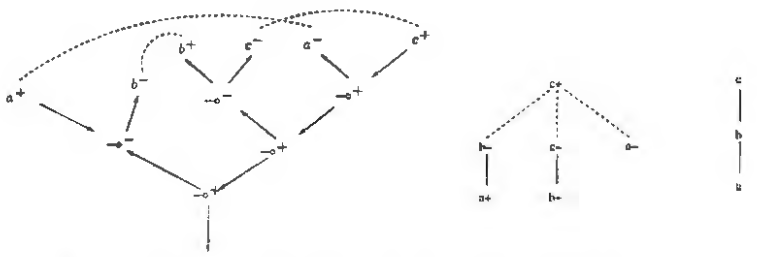


Figure 1: IILL proof of $\vdash (a \multimap b) \multimap (b \multimap c) \multimap (a \multimap c)$

are preserved: identification of dual nodes of a syntactic description corresponds to putting an axiom link in the corresponding proof net and the correctness criterion of the proof net is constantly guaranteed by the tree-like structure of the description. In order to extend these results to the whole multiplicative fragment of ILL, we have to relax the tree structure of descriptions to a DAG structure.

1.3. Planarity of Lambek proof nets and precedence order in syntactic descriptions

In the implicative fragment of the Lambek Calculus, linear implication is replaced by two implications, left and right, respectively denoted \ and / , which results from the non-commutativity of the calculus. Lambek proof nets differ from IILL proof nets by the fact that the premises of inference links are ordered and axiom links must not cross each other (Roo91).

This enrichment of IILL proof nets by a precedence order can be translated in the corresponding syntactic descriptions without difficulties: besides the two relations of dominance and immediate dominance, we add a *precedence* relation between atomic formulas. A difficulty comes then when we want to express the axiom links of a proof net with the merging of dual nodes in the corresponding syntactic description. This operation requires movement of nodes, which generally entails a violation of the precedence order. As a consequence, the monotonicity of the process of generating models from syntactic descriptions collapses. If we try to relax the precedence order, we obtain valid descriptions that correspond to non correct Lambek proof nets where some axiom links cross each other.

The fundamental reason of this difficulty lies in the intimate interweaving between the precedence and dominance orders in Lambek proof nets. The construction of a Lambek proof net can be viewed as the construction of an ordered tree from a syntactic description under the control of both dominance and precedence order. Whereas the initial dominance order is preserved in the final tree, this is not the case for the precedence order: it is only preserved between the children of negative nodes; for the rest, this order is used for bounding the movement of dual nodes in terms of good parenthesising, which corresponds to the planarity of Lambek proof nets.

2. Polarised tree descriptions: a framework for developing grammatical formalisms

2.1. Outline of Interaction Grammars

Even if Lambek Grammars (LG) do not fit in exactly with the framework of polarised tree descriptions, as we have just pointed out, their application to linguistics shows that this framework captures the essentials; the generation of syntactic trees driven by a mechanism of polarities from descriptions which use three kinds of relations: dominance, immediate dominance and precedence.

Concerning TAG, Vijay-Shanker (VS92) proposes their translation in terms of tree descriptions which can be completed with polarities for exactly recovering the shape defined in the previous section. This common shape highlights the main difference between TAG and LG: both definition and realisation of dominance relations are more constrained in TAG than in LG. Two nodes which participate in such a relation must have the same grammatical category in TAG and the relation can only be realized by insertion of another syntactic description between the two nodes, whereas, in LG, the only constraints are polarity and good parenthesising constraints.
By exploiting tree descriptions, some works aim to relax the TAG adjunction operation in order to capture linguistic phenomena which are beyond TAG (RVSW95; Kal99). Unfortunately, the counterpart of a more flexible framework is often over-generation and a loss of computational efficiency in the absence of control on the process of syntactic composition. IG are an attempt of exploiting the flexibility of tree descriptions as far as possible while keeping the notion of polarity as central for controlling syntactic composition.
A particular interaction grammar G, which is associated with a vocabulary $\mathcal{V}$, is defined from a finite set of labels C, which can be in a first approach a set of atomic categories. The basic objects of G are IG syntactic descriptions which are a variant of IILL syntactic descriptions.

Definition 2.1 *An IG syntactic description is a finite set of nodes structured by dominance, immediate dominance and precedence relations. Immediate dominance is defined in two ways: either classically with a binary relation between two nodes or with a parent-children relation which enumerate all children of a node. Every node is equipped with a label from C and a polarity -1, 0 or $+1$.*

IG are lexicalised so that G is completely defined by its lexicon which associates a set of syntactic descriptions to every word of $\mathcal{V}$.
With respect to the abstract IILL syntactic descriptions, IG descriptions present three differences: the use of precedence order in addition to dominance orders, the presence of neutral nodes and the possibility of closing the set of children for a node. These differences are reflected in the definition of a model.

Definition 2.2 *A model of an IG syntactic description D is an ordered tree T such that there exists an interpretation I which respects the following conditions:*

- *every node of the syntactic tree interprets a set of node variables labelled with the same labels; all these variables are neutral, otherwise, there is exactly one positive and one negative variables in this set;*
- *the interpretation respects dominance and precedence relations of D and the tree structure of T is totally realised by means of parent-children relations initially present in the description.*

IG differ from LG on two main points: precedence order between syntactic constituents is dissociated from dominance order and neutral nodes are used for pattern matching between syntactic structures. In this way, parsing amounts to generating models from syntactic descriptions and a parsing process can be viewed as an electrostatic process in which opposite charges attract themselves while charges with the same polarity repel each other, whence the name of Interaction Grammars.

Example 2.1 *Parsing the phrase* Marie que Jean voit *starts with extracting appropriate syntactic descriptions from a lexicon for all its words and gathering them in a unique syntactic description as Figure 2 shows it. The root of the description represents the request whereas*

each of its four children corresponds to a word of the phrase. Every syntactic node is labelled with its grammatical category and its polarity (polarity 0 is omitted). Contrary to IILL descriptions, we choose the opposite convention for polarities, which is better suited to linguistic reality: positive nodes represent actual constituents and negative nodes virtual constituents which are expected. Precedence order between syntactic nodes is denoted with dotted arrays and dominance order with dotted edges. Parsing succeeds in finding a model for this syntactic

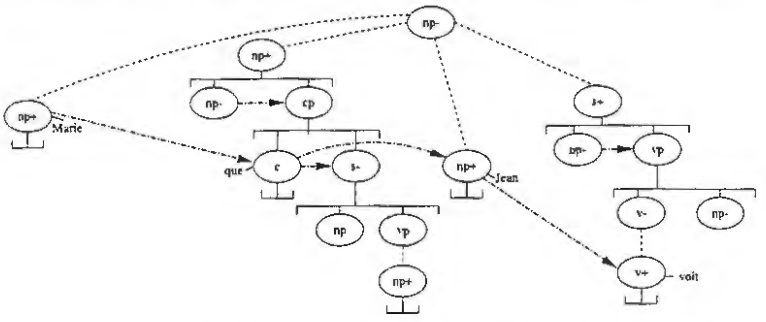


Figure 2: syntactic description of the phrase *Marie que Jean voit*

description: this model is the syntactic tree given by Figure 3.

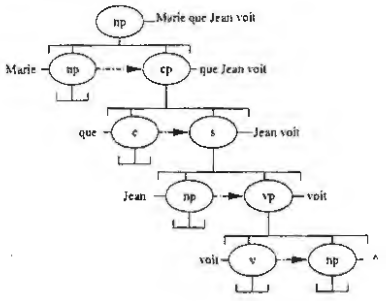


Figure 3: syntactic tree of the phrase *Marie que Jean voit*

In this first version, IG go beyond the expressivity of LG (for instance, middle extraction from relative clauses is representable in such a framework) but they are still too rigid.

2.2. *Polarised features and non-determinism in descriptions*

The outline of IG that was just presented encounters similar limitations to TAG for expressing the flexibility of word order in natural languages. For instance, the SVO order is sometimes relaxed like in the phrase *Marie que voit Jean*: the object of *voit* is provided by the relative pronoun *que* the form of which indicates the accusative case. As a consequence, there is no more ambiguity on the assignment of the subject and the object of *voit* and word order can be relaxed. Nevertheless, the phrase *Marie que voit il* is not acceptable because the position of the clitic *il* after the verb *voit* generates an interrogative type for the clause *que voit il*. Such a complex interaction between word order and grammatical features is not captured by the previous version of IG.

Another difficulty comes from the fact that a word can be used in several syntactic contexts which often differ only partially. For instance, the verb *voit* can be used without any object like

in the interrogative sentence *Jean voit-il ?*.
A answer to these problems consists in a refinement of our formalism on two main points:

- polarities are transferred from syntactic constituents to elementary features that are used for describing their properties, which gives a finer granularity to this notion;
- syntactic descriptions can be composed in two ways: in a product of two descriptions, the resources of both components are used whereas in a sum, either the resources of the first component or the resources of the second are used, but not both.

These refinements make the notion of model more complex: the neutrality condition refers now to features and not to nodes and for every choice point in a description exactly one alternative is used in a model.
Because of lack of space, we do not present the IG formal system in its complete shape; we prefer to give an example for illustrating the last refinements of IG. This formal system uses the framework of multiplicative and additive linear logic (MALL): "electrostatic" interaction is expressed by the resource sensitivity of MALL and non-determinism in descriptions by the additive part of this logic.

Example 2.2 *Figure 4 present the possible lexical entries for* Jean, voit *and* il. *Every entry is a combination of partial descriptions organized in a hierarchy according to a decision tree. Every node of this tree is a choice point between two partial descriptions. For instance, the lexical entry of* voit *includes two choice points: the left corresponds to the presence or not of an order subject-verb and the right to the presence or not of an object for the verb. All possible complete descriptions are built by making a choice at each choice point and by superposing all remaining descriptions. In this way, a single entry expresses four syntactic contexts for the verb* voit. *The entry for* il *also includes two choice points corresponding to the presence or not of an explicit subject in the sentence and to the order clitic-verb.*
Positive, negative and neutral features are respectively denoted $\rightarrow$, $\leftarrow$ *and* =. *A polarity which is not followed by a value means that this value is non determined. To remain readable, the figure includes only the most significant features of every node. With these entries, we succeed in parsing* il voit Jean, voit-il Jean ? *and* Jean voit-il ? *at once.*

The price for having a flexible model is a loss of computational efficiency but the monotonicity of the model generation process allows us to use the powerful tool of constraint solving for computing models from syntactic descriptions. Such an approach was inspired from the proposals of (DT99) and it gave rise to the implementation of a prototype in the constraint programming language *Oz* (Smo95). The first experiments show that polarities play a decisive role for computational efficiency and further validate our direction of research: exploiting in a same linguistic model the advantages of both under-specification and polarities.

References

V. M. Abrusci, C. Fouqueré, and J. Vauzeilles. Tree adjoining grammars in noncommutative linear logic. In C. Retoré, editor, *LACL'96. Nancy, France, September 1996*, volume 1328 of *LNCS*, pages 96–117. Springer Verlag, 1997.

P. Blache. *"Contraintes et théories linguistiques : des Grammaires d'Unification aux Grammaires de Propriétés"*. Thèse d'habilitation, Université Paris 7, 1999.

M.-H Candito. *Organisation modulaire et paramétrable de grammaires électroniques lexicalisées. Application au français et à l'italien,*. Thèse de doctorat, Université Paris 7, 1999.

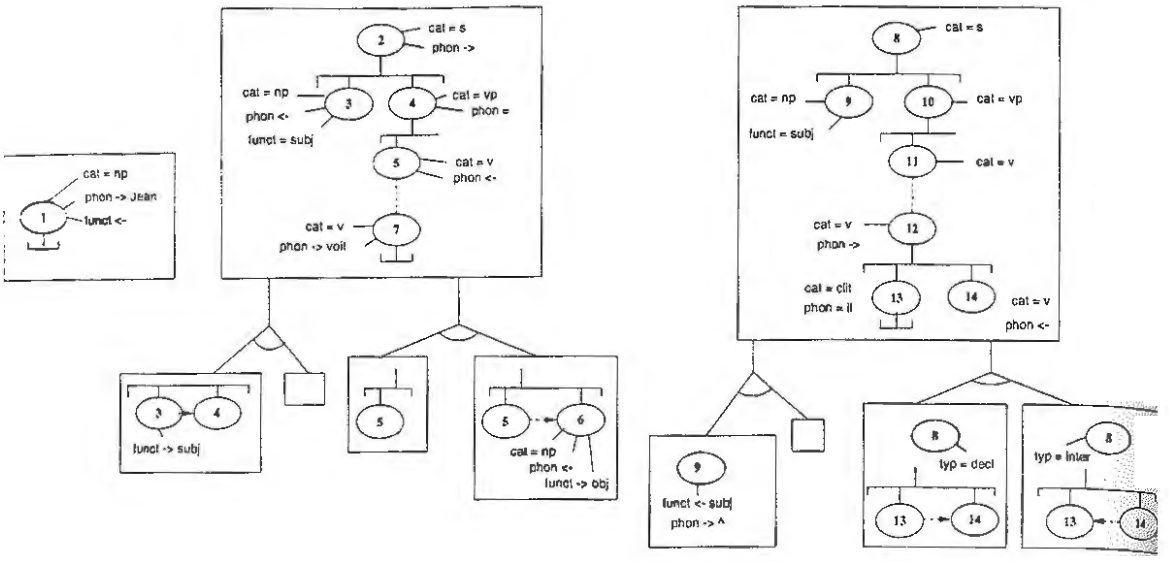


Figure 4: Lexical entries for *Jean*, *voit* and *il*

D. Duchier and S. Thater. Parsing with tree descriptions: a constraint based approach. In *NLULP'99,Dec 1999, Las Cruces, New Mexico*, 1999.

M. Egg, J. Niehren, P. Ruhrberg, and F. Xu. Constraints over lambda structures in semantic underspecification. In *COLING/ACL'98, Montreal,Quebec, Canada. August 1998*, 1998.

J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–102, 1987.

J. M. E. Hyland and L. Ong. Dialogue games and innocent strategies: an approach to intensional full abstraction for PCF. Manuscript, July 1993.

A. Joshi and S. Kulick. Partial proof trees as building blocks for a categorial grammar. *Linguistics and Philosophy*, 20(6):637–667, 1997.

L. Kallmeyer. *Tree Description Grammars and Underspecified Representations*. PhD thesis, Universität Tübingen, 1999.

F. Lamarche. From proof nets to games. *Electronic Notes in Theoretical Computer Science*, 3, 1996. Special Issue of Linear Logic'96, Tokyo Meeting, march 1996.

R. Muskens and E. Krahmer. Talking about trees and truth-conditions. In *LACL'98. Grenoble, France,December 1998.*

M. Moortgart. Categorial Type Logics. In J. van Benthem and A. ter Meulen, editors, *Handbook of Logic and Language*, chapter 2. Elsevier, 1996.

D. Roorda. Resource Logics. Proof-Theoritical Investigations. Phd Thesis, Universiteit van Amsterdam, 1991.

O. Rambow, K. Vijay-Shanker, and D. Weir. D-tree grammars. In *ACL'95*, pages 151–158, 1995.

Gert Smolka. The Oz programming model. In Jan van Leeuwen, editor, *Computer Science Today*, Lecture Notes in Computer Science, vol. 1000, pages 324–343. Springer-Verlag, Berlin, 1995.

K. Vijay-Shanker. Using description of trees in a tree adjoining grammar. *Computational Linguistics*, 18(4):481–517, 1992.

Workshop TAG+5, Paris, 24-27 May 2000

A Comparison of the XTAG and CLE Grammars for English

Manny Rayner, Beth Ann Hockey, Frankie James

Research Institute for Advanced Computer Science
Mail Stop 19-39
Moffett Field, CA 94035-1000
USA

1. Introduction

When people develop something intended as a large broad-coverage grammar, they usually have a more specific goal in mind. Sometimes this goal is covering a corpus; sometimes the developers have theoretical ideas they wish to investigate; most often, work is driven by a combination of these two main types of goal. What tends to happen after a while is that the community of people working with the grammar starts thinking of some phenomena as "central", and makes serious efforts to deal with them; other phenomena are labelled "marginal", and ignored. Before long, the distinction between "central" and "marginal" becomes so ingrained that it is automatic, and people virtually stop thinking about the "marginal" phenomena. In practice, the only way to bring the marginal things back into focus is to look at what other people are doing and compare it with one's own work.

In this paper, we will take two large grammars, XTAG and CLE, and examine each of them from the other's point of view. We will find in both cases not only that important things are missing, but that the perspective offered by the other grammar suggests simple and practical ways of filling in the holes. It turns out that there is a pleasing symmetry to the picture. XTAG has a very good treatment of complement structure, which the CLE to some extent lacks; conversely, the CLE offers a powerful and general account of adjuncts, which the XTAG grammar does not fully duplicate. If we examine the way in which each grammar does the thing it is good at, we find that the relevant methods are quite easy to port to the other framework, and in fact only involve generalization and systematization of existing mechanisms.

The paper is structured as follows. Section 2 presents a very brief overview of the CLE and XTAG grammars. In Section 3, we describe the CLE grammar from the XTAG grammar's point of view, following which Section 4 describes the XTAG grammar from a CLE perspective. Section 5 concludes.

2. An Overview of the XTAG and CLE Grammars

The CLE and XTAG grammars for English are extensively described elsewhere (Pulman, 1992; The XTAG-Group, 1995), and this section will only present the briefest possible summary. Both grammars make a serious attempt to cover all major syntactic phenomena of the language; the CLE grammar also associates each syntactic construction with a compositional scope-free semantics expressed in Quasi Logical Form notation (van Eijck & Alshawi, 1992). In particular, both grammars provide good coverage of the following:

NP structure: Pre- and post-nominal adjectival modification, postnominal modification by PPs, relative clauses, -ing and -ed VPs, comparative and superlative adjectives, possessives,

complex determiners, compound nominals, time, date and code expressions, numbers, "kind of" NPs, determiner and NBAR ellipsis, sentential NPs, apposition, conjunction of NP.
Clausal structure: A large variety of verb types, including intransitives, transitives, ditransitives, copula, auxiliaries, modals, verbs subcategorizing for PPs, particles, embedded clauses, raising and small clause constructions, and combinations of the above; VP modification by PPs, verbal ADVPs, -ing VP, "to" VP declaratives, imperative, WH-questions and Y-N questions; clefts; passives; sentential ADVPs; topicalization; negation; embedded questions; relative clauses; conjunction.

3. What XTAG Tells Us About the CLE Grammar

Both grammars are explicitly lexicalized in a way that makes it easy to define a wide variety of types of complement structure. The XTAG grammar defines complement structure through the very flexible and general mechanism of initial trees combined with the adjunction operation for introducing recursion. Very briefly, each initial tree defines one possible complement structure for its head. Complements can be specified as substitution nodes, with features constraining the possible constituents that can be substituted in; alternately, they can be specified as adjunction nodes, which allow auxiliary trees to be adjoined onto them.
CLE grammar, in contrast, defines complement structure through rule schemas. For example, the VP rule schema is of the form

VP --> V:[subcat=COMPS] | COMPS

the right hand side of which can be glossed as "V whose <subcat> feature has value COMPS, followed by a list of constituents which unify with COMPS". From a TAG perspective, COMPS is more or less equivalent to a list of substitution nodes; there is nothing corresponding to adjunction nodes. The CLE grammar can get along without the adjunction operation, which is absolutely central to XTAG, because it has a powerful mechanism for handling long-range dependencies based on the idea of "gap-threading" (Pereira, 1981; Karttunen, 1986; Pulman, 1992). From the XTAG point of view, it is none the less hard to believe that substitution nodes on their own will be capable of modeling an equally broad range of complement structures.
It does indeed appear to be the case that certain types of complements, particularly those related to idioms and light verbs, are difficult to capture in the CLE framework whereas there is an obvious way to treat these in XTAG. The most convincing example we have identified so far is the class of constructions, very common in English, involving a combination of a verb, a possessive, and a noun, for instance *shake one's head*, *close one's eyes*, *shrug one's shoulders*, *take one's time*, *have one's way*. In all of these constructions, the NP's determiner must be a possessive pronoun agreeing with the verb, and it is in general possible to modify the NP (*shake his pretty head*, *shrug her powerful shoulders*, *have his silly way*). It is obvious that *take one's time* and *have one's way* should be treated as light verb constructions and there are good arguments for modeling the less obvious cases such as *shake one's head*, *close one's eyes* and *shrug one's shoulders* as idioms or light verbs as well, rather than just viewing them as instances of the general transitive verbs shake, close or shrug. For instance, modeling them as idioms or light verbs would be an advantage in the context of a transfer-based machine translation system. Few languages express these concepts in the same way as English[1] and a straight forward compositional treatment will lead to serious complications in defining the associated transfer rules.

[1] for example, *close one's eyes* is *fermer les yeux* in French (transitive verb + definite NP) and *blunda* in Swedish (intransitive verb)

Coding the constraints needed to capture these constructions as idioms is unproblematic in XTAG: for example the initial tree for have one's way will be roughly of the form

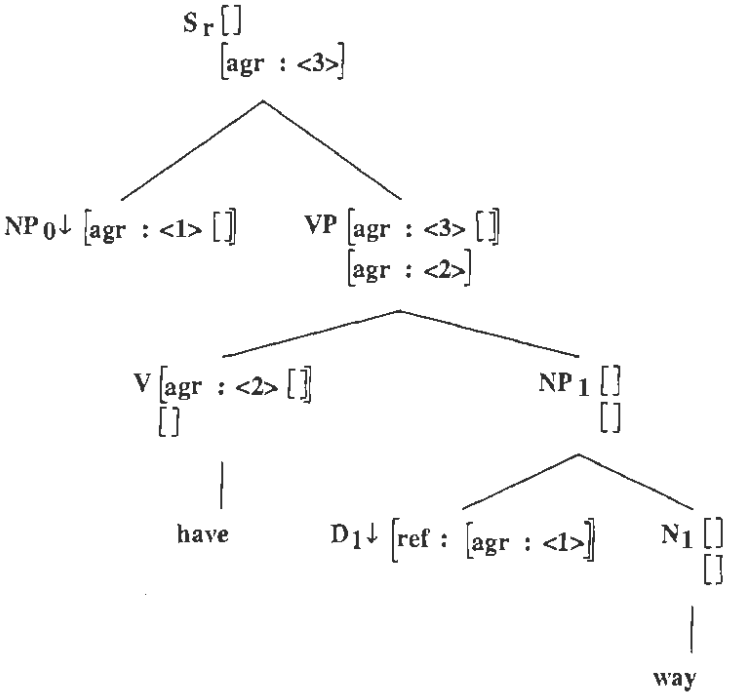


Figure 1: Initial Tree for have one's way

In the current XTAG grammar there is no possessive feature per se. In Figure 1 the determiner is forced to be a possessive pronoun by constraining node D1's <ref> feature to have the same <agr> values as the NP0 and V. Since only pronominal determiners have the <ref> feature, constraining it ensures that the determiner is both pronominal and agrees with the NP0 subject. Notice that because the determiner and the noun of the NP complement are leaves of the tree, it is trivial to state constraints on either of them.

The XTAG treatment cannot be duplicated directly in the CLE framework, since the constraints present in the value of the <subcat> feature are unable to directly reference the DET and N nodes in the complement NP; they can only access that NP's maximal projection. This means that the features on NPs must be such that the relevant information is percolated up through all NP modification rules. Concretely, the category NP needs a head feature which not only specifies whether the DET is a possessive, but also provides agreement information for that possessive; there is however no such feature in the current CLE grammar. We will return to this point in the final section.

4. What the CLE Tells Us About the XTAG Grammar

We now switch to looking at the XTAG grammar from the CLE's point of view. Perhaps the main strength of the CLE grammar is its handling of long-range dependencies, which as al-

ready noted is implemented using a gap-threading method. XTAG's main tools for dealing with long-range dependencies are the ability to state constraints within an elementary tree and the adjunction operation. This works very well for some things, in particular most phenomena involving movement of complements; the basic idea is to encode the movement in a suitable initial tree, and let adjunction take care of the rest. None the less, for someone used to the CLE's design philosophy, it is intuitively implausible that all movement phenomena can be captured in this way, and one expects problems with movement of adjuncts. Once again, we looked for a paradigmatic example of the problem; this time, the most clear-cut case appears to be preposition stranding in adjuncts, as illustrated in sentences like which lake did you swim in? The CLE's treatment is fairly straight forward. The sentence receives the phrase-structure

(1) [S [NP which lake]$_j$ [S did$_i$ you [VP [V t$_i$] [VP swim] [PP in [NP t$_j$]]]]]

in which the empty V constituent is linked to the inverted main verb did, and the empty NP node to the fronted WH+ NP which lake. Features are used to define both kinds of movement. The V is moved through the VP feature <sai> (subject-auxiliary inversion) down to the V gap in the main VP. The NP is moved further, using a gap-threading mechanism, successively through the inner S, the VP, and the PP, to end up coindexed with the NP gap. The mechanisms are described in more detail in (Pulman, 1992).

If we compare the CLE account with that provided by the XTAG grammar, an interesting point emerges. XTAG's treatment of inversion uses the notion of "multi-component adjunction" which is implemented by a feature mechanism. This feature mechanism, described in detail in (Hockey & Srinivas, 1993), forces two elementary trees to act as a "tree set"" by creating a feature clash with the adjunction of the first tree that is resolved by the adjunction of the second.

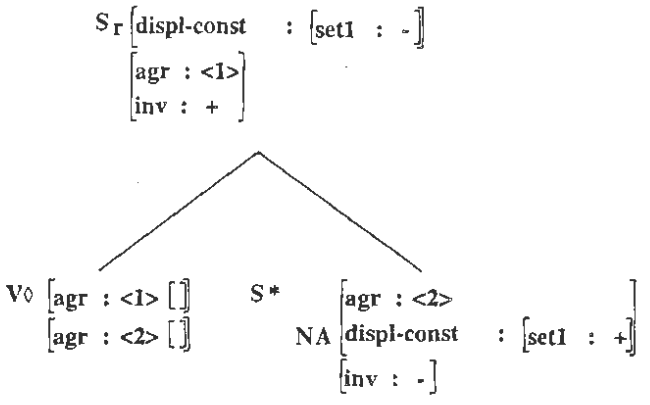


Figure 2: Inverted Verb Auxillary Tree

In the case of inversion the two trees are the tree anchored by the inverted verb shown in Figure 2 and the tree anchored by the verb's trace shown in Figure 3.

The adjunction of either tree individually creates a feature clash between top and bottom feature values of <displ_const> ("displaced constituent") ; however when both trees are adjoined the clash is resolved.

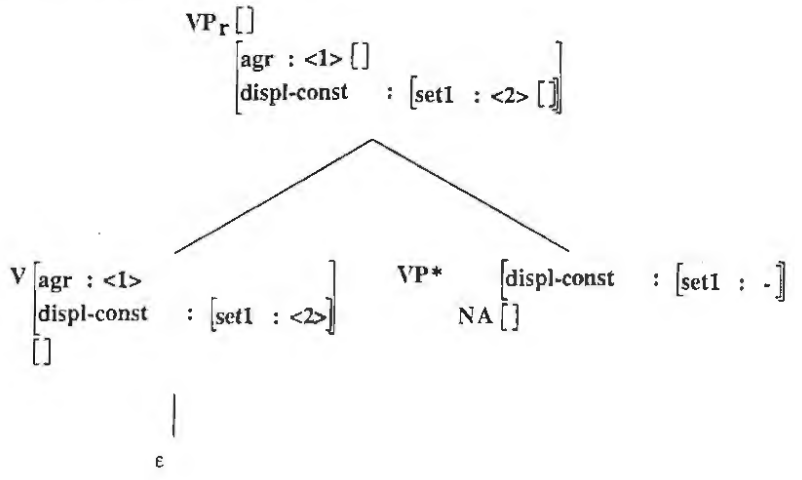


Figure 3: Verb Trace Auxillary Tree

Though formally different, one can see that the methods used by the two grammars to treat subject-verb inversion are essentially the same, and involve passing a feature that licenses the coindexing of the fronted main V and the V gap. This is the only type of adjunct movement permitted by the current version of the XTAG grammar. Since Hockey and Srinivas (1993) actually described how the same treatment could easily be used to account for other types of movement, we need to consider why this has not in fact been implemented.

The reason why it is not trivial to extend the current treatment to cover other types of movement is that the information passed by the <displ_const> feature is too coarse-grained; it says that a constituent has been moved, but fails to specify either the type of constituent or the type of movement. A minimal extension of the current framework to cover adjunct NP-movement cases would open the door to promiscuous filler-gap associations and the acceptance of such ungrammatical strings as *Can$_i$ they go to t_i* in which the inverted verb can associates with the gap in the PP adjunct to ti. It is clearly necessary to constrain the grammar so as to block these and similar incorrect associations of fillers and gaps.

At this point, it is useful to look at the details of the CLE treatment. The CLE grammar uses features to thread gaps, where the representation of the gaps are feature bundles encoding, among other things, the type of constituent being moved. This immediately suggests one refinement to the XTAG account: if a new feature is added which encodes the category of the moved constituent (call it <displ_cat>), then the worst types of incorrect filler-gap associations can be blocked. Unfortunately, this on its own is not enough since we have to take account of the fact that a constituent can contain more than one gap. The CLE grammar addresses this problem by letting the gap features be list-valued.

It is not clear that the CLE approach can be imported directly into XTAG; given the rather different way in which the two grammars thread features, the CLE's list-valued gap-threading mechanism is hard to combine with the TAG adjunction operator, which the CLE grammar lacks. There is however a straight forward solution. Since there are only a very small number of different types of movement in English that can involve adjuncts, it is possible to use a

separate set of features to mediate each type. Specifically, we need four sets of features, which respectively cover verb movement, WH-movement and topicalization, tough movement and right extraposition. (It is possible that passivization forms a fifth class (Pulman, 1987)).
There is nothing linguistically odd about the idea of threading different types of movement separately. It is obvious that subject-verb inversion, WH-movement and right extraposition have different constraints and in most cases operate on different types of constituents. In fact the CLE grammar handles subject-verb inversion and WH-movement through different features and does not cover right extraposition. It does however handle WH-movement and tough-movement through the same set of features, so the interesting question is whether these two should be merged. The most complex aspects of the CLE method are motivated by examples of double extractions involving both WH-movement and tough movement. The main consideration is to enforce the no-crossing dependencies (NCD) constraint as illustrated by the well known "sonata sentences" below; we want to allow (2) and block (3).

(2) Which violin$_i$ are these sonatas$_j$ hard to play t_j on t_i?
(3) *Which sonatas$_i$ is this violin$_j$ hard to play t_i on t_j?

This provides the main justification for using list as opposed to set valued features to implement gap threading (Pulman 1992, pp 71-73). Although a detailed discussion of the NCD constraint is beyond the scope of this paper, it is clear that it applies more strongly to extractions from complements than to extraction from adjuncts[2]. Since the gap threading mechanism would only be used by the XTAG grammar for adjuncts, the critical examples are those that involve double extraction from adjunct positions. Examples of this kind are first of all very rare, and it is not at all clear that the NCD constraint holds for them. For instance example (4) which breaks the constraint seems if anything more natural than the version with no crossing extractions in (5)

(4) Which articles$_i$ are men$_j$ most fun to shop for t_i with t_j?
(5) Which articles$_i$ are men$_j$ most fun to shop with t_j for t_i?

To sum up, it seems fair to say that the idea of using separate features to thread WH-movement gaps and tough movement gaps is at least no worse than the CLE's list-valued scheme, which merges them into a single set of features. Our overall conclusion is that the treatment we have sketched above represents a fully viable approach to adapting the CLE gap threading treatment to the problem of handling adjunct extractions in XTAG.

5. Summary and Conclusions

Looking at the examples in Sections 3 and 4, we see a common pattern. In each case, one grammar can do the job; the other one almost gets there, but falls over at the last moment. Intuitively, one feels that the problem is in neither case impossible to solve.
Let us first look at the example with have one's way from Section 3. As noted, the CLE could deal with this kind of construction if NPs just had the right head features. The reason these features aren't present is not particularly deep; no one saw a need for them, so they were never put in. Since they have to be trailed through a large number of rules involving NPs, the effort needed to add them is non-trivial, and without a concrete reason to attack the problem things stayed as they were. It would however be quite easy, in principle, to make a careful study of the types of features needed to cover the constructions which the XTAG grammar can deal

[2]We would like to thank Bob Levine for insightful discussion on this and other points relating to the NCD constraint.

with. If all of these features were added at once, using sensible macro invocations to do the threading, the work required would not in fact be very frightening. What is more, properly designed macros would make it easy to add new head features as and when they were found to be necessary. The biggest step to take is noting that there is a problem, and making the decision to do something about it.

The difficulties involving movement of adjuncts discussed in Section 4 are less trivial, but nonetheless quite soluble. Though these problems have been recognized for some time and suggestions made about how to provide the necessary additional constraint in the XTAG grammar, a system for doing this has not been implemented. As far as we can see, the real explanation is once again a combination of software engineering considerations and research sociology: a vague feeling that the solution was complex and inelegant, and would involve more effort that would be justified to cover a set of "marginal" phenomena. In actual fact, a comparison with the CLE grammar convinces us that the XTAG group was wrong on all counts. The solution appears fairly principled, and is not very hard to implement; and the phenomena in question, far from being marginal, are at least as central as many of those already covered.

To summarize, we have compared the CLE and XTAG grammars, and found some important and non-trivial problems. The CLE is unable to duplicate some of the complement structure phenomena handled by XTAG, and this appears to be due to an insufficiently detailed modeling of head features. Conversely, XTAG is unable to encode some types of constructs involving adjuncts and movement, and we have suggested that the CLE's gap-threading treatment could be adapted to a implement a more general version of multi-component adjunction. However, we think the real moral of the paper is much more fundamental: if people developing big grammars want to make serious progress, it would be in everyone's interest to carry out this kind of detailed comparison more regularly! We hope that our remarks will encourage other researchers to do so.

6. References

References

HOCKEY B. A. & SRINIVAS B. (1993). Feature-based tag in place of multi-component adjunction: Computational implications. In *Proceedings of the Natural Language Processing Pacific Rim Symposium*, Fukuoka, Japan.

KARTTUNEN L. (1986). D-patr: A development environment for unification-based grammars. In *Proceedings of the Eleventh International Conference on Computational Linguistics.*

PEREIRA F. C. N. (1981). Extraposition grammars. *Computational Linguistics*, 7, 243–256.

PULMAN S. G. (1987). Passives. In *Proceedings of the 3rd Meeting of the European Chapter of the Association for Computational Linguistics.*

PULMAN S. G. (1992). Unification-based syntactic analysis. In H. ALSHAWI, Ed., *The Core Language Engine*. MIT Press.

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.html.

VAN EIJCK J. & ALSHAWI H. (1992). Logical Forms. In H. ALSHAWI, Ed., *The Core Language Engine*. MIT Press.

Practical Experiments in Parsing using Tree Adjoining Grammars*

Anoop Sarkar

CIS Dept, University of Pennsylvania
200 South 33rd Street,
Philadelphia, PA 19104 USA
anoop@linc.cis.upenn.edu

Abstract

We present an implementation of a chart-based head-corner parsing algorithm for lexicalized Tree Adjoining Grammars. We report on some practical experiments where we parse 2250 *sentences from the Wall Street Journal using this parser. In these experiments the parser is run without any statistical pruning; it produces all valid parses for each sentence in the form of a shared derivation forest. The parser uses a large Treebank Grammar with* 6789 *tree templates with about* 120,000 *lexicalized trees. The results suggest that the observed complexity of parsing for LTAG is dominated by factors other than sentence length.*

1. Motivation

The particular experiments that we report on in this paper were chosen to discover certain facts about LTAG parsing in a practical setting. Specifically, we wanted to discover the importance of the worst-case results for LTAG parsing in practice. Let us take Schabes' Earley-style TAG parsing algorithm (Schabes, 1994) which is the usual candidate for a practical LTAG parser. The parsing time complexity of this algorithm for various types of grammars are as follows (for input of length n):

$O(n^6)$ - TAGs for inherently ambiguous languages

$O(n^4)$ - unambiguous TAGs

$O(n)$ - bounded state TAGs e.g. the usual grammar G where $L(G) = \{a^n\ b^n\ e\ c^n\ d^n \mid n \geq 0\}$ (see (Joshi *et al.*, 1975))

The grammar factors are as follows: Schabes' Earley-style algorithm takes $O(|A||I \cup A|Nn^6)$ worst case time and $O(|A \cup I|Nn^4)$ worst case space, where n is the length of the input, A is the set of auxiliary trees, I is the set of initial trees and N is maximum number of nodes in an elementary tree.

Given these worst case estimates we wish to explore what the observed times might be for a TAG parser. It is not our goal here to compare different TAG parsing algorithms, rather it is to discover what kinds of factors can contribute to parsing time complexity. Of course, a natural-language grammar that is large and complex enough to be used for parsing real-world text is typically neither unambiguous nor bounded in state size. It is important to note that in this paper we are not concerned with *parsing accuracy*, rather we want to explore *parsing efficiency*. This is why we do not pursue any pruning while parsing using statistical methods. Instead we produce a shared derivation forest for each sentence which stores, in compact form, all derivations for each sentence. This helps us evaluate our TAG parser for time and space efficiency. The experiments reported here are also useful for statistical parsing using TAG since discovering the source of grammar complexity in parsing can help in finding the right *figures-of-merit* for effective pruning in a sta-

*I would like to thank Aravind Joshi, Carlos Prolo and Fei Xia for their help and suggestions. This work was partially supported by NSF Grant SBR 8920230.

tistical parser.

2. Treebank Grammar

The grammar we used for our experiments was a Treebank Grammar which was extracted from Sections 02–21 of the Wall Street Journal Penn Treebank II corpus (Marcus *et al.*, 1993). We are grateful to Fei Xia for use of this grammar which was part of a separate study (Xia, 1999). The extraction converted the *derived* trees of the Treebank into *derivation* trees which represent the attachments of lexicalized elementary trees. There are 6789 tree templates in the grammar with 47,752 tree nodes. Each word in the corpus selects some set of tree templates. The total number of lexicalized trees is 123,039. The total number of word types in the lexicon is 44,215. The average number of trees per word is 2.78. However, the average gives a misleading overall picture of the syntactic lexical ambiguity in the grammar.[1] Figure 1 shows the syntactic lexical ambiguity of the 150 most frequent words in the corpus. We shall return to this issue of lexical ambiguity when we evaluate our TAG parser. Finally, some lexicalized trees from the grammar are shown in Figure 2.

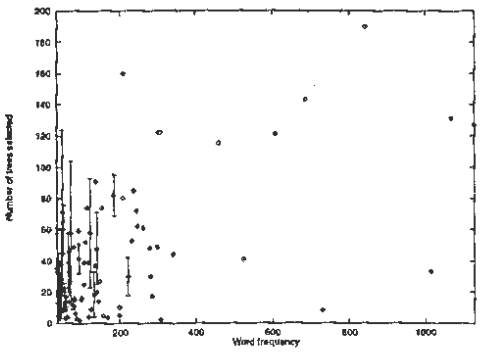


Figure 1: Number of trees selected by the 150 most frequent words in the input corpus. (x-axis: Word Frequency; y-axis: Number of Trees Selected)

3. The Parser

3.1. *Parsing Algorithm*

The parser used in this paper implements a chart-based head-corner algorithm. The use of head-driven prediction to enchance efficiency was first suggested by (Kay, 1989) for CF parsing (see (Sikkel, 1997) for a more detailed survey). (Lavelli & Satta, 1991) provided the first head-driven algorithm for LTAGs which was a chart-based algorithm but it lacked any top-down prediction. (van Noord, 1994) describes a Prolog implementation of a head-corner parser for LTAGs which includes top-down prediction. Significantly, (van Noord, 1994) uses a different closure relation from (Lavelli & Satta, 1991). The head-corner traversal for auxiliary trees starts from the footnode rather than from the anchor.

The parsing algorithm we use is a chart-based variant of the (van Noord, 1994) algorithm. We use the same head-corner closure as proposed there. We do not give a complete description of our parser here since the basic idea behind the algorithm can be grasped by reading (van Noord, 1994). Our parser differs from the algorithm in (van Noord, 1994) in some important respects: our implementation is chart-based and explicitly tracks *goal* and *item* states and does not perform any implicit backtracking or selective memoization, we do not need any additional variables to keep track of which words are already 'reserved' by an auxiliary tree (which (van Noord, 1994) needs to guarantee termination), and we have an explicit *completion* step.

3.2. *Parser Implementation*

The parser is implemented in ANSI C and runs on SunOS 5.x and Linux 2.x. Apart from the Treebank Grammar used in this paper, the parser has been tested with the XTAG English Grammar and also with a Korean grammar.

The implementation optimizes for space at the expense of speed, e.g. the recognition chart is

[1] We define the (syntactic) lexical ambiguity for a lexicalized TAG as the number of trees selected by a lexical item. Note that in a fully lexicalized formalism like LTAG, lexical ambiguity includes (to some extent) what would be considered to be a purely syntactic ambiguity in other formalisms.

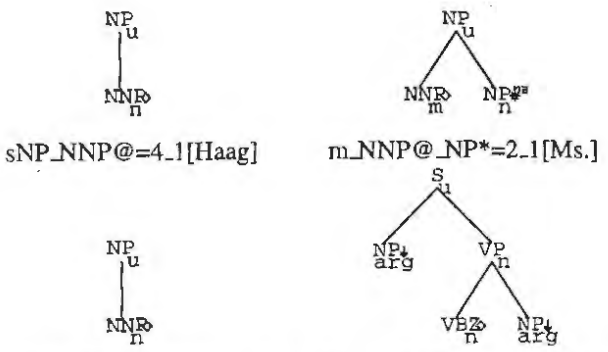


Figure 2: Example lexicalized elementary trees from the Treebank Grammar. They are shown in the usual notation: $\diamond$ = *anchor*, $\downarrow$= *substitution node*, $*$ = *footnode*, **na** = *null-adjunction constraint.* These trees can be combined using substitution and adjunction to parse the sentence *Ms. Haag plays Elianti.*

implemented as a sparse array thus taking considerably less than the worst case n^4 space and the lexical database is read dynamically from a disk-based hash table. For each input sentence, the parser produces as output a shared derivation forest which is a compact representation of all the derivation trees for that sentence. We use the definition of derivation forests for TAGs represented as CFGs, taking $O(n^4)$ space as defined in (Vijay-Shanker & Weir, 1993; Lang, 1994).

4. Input Data

The data used as input to the parser was a set of 2250 sentences from the WSJ Penn Treebank. The length of each sentence was 21 words or less. The average sentence length was 12.3 and the total number of tokens was 27,715. These sentences were taken from the same sections as the input Treebank Grammar. This was done to avoid any processing difficulties which are incurred for handling unknown words properly.

5. Results

In this section we examine the performance of the parser on the input data (described in §4).[2]

Figure 3 shows the time taken in seconds by the parser plotted against sentence length.[3] We see a great deal of variation in timing for the same sentence length, especially for longer sentences. This is surprising since all time complexity analyses reported for parsing algorithms assume that the only relevant factor is the length of the sentence. In this paper, we will explore whether sentence length is the only relevant factor.[4]

Figure 4 shows the median of time taken for each sentence length. This figure shows that for some sentences the time taken by the parser

[2]The data was split into 45 equal sized chunks and parsed in parallel on a Beowulf cluster of Pentium Pro 200Mhz servers with 512MB of memory running Linux 2.2.

[3]From the total input data of 2250 sentences, 315 sentences did not get a parse. This was because the parser was run with the start symbol set to the label S. Of the sentences that did not parse 276 sentences were rooted at other labels such as FRAG, NP, etc. The remaining 39 sentences were rejected because a tokenization bug did not remove a few punctuation symbols which do not select any trees in the grammar.

[4]A useful analogy to consider is the run-time analysis of *quicksort*. For this particular sorting algorithm, it was detemined the distribution of the order of the numbers in the input array to be sorted was an extremely important factor to guarantee sorting in time $\Theta(n \log n)$. An array of numbers that is already completely sorted has time complexity $\Theta(n^2)$.

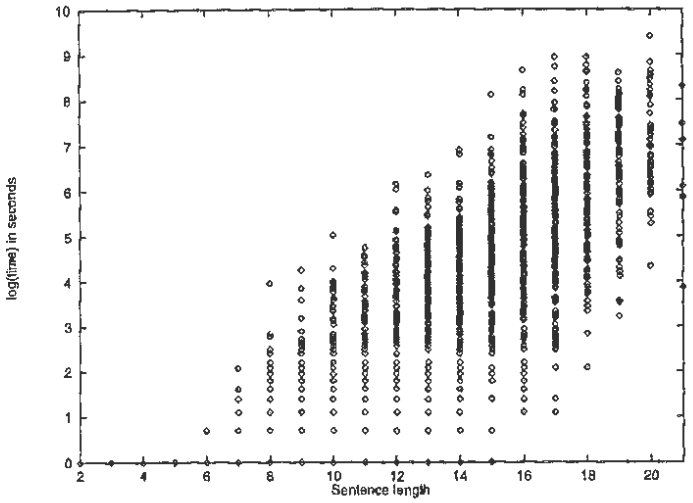


Figure 3: Parse times plotted against sentence length. (x-axis: Sentence length; y-axis: log(time) in seconds)

deviates by a large magnitude from the median case for the same sentence length. Next we considered each set of sentences of the same length to be a sample, and computed the standard deviation for each sample. This number ignores the outliers and gives us a better estimate of parser performance in the most common case. Figure 5 shows the plot of the standard deviation points against parsing time. The figure also shows that these points can be described by a linear function.

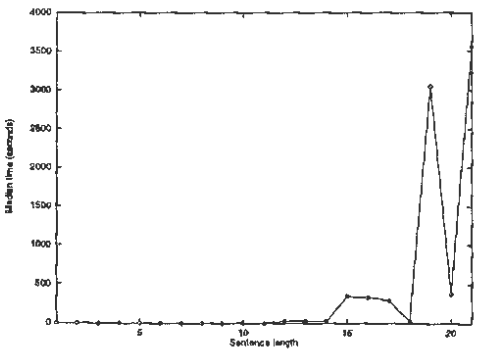


Figure 4: Median running times for the parser. (x-axis: Sentence length; y-axis: Median time in seconds)

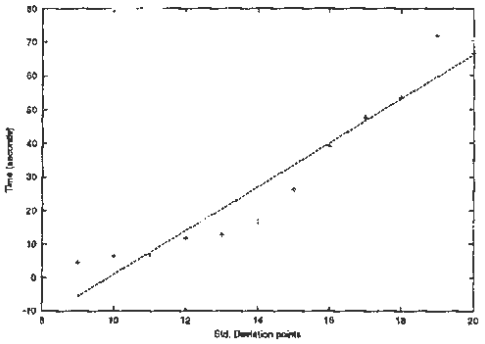


Figure 5: Least Squares fit over std. deviation points for each sentence length. Error was 9.078% and 13.74% for the slope and intercept respectively. We ignored sentences shorter than 8 words due to round-off errors; cf. Figure 3 (x-axis: Std. deviation points; y-axis: Time in seconds)

Figure 6 shows a plot of the number of derivations reported by the parser for each sentence plotted against sentence length. These derivations were never enumerated by the parser – the total number of derivations for each sentence was computed directly from the shared derivation forest reported by the parser. The

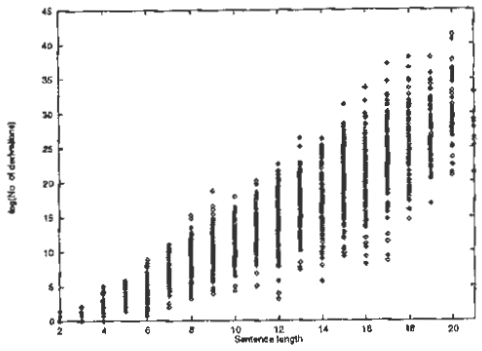


Figure 6: Log of number of derivations produced by the parser plotted against sentence length. (x-axis: Sentence length; y-axis: log(No. of derivations))

size of the grammar has a direct relation to the large number of derivations reported by the parser. However, in this figure just as in the figure for the parsing times while there is an overall increase in the number of derivations as the sentence length increases, there is also a large variation in this number for identical sentence lengths. We wanted to discover the relevant variable other than sentence length which would be the right predictor of parsing time complexity.[5] As our analysis of the lexicon showed us (see Figure 1), there can be a large variation in syntactic lexical ambiguity which might be a relevant factor in parsing time complexity. To draw this out, in Figure 7 we plotted the number of trees selected by a sentence against the time taken to parse that sentence. From this graph we see that the number of trees selected is a better predictor than sentence length of increase in parsing complexity. Based on the comparison of the graph in Figure 7 with Figure 3, we assert that it is the syntactic lexical ambiguity of the words in the sentence which is the major contributor to parsing time complexity. One might be tempted to suggest that instead of number of trees selected, the number of derivations reported by the parser might be a better predictor of parsing time complexity.[6] We tested this hypothesis by plotting the number of derivations reported for each sentence plotted against the time taken to produce them (shown in Figure 8). The figure shows that the final number of derivations reported is not a valid predictor of parsing time complexity.

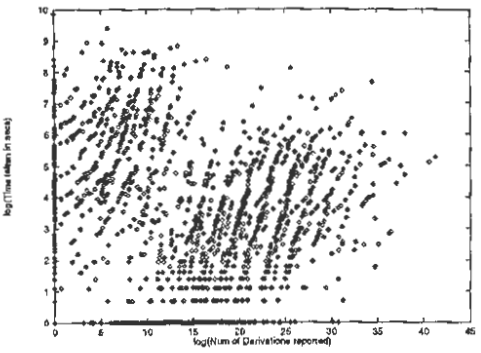


Figure 8: Number of derivations reported for each sentence plotted against the time taken to produce them (both axes are in log scale). (x-axis: log(Number of derivations reported); y-axis: log(Time taken) in seconds)

6. Conclusion

In this paper, we described an implementation of a chart-based head-corner parser for LTAGs. We ran some empirical tests by running the parser on 2250 sentences from the Wall Street Journal. We used a large Treebank Grammar to parse these sentences. We showed that the observed time complexity of the parser on these sentences does not increase predictably with longer sentence lengths. Looking at the derivations produced by the parser, we see a similar variation in the number of derivations for the same sentence length. We presented evidence that indicates that the number of trees selected by the words in the sentence (a measure of the syntactic lexical ambiguity of a sentence) is a better predictor of complexity in LTAG parsing.

[5] Note that this variable cannot be the number of active or passive edges proposed by the parser since these values can only be computed at run-time.

[6] One of the anonymous reviewers of this paper suggested that this might be a useful indicator.

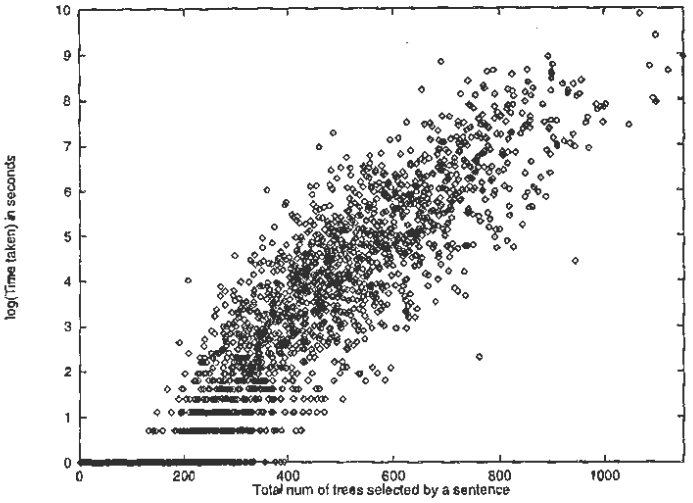


Figure 7: The impact of syntactic lexical ambiguity on parsing times. Log of the time taken to parse a sentence plotted against the total number of trees selected by the sentence. (x-axis: Total number of trees selected by a sentence; y-axis: log(Time taken) in seconds). Compare with Figure 3.

References

JOSHI A. K., LEVY L. & TAKAHASHI M. (1975). Tree Adjunct Grammars. *Journal of Computer and System Sciences.*

KAY M. (1989). Head driven parsing. In *Proc. of IWPT '89*, p. 52–62, Pittsburgh, PA.

LANG B. (1994). Recognition can be harder than parsing. *Computational Intelligence*, **10** (4).

LAVELLI A. & SATTA G. (1991). Bidirectional parsing of Lexicalized Tree Adjoining Grammars. In *Proc. 5th EACL*, Berlin, Germany.

MARCUS M., SANTORINI B. & MARCINKIEWIECZ M. (1993). Building a large annotated corpus of english. *Computational Linguistics*, **19** (2), 313–330.

SCHABES Y. (1994). Left to right parsing of lexicalized tree adjoining grammars. *Computational Intelligence*, **10** (4).

SIKKEL K. (1997). *Parsing Schemata.* EATCS Series. Springer-Verlag.

VAN NOORD G. (1994). Head-corner parsing for TAG. *Computational Intelligence*, **10** (4).

VIJAY-SHANKER K. & WEIR D. J. (1993). The use of shared forests in TAG parsing. In *Proc of 6th Meeting of the EACL*, p. 384–393, Utrecht, The Netherlands.

XIA F. (1999). Extracting tree adjoining grammars from bracketed corpora. In *Proceedings of 5th Natural Language Processing Pacific Rim Symposium (NLPRS-99)*, Beijing, China.

Lexicalized Grammar and the Description of Motion Events*

Matthew Stone*, Tonia Bleam†, Christine Doran‡ and Martha Palmer†

*Rutgers University
110 Frelinghuysen Road
Piscataway NJ 08854-8019
mdstone@cs.rutgers.edu

† University of Pennsylvania
3401 Walnut Suite 400A
Philadelphia PA 19104-6228
{tbleam,mpalmer}@linc.cis.upenn.edu

‡ The MITRE Corporation
202 Burlington Road
Bedford MA 01730-1420
cdoran@mitre.org

Abstract

In natural language generation, the use of a lexicalized grammar formalism and incremental syntactic and semantic processing places strong and specific constraints on the form and meaning of grammatical entries. These principles restrict which grammatical representations are possible and suggest examples an analyst can consult to decide among possibilities. We discuss and justify a number of such constraints, and describe how they inform the design of lexical entries for motion verbs. Our entries allow a generator to match the lexical choices found in a target corpus of action descriptions by assessing how the interpretation of a verb in context contributes towards the hearer's identification of the intended action.

1. Introduction

This paper originates in a project of tailoring a natural language generation system called SPUD, for sentence planning using description (Stone & Doran, 1997), to generate instructions for action in a concrete domain. The desired behavior for the system is specified by a corpus of edited, naturally-occurring action instructions whose form and content the system must mirror. The input to the system consists of three components: a representation of the context in which instruction is to be issued; a set of communicative goals describing the content that the instruction should make available to the audience; and a database of facts describing the GENERALIZED INDIVIDUALS such as paths, places and eventualities involved in the action (Bach, 1989; Hobbs, 1985). The task is further complicated because the content and organization of this input database must suit a variety of other tasks, such as animation (Badler *et al.*, 1998).

Such a generation task demands a detailed model of how the available input determines appropriate linguistic elements to arrange in output. The problem of LEXICAL CHOICE illustrates this. English offers a wide range of verbs to describe events in which an agent moves some object along a path; any motion instruction obliges the generator to choose just one. Uses of verbs differ syntactically in the kinds of optional elements that accompany them; they differ semantically both in the constraints they place on the motion event itself and in the links they establish between the event and the speaker and hearer's mutual knowledge of the environment. As we shall see, often many verbs, in many syntactic frames, can truly and appropriately describe

* The bulk of this work was performed while the authors were located at and supported through IRCS, Penn (NSF-STC SBR 8920230). Thanks to Aravind Joshi, Alistair Knott and Bonnie Webber.

each event. Nevertheless, we find a constrained and consistent pattern of lexical choice across naturally-occurring instructions. In order to mirror lexical choice in SPUD, we must provide a computational account of lexical items through which SPUD can exhibit the same consistency. SPUD is based on the widely-espoused view that sentence generation is goal-directed activity (Appelt, 1985; Dale, 1992; Moore, 1994; Moore & Paris, 1993); SPUD's repertoire of communicative action is determined by a declarative lexicalized grammar. To plan a sentence, SPUD searches among the derivations admitted by the grammar for a true sentence whose interpretation achieves the system's communicative goals in the current context. Clearly, then, to mirror a specified corpus of instructions, the grammar provided to SPUD must characterize the words and constructions used in the corpus accurately and comprehensively. It must describe forms syntactically, so that they are combined appropriately, but it must also describe them semantically and pragmatically, in order to support a useful assessment of interpretation.

In this paper we articulate a methodology for constructing lexicalized grammatical resources for generation systems such as SPUD, and show how this methodology allows us to ensure that SPUD deploys its lexical and syntactic options as observed in a corpus of desired output. Our methodology involves guidelines for the construction of syntactic structures, semantic representations and the interface between them, but the basic principle behind all of these guidelines is this: THE REPRESENTATION OF A GRAMMATICAL ENTRY MUST MAKE IT AS EASY AS POSSIBLE FOR THE GENERATOR TO EXPLOIT ITS CONTRIBUTION IN CARRYING OUT FURTHER PLANNING. This principle responds to two concerns. First, our research has revealed many characteristic uses of language in which a single entry helps achieve multiple communicative goals (Stone & Webber, 1998). This is an important way in which a generator needs to be able exploit the contribution of an entry it has already used, in line with our principle. Second, SPUD is currently constrained to greedy or incremental search for reasons of efficiency. At each step, SPUD picks the entry whose interpretation goes furthest towards achieving its communicative goals. As the generator uses its grammar to build on these greedy choices, our principle facilitates the generator in arriving at a satisfactory overall utterance.

2. Syntax

We collected occurrences of the verbs *slide*, *rotate*, *push*, *pull*, *lift*, *connect*, *disconnect*, *remove*, *position* and *place* in the maintenance manual for the fuel system of the American F16 aircraft; in this manual, each operation is described consistently and precisely. Syntactic analysis of instructions in the corpus and the application of standard tests allowed us to cluster the uses of these verbs into five syntactic classes; these classes are consistent with each verb's membership in a distinct Levin class (Levin, 1993). Differences among these classes include whether the verb lexicalizes a path of motion (*rotate*), an endpoint (*position*), or a change of state (*disconnect*); and whether a spatial complement is optional (as with the verbs just given) or obligatory (*place*). The data in (1) illustrate these alternatives.

(1) a Rotate valve one-fourth turn clockwise. [Path]
b Rotate halon tube to provide access. [No path]
c Position one fire extinguisher near aircraft servicing connection point. [Endpoint]
d Position drain tube. [No endpoint]
e Disconnect generator set cable from ground power receptacle. [Change of state]
f Disconnect coupling. [No source argument]
g Place grommet on test set vacuum adapter. [Endpoint, required]

We crafted syntactic entries for these verbs as trees in Lexicalized Tree-Adjoining Grammar, LTAG (Joshi *et al.*, 1975; Schabes, 1990). Our entries respect three requirements that reflect the analysis of the corpus and the generator's need to build on the syntax of entries it selects.

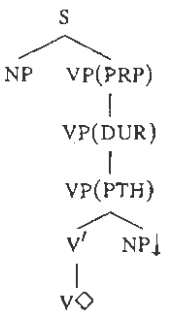
1. The grammar must associate each verb with its observed range of complements and modifiers, in the observed orders.

2. All optional elements, regardless of interpretation, must be represented in the syntax as modifiers, using the LTAG operation of adjunction. This allows the generator to select an optional element when it is needed to achieve communicative goals not otherwise satisfied. Recall that, in LTAG, a substitution site indicates a constituent that must be supplied syntactically to obtain a grammatical sentence; we call a constituent so provided a SYNTACTIC ARGUMENT. The alternative way of elaborating a sentence is to rewrite a node so as to include additional material (generally optional) specified by an auxiliary tree; we call material so provided a SYNTACTIC ADJUNCT. If optional elements are represented as syntactic adjuncts, it is straightforward to select one whenever its potential benefit is recognized. With other representations—for example, using alternative syntactic entries some of which include a syntactic argument position (substitution site) for the "optional" constituent—the representation can result in artificial dependencies or even dead-end paths in the search space in generation. To use this representation successfully, the generator would have to anticipate how the sentence would be fleshed out later in order to select the right entry early on.

3. The appropriate order of complements and modifiers for a verb must be represented using hierarchies of nodes in the verb's elementary tree. In a fixed word-order language like English, the nodes we add reflect different semantic classes which tend to be realized in a particular order; in a free word-order language, we might instead introduce ordering nodes based on information-structure status. Introducing such nodes decouples the generator's search space of derivations from the overt output word-order. It allows the generator to select complements and modifiers in any search order, while still realizing the complements and modifiers with their correct surface order. Again, alternative designs—representing word-order in the derivation itself or in features that clash when elements appear in the wrong order—introduce dependencies into the search space for generation that make it more difficult for the generator to build on its earlier choices successfully.

The latter requirements induce certain differences between our trees and other LTAG grammars for English, such as the XTAG grammar (Doran *et al.*, 1994), even in cases when the XTAG trees do describe our corpus. For example, we associate *slide* with the tree in (2); the structure reflects the optionality of the *path* constituent and makes explicit the observed characteristic order of constituents specifying path (PTH), duration (DUR) and purpose (PRP).

(2)



3. Syntax/semantics interface

SPUD adopts an ontologically promiscuous semantics (Hobbs, 1985): each entry used in the derivation of an utterance contributes a constraint to its overall semantics. The role of the syn-

tax/semantics interface is to determine when the constraints contributed by different grammatical entries describe the same generalized individuals. For example, take the phrase *slide the sleeve quickly*. The corresponding constraints describe an event e in which agent x *slides* object y along path p; describe an individual z that is a *sleeve*; and describe an event e' that is *quick*. The syntax/semantics interface provides the guarantee that $e = e'$ and $y = z$—i.e., that the sliding is what is quick and that the sleeve is what is slid. See (Hobbs, 1985; Hobbs *et al.*, 1993) for more details on ontologically promiscuous semantics.

Note that this strategy contrasts with other approaches to LTAG semantics, such as (Candito & Kahane, 1998), which describe meanings primarily in terms of function-argument relations. (It is also possible to combine both function-argument and constraint semantics, as in (Joshi & Vijay-Shanker, 1999; Kallmeyer & Joshi, 1999).) Like Hobbs, we use semantic representations as a springboard to explore the relationships between sentence meaning, background knowledge and inference—relationships which are easiest to state in terms of constraints. In addition, the use of constraints harmonizes with our perspective that a basic generation task is to construct extended descriptions of individuals (Stone & Webber, 1998; Webber *et al.*, 1999).

In general, to express the semantic links between multiple entries in a derivation, we associate each node in a syntactic tree with the individuals that the node describes. We refer to the collection of individuals that label the nodes in an entry as the SEMANTIC ARGUMENTS of the entry. When one tree combines with another by substitution or adjunction, a node in one tree is identified with a node in the other tree; at the same time the corresponding entities must be unified. Thus for example by labeling the foot VP node for *quickly* with e' and the corresponding VP node for *slide* with e, we can derive the identity $e = e'$ for *slide quickly*.

Our notion of semantic arguments is clearly distinguished from the notion of syntactic argument that we used in section 2 to characterize the syntactic structure of entries. Each syntactic argument position corresponds to one semantic argument (or more), since the syntactic argument position is a node in the tree which is associated with some individuals: semantic arguments. However, semantic arguments need not be associated with syntactic argument positions. For example, in a verb entry, we do not have a substitution site that realizes the eventuality that the verb describes. But we treat this eventuality as a semantic argument to implement a Davidsonian account of event modifiers, cf. (Davidson, 1980). Meanwhile, optional constituents that specify paths or places may be best modeled syntactically as modifiers, using the syntactic operation of adjunction. Optional constituents nevertheless can be taken to specify semantic arguments by associating their adjunction sites with references to the entities they specify (e.g., the paths or places). Because we count these implicit and unexpressed entities as semantic arguments, our notion is broader than that of (Candito & Kahane, 1998) and is more similar to Palmer's essential arguments (Palmer, 1990). It is a substantive question for grammar design WHICH entities SHOULD be acknowledged as semantic arguments for a given entry.

We make use of three tests to determine whether a particular syntactic modifier of a verb phrase describes the overall eventuality argument of the verb—this makes it an adjunct for the purposes of semantics as well—or whether it specifies some other semantic argument of the verb. The tests are: a DO SO test and an EXTRACTION test (explained here), and a PRESUPPOSITION test (explained in the following section). Together, these tests provide strong and specific guidance for designing the syntax/semantics interface in a generation grammar. (Of course, these tests are not perfect and may on occasion reveal difficult or ambiguous cases.)

1. The DO SO test succeeds when a modifier of a verb can be varied across ellipsis with *do so* naturally. The infinitivals in (3a), which provide different reasons for Kim and Sandy, pass the test; the locative PPs in (3b) fail the test, as they cannot be taken to describe Kim and Chris's separate destinations:

(3) a Kim left early to avoid the crowd. Sandy did so to find one.
b #Kim ran quickly to the counter. Chris did so to the kiosk.

A successful DO SO test suggests the modifier describes the event or action directly. A failed one suggests the modifier contributes a description of an entity that is independently related to the event or action—in other words, that the modifier specifies a semantic argument (e.g., the destination in (3b)). A theoretical explanation for the test can be given in terms of a semantic view of ellipsis such as (Hardt, 1999), where *do so* recovers an action discourse referent that has been introduced by an earlier predicate on events. When a modifier makes a predication on an event, there are two actions available for *do so*: the modified action and (as (3a) illustrates) the unmodified one. When a modifier instead makes a predication on a participant in the event, the only action referent for *do so* is that contributed by the main verb. In such cases, the DO SO test fails because we do not have a suitable action referent or a way of determining what role the new participant plays.

2. The EXTRACTION test applies to classes of syntactic modifiers of VP headed by a closed-class item. The test succeeds if it is grammatical to extract from inside the syntactic modifier (in a *wh*-question, for example), as in (4a); it fails otherwise, as in (4b).

(4) a What did you remove the rabbit from? (A: the hat)
b #What did you remove the rabbit at? (A: the magic show)

Passing the extraction test suggests that an optional constituent specifies a semantic argument. In LTAG, extraction describes a relation among trees in a tree family that have essentially the same meaning and differ only in syntax. On one formalization (Xia *et al.*, 1998), these relationships between trees are realized as descriptions of structure to add to elementary trees, or transformations. An "extraction transformation" that introduces the entity *l* in the syntax/semantics interface and relates *l* to the available entity *e* in the semantics cannot be represented this way. However, if some semantic argument *l* is referenced in the original tree, the extraction analogue to this tree can easily realize *l* differently. If we describe the source location as the semantic argument *l* in (4a) for example, the new realization involves an initial *wh*-NP substitution site describing the source *l*, and the corresponding stranded structure of the PP *from t*. (Note that failure of the extraction test would be inconclusive in cases where syntax independently ruled extraction out.)

4. Semantics

Semantic analysis of the instructions in the F16 corpus revealed that differences among verbs often involve links that the verbs impose between the action and what is known in the context about the environment in which the action is to be performed. The following illustration is representative. In the aircraft vent system, pipes are sealed together using a sleeve, which fits snugly over the ends of adjacent pipes; and a coupling, which snaps shut around the sleeve and holds it in place. At the start of maintenance, one *removes* the coupling and *slides* the sleeve away from the junction between the pipes. Afterwards, one *(re-)positions* the sleeve at the junction and *(re-)installs* the coupling around it. In the F16 corpus, these actions are always described using these verbs.

This use of verbs reflects the general design and function of the equipment as well as the motions themselves. For example, the motion involved in sliding the sleeve away is just the reverse of the motion involved in positioning the sleeve back. Since the verb *slide* indicates smooth motion ALONG A SURFACE (but not direction), *slide* seems to describe both actions equally well. The verb *position*, meanwhile, is used to describe a motion that leaves its object in some definite

location, where the object can perform some intended function. In the case of the sleeve, it would only be IN POSITION when straddling the pipes whose junction it seals.
We capture such distinctions in SPUD using a two-part lexical semantics.

1. The ASSERTION contributes new relationships among generalized individuals to the discourse. For example, the assertion of a motion verb might specify what manner of motion or what trajectory of motion is involved in an event.

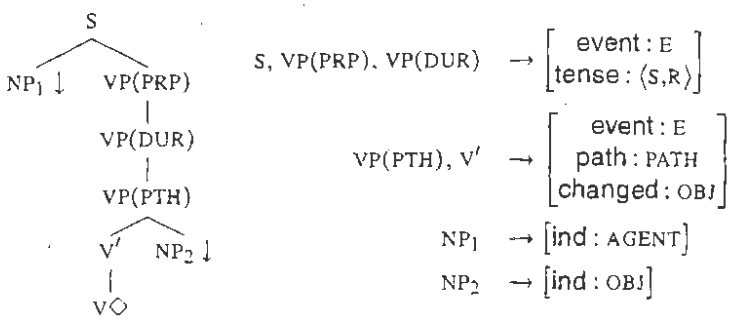
2. The ANAPHORIC PRESUPPOSITION is interpreted by a process of resolution linking it to salient facts and individuals from background knowledge and the conversational record. (Space precludes a description of the resolution process, but see (van der Sandt, 1992) for a theoretical account and (Stone, 2000) for the implementation.) Motion verbs generally carry such presuppositions: for instance, they presuppose a current location for the object (they assert this to be the beginning of the path traveled). But such presuppositions also distinguish lexical items. For example, *slide* presupposes a surface the object starts out in contact with; the object is asserted to remain in contact with this surface during the sliding. Meanwhile, *position* presupposes some "position" where the object carries out its intended function; the object is asserted to wind up at this position. Presuppositions can also evoke salient referents from the discourse history; for instance *reposition* presupposes a suitable prior motion event.

This formalism for presupposition is the basis for our third test for semantic arguments, the PRESUPPOSITION TEST. Any individual that is referenced in the presupposition of the verb must be treated as a semantic argument, even if a syntactic constituent that specifies that individual is optional. As suggested by (Saeboe, 1996), to apply the presupposition test in designing a lexical entry, we can compare the interpretation of a sentence with a modifier, such as *from the power adaptor* in (5a), to a corresponding sentence without the modifier, as in (5b):

(5) a (Find the power cable.) Disconnect it from the power adaptor.
b (The power cable is attached to the power adaptor.) Disconnect it.

If the entity specified by the modifier can be identified implicitly as discourse-bound—so that the sentence without the modifier can have the same interpretation as the sentence with the modifier, as in (5)—then the modifier must express a presupposed semantic argument. (Again, this is a partial diagnostic since semantic arguments need not always be presupposed.)
Let us pause to motivate our methodology of specifying lexical presuppositions as well as lexical assertions—and our tests for designing the syntax/semantics interface—in terms of our overriding goal: to allow the generator to build on its choices as easily as possible. The requirement to assert only what is true and to presuppose what is shared restricts which verbs are applicable in any context. At the same time, however, assertion and presupposition provide constraints on interpretation that can reduce ambiguity or trigger further inferences. They can thereby help the hearer identify the speaker-intended action. For example, the verb's presupposition may combine with other constraints contributed by the verb's complements to identify the participants in a described action (Stone & Webber, 1998). Of course, the generator can build on the presupposition of the verb this way only if it represents the interpretation of the presupposition and keeps track of the semantic arguments of the verb in order to model further elements as providing additional constraints on these arguments.
(6) fleshes out our earlier sample entry, for *slide*. The tree gives the syntax for one element in the tree family associated with *slide*; the feature structures associated with nodes show the syntax/semantics interface for this tree; the associated formulas describe the semantics of the entry in terms of presuppositions and assertions about the individuals referenced in the tree.

(6) a Syntax and syntax/semantics interface:



b Presupposition: *located-at-start*(R,OBJ,PATH), *along-surface*(PATH)

c Assertion: *caused-motion*(R,E, AGENT,OBJ,PATH)

5. Conclusion: grammar and lexical choice

The manual's consistent alternation between *slide* and *position* casts into relief the problem of lexical choice with which this paper opened. We close by suggesting how the methodology we have outlined here—formulating a grammar that matches a corpus and allows the generator to build on and exploit the entries it selects—leads to the construction of generation resources that can account for such alternation.

First, observe that the syntax and the syntax/semantics interface put *slide* and *position* on an equal footing. We can settle on a syntactic tree for each verb that best fits the context as in (Stone & Doran, 1997); we have designed these trees so that either choice can be fleshed out by further constituents into a satisfactory utterance.

To choose one verb in construction over the other, we must look at the INTERPRETATION of the two entries. A key part of this interpretation is the way the hearer resolves the presupposition. For example, the hearer resolves *position the sleeve* by finding in the common ground SOME sleeve and SOME position where it belongs. Part of SPUD's task is to ensure that the hearer will arrive at the SAME resolution that the generator intends; for *position the sleeve*, that of course means identifying the INTENDED sleeve and the INTENDED position for it. Depending on the context, it may be necessary to elaborate the description of an action, by adding additional words and additional presuppositions with them, to make the hearer's resolution of the presupposition unique. (Such an elaboration might yield *position the wing-vent sleeve*.)

This characterization of the speaker's communicative goals and the hearer's interpretation directly informs our lexical choice. Different presuppositions determine different possible resolutions, depending on the properties of salient objects in the common ground. The fewer resolutions that there are after selecting a verb, the more the verb assists the hearer in identifying the needed action. This gives a reason to prefer one verb over another. In our example, general background indicates that each sleeve only has a single place where it belongs, at the joint; meanwhile, there may be many "way points" along the pipe to slide the sleeve to. This makes the anaphoric interpretation of *position* less ambiguous than that of *slide*; to obtain an equally constrained interpretation with *slide*, an additional identifying modifier like *into its position* would be needed. This favors *position* over *slide*.

References

APPELT D. (1985). *Planning English Sentences*. Cambridge.

BACH E. (1989). *Informal Lectures on Formal Semantics*. SUNY.

BADLER N., BINDIGANAVALE R., BOURNE J., PALMER M., SHI J. & SCHULER W. (1998). A parameterized action representation for virtual human agents. In *Embodied Conversational Characters*.

CANDITO M. & KAHANE S. (1998). Can the TAG derivation tree represent a semantic graph? an answer in the light of Meaning-Text Theory. In *TAG+4*.

DALE R. (1992). *Generating Referring Expressions*. MIT.

DAVIDSON D. (1980). The logical form of action sentences. In *Essays on actions and events*, p. 105–148. Clarendon.

DORAN C., EGEDI D., HOCKEY B. A., SRINIVAS B. & ZAIDEL M. (1994). XTAG System—a wide coverage grammar for English. In *COLING*.

HARDT D. (1999). Dynamic Interpretation of Verb Phrase Ellipsis. *Linguistics and Philosophy*, **22**, 187–221.

HOBBS J., STICKEL M., APPELT D. & MARTIN P. (1993). Interpretation as abduction. *Artificial Intelligence*, **63**, 69–142.

HOBBS J. R. (1985). Ontological promiscuity. In *ACL*, p. 61–69.

JOSHI A. & VIJAY-SHANKER K. (1999). Compositional semantics with lexicalized tree-adjoining grammar (LTAG). In *International Workshop on Computational Semantics*, p. 131–145.

JOSHI A. K., LEVY L. & TAKAHASHI M. (1975). Tree adjunct grammars. *J. of the Computer and System Sciences*, **10**, 136–163.

KALLMEYER L. & JOSHI A. (1999). Factoring predicate argument and scope semantics: underspecified semantics with LTAG. In *12th Amsterdam Colloquium*.

LEVIN B. (1993). *English Verb Classes and Alternations*. University of Chicago.

MOORE J. (1994). *Participating in Explanatory Dialogues*. MIT.

MOORE J. D. & PARIS C. L. (1993). Planning text for advisory dialogues: capturing intentional and rhetorical information. *Computational Linguistics*, **19**, 651–695.

PALMER M. (1990). *Semantic Processing for Finite Domains*. Cambridge.

SAEBOE K. J. (1996). Anaphoric presuppositions and zero anaphora. *Linguistics and Philosophy*, **19**, 187–209.

SCHABES Y. (1990). *Mathematical and Computational Aspects of Lexicalized Grammars*. PhD thesis, University of Pennsylvania.

STONE M. (2000). Towards a computational account of knowledge, action and inference in instructions. *J. of Language and Computation*.

STONE M. & DORAN C. (1997). Sentence planning as description using tree-adjoining grammar. In *ACL*, p. 198–205.

STONE M. & WEBBER B. (1998). Textual economy through close coupling of syntax and semantics. In *INLG*, p. 178–187.

VAN DER SANDT R. (1992). Presupposition projection as anaphora resolution. *J. of Semantics*, **9**, 333–377.

WEBBER B., KNOTT A., STONE M. & JOSHI A. (1999). Discourse relations: A structural and presuppositional account using lexicalised TAG. In *ACL*, p. 41–48.

XIA F., PALMER M., VIJAY-SHANKER K. & ROSENZWEIG J. (1998). Consistent grammar development using partial tree-descriptions for lexicalized tree-adjoining grammars. In *TAG+4*.

Extending Linear Indexed Grammars

Christian Wartena

Universität Potsdam
Institut für Linguistik/Allgemeine Sprachwissenschaft
Postfach 601553, 14415 Potsdam, Germany
wartena@ling.uni-potsdam.de
WWW home page: http://www.ling.uni-potsdam.de/~wartena

Abstract

This paper presents a possibility to extend the formalism of linear indexed grammars. The extension is based on the use of tuples of pushdowns instead of one pushdown to store indices during a derivation. If a restriction on the accessibility of the pushdowns is used, it can be shown that the resulting formalisms give rise to a hierarchy of languages that is equivalent with a hierarchy defined by Weir. For this equivalence, that was already known for a slightly different formalism, this paper gives a new proof. Since all languages of Weir's hierarchy are known to be mildly context sensitive, the proposed extensions of LIGs become comparable with extensions of tree adjoining grammars and head grammars.

1. Introduction

It is well known that tree adjoining grammars (TAGs), head grammars (HGs) and linear indexed grammars (LIGs) are weakly equivalent (Vijay-Shanker & Weir, 1994). Each of these formalisms was developed independently for the description of natural languages. For TAGs and HGs hierarchies of extensions were defined by increasing the number of auxiliary trees that are inserted in one step and by increasing the size of the tuples that are handled, resp. (cf. (Weir, 1988)). The extensions of TAGs, multi–component TAGs (MCTAGs) (Joshi, 1987), were argued to be useful for the description of natural languages by Kroch (1987) and Kroch and Joshi (1987). For LIGs a linguistically motivated extension is defined by Rambow (1994) that is however of a rather different nature than the extensions of HGs and TAGs and does not give rise to a hierarchy of formalisms and language classes. Weir (1988; 1992) defines a hierarchy of linear controlled grammars that are strongly related to LIGs. It is however not immediately apparent what use these formalisms could have for linguistics. In (Wartena, 1998) recently extensions of LIGs, called *context–free linear multi–pushdown grammars (CFL–MPD–Gs)*, were defined that use tuples of pushdowns to store indices instead of a single pushdown. The use of tuples was motivated by linguistic needs. These extensions form a hierarchy of formalisms with an increasing number of pushdowns. If no pushdown is available the grammars are strongly equivalent to context–free grammars. If one pushdown is used we obtain LIGs. The nth element of the hierarchy can be shown to be a subclass of the nth class of Weir's hierarchy of controlled languages.

CFL–MPD–Gs seem to fill up an apparent gap in the square formed by TAGs, HGs and LIGs on the first axis and their extensions on the other axis. In order to formally justify this square we have to show that CFL–MPD–Gs and MCTAGs[1] or the extensions of head grammars are equivalent. (The equivalence between the last two was shown by Weir (1988)). We will go

[1] There are two variants of MCTAGs, the first of which allows only for simultaneous adjunction in one elemen-

the following way to show this equivalence. First we will prove the equivalence between the hierarchy of CFL–MPD–Gs and Weir's hierarchy of linear controlled grammars. Subsequently the equivalence between the latter hierarchy and MCTAGs has to be shown. In this paper we will do the first of the two steps.

2. Grammars with storage

LIGs store their indices in pushdowns. For the description of non–local dependencies in natural languages this organization can be argued to be too restrictive. Thus we might want to define formalisms similar to LIGs but with a more liberal stack structure. We start defining abstract storages, that will form the base of the subsequent extensions.

Definition 1 (storage) A *storage* is a 6–tuple $S = (C, C_0, C_F, P, F, m)$, where C is a set of configurations, $C_0 \subseteq C$ and $C_F \subseteq C$ the sets of initial and final configurations, respectively, P is a set of predicate symbols, F a set of instruction symbols. m is the meaning function, which associates every $p \in P$ with a mapping $m(p) : C \to \{\mathsf{true}, \mathsf{false}\}$ and every $f \in F$ with a partial function $m(f) : C \to C$.

Usually we are interested in properties of classes of storages rather than in properties of individual ones. Classes of storages are often called *storage types*.

Example 1 A *trivial storage* is defined as $S_{\mathrm{triv}} = (\{c\}, \{c\}, \{c\}, \emptyset, \{\mathsf{id}\}, m)$, where c is an arbitrary object and $m(\mathsf{id})(c) = c$. The class of all trivial storages is denoted $\mathfrak{S}_{\mathrm{triv}}$.

Example 2 A *pushdown* over some finite alphabet Γ can be defined as a storage[2] $S_{\mathrm{pd}}(\Gamma) = (\Gamma^*, \{\epsilon\}, \{\epsilon\}, P, F, m)$ with $P = \{\mathsf{top}(\gamma) | \gamma \in \Gamma\}$, $F = \{\mathsf{push}(\gamma) \mid \gamma \in \Gamma\} \cup \{\mathsf{pop}\} \cup \{\mathsf{id}\}$ and for every $a \in \Gamma$ and $\beta \in \Gamma^*$,

$$m(\mathsf{top}(\gamma))(a\beta) = (a = \gamma) \qquad m(\mathsf{pop})(a\beta) = \beta$$
$$m(\mathsf{push}(\gamma))(\beta) = \gamma\beta \qquad m(\mathsf{id})(\beta) = \beta$$

The class of all pushdowns is denoted $\mathfrak{S}_{\mathrm{pd}}$.

On the base of this notion of storages we can define *context–free linear–S grammars (CFL–S–Gs)* as a generalization of LIGs.

Definition 2 (CF linear S–grammar) If $S = (C, C_0, C_F, P, F, m)$ is a storage then a *context–free linear S–grammar* is a five tuple $G = (N, \Sigma, R, A_{\mathrm{in}}, c_0)$, where N, Σ denote the sets of nonterminal and terminal symbols, respectively. $A_{\mathrm{in}} \in N$ is a distinguished symbol, called the *start symbol*, $c_0 \in C_0$ is the *initial configuration*, and R is a set of *production rules* of the following two forms:

$$A \to \mathsf{if}\ \pi\ \mathsf{then}\ \zeta_1(B, f)\zeta_2$$
$$A \to \mathsf{if}\ \pi\ \mathsf{then}\ w$$

where $A, B \in N$, $\pi \in BE(P))$ and $\zeta_1, \zeta_2 \in (N \cup \Sigma)^*$, $f \in F$, $w \in \Sigma^*$. BE(P) denotes the set of boolean expressions over P.

tary tree, the second one of which allows for adjunction of a tuple in a tuple of elementary trees as well. The first variant is equivalent to (simple) TAGs, the second one gives rise to an hierarchy of languages. In this paper we will only consider these more powerful MCTAGs.

[2]Throughout the paper the following notational conventions are used. The empty string is denoted by ϵ. For each set V the notation V_ϵ is used as an abbreviation for $V \cup \{\epsilon\}$.

A string $\sigma \in O^*$ is said to derive a string $\tau \in O^*$, written $\sigma \underset{G}{\Rightarrow} \tau$, if either (1) or (2).

(1) $\sigma = \alpha(A, c)\beta$
$A \to \text{if } \pi \text{ then } \zeta_1 B f \zeta_2 \in R$
$m(\pi)(c) = \text{true}$[3]
$m(f)$ is defined on c
$\tau = \alpha\zeta_1'(B, c')\zeta_2'\beta$

(2) $\sigma = \alpha(A, c)\beta$
$A \to \text{if } \pi \text{ then } w \in R$
$m(\pi)(c) = \text{true}$
$\tau = \alpha w \beta$

where $A, B \in N$, $\alpha, \beta \in O^*$, $w \in \Sigma^*$, $c \in C$, $c' = m(f)(c)$ and ζ_1', ζ_2' are obtained from ζ_1 and ζ_2, respectively, by replacing every nonterminal D by (D, c_0). The reflexive and transitive closure of $\underset{G}{\Rightarrow}$, denoted by $\underset{G}{\Rightarrow}^*$, is defined as usual. The language generated by G is defined as $L(G) = \{w \in \Sigma^* | (A_{\text{in}}, c_0) \underset{G}{\Rightarrow}^* w\}$. If $\mathfrak{S}$ is a storage type and $S \in \mathfrak{S}$ then a CFL–S–G is called a CFL–$\mathfrak{S}$–G as well. The class of languages generated by CFL–$\mathfrak{S}$–Gs is denoted $\mathfrak{L}_{\text{CFL}(\mathfrak{S})}$.
The way in which the storage is passed on during a derivation is the same as in LIGs. It is easy to check that CFL–$\mathfrak{S}_{\text{pd}}$–Gs are equivalent to LIGs. Now we can easily define extensions of LIGs by choosing other storage types. The generative capacity of variants that are defined in this way crucially depends on the storage type. In order to investigate the typical complexity and generative capacity of a storage type we will use storage–automata.

Definition 3 (S–automaton) If $S = (C, C_0, C_F, P, F, m)$ is a storage, then an *S–automaton* M is a tuple $(Q, \Sigma, \delta, q_0, c_0, Q_F)$, where Q is a finite set of states, Σ is the input alphabet, $q_0 \in Q$ the initial state, $c_0 \in C_0$ the initial configuration, $Q_F \subseteq Q$ the set of final states, and δ, the transition relation, a finite subset of $Q \times \Sigma_\epsilon \times \text{BE}(P) \times Q \times F$.

The set $\text{ID}(M) = Q \times \Sigma^* \times C$ is called the set of instantaneous descriptions. For each (q_1, xw, c_1), $(q_2, w, c_2) \in \text{ID}(M)$ with $x \in \Sigma_\epsilon$ we write $(q_1, xw, c_1) \vdash_M (q_2, w, c_2)$ if there exists $(q_1, x, \pi, q_2, f) \in \delta$ such that $m(\pi)(c_1) = \text{true}$, $m(f)$ is defined on c_1 and $m(f)(c_1) = c_2$. The transitive and reflexive closure $\vdash_M^*$ of $\vdash_M$ is defined as usual. Sometimes conjunction of function symbols is used. For two function symbols f_1 and f_2 the meaning of the composed function symbol $f_1 \& f_2$ is defined as $m(f_1 \& f_2) = m(f_2) \circ m(f_1)$. The language accepted by M is defined $L(M) = \{w \,|\, (q_0, w, c_0) \vdash_M^* (q, \epsilon, c)$ for some $c \in C$ and $q \in Q_F\}$ if $Q_F \neq \emptyset$ and $L(M) = \{w \,|\, (q_0, w, c_0) \vdash_M (q_1, w', c_1) \vdash_M \ldots (q_n, \epsilon, c_n)$ for some $c_n \in C_F$, $q_n \in Q$, $c_i \in C - C_F$ and $q_i \in Q$ with $1 \leq i < n\}$ otherwise. In the first case we say that M accepts by final state. In the second case M accepts by final configuration. Let $\mathfrak{S}$ be some storage type. If M is an S–automaton and $S \in \mathfrak{S}$ we say as well that M is an $\mathfrak{S}$–automaton. Take e.g. $\mathfrak{S} = \mathfrak{S}_{\text{pd}}$, then we can say as usual that an automaton M is an $\mathfrak{S}_{\text{pd}}$–automaton or a pushdown–automaton without reference to the specific pushdown that is used. Finally we set $L_C(\mathfrak{S}) = \{L(M) \mid M$ is an $\mathfrak{S}$–automaton accepting by final configuration$\}$ and $L_Q(\mathfrak{S}) = \{L(M) \mid M$ is an $\mathfrak{S}$–automaton accepting by final state$\}$. For some important storage types (like pushdowns and concatenations of pushdowns) $L_Q(\mathfrak{S}) = L_C(\mathfrak{S})$. In these cases we drop the subscript. In (Wartena, 2000) these storage types are called *well–behaved*. The reader is referred there for details. A subclass of the well–behaved storage types is constituted by the *concatenating* storages types. In a concatenating storage $C_0 = C_F$ and the cardinality of C_0 is 1.

3. Concatenation of storages

It was argued in (Wartena, 1998) that a tuple of pushdowns would be an adequate storage type to describe non–local dependencies in a number of constructions in various languages. The

[3] In fact only $m(\pi)$ for $\pi \in P$ has been defined so far. It is straightforward to extend the domain of m to $\text{BE}(P)$.

motivation there was that we have to distinguish between different types of non–local dependencies, as proposed in theories like that of relativized minimality (Rizzi, 1990), and that there has to be one stack for each type.

Definition 4 (product of storages) For arbitrary storages $S^1 = (C^1, C_0^1, C_F^1, P^1, F^1, m^1)$ and $S^2 = (C^2, C_0^2, C_F^2, P^2, F^2, m^2)$ the *product* of S^1 and S^2 is defined as $S^1 \circ S^2 = (C^1 \times C^2, C_0^1 \times C_0^2, C_F^1 \times C_F^2, P, F, m)$ where $P = P^1 \cup \{\mathsf{test}(p) \mid p \in P^2\}$, $F = F^1 \cup \{\mathsf{do}(f) \mid f \in F^2\}$ and for every $c^1 \in C^1$ and $c^2 \in C^2$

$$\begin{aligned} m(p)((c^1,c^2)) &= m^1(p)(c^1) \text{ if } p \in P^1 \\ m(f)((c^1,c^2)) &= (m^1(f)(c^1), c^2) \text{ if } f \in F^1 \\ m(\mathsf{test}(p))((c^1,c^2)) &= m^2(p)(c^2) \text{ if } p \in P^2 \\ m(\mathsf{do}(f))((c^1,c^2)) &= (c^1, m^2(f)(c^2)) \text{ if } f \in F^2 \end{aligned}$$

The set of *semi–final* configurations of $S^1 \circ S^2$ is defined as $C_{SF} = (C_0^1 \cup C_F^1) \times C^2$. For two storage types $\mathfrak{S}^1$ and $\mathfrak{S}^2$ the product is defined straightforwardly as $\mathfrak{S}^1 \circ \mathfrak{S}^2 = \{S^1 \circ S^2 \mid S^1 \in \mathfrak{S}^1 \text{ and } S^2 \in \mathfrak{S}^2\}$.

Tuples or products of pushdowns are from a formal point of view to powerful. Following an idea of Breveglieri et al. (1996) we can reduce this power by restricting the operations that can be applied to the components. A similar idea to restrict the power of tuples of stacks was proposed by Becker (1994). Here we will define concatenation by means of an explicit restriction on a product of two storages. This general definition was suggested by Lothar Budach (p.c.).

Definition 5 (K–product of storages) For arbitrary storages S^1 and S^2 such that $S^1 \circ S^2 = (C, C_0, C_F, P, F, m)$ and for a mapping $K : F^2 \to \{\text{true}, \text{false}\}$ the *K–product* of S^1 and S^2 is defined as $S^1 \circ_K S^2 = (C, C_0, C_F, P, F, m')$ with[4]

$$\begin{aligned} m'(\varphi) &= m(\varphi)|_{C_{SF}} && \text{if } \varphi = \mathsf{do}(\varphi') \text{ and } K(\varphi') = \text{true} \\ m'(\varphi) &= m(\varphi) && \text{otherwise.} \end{aligned}$$

For two storage types $\mathfrak{S}^1$ and $\mathfrak{S}^2$ and any predicate K the K–product is defined as $\mathfrak{S}^1 \circ_K \mathfrak{S}^2 = \{S^1 \circ S^2 \mid_K S^1 \in \mathfrak{S}^1 \text{ and } S^2 \in \mathfrak{S}^2\}$.

Note that $m(\mathsf{do}(f))((c^1,c^2))$ is undefined for $f \in F^2$ if $K(f)$ is true and c^1 is not initial or final. The K–products for two predicates K are of special interest. The predicate r determines what operations are considered as reading operations. For any pushdown let $r(\mathsf{pop}) = \text{true}$ and let $r(\mathsf{push}) = r(\mathsf{id}) = \text{false}$. The r–product of two stores corresponds exactly with the *concatenation with regard to reading* defined in (Wartena, 1998). The counterpart of the predicate r is w which is defined by $w(\mathsf{push}) = \text{true}$ and $w(\mathsf{pop}) = w(\mathsf{id}) = \text{false}$ for any (n–turn) pushdown. The product $\circ_w$ is the same as *concatenation with regard to writing*.

Example 3 S_{pd} denotes a pushdown storage. Consequently, $(S_{\mathrm{pd}} \circ_r S_{\mathrm{pd}}) \circ_r S_{\mathrm{pd}}$ denotes the concatenation w.r.t. reading of three pushdowns. Each component behaves like a pushdown. At each point in the computation elements can be pushed on each of the three pushdowns, popping, however, is only possible from the first non empty one.

[4] For any (partial) function $f : A \to B$ and any $U \subseteq A$ the *restriction of f to U*, denoted $f|_U$, is defined as $f|_U(u) = f(u)$ if $u \in U$ and undefined otherwise.

Using r– and w–products we can recursively define the following hierarchies of storage types and corresponding classes of languages. The hierarchy established in (Breveglieri *et al.*, 1996) can now be defined as $\mathfrak{C}^{\mathfrak{R}}_i = L(\mathfrak{S}_i)$ for each $i \geq 0$ where $\mathfrak{S}_0 = \mathfrak{S}_{\text{triv}}$ and $\mathfrak{S}_i = \mathfrak{S}_{i-1} \circ_r \mathfrak{S}_{\text{pd}}$. ($\mathfrak{C}$ is intended as a mnemonic for concatenation, the superscript $\mathfrak{R}$ indicating that concatenation w.r.t. reading is meant.) For natural language syntax (Wartena, 1998) argues that concatenation w.r.t. writing is more appropriate. Thus the hierarchy, that is defined by setting $\mathfrak{C}^{\mathfrak{W}}_i = L(\mathfrak{S}_i)$ for each $i \geq 0$ where $\mathfrak{S}_0 = \mathfrak{S}_{\text{triv}}$ and $\mathfrak{S}_i = \mathfrak{S}_{i-1} \circ_w \mathfrak{S}_{\text{pd}}$, is of more interest for us. It can however be shown that $\mathfrak{C}^{\mathfrak{R}}_i \subseteq \mathfrak{C}^{\mathfrak{W}}_{i+1}$ and that $\mathfrak{C}^{\mathfrak{W}}_i \subseteq \mathfrak{C}^{\mathfrak{R}}_{i+1}$. Thus both hierarchies are very similar. An interesting fact is that $\mathfrak{C}^{\mathfrak{R}}_2$ and $\mathfrak{C}^{\mathfrak{W}}_2$ are the classes of extended left and extended right linear indexed languages (ELLILs and ERLILs), resp., defined in (Michaelis & Wartena, 1999). ERLILs were proposed as an appropriate restriction of linear indexed languages w.r.t. the paths along which a stack can be inherited.

4. Linear controlled grammars

The hierarchy of Weir can be expressed most easily in terms of linear control. Linear control of context–free grammars (CFGs) is defined in (Weir, 1992).

Definition 6 (linear controlled grammar) A *linear distinguished grammar (LDG)* is a quadruple $G = (N, \Sigma, R, A_{\text{in}})$, where N and Σ are finite sets of non–terminal and terminal symbols, respectively, $A_{\text{in}} \in N$ is the start symbol and where R is a finite set of production rules of the form: $A \to \beta_1 X ! \beta_2$ with $A \in N$, $X \in N \cup \Sigma$, called the *distinguished symbol*, $\beta_1, \beta_2 \in (N \cup \Sigma)^*$, and ! a special symbol not in $(N \cup \Sigma)$. A *linear controlled grammar (LCG)* is a pair $K = (G, H)$, where G is an LDG and H is a language over R, called the control language.

The set of (nonterminal and terminal) objects in K is defined as $O(K) = (N \cup \Sigma) \times R^*$. A string $\sigma \in O(K)^*$ is said to derive a string $\tau \in O(K)^*$, written $\sigma \underset{G}{\Rightarrow} \tau$, if

$$\begin{gathered} \sigma = \gamma(A, \omega)\delta \\ r = A \to \beta_1 X ! \beta_2 \in R \\ \tau = \gamma\beta_1'(X, \omega r)\beta_2'\delta \end{gathered}$$

where $A \in N$, $X \in N \cup \Sigma$, $\beta_1, \beta_2 \in (N \cup \Sigma)^*$, $\gamma, \delta \in O(K)^*$, $\omega \in R^*$, and β_1' and β_2' are obtained from β_1 and β_2, resp. by replacing every symbol $Y \in N \cup \Sigma$ by (Y, ϵ). In this case $(X, \omega r)$ is called the *distinguished child* of (A, ω). The reflexive and transitive closure of $\underset{G}{\Rightarrow}$, denoted $\underset{G}{\Rightarrow}^*$, is defined as usual. The language generated by K is defined as $L(K) = \{a_1 \dots a_n | (S, \epsilon) \underset{G}{\Rightarrow}^* (a_1, \omega_1) \dots (a_n, \omega_n)$ and $a_i \in \Sigma, \omega_i \in H$ for $1 \leq i \leq n\}$. The class of all LDGs is denoted by $\mathfrak{G}_{\text{LD}}$. Furthermore, for any class of grammars $\mathfrak{G}$ for which control is defined let $\mathfrak{G}/\mathfrak{L} = \{(G, H) \mid G \in \mathfrak{G}$ and $H \in \mathfrak{L}\}$ and for any class of grammars $\mathfrak{G}$ let $L(\mathfrak{G}) = \{L(G) \mid G \in \mathfrak{G}\}$. The obvious relation between linear controlled grammars and CFL–$\mathfrak{S}$–grammars was shown in (Wartena, 1998).

Proposition 7 $L(\mathfrak{G}_{\text{LD}}/L_Q(\mathfrak{S})) = \mathfrak{L}_{\text{CFL}(\mathfrak{S})}$ □

In order to refer to objects in a derivation it is sometimes assumed that the objects have addresses in $\mathbb{N}^*$.[5] In the following we will use two different address assignments, *leftmost* and *inside–out* address assignment. Suppose a string $\sigma = \alpha X \beta \in O(K)^*$ derives a string τ rewriting the object X with address ξ into new objects $Y_0 Y_1 \dots Y_i \dots Y_n$ with Y_i the distinguished child of X. If the address assignment is leftmost then the address of each Y_k is ξk with $0 \leq k \leq n$. In the case of inside–out assignment the address of Y_k is $\xi(i-k)$ for $0 \leq k \leq i$ and ξk for

[5] $\mathbb{N}$ denotes the set of all non–negative integers.

$i < k \leq n$. For each object in α and β the address in σ and τ is the same. A sequence of strings of objects $\delta = \sigma_0 \ldots \sigma_n$ such that $\sigma_i \Rightarrow \sigma_{i+1}$ is called a derivation If in each step the nonterminal object with the lexicographic[6] smallest address is rewritten then the derivation is called leftmost in case the address assignment is leftmost and inside–out in case the address assignment is inside–out.

5. The hierarchy of Weir

The classes of languages constituting Weir's hierarchy can be defined by setting $\mathfrak{W}_0 = \mathfrak{L}_{\text{reg}}$ and $\mathfrak{W}_i = L(\mathfrak{G}_{\text{LD}}/\mathfrak{W}_{i-1})$ for each $i > 0$. The following proposition was already shown in (Wartena, 1998).

Proposition 8 $\mathfrak{C}^{\mathfrak{R}}{}_i \subseteq \mathfrak{W}_i$ □

The languages of Weir's hierarchy of controlled grammars are accepted by concatenated pushdowns as well. Below we will show that the derivation of an LDG controlled by some $\mathfrak{S}$–automaton can be executed by an $(\mathfrak{S} \circ_r (\mathfrak{S}_{\text{pd}} \circ_r \mathfrak{S}_{\text{pd}}))$–automaton. The idea is that the automaton follows one spine using its finite control to store the element actually being expanded and using the first component to compute the control word. Everything that is generated to the right of the spine is written on the third pushdown, terminal and nonterminal symbols generated to the left of the spine are written on the second pushdown. If the foot of the spine is reached the second pushdown contains the left part of the derived sentential form in reversed order. The automaton now continues expanding the nonterminals on that pushdown, starting with the nonterminal directly to the left of the foot of the spine that just is reached. The automaton can read that symbol, since the first component is empty, just having accepted a control word. Thus the automaton simulates an inside–out derivation.

Lemma 9 *Let S be a concatenating storage. Then the following holds.*

$$L(\mathfrak{G}_{\text{LD}}/L(\mathfrak{S})) \subseteq L(\mathfrak{S} \circ_r (\mathfrak{S}_{pd} \circ_r \mathfrak{S}_{pd}))$$

Proof. Let $\mathfrak{S}$ be a concatenating storage type, let $S = (C, C_0, C_F, P, F, m) \in \mathfrak{S}$ and let $K = (G, L(M))$ be an LCG with $G = (N, \Sigma, R, A_{\text{in}})$ an LDG and $M = (Q, R, \delta, c_0, \emptyset)$ an S–automaton. Assume w.l.o.g. that each production of G is of the form $A \to B_1 B_2 ! B_3$ or $A \to a$ with $A, B_1, B_2, B_3 \in N$ and $a \in \Sigma_\epsilon$. Construct an $(S \circ_r (S_{pd}(\Gamma) \circ_r S_{pd}(\Gamma))$–automaton $M' = (Q \times N_\epsilon, \Sigma, \delta', (q_0, A_{\text{in}}), (c_0, \epsilon, \epsilon), \emptyset)$ with $\Gamma = N \cup \Sigma$, by setting

$$\begin{aligned}
\delta' &= \{((q_1, A), \epsilon, \pi, (q_2, B_2), f\&\mathsf{do}(\mathsf{push}(B_1)\&\mathsf{do}(\mathsf{do}(\mathsf{push}(B_2)))) \mid \\
&\qquad r = A \to B_1 B_2 ! B_3 \in R \text{ and } (q_1, r, \pi, q_2, f) \in \delta\} && (1a)\\
&\cup \{((q_1, A), \epsilon, \pi, (q_2, \epsilon), f\&\mathsf{do}(\mathsf{do}(\mathsf{push}(a))) \mid \\
&\qquad r = A \to a \in R \text{ and } (q_1, r, \pi, q_2, f) \in \delta\} && (1b)\\
&\cup \{((q_1, A), \epsilon, \pi, (q_2, \epsilon), f) \mid (q_1, \epsilon, \pi, q_2, f) \in \delta\} && (2)\\
&\cup \{((q, \epsilon), \epsilon, \mathsf{test}(\mathsf{top}(A)), (q, A), \mathsf{do}(\mathsf{pop})) \mid A \in N\} && (3a)\\
&\cup \{((q, \epsilon), \epsilon, \mathsf{test}(\mathsf{test}(\mathsf{top}(A))), (q_0, A), \mathsf{do}(\mathsf{do}(\mathsf{pop}))) \mid A \in N\} && (3b)\\
&\cup \{((q, \epsilon), a, \mathsf{test}(\mathsf{test}(\mathsf{top}(a))), (q, \epsilon), \mathsf{do}(\mathsf{do}(\mathsf{pop}))) \mid a \in \Sigma\} && (4)
\end{aligned}$$

It can be shown by induction on the number of steps in a computation and in a derivation, respectively, that

$$((q_1, A), w, (c_1, (\epsilon, \epsilon))) \vdash_{M'}^* ((q_2, B), \epsilon, (c_2, (\alpha^R, \beta)))$$
iff
$$(A, \epsilon) \overset{*}{\underset{G}{\Rightarrow}} w\alpha'(B, \omega)\beta' \text{ (inside–out) and } (q_1, \omega, c_1) \vdash_M^* (q_2, \epsilon, c_2)$$

[6] The lexicographic ordering relation $<_{\text{lex}}$ on $\mathbb{N}^*$ is defined by: $\chi <_{\text{lex}} \chi j \omega$ and $\chi i \psi <_{\text{lex}} \chi j \omega$ if $i < j$ for all $\chi, \psi, \omega \in \mathbb{N}^*$ and $i, j \in \mathbb{N}$.

where α', β' are obtained from α and β respectively by replacing every nonterminal $A \in N$ by (A, ϵ). In case $A = A_{\text{in}}$ and $\alpha = \beta = B = \epsilon$ we see that $L(M') = L(K)$. □

Proposition 10 *for each* $i > 1$: $\mathfrak{W}_i \subset \mathfrak{C}^{\mathfrak{R}}{}_{2i-1}$

Proof. First we show by induction that $\mathfrak{W}_i \subseteq \mathfrak{C}^{\mathfrak{R}}{}_{2i-1}$ for $i \geq 1$. For $i = 1$ the proposition is trivially true, since $\mathfrak{W}_1 = \mathfrak{C}^{\mathfrak{R}}{}_1 = \mathfrak{L}_{\text{CF}}$, the class of all context–free languages. Suppose that the assertion is true for some $i \in \mathbb{N}$. Let $\mathfrak{S}_i$ denote the concatenation w.r.t. reading of i–pushdowns, for some $i \in \mathbb{N}$. thus $L(\mathfrak{S}_i) = \mathfrak{C}^{\mathfrak{R}}{}_i$. For $i + 1$ we find

$$\begin{array}{rcll} \mathfrak{W}_{i+1} & = & L(\mathfrak{G}_{\text{LD}}/\mathfrak{W}_i) & \text{(by definition)} \\ & \subseteq & L(\mathfrak{G}_{\text{LD}}/\mathfrak{C}^{\mathfrak{R}}{}_{2i-1}) & \text{(by induction)} \\ & \subseteq & L(\mathfrak{S}_i \circ_r (\mathfrak{S}_{pd} \circ_r \mathfrak{S}_{pd})) & \text{(by Lemma 9)} \\ & = & \mathfrak{C}^{\mathfrak{R}}{}_{2i+1} = \mathfrak{C}^{\mathfrak{R}}{}_{2(i+1)-1} & \text{(by definition)} \end{array}$$

For $i > 1$ the inclusion is proper since it is known that both $\mathfrak{C}^{\mathfrak{R}}{}_i$ and $\mathfrak{W}_i$ contain the language $\{a_1^n \ldots a_{2i}^n \mid n \in \mathbb{N}\}$ but not the language $\{a_1^n \ldots a_{2i+1}^n \mid n \in \mathbb{N}\}$. □

This result combined with Proposition 8 implies that the languages from the multi–pushdown hierarchy are the same as those in Weir's hierarchy.

Corollary 11 $\bigcup_{i=0}^{\infty} \mathfrak{W}_i = \bigcup_{i=0}^{\infty} \mathfrak{C}^{\mathfrak{R}}{}_i$ □

A similar result was found by Cherubini and San Pietro (1999a; 1999b), using different proofs. Finally, let us return to the context–free linear $\mathfrak{S}$–grammars. The extensions of LIGs we are interested in are CFL–$\mathfrak{S}$–Gs with $\mathfrak{S}$ a concatenation of pushdowns. Calling each storage type formed by concatenation of pushdowns a *multiple pushdown (MPD)* we can refer to these grammars as CFL–MPD–grammars. It is straightforward to check that the languages generated by CFL–MPD–Gs are included in the hierarchy of Weir as well. Let $\mathfrak{S}_i$ be the storage that arises from concatenation w.r.t. reading from i–pushdowns, for some $i \in \mathbb{N}$. Then we find

$$\begin{array}{rcll} \mathfrak{L}_{CFL(\mathfrak{S}_i)} & = & L(\mathfrak{G}_{\text{LD}}/\mathfrak{C}^{\mathfrak{R}}{}_i) & \text{(by Proposition 7)} \\ & \subseteq & L(\mathfrak{G}_{\text{LD}}/\mathfrak{W}_i) & \text{(by Proposition 8)} \\ & \subseteq & \mathfrak{W}_{i+1} & \text{(by definition)} \end{array}$$

The inclusion of the classes Weir's hierarchy in the classes of languages generated by CFL–MPD–Gs is even simpler, since it can be shown that $\mathfrak{L}_{CFL(\mathfrak{S})} \supset L(\mathfrak{S})$.

6. Conclusion

In this paper two hierarchies of storage types were presented that are based on tuples of pushdowns with restrictions on the accessibility of the components. These storage types can be used to make new and linguistically interesting extensions of LIGs and besides for the construction of automata. It can be shown that automata based on a concatenation of two pushdowns accept only a subset of the linear indexed languages (LILs). Automata based on a triple of pushdowns accept already languages that cannot be generated by any LIG. A storage type corresponding to LIGs, the *nested pushdown*, was defined by Weir (1994). Though this storage type is rather different from ours, in a nested pushdown there are as well various possibilities for writing but only one for popping symbols. Becker (1994) defined automata, as well accepting LILs, that use two nested stacks with an explicit restriction on popping symbols from the second one, similar to the restrictions defined above. Reading from the second nested stack is possible if the

top of the first one is a bottom of stack symbol of an embedded stack. Thus reading from the second component is restricted but not only to situations in which the first component is empty. Storage types based on tuples with restricted possibilities for writing to our knowledge were not considered in any form up to now.

The main result of this paper is a new proof for the equivalence of the hierarchies of concatenated pushdowns and a hierarchy of controlled languages established by Weir (1988; 1992). By this equivalence we know that the languages generated by the extensions of LIGs presented here are mildly context sensitive and therefore are comparable with extensions of TAGs and head grammars. Whether the languages studied in this paper and the languages generated by MCTAGs coincide, is a question that remains for future research.

References

BECKER T. (1994). A new automaton model for TAGs. *Computational Intelligence*, **10** (4), p. 422–430.

BREVEGLIERI L., CHERUBINI A., CITRINI C. & CRESPI REGHIZZI S. (1996). Multi-push-down languages and grammars. *International Journal of Foundations of Computer Science*, **7** (3), p. 253–291.

CHERUBINI A. & SAN PIETRO P. (1999)a. On the relation between multi-depth grammars and tree-adjoining grammars. *Publ. Math. Debrecen*, **54** (Supplement), p. 625–640.

CHERUBINI A. & SAN PIETRO P. (1999)b. On the relations between multi-depth grammars and label-distinguished control grammars. In C. L. NEHANIV & M. ITO, Eds., *Algebraic Engineering*, Singapore: World Scientific.

JOSHI A. K. (1987). An introduction to tree adjoining grammars. In A. MANASTER-RAMER, Ed., *Mathematics of Language*, p. 87–113. Amsterdam/Philadelphia: John Benjamins.

KROCH A. S. (1987). Unbounded dependencies and subjacency in a tree adjoining grammar. In A. MANASTER-RAMER, Ed., *Mathematics of Language*, p. 143–172. Amsterdam/Philadelphia: John Benjamins.

KROCH A. S. & JOSHI A. K. (1987). Analyzing extraposition in a tree adjoining grammar. In G. J. HUCK & A. E. OJEDA, Eds., *Discontinuous Constituency*, volume 20 of *Syntax and Semantics*, p. 107–149. Academic Press.

MICHAELIS J. & WARTENA C. (1999). LIGs with reduced derivation sets. In G. BOUMA, G.-J. M. KRUIJFF, E. HINRICHS & R. T. OEHRLE, Eds., *Constraints and Resources in Natural Language Syntax and Semantics*, volume II of *Studies in Constrained Based Lexicalism*, p. 263–279. Stanford, CA: CSLI Publications.

RAMBOW O. (1994). *Formal and Computational Aspects of Natural Language Syntax*. PhD thesis, University of Pennsylvania.

RIZZI L. (1990). *Relativized Minimality*. Cambridge, MA: MIT Press.

VIJAY-SHANKER K. & WEIR D. J. (1994). The equivalence of four extensions of context-free grammars. *Mathematical Systems Theory*, **27** (6), p. 511–546.

WARTENA C. (1998). Grammars with composite storages. In *Proceedings of the Conference on Logical Aspects of Computational Linguistics (LACL '98)*, p. 11–14, Grenoble.

WARTENA C. (2000). Operations on storage types. Submitted.

WEIR D. J. (1988). *Characterizing Mildly Context-Sensitive Grammar Formalisms*. PhD thesis, University of Pennsylvania.

WEIR D. J. (1992). A geometric hierarchy beyond context-free languages. *Theoretical Computer Science*, **104** (4), p. 235–261.

WEIR D. J. (1994). Linear iterated pushdowns. *Computational Intelligence*, **10** (4), p. 422–430.

A Corpus-based Evaluation of Syntactic Locality in TAGs*

Fei Xia and Tonia Bleam

Institute for Research in Cognitive Science
University of Pennsylvania
Philadelphia, PA 19104,USA
{fxia/tbleam}@linc.cis.upenn.edu

Abstract

This paper presents a new methodology for examining cases of non-locality. The algorithm presented here allows us to extract from a large annotated corpus sentences that appear to require non-local MCTAG. We examine one such case, extraposition from NP, and argue that the dependency involved is not syntactic and therefore does not require non-local MCTAG.

1. Introduction

Mnch important work has been done to investigate the adequacy of local TAGs to account for various linguistic phenomena, see, e.g., (Heycock, 1987; Becker *et al.*, 1992; Abeillé, 1994; Bleam, 1994; Kulick, 1998; Joshi *et al.*, 2000). This paper presents a new methodology for doing this kind of research. The algorithm presented here allows us to extract from a large annotated corpus (the Penn Treebank) constructions that seem to require non-local[1] derivations. We propose that, in fact, these non-local dependencies should not be represented syntactically, and therefore do not constitute a problem for maintaining tree-local MCTAG.

*We would like to thank Aravind Joshi, Jeff Lidz, Anoop Sarkar and the XTAG research group for their help and suggestions. This work was supported by NSF Grant SBR 8920230.

[1]By *non-local*, we mean *non-tree-local*.

2. Extracting MC sets from the Treebank

Extracting multi-component (MC) tree sets from Treebanks is one of the tasks performed by a grammar development system named LexTract, whose structure is shown in Figure 1, with the components relevant to the MC extraction task marked in bold. There are three main steps in the MC extraction procedure: first, a bracketed structure in a Treebank (*ttree*) is decomposed into a set of elementary trees (*etrees*); second, a derivation tree is built to show how the *etrees* are combined; third, any pair of *etrees* that coutain co-indexed components are placed in a trees set with the *etrees* that connect them in the derivation tree. If the size of the set is more than three, the relation between the co-indexed components is not tree-local, assuming the correctness of Treebank annotations. For lack of space, we will use an example to demonstrate these main steps without going into the details of the algorithms (see (Xia, 1999) for details).

2.1. *The extracted grammar*

To ensure that the extracted *etrees* are compact and linguistically sound, we require that each *etree* in the grammar fall into one of three types determined by the relations between the anchor of the *etree* and other nodes in the tree, as shown in Figure 2:

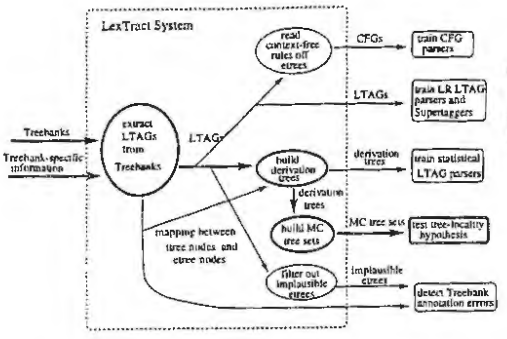


Figure 1: The structure of LexTract

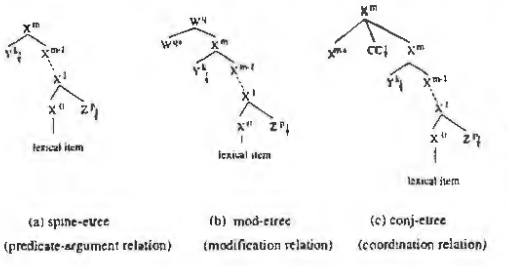


Figure 2: Forms of extracted *etrees*

2.2. *Extracting etrees from ttrees*

The first step of the MC extraction procedure is to extract *etrees* from *ttrees*. A *ttree* from Penn English Treebank is shown in Figure 3, where reference indices (e.g. -1 and -2) mark co-indexed constituents.

```
(S (NP-SBJ (NN supply) (NNS troubles))
   (VP (VBD were)
       (PP-LOC-PRD (IN on)
           (NP (NP (DT the) (NNS minds))
               (PP (IN of)
                   (NP (NP (NNP Treasury) (NNS investors))
                       (SBAR (-NONE- *ICH*-2) )))))
       (NP-TMP (RB yesterday))
       (, ,)
       (SBAR-2 (WHNP-1 (WP who) )
               (S (NP-SBJ (-NONE- *T*-1) )
                  (VP (VBD worried)
                      (PP-CLR (IN about)
                              (NP (DT the) (NN flood) ))))))
   (. .) ))
```

Figure 3: An example from the Treebank

The *ttrees* in the Treebank are partially bracketed in a way that does not explicitly distinguish arguments from adjuncts.

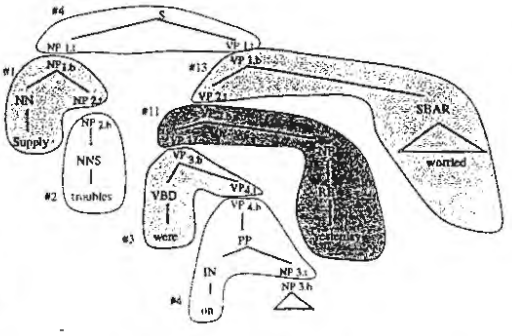


Figure 4: The *etree* set is a decomposition of the fully bracketed *ttree*.

In LTAGs, on the other hand, arguments and adjuncts are distinguished. To overcome this difference in notation, the algorithm first fully brackets *ttrees* by adding intermediate nodes so that *etrees* express one of three relations: a predicate-argument relation, a modification relation, or a coordination relation.

The next step is to extract the component *etrees* from a fully bracketed *ttree*. Recursive structures become mod-etrees or conj-etrees, and the remaining structures become spine-etrees. For instance, in the fully bracketed *ttree* in Figure 4,[2] along the path $S \rightarrow VP_1 \rightarrow VP_2 \rightarrow VP_3 \rightarrow VP_4 \rightarrow PP \rightarrow IN$, three adjuncts (the relative clause, the NP *yesterday* and the auxiliary verb *were*) are factored out and each forms a mod-*etree* (#13, #11 and #3 resp.), while the remaining structures become a spine-*etree* #4. The whole *ttree* yields the fifteen *etrees* shown in Figure 5.

[2]Some nodes in the *ttree* are numbered and split into the top and bottom pairs. Recall that when a pair of *etrees* are combined during parsing, the root of one *etree* is merged with a node in the other *etree*. Splitting nodes into top and bottom pairs during the decomposition of the fully bracketed *ttree* is the reverse process of merging nodes during parsing. For the sake of simplicity, we show the top and the bottom parts of a node only when the two parts will end up in different *etrees*.

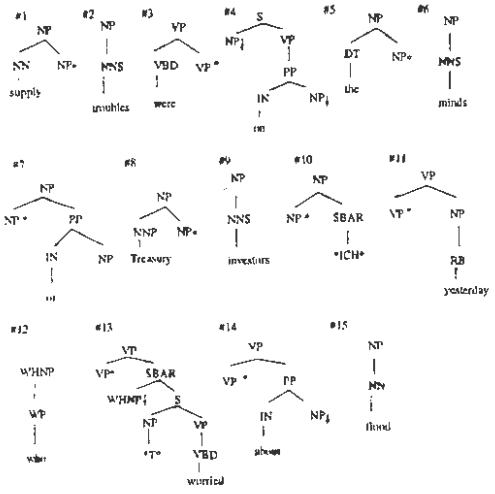


Figure 5: The extracted *etrees*

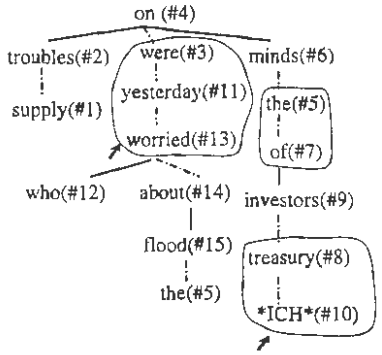


Figure 6: The derivation tree

2.3. *Building derivation trees*

Having extracted the *etrees* from a *ttree*, the next step for MC extraction is to build the derivation tree. Under the assumptions that no adjunctions are allowed at the foot nodes and at most one adjunction at any one node, and given the *etrees*, the mapping between the fully bracketed *ttree* and the derivation tree is one-to-one. The derivation tree for the *ttree* in Figure 4 is shown in Figure 6.

2.4. *Building MC tree sets*

We construct MC sets using the derivation trees and the reference indices in the *ttrees*. Given a pair of constituents that are co-indexed in a *ttree*, let e_g and e_f be the two *etrees* that the two constituents belong to. There exists a unique path that connects the two *etrees* in the derivation tree. The *etrees* on the path form a tree set.[3] If the size of the set is more than three, the relation between the co-indexed components is not tree-local, assuming the correctness of Treebank annotations. In our example, the relation between WHNP-1 and *T*-1 (both are in tree #13) is tree-local, whereas the relation between *ICH*-2 (in tree #10) and SBAR-2 (in tree #13) is not.

3. Experiments

We ran the algorithm on the Penn Treebank II (Marcus *et al.*, 1994). Table 1 gives the breakdown of MC sets by size. Out of 3151 MC sets, 999 sets (31.7%) had more than three *etrees* and were thus not tree-local. Table 2 shows the classifications of these non-local sets.

(1) That is [a skill]$_i$ Sony badly needs t_i and Warner is loath to lose t_i.

(2) It t_i would be my inclination [to advise clients not to sell]$_i$.

(3) Federal Express goes further t_i in this respect [than any company]$_i$.

(4) [Of all the ethnic tensions in America]$_i$, which t_i is the most troublesome right now ?

(5) [JMB officials are expected to be hired to represent the pension fund on the Santa Fe Pacific Realty board, Mr Roulac said t_i, to insulate the fund from potential liability problems.]$_i$

(6) The Diet doesn't normally even debate bills because the opposition parties are so often t_i opposed to whatever LDP does [that it would be a waste of time]$_i$.

[3] Notice if a list *etrees* E_i all modify the same *etree* E, E_i will form a chain in the derivation tree, as circled in Figure 6. Those intermediate mod-*etrees* are not included in the MC tree set.

size of MC sets	≤ 3 (tree-local sets)	4	5	6	7	8	subtotal	total
# of MC sets (type)	2152(68.3%)	874	94	26	4	1	999(31.7%)	3151
# of MC sets (token)	19994(91.3%)	1772	102	26	4	1	1905(8.7%)	21899

Table 1: Numbers of extended MC sets and their frequencies in PTB

PTB errors	LexTract errors	NP-EXP	extraction from coord.	*it*-EXP	comparative construction	*of*-PP	paren-thetical	*so .. that*	others
71	65	337	209	176	50	31	30	11	19

Table 2: Classification of 999 extended MC sets that look non-local

In each of these "non-local" cases, the Treebank notation establishes a dependence between two elements, as shown in (1) – (6). We suggest that, in fact, in all of the cases, the dependence is not syntactic, and so these examples do not constitute cases where non-local MCTAG would be required. Due to space considerations, however, we cannot address each case independently. Instead, we focus on one construction, that of Extraposition (EXP) from NP, both because this was the most common type of "non-local" example found by the algorithm and because it is potentially the strongest case against tree-locality. We will show that even for this difficult case, tree-locality can be maintained.

4. Extraposition

One example of EXP was discussed in Section 2 (cf. Figure 3-6). Further examples are illustrated in (7) and (8), where the bracketed prepositional phrase is construed as an argument (7) or a modifier (8) of the NP in bold.[4]

(7) Younkers rang up **sales** in 1988 [of $313 million].

(8) The company gave us **discounts** all last year [on their premium brands].

Most generative analyses of this phenomenon associate the extraposed phrase (EXP phrase) with a gap in the NP with which it is interpreted. See, e.g., (Guéron, 1980; Baltin, 1981; Pollard & Sag, 1994). These accounts can be referred to as "syntactic-dependence" analyses, since they require that the extraposed phrase and its "antecedent" noun be coindexed or associated *in the syntax*. This coindexation is shown in Figure 7. Other authors, on the other hand, argue for a semantic dependence, or non-gap analysis (Andrews, 1975; Culicover & Rochemont, 1990).

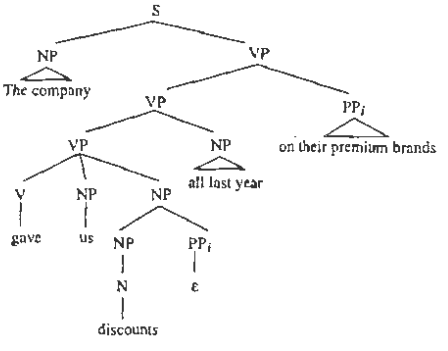


Figure 7: Gap analysis of Extraposition

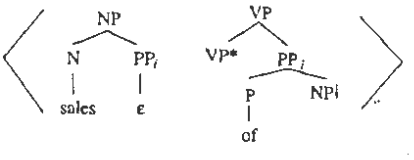


Figure 8: Gap analysis of argument EXP

Within TAG, the syntactic-dependence analysis can be modeled using MC tree sets (Kroch & Joshi, 1987), as in Figures 8 and

[4] Adjunct status was determined using two tests: *one*-substitution and *wh*-extraction.

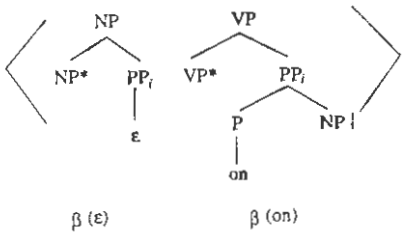


Figure 9: Gap analysis of adjunct EXP

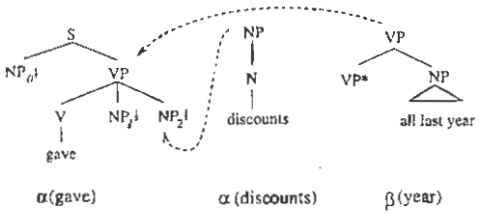


Figure 10: Elementary trees for (8)

9.[5] This approach works for argument EXP, but it faces two problems when applied to adjunct EXP. The first problem is that, given current assumptions, the derivation of even the simplest cases requires non-local MCTAG (Weir, 1988). The trees required to derive (8) are given in Figures 9 and 10. In this derivation $\beta(\varepsilon)$ adjoins to the NP of α(discounts), and β(on) adjoins to the VP node[6] in α(give).

A second problem with the gap analysis is pointed out by (Abeillé, 1994) citing (Gunnarson, 1982). Extraposed adjunct phrases (9) allow pronominalization of the head noun, something that is not allowed if the adjunct phrase is not extraposed (10). This is clear evidence that there is no movement since the putative underlying representation is impossible.

(9) John makes **lists** every day [with names of people who owe us money], and I make **them** every day [with names of people who we owe money to].

(10) * I make them with names of people every day.

(Abeillé, 1994) thus proposes that the relationship between adjunct extraposition and the head noun should be a semantic one rather than a syntactic one. These "base generated" cases are handled using synchronous TAG (S-TAG), where the syntax and semantics are represented by parallel TAG derivations. Representing the semantics with a TAG allows Abeillé to preserve the locality effects that we find in argument EXP, which do require a syntactic dependence. We refer to this locality property as *etree boundedness* (ETB). As Abeillé notes, her analysis predicts that EXP is *NP-bounded*; that is, the extraposed element "has to be a complement of the top N, and cannot be a dependent of an embedded N". While ETB holds of argument EXP, we have found that adjunct EXP does not obey this condition, and hence cannot be accounted for in the S-TAG analysis. (11) and (12) are examples from the Treebank of non-NP-bounded EXP. In (11), the extraposed relative clause *who worried...* is not associated with an argument of the *etree* to which it attaches, but rather to a more deeply embedded NP, thus violating ETB.

(11) Supply troubles were on the minds of **Treasury investors** yesterday [who worried about the flood of new government securities].

(12) Major rivals have been following a policy of continuous and **deep discounting** for at least the past 18 months [on their premium brands].

These examples show that the S-TAG analysis of the semantic dependence is too re-

[5] Notice that positing a dependence in the syntax would not necessarily require an explicit gap in the case of extraposition of an argument PP. When the extraposed phrase is an adjunct, however, syntactic dependence must be represented by adjoining a trace onto the head noun phrase (or alternatively coindexing with features).

[6] Alternatively, the extraposed element could adjoin to the S node. See (Kroch & Joshi, 1987; Culicover & Rochemont, 1990) for discussion.

strictive for adjunct EXP. Instead, we propose that the semantic dependency must be calculated post-derivationally, as, for example, in (Joshi & Vijay-Shanker, 1999), where the semantic representation is read off the derivation tree. The process of calculating this dependency must make reference to structure, but it does not adhere to the strict locality that the S-TAG analysis requires.

5. Conclusions

We have presented an algorithm to extract from the Penn Treebank constructions that seem to require non-local MCTAG. We propose that all these non-local dependencies should not be represented syntactically, and therefore do not require non-local MCTAG. One such example is NP-EXP, which has been previously argued to be a locally-bounded dependency. Our algorithm has revealed that adjunct EXP does not obey the locality constraints previously posited by linguists. If these examples are to be derived syntactically, they would require an LTAG more powerful than Tree-local MCTAG. We show, however, that the dependency between the head noun and the EXP phrase is not a syntactic one, but a semantic one. We conclude that extraposition does not constitute a case for using non-local MCTAG; tree-locality can be maintained.

References

ABEILLÉ A. (1994). Syntax or Semantics? Handling Nonlocal Dependencies with MCTAGs or Synchronous TAGs. *Computational Intelligence*, **10**, 471–485.

ANDREWS A. (1975). *Studies in the Syntax of Relative and Comparative Clauses.* PhD thesis, MIT.

BALTIN M. (1981). Strict Bounding. In C. BAKER & J. MCCARTHY, Eds., *The Logical Problem of Language Acquisition.* MIT Press.

BECKER T., RAMBOW O. & NIV M. (1992). *The Derivational Generative Power, or, Scrambling is Beyond LCFRS.* Technical Report IRCS-92-38, University of Pennsylvania.

BLEAM T. (1994). Clitic Climbing in TAG: A GB Perspective. In *Proc. of TAG+3.*

CULICOVER P. & ROCHEMONT M. (1990). Extraposition and the Complement Principle. *Linguistic Inquiry*, **21**, 23–47.

GUÉRON J. (1980). On the Syntax and Semantics of PP Extraposition. *Linguistic Inquiry*, **11**, 637–678.

GUNNARSON A. K. (1982). Trois constructions à dépendance entre sujet et pp. *Lingvisticae Investigationes.*

HEYCOCK C. (1987). The Structure of the Japanese Causative. University of Pennsylvania.

JOSHI A., BECKER T. & RAMBOW O. (2000). The complexity of scrambling and the competence performance distinction. In A. ABEILLÉ & O. RAMBOW, Eds., *Tree Adjoining Grammar: Formalism, Computation, Applications.* CSLI Publications.

JOSHI A. & VIJAY-SHANKER K. (1999). Compositional Semantics with LTAG: How Much Underspecification is Necessary? In *Proc of 3nd International Workshop on Computational Semantics.*

KROCH A. S. & JOSHI A. K. (1987). Analyzing Extraposition in a Tree Adjoining Grammar. In G. HUCK & A. OJEDA, Eds., *Discontinuous Constituents, Syntax and Semantics*, volume 20. Academic Press.

KULICK S. (1998). TAG and Clitic Climbing in Romance. In *Proc. of TAG+4.*

MARCUS M., KIM G., MARCINKIEWICZ M. A. *et al.* (1994). The Penn Treebank: annotating predicate argument structure. In *Proc of ARPA speech and Natural language workshop.*

POLLARD C. & SAG I. A. (1994). *Head-Driven Phrase Structure Grammar.* The University of Chicago Press.

WEIR D. (1988). *Characterizing mildly context-sensitive grammar formalisms.* PhD thesis, University of Pennsylvania.

XIA F. (1999). Extracting tree adjoining grammars from bracketed corpora. In *Proc. of NLPRS-99*, Beijing, China.

Workshop TAG+5, Paris, 25-27 May 2000

Customizing the XTAG System for Efficient Grammar Development for Korean*

Juntae Yoon†, Chung-hye Han†, Nari Kim‡ and Meesook Kim†

†IRCS, University of Pennsylvania
3401 Walnut St., Suite 400
Philadelphia, PA 19104
USA

‡Konan Technology, Inc.
Simone Building, 144-1
Samsung-Dong, Kangnam-Gu
Seoul 135-090, Korea

{jtyoon,chunghye,nari,meesook}@linc.cis.upenn.edu

Abstract

This paper addresses linguistic and implementation problems for a practical LTAG parser raised by rich morphology in Korean. We propose a way of representing the Korean inflectional system as feature structures in lexicalized elementary trees, and describe our implemented modifications on the XTAG system for a more efficient grammar development for Korean.

1. Issues

Korean is an agglutinative language with a very productive inflectional system. Inflections include postpositions on nouns; tense morphemes and endings that indicate sentence types on verbs and adjectives; among others. Furthermore, these inflections can combine with each other to form compound inflections.

(1) Noun

a. *hakkyo-ka*
school-Nom

b. *hakkyo-eyse-ka*
school-from-Nom

c. *hakkyo-eyse-man*
school-from-only

d. *hakkyo-eyse-man-un*
school-from-only-Topic

(2) Verb

a. *ka-ss-ta*
go-Past-Decl

b. *ka-si-ess–ta*
go-Honor-Past-Decl

c. *ka-ki-ka*
go-Nominalizer-Nom

d. *ka-si-ess-ki-ey-nun*
go-Honor-Past-Nominalizer-to-Topic

This implies that a word in Korean can have a very large number of morphological variants. For example, verbs can be followed by honorific and tense morphemes which can then be followed by endings indicating clause-type which then can be followed by case postpositions. Similarly, adverbial postpositions which correspond to English prepositions, can be followed by other case postpositions such as nominative or accusative case markers and auxiliary postpositions such as *to* ('also') and *man* ('only'), which then can be followed by a topic marker. Accordingly,

*We thank Martha Palmer, Aravind Joshi, Anoop Sarkar and the XTAG Group at Penn for their support and discussion. We also acknowledge the two anonymous reviewers. This project has been partially funded by the Army Research Lab via a subcontract from CoGenTex, Inc. and NSF Grant SBR 8920230. The third author's contribution to this work was made while she was a visiting researcher at IRCS.

the number of possible morphological variants of a word can in principle be in the tens of thousands.

This property of Korean raises two issues within the context of developing and implementing a Feature Based Lexicalized Tree Adjoining Grammar (FB-LTAG) for Korean using the XTAG system (The XTAG-Group, 1998): (1) adequate linguistic description of the inflections and (2) efficient lexicon development. From a linguistic point of view, describing a grammar of a language is to construct rules that generate sentences in the language at a formal level. From an implementational point of view, the grammar should be described in a consistent and efficient way. The XTAG system helps us to pursue both these goals, but the complicated inflection system mentioned above leads to difficulties in building a grammar for Korean.

In this paper, we provide our solution to the linguistic and implementational issues raised by these morphological properties of Korean. We first provide a way of handling the Korean inflectional system using feature structures in lexicalized elementary trees in section 2. We impose a hierarchy on various types of inflections in order to handle all possible ways of combining inflections, and we represent this by assigning different feature attributes to different types of inflections.

In section 3, we then point out that the current XTAG system as it is forces us to construct a lexicon (i.e., syntactic database) that lists all possible morphological variants of words. A lexicon must contain all possible *eojeols*, where an *eojeol* is a term in Korean for denoting a spacing unit which consists of a content word and associated functional words. However, this is highly impractical and inefficient given the rich inflectional system in Korean. We would end up with a very large (even unbounded) lexicon. Therefore, we found it necessary to develop an alternative method for constructing the lexicon in order to continue to use the XTAG parser for developing a Korean grammar . One possible solution to the problem is to incorporate morphological rules in the grammar that regulate the generation of *eojeols* with several morphemes combined. However, doing so will mix up morphological generative rules with syntactic rules, complicating the TAG grammar tremendously. Instead, we have chosen to pursue an approach in which morphological regularities are handled by a separate morphological component using a morphological analyzer (Yoon *et al.*, 1999). The output of this analysis then interacts with our Korean TAG grammar which handles syntactic regularities. As a way of implementing this approach, we modified the XTAG system by dividing up the syntactic database into elementary syntactic database (ESDB) and local syntactic database (LSDB). ESDB is a general lexicon that contains stems with the elementary trees associated with them. LSDB is a partial lexicon dynamically generated for each input sentence using information from ESDB and the output of a morphological analyzer. That is, it contains only entries for *eojeols* occurring in the input sentence. The morphological analyzer produces the morphological analysis of each *eojeol* in the input sentence identifying its stem and inflections. Then, the stem of each *eojeol* is associated with elementary trees or tree families by looking up the ESDB and stored in the LSDB. The inflections of each *eojeol* are converted into features and are also stored in the LSDB. This modification to the XTAG system allows us to build a lexicon efficiently and develop a grammar for Korean that is compatible with the XTAG system.

2. Handling inflectional morphology

In our current Korean grammar, the inflectional morphology on an *eojeol* that are relevant for syntactic analysis is represented as features on the tree node. For instance, a noun with a nominative case marker is associated with the feature <case:nom> and when this lexical item is anchored by an NP tree, the feature <case:nom> is passed up to the NP node.

In Korean, combining inflections is a highly productive process with some restrictions. For

example, nominative, accusative and genitive CASE postpositions occur in a complementary distribution, but ADVERBIAL postpositions (which correspond to English prepositions) such as *-ey* ('at'), *eykey* ('from'), *-kkaci* ('to'), etc. can be followed by nominative case or genitive case. Case and adverbial postpositions are assumed to be assigned by the predicate of the sentence. Moreover, AUXILIARY postpositions which have semantic content such as *-man* ('only') and *-to* ('even') can combine with an adverbial postposition and the topic marker *-(n)un* can combine with an adverbial postposition and/or an auxiliary postposition but not the case postposition.
Moreover, predicates[1] in Korean are inflected with several morphemes. They carry CLAUSE-TYPE morphemes that indicate whether the clause is a main, coordinate, subordinate, relative clause, or nominalized clause. If a clause is a main clause, the verb carries a MODE morpheme that indicates whether the clause is a declarative, imperative, interrogative, exclamation, or propositive, etc. Clause-type morphemes and mode morphemes occur at the end of the verb. In addition, verbs also carry TENSE inflections right before the clause-type and mode morphemes. Further, all these inflections can be expressed in many different ways.
In order to handle all possible ways of combining inflections, we imposed a hierarchy among various types of inflections and represented this by assigning different types of inflections to different feature attributes. Table 1 summarizes the list of inflectional feature attributes and the corresponding feature values currently being used by our grammar. The label 'pp' on <adv-pp> and <aux-pp> stand for postpositions. Note that verbal features include <ending> which allows us to store the string values of mode and clause-type morphemes in the tree node for later semantic interpretation. Examples of an NP tree that anchors a noun (*hakkyo* 'school') with compound inflections, and an S tree that anchors a verb (*ka* 'go') with some verbal inflections are given in Figure 1.

On nouns		
⟨**case**⟩	a case feature assigned by predicate	nom, acc, gen, adv
<**adv-pp**>	a feature assigned by predicate only if <case:adv>, which corresponds to English prepositions such as *to, from, in*	string values such as *ey, eyse, lo, wa, ya, kkaci, pwute, pota, lako, losse, ...*
<**topic**>	presence/absence of topic marker	+, -
<**aux-pp**>	adds specific meaning e.g., *only, also*	string values e.g., *to, man*
On predicates		
<**clause-type**>	a feature that indicates the type of the clause that contains the predicate	main, coord, subord, adnom, nominal, aux-connect
<**mode**>	a feature on a predicate only if <clause-type:main>	decl, imp, int, excl, propos
<**tense**>	encodes temporal interpretation	pres, past, future
<**ending**>	a feature marked for different ways of instantiating mode and the clausal type	string values e.g., *ta, nunka, ela, ki, nun, tako, ...*

Table 1: Features for Inflectional Morphology

3. Local Syntactic Database

Our Korean XTAG system uses the LTAG parser developed by Anoop Sarkar (Sarkar, 2000). Written in C, it can process Korean characters represented as 2-byte codes. This parser was meant to use the XTAG English grammar (The XTAG-Group, 1998), and so it uses the lexical

[1] In Korean, both verbs and adjectives play the role of a predicate in a sentence.

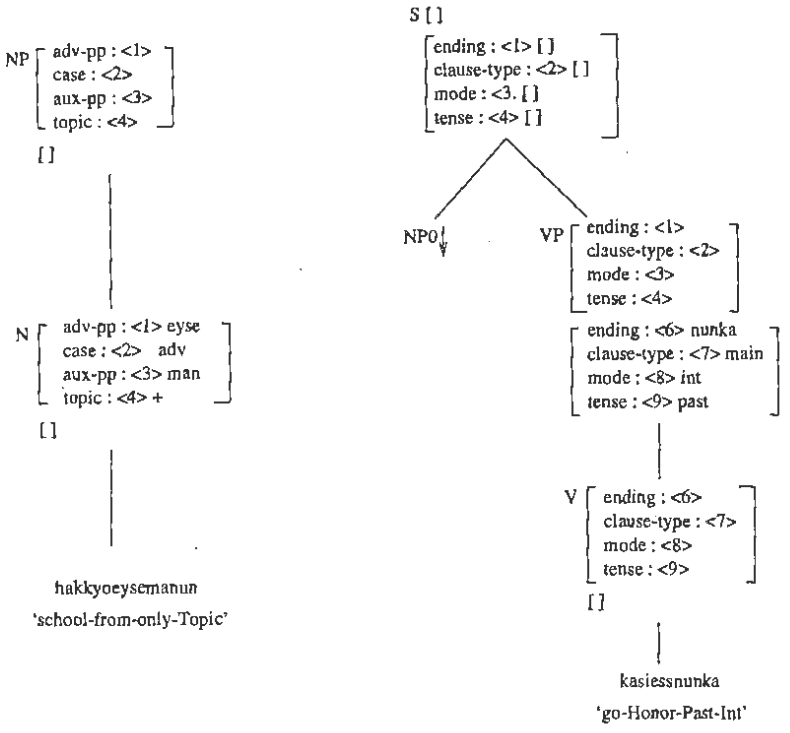


Figure 1: Instantiating Inflections as Features

databases that are part of the English grammar. In the English grammar, all the morphological variants of each word are listed in the morphological database (Morph DB), where they are mapped to a stem and lexical feature structures. The stem is then used to select a set of elementary trees in the syntactic database. The older Common Lisp XTAG parser keeps these databases separate, but the C parser combines them into a single database. The C parser uses this database (Syn DB) in order to select appropriate trees for the words in the input sentence.

Since a word in English has a small number of inflections, it is possible to describe as separate entries all the inflected forms in the Syn DB. However, this way of describing lexicons for Korean is impractical and inefficient, due to its rich morphology. To resolve this problem, we separate the Syn DB into Elementary Syn DB (ESDB) and Local Syn DB (LSDB). Only stems of *eojeols* are listed in the ESDB. This means that we can construct a lexicon for the stem words without considering all the morphological variants, making the life of grammar developers much easier. LSDB only contains *eojeols* of an input sentence as entries with associated elementary trees and lexical feature structures. LSDB is generated dynamically through the use of *Lexicon Extractor*. Given an input sentence, the lexicon extractor takes the result of morphological analysis produced by a morphological analyzer (and the POS tagger) developed at Yonsei University (Yoon *et al.*, 1999), and generates an LSDB by making reference to the ESDB and converting inflections on each *eojeol* to feature structures.

The step of generating an LSDB is as follows:

Firstly, the input sentence goes through the morphological analyzer and the POS tagger. If the morphological analyzer or the POS tagger makes errors, the user can manually input the tagged

sotaycangi	*sotaycang*/N+*i*/Pn
mwucenkilul	*mwucenki*/N+*lul*/Pa
swulihayessta	*swuliha*/V+*yess*/PE+*ta*/Ei+./SC

Table 2: Results of the morphological analyzer and the POS tagger for example (3)

form. We consider the example sentence in (3) to show the execution of the system.

(3) *Sotaycang-i mwucenki-lul swuliha-yess-ta.*
platoon-leader-Nom radio-Acc repair-Past-Decl
'The platoon leader repaired the radio.'

The morphological and tagging results of (3) are as shown in Table 2. Here, N, V, PE, Ei, Pn, Pa, and SC are tags for noun, verb, pre-ending (i.e., tense), indicative ending, nominative postposition, accusative postposition and period punctuation respectively.[2]
Secondly, the lexicon extractor extracts syntactic information from ESDB for the content words and function words which appear in the given sentence, i.e. 'sotaycang/N', 'i/Pn', 'mwucenki/N', 'lul/Pa', 'swuliha/V', 'yess/PE', 'ta/Ei' and './SC'. From the results of morphological analysis, the lexicon extractor selects elementary trees or tree families for the content words (i.e., 'sotaycang/N', 'mwucenki/N', 'swuliha/V') by looking up the ESDB. The function words (i.e., 'i/Pn', 'lul/Pa', 'yess/PE' and 'ta/Ei') are converted to feature structures, which will appear in tree nodes. With this data collected, the LSDB is generated listing all the *eojeols* in the input sentence with associated elementary trees, tree families and features. Crucially, the LSDB contains only the *eojeols* of the input sentence as entries. Table 3 shows the LSDB generated from the morphological analysis results and the ESDB. In Table 3, @nom is a template for <case>=nom, @acc for <case>=acc, @past for <tense>=past, @cls-main for <clause-type>=main and @end-*ta* for <ending>=ta.

⟨⟨INDEX⟩⟩*sotaycangi* ⟨⟨ENTRY⟩⟩*sotaycangi* ⟨⟨POS⟩⟩N ⟨⟨TREES⟩⟩αNP βNP βNP-V βNP-S ⟨⟨FEATURES⟩⟩@nom
⟨⟨INDEX⟩⟩*mwucenkilul* ⟨⟨ENTRY⟩⟩*mwucenkilul* ⟨⟨POS⟩⟩N ⟨⟨TREES⟩⟩αNP βNP βNP-V βNP-S ⟨⟨FEATURES⟩⟩@acc
⟨⟨INDEX⟩⟩*swulihayessta* ⟨⟨ENTRY⟩⟩*swulihayessta* ⟨⟨POS⟩⟩V
⟨⟨FAMILY⟩⟩Tnx0nx1V ⟨⟨FEATURES⟩⟩@past @cls-main @end-*ta*

Table 3: LSDB generated from the example

The parser uses this LSDB and generates the derived tree shown in Figure 2.

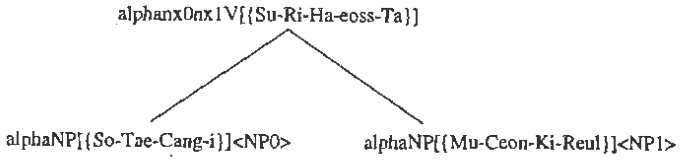


Figure 2: Derivation tree for the example sentence (3)

[2]Although our system reads and generates Hangul (Korean characters), we use romanized examples in this paper for convenience.

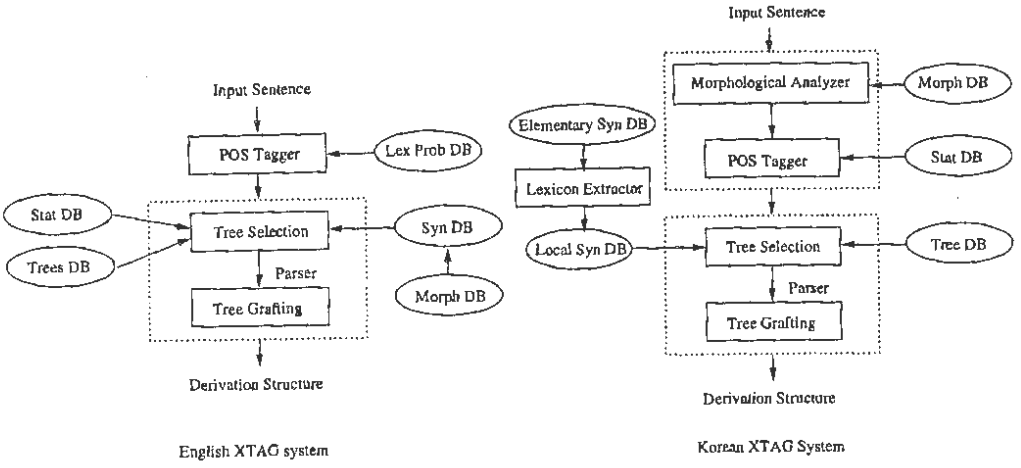


Figure 3: Korean XTAG System and English XTAG System

The overall flow of our Korean XTAG system is represented in Figure 3 in comparison to the English XTAG system. In the Korean XTAG system, we added the lexicon extractor, morphological analyzer and POS tagger which run independently of the XTAG parser. Our Korean grammar currently has 15 tree families and 289 elementary trees that handle various syntactic phenomena: e.g., adverb modification, sentences with empty arguments, relative clauses, complex noun phrases, auxiliary verbs, gerunds, adjunct clauses.

4. Conclusion

In this paper, we addressed linguistic and implementation problems raised by rich morphology in Korean. We first motivated a feature hierarchy on various types of inflections in order to handle all possible ways of combing them. We then described the modifications we have implemented on the English XTAG system, enriching it with a morphological analyzer (which also does POS tagging) and lexicon extractor. These modifications enable us to get rid of a syntactic database from the system that would require listing of all possible morphological variants of words. Instead, we divide up the syntactic database into ESDB and LSDB, where ESDB contains stems with associated elementary trees and tree families, and LSDB only contains *eojeols* of a given input sentence with associated elementary trees and feature structures to represent inflections. Furthermore, by incorporating a morphological analyzer to the system, we are able to separate out morphological generative rules from syntactic rules in the description of LTAG grammar for Korean. Our approach can be applied to FB-LTAG development for other languages with rich morphology.

References

SARKAR A. (2000). A probabilistic head-corner chart parser for TAGs. Ms. UPenn.

THE XTAG-GROUP (1998). *A Lexicalized Tree Adjoining Grammar for English.* Technical Report IRCS 98-18, UPenn.

YOON J., LEE C., KIM S. & SONG M. (1999). Morphological analyzer of Yonsei univ., Morany: Morphological analysis based on large lexical database extracted from corpus. In *Proceedings of Korean Language Information Processing* (In Korean).

POSTERS - AFFICHES

Deriving polarity effects

Raffaella Bernardi

UiL-OTS
University of Utrecht
Trans, 10
3512 Utrecht, NL

Abstract

Polarity Items are linguistic expressions known for being a 'lexically controlled' phenomenon. In this paper we show how their behavior can be implemented in a deductive system. Furthermore, we point out some possible directions to recast the deductive solution into a Tree Adjoining Grammar system. In particular, we suggest to compare the proof system developed for Multimodal Categorial Grammar (Moot & Puite, 1999) with the Partial Proof Trees proposed in (Joshi & Kulick, 1997).

Introduction

In this paper we discuss how polarity effects can be derived from controlled lexical items. Polarity Items (PIs) are linguistic expressions which depend on the polarity of their context for grammaticality (Ladusaw, 1979). Moreover, both in the syntactic and semantic traditions their distribution is considered to be 'lexically controlled'. Combining these two claims we can look at PIs as lexical items carrying some sensitivity features from which their restricted distribution derives. Reading out this observation, we can deduce that the needed ingredients to formalize PIs' behavior are: (i) lexically anchored structures, and (ii) operations to compose them. These two points are what is required by the definition of 'lexicalized grammar'. Several are the formalisms which satisfy these properties, among them we distinguish two main groups: Phrase Structure Grammars (e.g. Tree Adjoining Grammars –TAG), and Deductive Grammars (e.g. Multi Modal Categorial Grammar –MMCG). In (Bernardi, 1999) PIs have been studied from a proof theoretical perspective using MMCG as framework.

An interesting question to ask is how the derivations of polarity effects can be recast into Phrase Structure Grammars. Working out a comparison in this sense, will clarify the linguistic meaning of the logical principles at work in the deductive approach, and will open new possibilities of interaction between the two groups. From the one hand, Phrased Structure Grammars are known for being linguistically sensitive formalisms which, however, lack some of the inferential power inherent in the deductive approaches. On the other hand, the latter, are logically well defined, but the formal behavior of its operators might result less intuitive from a linguistic perspective. We believe that a communication between the two families would be productive for both approaches.

In this paper we suggest some possible lines of research which could be worked out to recast the deductive implementation of PIs into TAG. In order to reduce the gap between the two systems we consider the works carried out in (Joshi & Kulick, 1997) and (Joshi *et al.*, 1999), which build a bridge between TAG and MMCG. In the former

paper, categorial grammar proofs are used as building blocks resulting in a 'middle ground' system known as PPTS. In the latter, the comparison is extended to the structural modalities which characterize MMCG.

1. Polarity Items

For reasons of space we limit our analysis to Negative Polarity Items (NPIs), i.e. expressions as *yet, at all, anything,* licensed by downward-entailing operators, e.g. *nobody, rarely,* (Ladusaw, 1979). In the examples below NPIs are emphasized and licensers are marked by bold characters.

Linguistic data

(ia.)	Somebody left.	(iiia.)	Kim **rarely** says *anything at all.*
(ib.)	**Nobody** left.	(iiib.)	*Kim says *anything at all.*
(iia.)	**Nobody** left *yet.*	(iva.)	**Nobody rarely** says *anything.*
(iib.)	*Somebody left *yet.*	(ivb.)	**Nobody** says *anything.*

These data show that: although NPIs require a negative licenser the converse is not the case (i,ii); the negative context created by a licenser can license more than one NPI within its scope (iii); and NPIs can occur in sentences with more then one licenser (iv). Furthermore, NPIs can occur in more complex structures as well, as shown below:

(va.) **Nobody** thinks Peter did *anything* wrong.
(vb.) *Somebody thinks Peter did *anything* wrong.

(va.) A doctor who knew *anything* about acupuncture was **not available**.
(vb.) *Some doctor who knew *anything* about acupuncture was **not found**.

These example show that NPIs can occur in an embedded sentence while licensed by an expression in the main sentence (v); and that they are felicitous when part of a relative construction which allows to escape the syntactic scope of the licenser, but still force them to be interpreted in its semantic scope (vi). See (de Swart, 1998), where the last example has been proposed and discussed.

2. Polarity Items in MMCG

Two well known facts regarding MMCG and PIs are that: MMCG belongs to the family of resource sensitive logic, where the resources are meant as linguistic signs; and PIs are linguistic expressions sensitive to the polarity of their context. We suggest to consider the polarity as a particular feature required by the NPI and produced by the licenser. This idea has been independently implemented in two different resource logics, namely MMCG (Bernardi, 1999), and Multiplicative Linear Logic (Fry, 1999). In the latter the 'polarity feature' is represented as a proposition ℓ assigned to the linguistic categories, of the NPIs and licensers, by means of the tensor operator $\otimes$. The proper function of this operator is to concatenate logical types, or in other words the linguistic resources the logic is reasoning about. When employing it to concatenate the polarity feature to a linguistic category the former is treated as a 'phantom resource'. The language of MMCG is expressive enough to avoid this improper use of the concatenation operator, and of the resource management. A detailed comparison of the two proposals is given in (Bernardi, 2000). In the following we briefly introduce MMCG system and then we show its application to NPI.

Classical Categorial Grammar (CG), has its logical counterparts in the Lambek Calculus (Lambek, 1958). The formal language of this calculus is built on the binary opera-

tors, $\backslash$, / and $\bullet$, viz. the directed implication operators and the product one, and a finite set $\mathcal{A}$ of atomic formula, e.g. $\mathcal{A}$= $\{np, s, n\}$. MMCG is obtained extending this language with unary operators $\Box^{\downarrow}$ and $\Diamond$. We refrain from presenting the logical rules of the whole system which can be found in (Moortgat, 1997) and we comment the logical behavior of the unary operator on which the PIs account is based. Let $\Gamma \vdash A$ stand for the assignment of the category A to the linguistic structure Γ,

Logical Rules

$$\frac{\Delta \vdash \Diamond A \quad \Gamma[\langle A \rangle] \vdash B}{\Gamma[\Delta] \vdash B}\,[\Diamond \mathrm{E}] \qquad \frac{\Gamma \vdash A}{\langle \Gamma \rangle \vdash \Diamond A}\,[\Diamond \mathrm{I}]$$

$$\frac{\Gamma \vdash \Box^{\downarrow} A}{\langle \Gamma \rangle \vdash A}\,[\Box^{\downarrow} \mathrm{E}] \qquad \frac{\langle \Gamma \rangle \vdash A}{\Gamma \vdash \Box^{\downarrow} A}\,[\Box^{\downarrow} \mathrm{I}]$$

Notation: [$*$E] and [$*$I] stand for the elimination and introduction of the operator $*$. For our goal the attention should be focused on the introduction rules, which imply that if a structure Γ is proved to be of category A, then it is of category $\Box^{\downarrow}\Diamond A$ as well, viz. $A \Rightarrow \Box^{\downarrow}\Diamond A$

$$\frac{\dfrac{\Gamma \vdash A}{\langle \Gamma \rangle \vdash \Diamond A}\,[\Diamond \mathrm{I}]}{\Gamma \vdash \Box^{\downarrow}\Diamond A}\,[\Box^{\downarrow} \mathrm{I}]$$

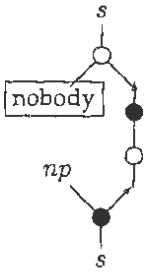
We will profit of this logical property of the system to deal with NPIs. Recolling the information deduced from the linguistic data given above, we know that while a NPI requires a negative licenser, the converse is not true. In our framework this means that the type assigned to the licenser has to derive the type of a lexical item of the same linguistic category but lacking the polarity effect, e.g. if the standard type for general quantifier (GQ) is $s/(np\backslash s)$, then a licenser GQ, as *nobody*, is typed $s/\Box^{\downarrow}\Diamond(np\backslash s)$, this type satisfies the requirement above, namely $s/\Box^{\downarrow}\Diamond(np\backslash s) \Rightarrow s/(np\backslash s)$. The 'polarity feature' is properly represented as a 'property' of the linguistic category by means of $\Box^{\downarrow}\Diamond$. The logical type assigned to NPIs will require to be in a context where this property is provided. Moreover, it will have to account for cases as (iiia), where more then one NPI is licensed by the same licenser. Let us consider the adverb *yet* as an example. The standard adverbial type is $(np\backslash s)\backslash(np\backslash s)$, we enrich it with the the polarity feature obtaining $\Box^{\downarrow}\Diamond(np\backslash s)\backslash\Box^{\downarrow}\Diamond(np\backslash s)$, where the modalities on the goal formula will require the context to be of the right polarity, and the ones on the argument will account for multiple NPIs occurrences.

Example 2.1 Nobody left yet.

$$\frac{\mathrm{nobody} \vdash s/\Box^{\downarrow}\Diamond iv \qquad \dfrac{\dfrac{\dfrac{\mathrm{left} \vdash iv}{\langle \mathrm{left} \rangle \vdash \Diamond iv}\,[\Diamond \mathrm{I}]}{\mathrm{left} \vdash \Box^{\downarrow}\Diamond iv}\,[\Box^{\downarrow} \mathrm{I}] \qquad \mathrm{yet} \vdash \Box^{\downarrow}\Diamond iv\backslash\Box^{\downarrow}\Diamond iv}{\mathrm{left} \circ \mathrm{yet} \vdash \Box^{\downarrow}\Diamond iv}\,[\backslash \mathrm{E}]}{\mathrm{nobody} \circ (\mathrm{left} \circ \mathrm{yet}) \vdash s}\,[/\mathrm{E}]$$

3. A possible interaction

In (Joshi & Kulick, 1997) it is acknowledged that the bridge connecting TAG with deductive approaches fails to incorporate the elimination rule for the tensor operator. This might be a problem when trying to recast the way PIs are treated in (Fry, 1999). We have shown that MMCG has the right expressiveness for dealing with this linguistic phenomenon and that the solution is strongly based on the logical properties of the unary modalities. In (Joshi *et al.*, 1999) a translation of the behavior of MMCG modalities into Partial Proof Trees (PPTs) is given and it is claimed that by using PPTs the linguistic phenomena motivating the introduction of these modalities can be handled eliminating them. It could be interesting to see whether this claim hold with respect to the linguistic application here described. A possible way to tackle this question could be to look at the proof nets developed for MMCG and presented in (Moot & Puite, 1999), where they are proved to be sound and complete. In this graph-based proof system, the lexical items are anchored to trees, which are the result of the unfolding of the original types. This remind quite straightforward the idea on which PPTS is based. Below we give the tree assigned to nobody as an example.



References

BERNARDI R. (1999). Monotonic Reasoning from a Proof-Theoretic Perspective. In G. KRUIJFF & R. OEHRLE, Eds., *Proceedings of Formal Grammar 1999*, p. 13–24.

BERNARDI R. (2000). Polarity items in resource logics. A comparison. Submitted to Esslli00-Student Session.

DE SWART H. (1998). Negation, polarity and inverse scope. *Lingua*, 105 (3 p.), 175–200.

FRY J. (1999). Proof nets and negative polarity licensing. In M. DALRYMPLE, Ed., *Semantics and Syntax in Lexical Functional Grammar. The Resource Logic approach.*, chapter 3, p. 91–116. MIT Press.

JOSHI A. & KULICK S. (1997). Partial proof trees as building blocks for categorial grammar. *Linguistics and Philosophy*, 20, 637–667.

JOSHI A., KULICK S. & KURTONINA N. (1999). Partial proof trees and structural modalities. TAG+4 Workshop.

LADUSAW W. (1979). *Polarity sensitivity as inherent scope relations*. PhD thesis, Texas.

LAMBEK J. (1958). The Mathematics of Sentence Structure. *American Mathematical Monthly*, 65, 154–170.

MOORTGAT M. (1997). Categorial Type Logics. In J. VAN BENTHEM & A. TER MEULEN, Eds., *Handbook of Logic and Language*, p. 93–178. Cambridge: The MIT Press, Cambridge, Massachusetts.

MOOT R. & PUITE Q. (1999). *Proof Nets for the Multimodal Lambek Calculus*. Technical report, Department of Mathematics, Utrecht University.

Un outil pour calculer des arbres de dépendance à partir d'arbres de dérivation

Lionel Clément

TALANA, UFRL case 7003 - Université Paris 7
2, pl. Jussieu 75005 Paris FRANCE - Lionel.Clement@linguist.jussieu.fr

Résumé

Nous présentons un outil permettant de calculer un arbre de dépendance sémantique à partir d'un arbre de dérivation TAG.
Cette opération est rendue possible grâce à un algorithme de filtrage des arbres de dérivation et à une étude des propriétés sémantiques liées aux opérations de substitution et adjonction sur une grammaire lexicalisée.

Introduction

Les principes de cooccurrence prédicat-argument (PCPA) et de minimalité sémantique associés au principe de lexicalisation nous permettent d'interpréter l'arbre de dérivation en représentation sémantique de la phrase comme le font (Rambow & Joshi, 1992), (Vijay-Shanker *et al.*, 1995), (Candito & Kahane, 1998) et (Candito, 1999).
PCPA indique que tout arbre élémentaire ancré par un prédicat comporte au moins un nœud pour chacun des arguments ((Kroch & Joshi, 1985), (Abeillé, 1991)). Le principe de lexicalisation pose que tout arbre élémentaire comporte au moins un nœud feuille lexical et le principe de minimalité sémantique que tout arbre élémentaire corresponde à une unité sémantique non vide (Abeillé, 1991), (Candito, 1999).
Ainsi l'arbre de dérivation est une représentation possible de dépendances sémantiques dans la mesure où les adjonctions et substitutions constituent des opérations entre gouverneurs sémantiques et dépendants sémantiques.
Mais, même en prenant la définition de (Schabes & Shieber, 1994) des arbres de dérivations, cette représentation syntaxique ne permet pas toujours de calculer les dépendances immédiatement.
Les cas suivants semblent poser particulièrement problème :

- L'adjonction d'arbres élémentaires non modificateurs comme les auxiliaires verbaux. (fig.1)
- L'inversion de l'ordre de dépendance entre gouverneur sémantique et dépendant sémantique lors d'adjonction. C'est par exemple le cas pour une infinitive ou une complétive dominant la principale dans l'arbre de dérivation (fig.2).

Et dans le cas de l'analyse selon (Vijay-Shanker, 1987)

- Les ambiguïtés artificielles dues aux ordres multiples d'adjonctions dans la dérivation. (fig.3).

Nous avons donc développé, dans le cadre du projet FTAG (Abeillé *et al.*, 1999), un outil qui permet de calculer un arbre de dépendance sémantique de la suite analysée en tirant-parti, d'une part des informations lexicales de la grammaire lexicalisée et, d'autre part d'un ensemble de règles générales.

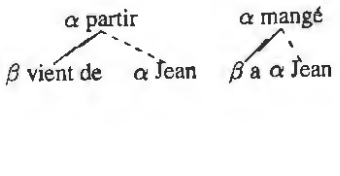


FIG. 1 – *Arbre de dérivation de "Jean vient de partir", "Jean a mangé"*

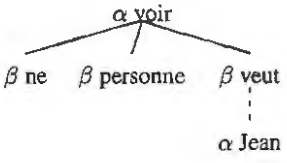


FIG. 2 – *Arbre de dérivation selon (Vijay-Shanker, 1987) de "Jean ne veut voir personne"*

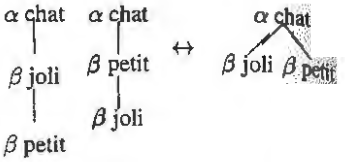


FIG. 3 – *Arbre de dérivation selon (Vijay-Shanker, 1987) puis (Schabes & Shieber, 1994) de "joli petit chat"*

1. Principe général

Le logiciel filtre les arbres de dérivation en fonction d'un schéma général (par exemple celui de l'adjonction d'un auxiliaire verbal ou d'une infinitive), puis applique de façon incrémentale un ensemble de règles permettant de dresser une représentation sémantique correspondante.
Les fonctions effectives données par la grammaire LTAG sont directement attribuées aux arguments sémantiques.
Nous avons décrit trois règles générales qui s'appliquent pour un ensemble de familles donné :

- Règle faisant remonter la principale d'une complétive ou d'une infinitive.
- Règles aplatissant les modifieurs.
- Règle éliminant l'adjonction d'un auxiliaire de temps.

Nous expliciterons infra les deux premières.

2. Règles de calcul d'un arbre de dépendance sémantique à partir d'un arbre de dérivation TAG

2.1. Règle faisant remonter la principale d'une complétive ou d'une infinitive

A l'exception des complétives sujet, les phrases enchâssées sont décrites comme des adjonctions sur un nœud phrastique pour les complétives et infinitives et sur un nœud nominal pour les relatives.
Dans le premier cas, il en résulte naturellement que l'arbre élémentaire correspondant à la phrase matrice est dominé par l'arbre élémentaire correspondant à la phrase enchâssée dans l'arbre de dérivation comme montré fig.4

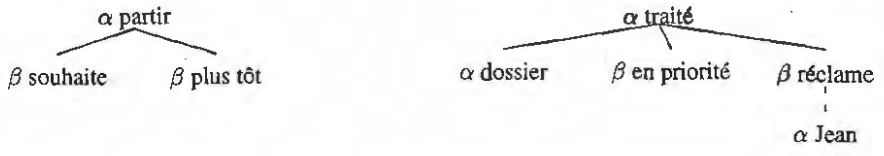


FIG. 4 – *Arbres de dérivation de "Jean souhaite partir plus tôt" et "Jean réclame que son dossier soit traité en priorité"*

Dans une représentation sémantique, nous voudrions voir cet ordre - s'il correspond à un ordre de dépendance sémantique - respecté pour les relatives mais pas pour les complétives et infinitives.
Nous présentons fig.5 la règle générale qui permet de traduire l'adjonction sur un nœud phrase d'un arbre élémentaire correspondant à une complétive ou infinitive. Cette règle ne s'appliquera pas aux relatives car l'arbre de dérivation ne sera pas filtré.

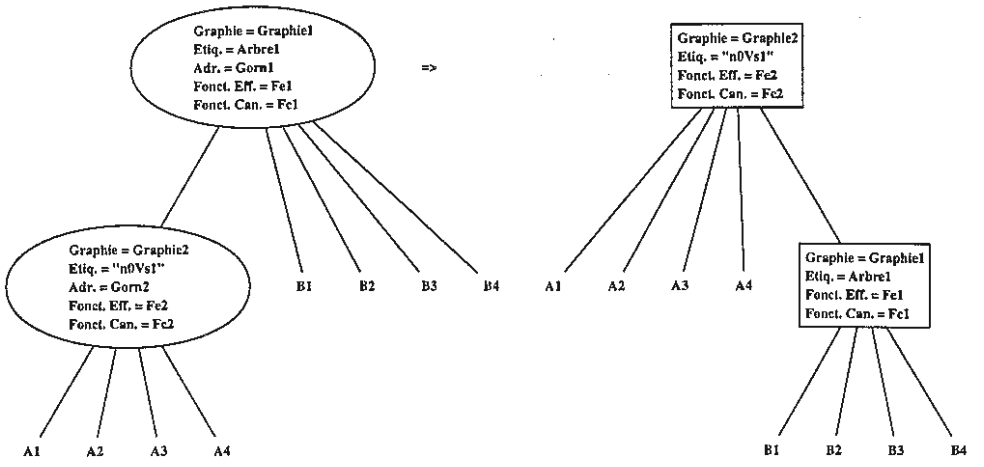


FIG. 5 – *Règle faisant remonter la principale d'une complétive ou d'une infinitive. Les termes $A_1, A_2, \ldots, A_k$ et $B_1, B_2, \ldots, B_k$ correspondent à des variables libres pouvant être instanciées par un nœud.*

Le résultat de l'application de la règle est simplement une réorganisation des nœuds comme le montre la fig.5.

2.2. Règles aplatissant les modifieurs

L'analyse de (Schabes & Shieber, 1994) permet de construire plus immédiatement une représentation sémantique pour les modifieurs multiples. En effet, plusieurs adjonctions peuvent avoir lieu sur le même nœud en les ordonnant, ce qui produit un arbre de dérivation "plat" comme montré fig.6.

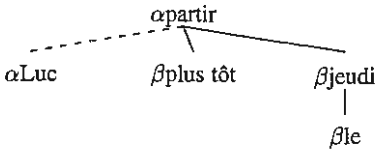


FIG. 6 – *Arbre de dérivation selon (Schabes & Shieber, 1994) de "Luc part plus tôt le jeudi."*

Dans le cas d'une analyse selon (Vijay-Shanker, 1987), il est possible "d'aplatir" les modifieurs de telle manière qu'il puissent être dépendants sémantiques au même niveau.
L'arbre de dérivation correspondant à une adjonction multiple contient nécessairement une adjonction sur la racine d'un arbre auxiliaire. Cette condition étant par ailleurs suffisante, elle permet de décrire la structure filtrante.

Nous présentons fig.7 la règle générale qui permet d'aplatir les modifieurs. Cette règle sera appliquée autant de fois que des modifieurs artificiellement enchâssés apparaîtront.

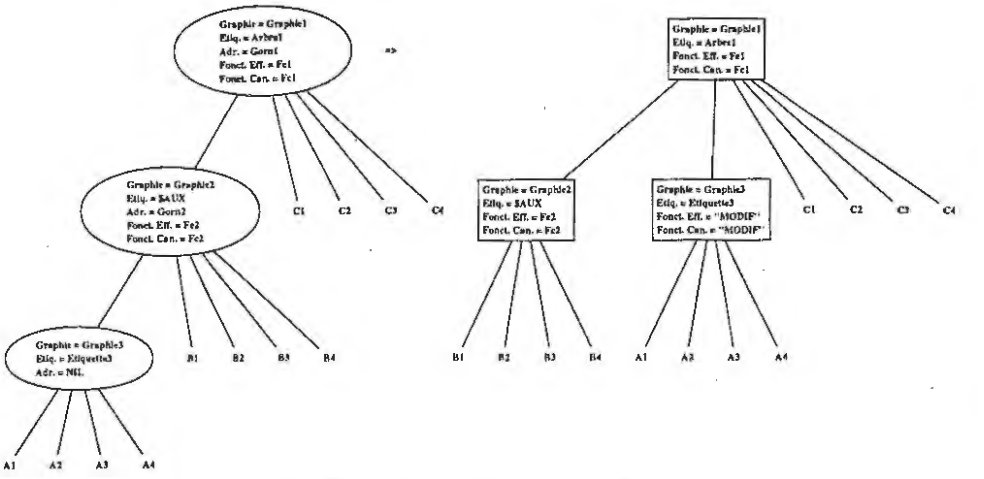


FIG. 7 – *Règle aplatissant les modifieurs*

Conclusion

Cet algorithme a été implémenté et fournit pour chaque analyse effectuée un arbre de dépendance sémantique. C'est donc une interface utile pour lier un niveau syntaxique à un niveau conceptuel.

Ce travail a été fait dans la perspective de l'analyse en TAG. Nous pouvons le relier aux travaux de (Danlos, 1998) où l'arbre de dérivation est calculé à partir d'un tel niveau conceptuel.

Références

ABEILLÉ A. (1991). *Une grammaire lexicalisée d'arbres adjoints pour le français: application à l'analyse automatique*. PhD thesis, University Paris 7.

ABEILLÉ A., CANDITO M.-H. & KINYON A. (1999). Ftag : current status and parsing scheme. In *VEXTAL'99*, Venise.

CANDITO M.-H. (1999). *Représentation hiérarchique de grammaires lexicalisées: application au français et à l'italien*. PhD thesis, University Paris 7.

CANDITO M.-H. & KAHANE S. (1998). Can the tag derivation tree represent a semantic graph? an answer in the light of meaning-text theory. In *TAG+4*.

DANLOS L. (1998). G-tag : un formalisme lexicalisé pour la génération de textes inspiré de tag. In *Traitement Automatique des Langues (TAL)*, volume 39.

KROCH A. S. & JOSHI A. K. (1985). *The linguistic relevonce of tree adjoining grammars*. Technical report MS-CIS-85-16, Department of Computer and Information Science, University of Pennsylvania.

RAMBOW O. & JOSHI A. (1992). A formal look at dependency grammars and phrase-structure grammars, with special consideration of word-order phenomena. In *International Workshop on The Meaning-Text Theory*, Darmstadt. Arbeitspapiere der GMD 671. To appear in *Current Issues in Meaning-Text Theory*, Leo Wanner, editor.

SCHABES Y. & SHIEBER S. M. (1994). An alternative conception of tree-adjoining derivation. In *Computational Linguistics*.

VIJAY-SHANKER K. (1987). *A study of Tree Adjoining Grammars*. PhD thesis, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA.

VIJAY-SHANKER K., WEIR D. & RAMBOW O. (1995). Parsing d-tree grammars. In *International Workshop on Parsing Technologies*.

ELEMENTARY TREES FOR SYNTACTIC AND STATISTICAL DISAMBIGUATION

Rodolfo Delmonte, Luminita Chiran, Ciprian Bacalu
Ca' Garzoni-Moro, San Marco 3417
Università "Ca Foscari"
30124 - VENEZIA
Tel. 39-41-2578464/52/19
E-mail: delmont@unive.it
Website: http//byron.cgm.unive.it

ABSTRACT

In this paper we argue in favour of an integration between statistically and syntactically based parsing, where syntax is intended in terms of shallow parsing with elementary trees. None of the statistically based analyses produce an accuracy level comparable to the one obtained by means of linguistic rules [1]. Of course their data are strictly referred to English, with the exception of [2, 3, 4]. As to Italian, purely statistically based approaches are inefficient basically due to great sparsity of tag distribution – 50% or less of unambiguous tags when punctuation is subtracted from the total count as reported by [5]. We shall discuss our general statistical and syntactic framework and then we shall report on an experiment with four different setups: the first two approaches are bottom-up driven, i.e. from local tag combinations:
A. Statistics only tag disambiguation; B. Stastistics plus syntactic biases; C. Syntactic-driven disambiguation with no statistics; D. Syntactic-driven disambiguation with conditional probabilities computed on syntactic constituents.
The second two approaches are top-down driven, i.e. driven from syntactic structural cues in terms of elementary trees:
In a preliminary experiment we made with automatic tagger, we obtained 99% accuracy in the training set and 98% in the test set using combined approaches: data derived from statistical tagging is well below 95% even when referred to the training set, and the same applies to syntactic tagging.

1. INTRODUCTION

We assume, together with [1] that POS tagging is essentially a syntactically-based phenomenon and that by cleverly coupling stochastic and linguistic processing one should be able to remedy some if not all of the drawbacks usually associated with the two approaches, when used in isolation. However, as will be shown in detail in the following section, rather than using FSA we use Elementary Trees organized in an RTN both for training and for parsing. As to the statistical part, we don't use HMMs but only conditional probabilities on the basis of trigram information as discussed below.
Syntactic driven disambiguation is accomplished by using an RTN made up of 1700 arcs and 22 nets, which we use in a non-recursive way, as explained below. Data for the construction of the RTN were derived from the manual annotation of 60,000 token corpus suite which is then used as test set. Frequency of occurrence associated to each rewrite rule is used as organizing criteria in the ordering of the arcs contained in each node of each net. However, in the experiment, we let conditional probabilities at the level of major constituent, or net, do the choice for the best path.

Rather than flattening the Phrase Structure Grammar as [8] suggest in their shift-reduce algorithm, we only check for reachability in nonterminal symbols. So, even though the formal structure of RTN is recursive, the disambiguating algorithm does not use recursive calls and all computation is flattened down to one level, that of tags corresponding to preterminals in the RTN. The syntactic-statistical disambiguator (hence SSD) can be defined as a slightly augmented finite state transducer which works at a single level of computation and has access to higher level information when needed. For the details of the implementations the reader should look at [10].

2 .STATISTICAL VS. SYNTACTIC DISAMBIGUATION

The SSD is the final module of our syntactic tagger of Italian. Input to the SSD is the complete and redundant output of the morphological analyser and lemmatizer, IMMORTALE [10]. IMMORTALE finds all possible and legal tags for the word/token under analysis on the basis of morphological generation from a root dictionary of Italian made up of 80,000 entries and a dictionary of invariant words - function words, polywords, names and surnames, abbreviations etc. - of over 12,000 entries.

As commented by [6], the application of stochastic techniques in automatic part-of-speech tagging is particularly appealing given the ease with which the necessary statistics can be automatically acquired and the fact that very little handcrafted knowledge need to be built into the system(ibid., 152). However both probabilistic models and Brill's algorithm need a large tagged corpus where to derive most likely tagging information. It is a well known fact that in lack of sufficient training data, sparsity in the probabilistic matrix will cause many bigrams or trigrams to be insufficiently characterized and prone to generate wrong hypotheses. This in turn will introduce errors in the tagging prediction procedure. Italian is a language which has not yet made available to the scientific community such large corpus. In lack of such an important basic resource, there are two possibilities:
3. manually building it by yourself;

4. using some automatic learning procedure which in our case corresponds to the use of a syntactic tagger.

We have been working on such a corpus of Italian with the aim of achieving the above-mentioned final goal, without having to manually build it. The algorithm that we will present in this paper is partly based on stochastic techniques: this is however coupled with linguistic processing by means of a Context Free grammar of Italian formalized as an RTN, which filters it. Statistics is usefully integrated into the syntactic disambiguator in order to reduce recursivity and allow for better predictions.

After a first fully automatic phase, we started building BIASES which are used to correct most common errors. This second phase has taken us 3 man/months work to complete. The final result is a 95% accuracy analysis on the whole corpus. The final output has then been used to collect trigrams for the statistical tagger. Statistics and syntactic disambiguation have then been fully integrated in order to reduce recursivity and allow for better predictions and higher efficiency. Fully stochastic taggers, in case no large tagged corpora are available, may make use of HMMs. However, HMMs show some of the disadvantages present in more common Markov models: they lack perspicuity basically because they impose that the data related to tags are all treated on a par. Even though they allow for biases to be implemented – very similar to patches in Brill's tagger – they are inherently incapable of capturing higher level dependencies present in natural language, and are always prone at generating wrong interpretations, i.e. accuracy never goes higher than 96-97%. Of course it is a good statistical result, but a poor linguistic result, seen the premises, i.e. the need to use tagging information for further syntactic processing.

2.1 Tagset and Statistical Processing

Our tagset is made up of 91 tags thus subdivided: 10 for punctuation; 4 for abbreviations, titles, dates, numbers; 19 for verbs including three syntactic types of subcategorization – transitive, intransitives, copulatives – and tensed cliticized verbs; 47 for function closed class words subdivided into 18 for pronouns, clitics, determiners and quantifiers - 18 for adverbs conjunctions and prepositions - 11 for auxiliaries and modals; 11 for adjectives and nouns, including special labels for colour nouns, time nouns, factive nouns, proper nouns, person names - this list includes special labels for guessed proper nouns, foreign words and misspelled words. Twenty categories from the general tagset never occur single, so they had to be converted into distributionally equivalent ones, in the statistical table.

We refer to the tagset of LOB corpus which uses 157 tags for English: however, they include in their set special tags for plural forms, genitive forms both for nouns and verbs, and with tags for comparative and superlative forms of adjectives. In case we eliminate these duplicate forms the total number of tags is 107.

The disambiguator is made up of two separate modules: the Probabilistic Transition Table for local tag disambiguation; the syntactic transition network where the learning phase is situated. We use a Viterbi-like algorithm to find and select the best candidates in any given context, given the trigram matrix information. However, since we only computed trigram for a comparable small quantity of training data - we would need 700K trigrams for our 90 tags, but we only use 30K! - we often find no data available. In a similar way to the reductionist statistical approach proposed by [2,7] we induce the best tag from the set of available tags in the context of an unambiguous tag by recursively calling all contextually allowable combinations, from where we select the ones corresponding to the current ambiguity class: we then compute trigram conditional probabilities, according to the formula suggested in [2]. We remove low-probability candidate tags by ignoring the tail of the Viterbi output list, on the basis of a fixed threshold. In case no data are available, rather than computing zero probability we let the current procedure fail - the algorithm is implemented in Prolog - and use information coming from Elementary Trees (ETs) or Networks which can be superimposed on each tag in a given context: the most adequate ETs will be chosen in the top-down syntactically driven disambiguating procedure. The final aim of the disambiguation is to produce information reusable by the following shallow parser, which will then be in charge of combining ETs previously assigned by the SSD.

3. SYNTACTIC CONSTITUENCY ANNOTATION

The first problem to be solved when starting work on a corpus in order to produce a syntactic structure annotation, is the choice of representation, or the syntactic annotation scheme. As with tagging, the scheme must be consistent, it could be used as gold standard for parser testing or as a basis for the induction of stochastic grammars and lexical representations. The main sources of information in the field of syntactic annotation scheme are related to the Penn Treebank (hence PT) [11], which is remarkable as to extension of the coverage and documentation of linguistic phenomena. The PT uses a generativist constituency which is related to chomskian syntax of the '60s/'70s which we do not share: as a result, much of the bracketing is non comparable. In addition, syntactic constituency has been enriched with functional labels and other non standard additional labels which increased the overall number of constituents but reduced its perspicuity. As a result, PT uses 22 symbols for main constituent and 32 more for functional annotation. We also use 22 symbols for syntactic constituency but they are different from the PT's ones.

The inventory we use follows the basic intuitions of the XBAR syntax, while having as its main goal that to serve as an interface as simple as possible to the following levels of representations: the functional, LFG-style, and the semantic ones. In particular, whereas PT uses Chomsky-adjunction and VP, we opted for a separated IBAR constituent with all tensed verbal constituents and its adjoined minor constituents, like negation, clitics and certain adverbials. We then qualify all verbal complements according to their lexical subcategorization frame. Seen that they only have

one layer of syntactic representation, whereas we allow for two, they include all semantic information at constituent level. In particular, they introduce all possible empty categories in the syntactic constituents with coindexation. In case of discontinuous or non canonical order of constituents, they use special constituent names, like SINV (Inverted Sentence), to allow for the subject NP to be automatically recovered. We introduce no empty category at syntactic level, while leaving their computation for the functional and semantic level. . As an example we report the bracketing for "John's decision to leave":

```
(NP (NP John 's)
        decision
                (S (NP-SBJ *)
                        (VP to
                        (VP leave))))
```

compared to the Italian, "la decisione di Gino di partire"

```
SN-[la-art, decisione-n,
                SPD-[di-pd, SN-[Gino-nh] ]
                SV2-[di-pt, partire-viin] ]
```

where we can see that the level of embedding in PT is 4 brackets, whereas it is 2 brackets in our representation. We report here below the list of constituents in our representation for Italian corpora.

TABLE 1. List of Syntactic Constituents and their meaning

F	sentence, starting with subject SN or SV2; or in case subject is missing starting with IBAR
SN	noun phrase, including its complements and/or adjuncts
SA	adjectival phrase, including its complements and/or adjuncts
SP	prepositional phrase
SPD	prepositional phrase DI / "of"
SPDA	prepositional phrase DA / "by,from"
SAVV	adverbial phrase, including its complements and/or adjuncts
IBAR	verbal nucleus with finite tense and all adjoined elements like clitics, adverbs and negation
SV2	F for infinitival clause
SV3	F for participial clause
SV5	F for gerundive clause
FAC	CP for sentential complement
FC	CP for Coordinate sentences (also ellipsed and gapped)
FS	CP for Subordinate sentence
FINT	CP for +wh interrogative sentence
FP	CP for punctuation marked parenthetical or appositional sentence
F2	CP for relative clause
CP	Generically for dislocated or fronted, sentential adjuncts
COORD	Coordination with coordinating conjunction as head
COMPT	Transitive/Passive/Ergative/Reflexive Complement
COMPIN	Intransitive/Unaccusative Complement
COMPC	Copulative/Predicative Complement

4. AUTOMATIC SYNTACTIC TAGGING

Being language-dependent the tagger needs to be based on an accurate analysis of corpora with an as broad as possible coverage of genre, style and other social and communicative variables. To answer these needs we built our syntactic shallow parser on the basis of manually annotated texts for 60,000 words chosen from different corpora and satisfying the above-mentioned criteria. The annotation was carried out twelve years ago to be used for a text-to-speech system for Italian (DecTalk Italian version) with unlimited vocabulary.

We report here below the list of the 10 main constituents or net labels used by the annotators, which are a superset of our current syntactic tagset which is subsumed by it. As can be easily seen, lexical subcategorization information for verbs was not included: also, no information was available as to DI/DA (of/by-from) PPs, nor a subdivision of sentences in simplex and complex with subordination. Sequences of preterminal symbols, category labels or simply POS tags may reasonably belong to three levels of constituency: in the most desirable case, they may be part of the same constituent, e.g. NP(art, quant, noun); else, they may belong to a parent node, whose head is followed by the Complement node, any head dependent constituent in a daughter node, e.g. NP(art, noun (AP(adj)); finally, it may belong to two sibling nodes from a common higher parent node, as for instance in the case of CP(AdvP(adv, NP), IP(NP, VP)).

However, our tagset of elementary trees is different from the one used within the LTAG approach [12], where they are called Supertags: in our framework, elementary trees only belong to the syntactic constituency domain. On the contrary, in the LTAG framework they are constituted by both syntactic and functional constituent labels.

Table 2. Net Accessibility Preterminals and their Frequency

NET	TAG	FREQ	NET	TAG	FREQ	NET	TAG	FREQ	NET	TAG	FREQ
F	PK	235	SN	Q	189	SP	P	6160	SV	VG	147
F	CONG	218	SN	PRON	338	SV	V	656	SV	VPP	814
F	COSU	294	SN	ART	3792	SV	AUSA	244	SV	VSUP	518
SA	A	353	SN	DIM	117	SV	AUSE	363	SV2	P	173
SA	Q	239	SN	N	1662	SV	CLIT	388	SV2	PT	529
SAVV	AVV	1479	SP	PART	5234	SV	NEG	318	SV2	VI	217

Disambiguation proceeds as follows. Fully ambiguous cases such as the following:**Tag1=[ag, n], Tag2=[ag, n],** cannot be solved by relying on frequency of occurrence given the fact that 75% of all NP rules take the pair Noun/Adjective, and only 25% take Adjective/Noun.

We use biases which take into account a list of exceptions - ambiguous cases which prefer Prenominal position and only then to use local cues provided by the RTN.

At first we try to traverse the network by continuing in the network accessible from the left highest score tag, as explained below. Net traversal is worked out trying to proceed from the arc associated with that tag onto a following one as encoded in the RTN and extracted from the current tag-list. The arc in question is called from the pair (Net, Tag). The output is the associated arc, which is represented as follows,

- arc(Net,Category,InputNode,OutputNode).

In case the current tag(list) is accepted by the RTN no further computation is needed: the associated network will be used for further processing.

2. In case of failure, we execute in turn the following procedures:

a. The two tags belong to two separate networks which are in an inclusion relation;

b. The two tags belong to non inclusive networks.

Case a. is further expanded as follows:

Tag 1 belongs to a network which includes the network to which Tag 2 belongs. Network for Tag 2 is then simply asserted as the first network that Tag 2 may be a proper starting category for.

This information is recovered from a Network Accessibility Table Lookup (NATL) as indicated in Table 2, where all category symbols are cross-tabulated against the network they may provide access for. NATLs are compiled at runtime and are encoded as sets of starting symbols for each network with a given probability.

Match for tags is a simple membership check.

```
Tag1/Tag2 => Network1/Tag1 ⊇
                Network2/Tag2
```

Tag 1 and Tag 2 belong to two separate networks which are both included in another network. Whereas in i. above it was between terminal and nonterminal, this time, the inclusive relationship is between nonterminals,. Network for Tag1 and network for Tag2 are both included in the set of Networks accessible from a higher Network. NATLs used in this case are for nonterminals.

```
Tag1/Tag2=>
{Network1/Tag1,Network2/Tag2} ⊇
        HigherNetwork
```

Tag1 and Tag2 cannot be regarded a legal continuation as can be computed from the available grammar encoded in the RTN. The parsing process is reverted from Top-down to Bottom-up. The first network associated to Tag2 as recovered from NATLs.

5.THE EXPERIMENT

As said above, we took two subparts in order to check the effect of training separately. The benchmark test corpus was constituted by a segment from the School-Administration corpus which amounts to approximately 10,000 tokens and is not included in the training set.

We constrained the choice of the statistical tagger by the matrix of actually occurring combinations as determined by the syntactic disambiguator. Thus the training set should have granted similar results, but as Table 4. clearly show, this is not the case. Some improvements are obtained by the addition of Biases, which in one case that advantage of local syntactic accessibility information.

Table 3. EXPERIMENTAL RESULTS

	Tot. Toks	%Tot	Syntax Only	Syntax + Biases	Trigrams only	Trigr. + Biases	Syntax + Statistics
Culture	28,000	62%	90%	97.5%	93.5%	96.5%	98.97
Politics	7,000	16%	87%	95.5%	92.5%	94.5%	98.05
S.Admin.	10,000	22%	86,5%	93.5%	90.5%	93.5%	97.85
Total	45,000	100%					

6. REFERENCES

[1] P. Tapanainen and Voutilainen A.(1994), Tagging accurately - don't guess if you know, *Proc. of ANLP '94*, pp.47-52, Stuttgart, Germany.

[2] Brants T. & C.Samuelsson(1995), Tagging the Teleman Corpus, in *Proc.10th Nordic Conference of Computational Linguistics*, Helsinki, 1-12.

[3] Lecomte J.(1998), Le Categoriseur Brill14-JL5 / WinBrill-0.3, INaLF/CNRS,

[4] Chanod J.P., P.Tapanainen (1995), Tagging French - comparing a statistical and a constraint-based method". *Proc. EACL'95*, pp.149-156.

[5] Delmonte R., E.Pianta(1999), Tag Disambiguation in Italian, *in Proc.Treebanks Workshop ATALA*, Paris, pp.18-25.

[6] Brill E. (1992), A Simple Rule-Based Part of Speech Tagger, in *Proc. 3rd Conf. ANLP*, Trento, 152-155.

[7] Voutilainen A. and P. Tapanainen,(1993), Ambiguity resolution in a reductionistic parser, *in Sixth Conference of the European Chapter of the ACL*, pp. 394-403. Utrecht.

[8] Pereira F., R.Wright, (1991), Finite-State Approximation of Phrase Structure Grammars, in *Proc. 29th ACL*, Berkeley, 246-255.

[10] Delmonte R., E.Pianta(1996), "IMMORTALE - Analizzatore Morfologico, Tagger e Lemmatizzatore per l'Italiano", in *Atti V Convegno AI*IA*, Napoli, 19-22.

[11] Marcus M. et al.(1993), Building a Large Annotated Corpus of English: The Penn Treebank, *Computational Linguistics*, Vol.19.

[12] Bangalore S. & A.K.Joshi(1998), Supertagging: An Approach to Almost Parsing, in Computational Linguistics, 22.

How Problematic are Clitics for S-TAG Translations?

Mark Dras and Tonia Bleam

Institute for Research in Cognitive Science, University of Pennsylvania
Suite 400A, 3401 Walnut St, Philadelphia PA 19104 USA
{madras,tbleam}@linc.cis.upenn.edu

Abstract

We show that clitics are not as problematic for Synchronous TAG as has been supposed, and give two solutions; and, in doing so, demonstrate that 'unbounded relations', such as it is argued clitics induce between dependency trees, are only an artefact of particular analyses.

1. Introduction

In this paper we investigate Synchronous TAG as defined in Shieber (1994) (hereafter just S-TAG). This formalism has attractive characteristics such as the weak language preservation property (WLPP), whereby the power of the component TAGs is not altered by their synchronisation. A canonical example of the (potential) limitations of S-TAG is translation between languages with pronominal clitics and those without: because of unbounded clitic dependencies, the argument goes, radically different derivation structures are produced for each language, in violation of the isomorphism required by S-TAG. We illustrate the problem using inalienable possession constructions in Spanish, and then present one possible solution using a metagrammar, as in Dras (1999a).

However, this is not the only possible solution; and in examining a variant analysis, this paper demonstrates that the problematic 'unbounded relations' between trees that Shieber mentions are not an innate characteristic of constructions, but rather are artefacts of the analysis. Further, it suggests that the two solutions for the behaviour of clitics presented here reflect a common concept of 'grouping' in grammars.

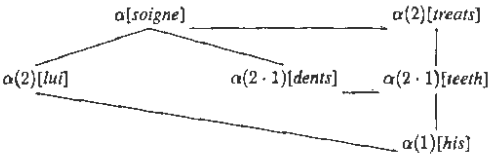


Figure 1: Shieber partial derivation tree pair

2. An Initial Analysis

Shieber (1994) sketches an analysis of clitics (based on a suggestion by Abeillé) giving it as a potential problem for S-TAG, which requires an isomorphism between derivation trees. In this section we discuss Shieber's analysis and show that his class of examples does not, in fact, require non-isomorphic derivation trees. However, such non-isomorphic constructions do exist in other languages and are thus problematic. We go on to argue that the unboundedness in these structures can be handled through the relaxation of the isomorphism requirement via a metagrammar (Dras, 1999a).

2.1. *Shieber's Analysis*

Shieber's example is in (1), with the clitic *lui* indicating possession of the body part by the patient. A partial derivation tree pair for (1) is given in Figure 1, reproduced from Figure 10 of Shieber (1994).

(1) a. Le docteur lui soigne les dents.

b. The doctor treats his teeth.

The trees are clearly not isomorphic. If they represent a fixed relation—i.e. each node is always immediately dominated by its parent, with no possibility of intervening nodes—this could be handled by Shieber's suggestion of 'bounded subderivation', where the fixed relations are treated as single nodes. However,

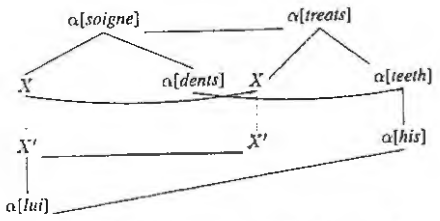


Figure 2: Unbounded relation, variant 1

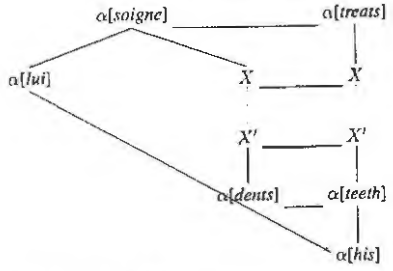


Figure 3: Unbounded relation, variant 2

Shieber also suggests that the "relation between the clitic and the NP which it is semantically related to seems to be potentially unbounded". In terms of tree relations, this suggests that there is unbounded material intervening in the trees between where $\alpha[lui]$ and $\alpha[his]$ attach, hence no possible isomorphism. Given the tree configuration of Figure 1, there are two possible cases where the relation between the trees is unbounded. The first is in Figure 2: the X and X' connected by vertical dots indicate the unbounded material. The derivation represented by Figure 2 is exemplified in (2). In this example, there is an unbounded number of verbs which can be adjoined into $\alpha[soigne]$; $\alpha[lui]$ is adjoined into the lowermost of these nodes (X'). However, expressions such as (2) are unattested in French, since the clitic must occur immediately before *soigner* (and auxiliaries).

(2) a. * Le docteur lui veut pouvoir ... soigner les dents.

b. The doctor wants to be able ... to treat his teeth.

The second possible case is illustrated by Figure 3. This derivation is exemplified by (3), which has an unbounded number of NPs between clitic and body part.

(3) a. * Le docteur lui regarde une copie d'une photo ... des dents.

b. The doctor is looking at a copy of a photo ... of his teeth.

These examples are also ungrammatical in French. Thus, neither possibility for establishing an unbounded relation applies, and hence, contra Shieber's footnote (and accepted folklore) they do not appear to be problematic for isomorphic S-TAG, although they do raise other problems (Abeillé, 1994).

2.2. *A Spanish Example*

Spanish, however, does allow clitic climbing over a potentially unbounded number of 'trigger' verbs (Aissen & Perlmutter 1976). The example in (4) parallels the French example in (2), with clitic *le*, but is acceptable.

(4) El médico le quería poder ... examinar los dientes.

In analysing clitic behaviour in (4), either syntax-dependent or syntax-independent analyses are possible. In a syntax-dependent analysis, there would be a coindexing (in the derived tree) between the clitic and its corresponding NP. In a syntax-independent analysis, the relationship would be handled by some other mechanism which remains to be specified. Our reconstruction of Shieber's analysis is syntax-independent, with $\alpha[lui]$ a single tree.

2.3. *A Metagrammar*

We propose to handle the unboundedness shown in (4), with its derivation tree pair in Figure 4, using a metagrammar (Dras, 1999a). A metagrammar specifies a relation between derivation trees by means of a TAG grammar of derivation structures. A minimal metagrammar for (4) is shown in Figure 5.

The pair $\mathfrak{A}$ does the essential grouping of the clitic and slot for a recursively-addable verb (the X to X' material), mapping to the English substructure. The unbounded intervening material is given by tree pair $\mathfrak{B}$, and clearly there is an isomorphism at the level of the derivation of the derivation (the 'meta-derivation'). This metagrammar is in Rogers' (1994) regular form (it is not possible to adjoin into the

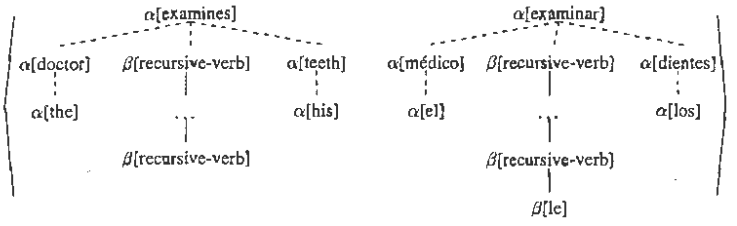


Figure 4: Derivation tree pair

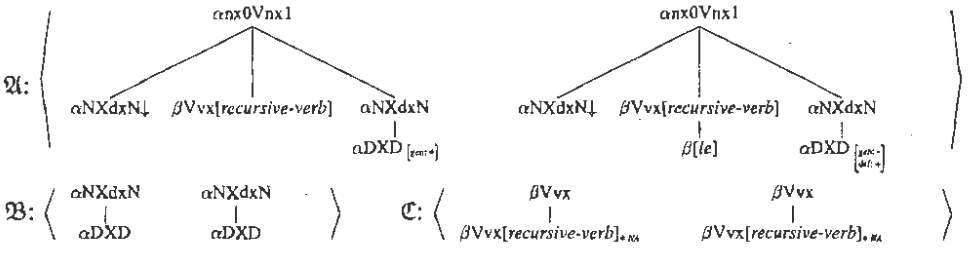


Figure 5: A metagrammar for Figure 4

spine of an auxiliary tree in this metagrammar) and so the results from Dras (1999a) apply: the WLPP holds, and the object-level formalisms still have TAG weak generative capacity.

Note that this analysis is compatible with the spirit of Abeillé (1994). There, the behaviour of the clitic is constrained by an S-TAG which pairs a syntactic and a semantic grammar. The S-TAG there is the earlier, non-isomorphic S-TAG of Shieber & Schabes (1990), so the precise analysis is not of use for investigating isomorphic S-TAG, and moreover its mathematical properties are not well understood. What we have done here, however, is compatible with Abeillé's syntax-semantics idea. There is a parallel between the English side of our grammar and the semantic side of Abeillé's grammar, with the metagrammar pairing the nodes in such a way that the clitic must be interpreted as an inalienable possessor.

3. An Alternative Analysis

Taking an individual Romance syntactic grammar by itself (that is, not constraining it through pairing with another grammar), the analysis above is insufficiently restrictive. For example, if there is a standard bridge verb tree adjoined, as in (5), there is nothing in this analysis preventing the clitic from climbing over

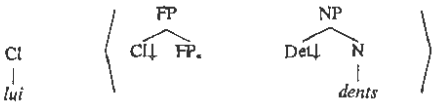


Figure 6: New clitic analysis

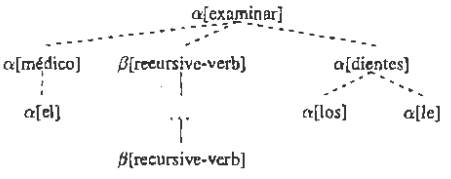


Figure 7: Reanalysed Spanish derivation tree

the bridge verb (*piensa*, thinks).

(5) * Juan le piensa que el médico examinó los dientes.

To account for Spanish clitic climbing, Bleam (1994) adopts a syntax-dependent analysis in which the coindexing between the clitic and the NP is represented by an MCTAG sequence. For us, the important aspect of this analysis is that the clitic is prevented from moving past particular constituents, such as negation and complementizers, and examples like (5) are not generated.

We analyze (4) using the tree sequence shown in Figure 6. The Spanish derivation tree is as

in Figure 7, with the English tree as before.[1,2]

In the Spanish tree, *los* and *le* are inserted into the tree sequence for *dientes*. A bounded relation between the English and Spanish trees is now induced, treating *los* and *le* as a bounded subderivation.[3]

4. Discussion

In the analysis presented in Section 3, the relation between the clitic and its associated NP is local, so we do not need to represent unbounded relations in a metagrammar. In addition, it not only rules out ungrammatical structures that our first approach does not, but also captures the intuition that the clitic is as much a part of the *dientes* structure as *his* is of *teeth*.

Both analyses discussed here draw attention to the fact that the 'unbounded' nature of constructions is not fixed. What constitutes an unbounded relation at the derivation level for a given object-level grammar becomes a bounded relation for a slightly different object-level grammar. To explain this, a notion of 'grouping' is useful here. Grouping is related to the concept of domain of locality: MCTAG group entities by associating trees together in multi-component tree sequences; a metagrammar groups elements by associating nodes in the derivation tree. So the role of grouping elements so that a relation between derivation trees is established can be traded off between the object-level grammar and a metagrammar. As an obvious rule of thumb, grouping should occur in the object-level grammar when justified by linguistic reasons, such as a preference for a syntax-dependent analysis of clitics; a metagrammar can group items that are related in some other way, such as if a syntax-independent (semantic) analysis of clitics were preferred, or in cases such as the structurally-rearranging paraphrases of Dras (1999a).

In sum, we have shown that problematic cases in S-TAG models of Romance-English translation can be resolved by using either a metagrammar or an MCTAG analysis of the clitic-body part relationship; and in doing so, we have demonstrated that unbounded relations between derivation trees in S-TAG are only an artefact of the analysis.

[1]Note that there are some changes in the Spanish derivation tree. There is a new location for the *le* node, and α[*dientes*] is a two-element sequence. In addition, the *examinar* tree is modified slightly as well, now including a functional projection (FP) node. This is necessary for two reasons: to prevent multiple adjunctions at the VP node (clitic and recursive verb); and to account for the effects discussed in Bleam (1994).

[2]Note that a synchronous relation between a TAG and an MCTAG is formally well-defined (Dras, 1999b), working in essentially the same manner as S-TAG, but pairing trees with sequences rather than with trees.

[3]Other alternative TAG-based analyses are possible here also (e.g. Abeillé, 1994; Kulick, 1998; Candito, 1999). However, we have chosen the analysis given here because, as we are examining the relation between English and Romance derivation trees, we would like to have 'minimal tree pairs', to concentrate on the one phenomenon of unboundedness; the other analyses give substantially different derivations.

References

1. Abeillé, A. 1994. Two cases of clitic-noun dependencies in French. *TAG+3*, Paris. To appear: Abeillé & Rambow, eds., *Tree Adj. Gram.: Formal, Computational, and Linguistic Aspects.*
2. Aissen, J. and D. Perlmutter. 1976. Clause Reduction in Spanish. *Proc. of the Second Annual Meeting of Berkeley Linguistics Society.*
3. Bleam, T. 1994. Clitic Climbing and The Power of Tree Adjoining Grammar. *TAG+3*, Paris. To appear: Abeillé & Rambow, eds., *Tree Adj. Gram.: Formal, Computational, and Linguistic Aspects.*
4. Candito, M-H. 1999. *Organisation modulaire et paramétrable de grammaires électroniques lexicalisées.* PhD thesis. Univ. of Paris 7, France.
5. Dras, M. 1999a. A Meta-Level Grammar: Redefining Synchronous TAG for Translation and Paraphrase. *ACL'99.*
6. Dras, M. 1999b. Synchronous Parallelisms Between Different Grammar Formalisms. *MoL6.*
7. Kulick, S. 1998. Clitic climbing in Romance: "Restructuring", causatives, and object-control verbs. *TAG+4*, 88–91.
8. Rogers, J. 1994. Capturing CFLs with Tree Adjoining Grammars. *ACL'94*, 155–162.
9. Shieber, S. & Y. Schabes. 1990. Synchronous Tree-Adjoining Grammars. *COLING-90*, 253–258.
10. Shieber, S. 1994. Restricting the Weak-Generative Capacity of Synchronous Tree-Adjoining Grammars. *Computational Intelligence*, 10(4), 371–386.

Reuse of Plan–Based Knowledge Sources in a Uniform TAG–based Generation System

Karin Harbusch & Jens Woch

Universität Koblenz–Landau; Abt. Koblenz
Rheinau 1, D–56075 Koblenz
e–mail: {harbusch, woch}@uni–koblenz.de

Abstract

In an uniform generation system all knowledge bases are specified in the same formalism and run the same processing component. The advantage of this behavior is that any order of applying the knowledge bases, i.e. a negotiation on revisions between the individual components, can easily be imposed on the system. Furthermore, the implementation of the overall system is simpler because only one algorithm must be developed and tested. In the project INTEGENINE *we specify all knowledge sources in the formalism of Schema–TAGs with Unification (SU–TAGs). A general paradigm of our work is to reuse existing knowledge bases, i.e. to transform various formats into a SU–TAG. For the syntactic and lexical knowledge the existing XTAG system has already automatically been transformed. In this paper we address the general question how to transform plan–based knowledge–sources which are frequently used in the what–to–say part of a generation system. As an instance of the general transformation model presented here, we show how to transform the knowledge sources of the plan–based system VOTE.*

The transformation component we describe in the following appertain to a *uniform generation system* based on *Schema–TAGs*. Let us first briefly address this system in order to motivate the serviceableness of the transformation component in the general system.

The idea of *uniform* or so called *integrated generation* was basically described in the system KAMP (Appelt, 1985). In this system a hierarchical action planner explores expressions based on the formalism of intensional modal logic. KAMP was not intended to be a psycholinguistic model of human behavior, although is reflects some aspects of human language production such as *incrementality*. This behavior directly results from the integrated model. Any knowledge base is supposed to become active at any time, i.e. as early as possible.

From this observation the question arises whether the uniform model can serve as a basis to remedy the generation gap (Meteer, 1990), i.e. the situation in which a sequential process (first what–to–say, then how–to–say) leads to dead end situations which cannot be solved by local modifications in the component in which the problem occurs. Our assumption is to extend the — in a sense demon–like — activation of knowledge bases towards a *parametrised model* which allows for recovery strategies to escape from local dead ends by imposing revisions of parameter–defined components. This means that parameters trigger the activation of specific knowledge bases and hence initiate overall revisions. Our claim is that this approach is able to build up any kind of communication model in a generation system.

As underlying formalism of our integrated generation model we have chosen *Schema–TAGs with Unification (SU–TAGs)*[1] because TAGs provide the necessary complexity to express any kind of concept in the what–to–say and how–to–say component (cf., e.g., (Stone & Doran, 1997), (Webber & Joshi, 1998), (Becker *et al.*, 1998), (Nicolov, 1998)). Schema–TAGs are especially

[1] In a *schematic elementary tree*, a *regular expression (RX)*, is annotated at each inner node of an elementary tree. This means, that the elementary schemata enumerate a possibly infinite set of elementary trees. RXs are inductively defined. Let α, β and $\beta_1, ..., \beta_n$ ($n \geq 2$) be Gorn addresses uniquely referring to daughters or arbitrarily complex RXs, then $\alpha.\beta$ (concatenation of branches), $(\beta_1 + ... + \beta_n)$ (enumeration of alternatives), α^* (Kleene Star) and α^+ (α^* without the empty repetition) are RXs. Finally, "–" allows to cut off a subtree at the end of a path. As an abbreviation we write $\alpha^{(n|m)}$ which enumerates $\sum_{i=0}^{m} \alpha^{n+i}$ ($n, m \geq 0$). Notice, "." binds stronger than "+". Notice also, that here the feature specifications are attached to the regular expression because the branches are licensed by RXs (cf. (Harbusch & Woch, 2000)).

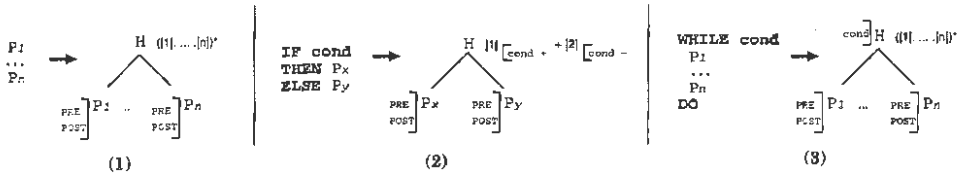


Figure 1: Transformation of sequences, alternatives, and repetitions into SU–TAGs

advantageous because they compress grammars in a manner that allows for the underspecified generation of substructures (cf. (Harbusch, 2000), (Harbusch & Woch, 2000)).
In order to provide a flexible generation system, the example domain is not of particular interest but only a necessary prerequisite of a demonstration system. On that account, we decide to *reuse* existing knowledge sources to circumvent the time–consuming task of developing a knowledge base from scratch. Thus, transformation algorithms for the individual knowledge bases of a generation system must be provided. Any TAG can automatically be transformed into a SU–TAG (Harbusch *et al.*, 1998). This is already done for the syntactic knowledge base XTAG (Doran *et al.*, 1994). The knowledge sources of SPUD (Stone & Doran, 1997), those of anchored L–TAGs (cf. (Webber & Joshi, 1998)), as well as the TAG transformed from an HPSG (Becker *et al.*, 1998) will be rewritten as SU–TAG next. Doing so, the generation system is extended towards a *generation workbench* which provides libraries with knowledge sources from which the user can select a personal generation system with self–defined parameters.
In this paper, we describe the transformation of *plan–based knowledge bases* into a SU–TAG. Here we only concentrate on the particular class of plans which is widely applied in what–to–say components of generation systems (e.g. VOTE (Slade, 1994)), i.e. the classical *plan–based plans* (cf. (Yang, 1997)). As an illustration a concrete plan of the system VOTE is transformed and the decision making on the basis of VOTE's further knowledge bases is presented. Finally, the interaction of plans with the system's knowledge about the domain — also specified as SU–TAG — is outlined in order to demonstrate basically, how the uniform generation works.

A *plan*[2] consists of n steps, any of them in turn may be an *action* or a plan again. Each step consists of *pre–* and *postconditions*, as well as controlling elements of a programming language (e.g. `IF-THEN-ELSE`, `WHILE`). A plan can be applied iff the overall *goal*, i.e. the input specification, matches the preconditions of the first step. A plan step can be applied iff the *current situation*, i.e. the postconditions of the previous step, or the input specifications respectively, match the preconditions of the currently considered plan step. If a plan step is atomic, i.e. an action, it is *performed* by replacing the preconditions with the postconditions, resulting in the new current situation. An overall plan can *successfully be applied in the current situation* iff the final postconditions can be computed according to the overall goal and the initial situation. Given that, the general idea of the transformation into a SU–TAG is as follows:

1. Each plan step in a sequence becomes an individual node of an elementary scheme under a common root node.
2. The chronological sequence of plan steps is rewritten via concatenation in the RX.
3. Pre– and postconditions at each node are wrapped up in feature specifications.
4. The conditions of concepts of the programming language are realized by unification too, whilst the branches and repetitions itself are transformed into RXs.

In the transformation of Fig. 1-1, the first three steps are illustrated. Each plan step P_1, ..., P_n is transformed into a daughter node. The regular expression at the root node enumerates the concatenation of all daughters from left to right and all pre– and postconditions are rewritten as feature specifications. Step 4 is illustrated by two example statements in Fig. 1-2 and 1-3. Basically, the conditions in the statements are checked by a feature "cond". For instance,

[2]For an illustration of a plan, see the strategy for decision making in VOTE (abstracting from technical notations, cf. (Slade, 1994), p. 140) on the left side of Fig. 2.

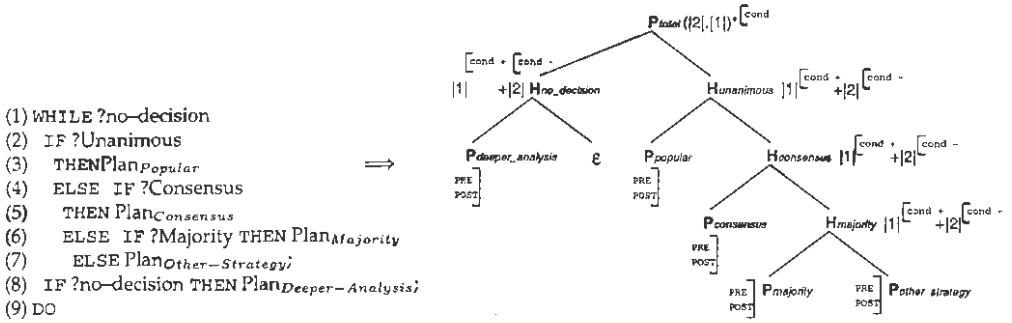


Figure 2: Transformation of the VOTE–Plan

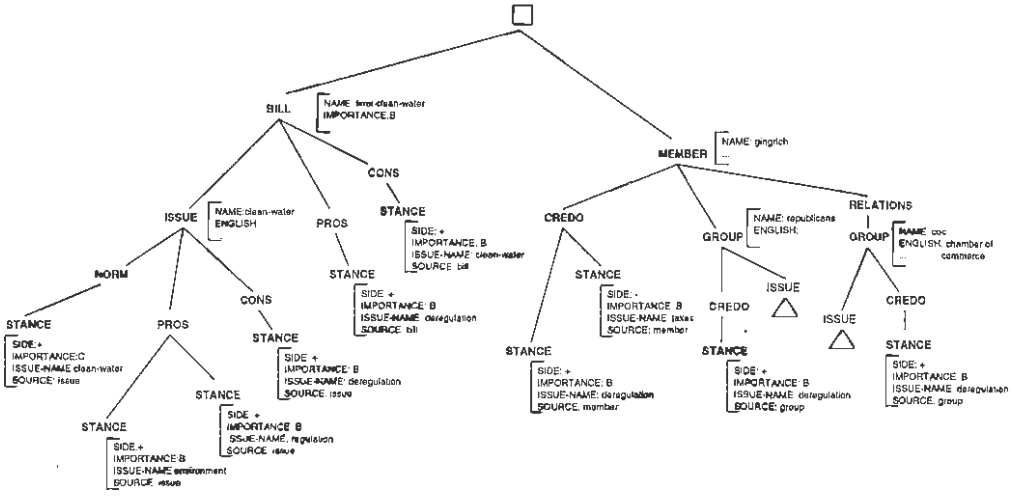


Figure 3: Part of VOTE's knowledge for MEMBER:gingrich and BILL:limit–clean–water

in the IF-THEN-ELSE statement a positive value for "cond" activates the THEN part and a negative value the ELSE part. The behavior of the steps 1, 2 and 4 is exemplified in Fig. 2. This plan describes the decision making process in the system VOTE. The actual knowledge in the pre– and postconditions is suppressed here for reasons of simplicity. As outlined in step 4, IF-THEN-ELSE statements are rewritten as sums, (i.e. representing the choice of one of the branches according to the instantiation of the condition) and the WHILE construction is rewritten as Kleene Star which stops according to the instantiation of the respective condition represented as feature specification. At the root node the concatenation represents the sequence of the two IF-THEN-ELSE statements in line (2) and (8) (step 1 and 3 result in |2|.|1| according to the order of branches). Here, the Kleene Star in the RX rewrites line (1).

Now we explain how pre– and postconditions are specified and tested in order to apply plans in this particular example. For this reason we must describe VOTE's further knowledge bases in more detail: VOTE consists of ISSUES (e.g. *gun control*), STANCES (PRO, CON, normal case), GROUPS (e.g. *ACLU*), RELATIONSHIPS, MEMBERS, BILLS and STRATEGIES. Fig. 3 shows the structure of what VOTE knows about a concrete BILL:limit–clean–water and the attitude of a concrete MEMBER:gingrich towards this bill. Let us presuppose here that the structures described in Fig. 3 can be produced by SU–TAG structures of the form outlined in Fig. 4. This is directly obvious because all mother–daughter relations in the instantiation are represented as elementary schemata. Furthermore, any scheme licenses the specification of any number of such relations by Kleene Star. In any plan of VOTE the pre– and postconditions are yet specified by unification about STANCEs of bills and members. For instance in Plan$_{popular}$, PRE = {Unify

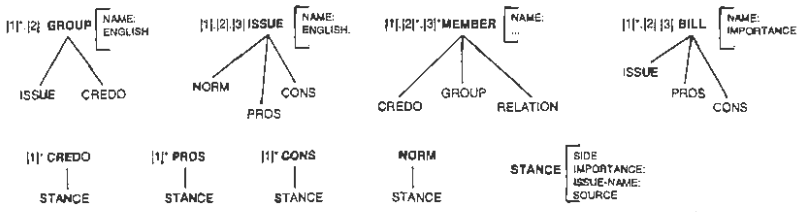


Figure 4: SU–TAG representation of the knowledge base classes BILL and MEMBER

all SIDE features of all STANCEs}, i.e. test whether they have the same value and POST = {Unify the feature DECISION at the root node with the SIDE feature of the uppermost STANCE of BILL}, i.e. vote in a popular manner. Hence, in the uniform framework the application of a plan imposes further constraints on the knowledge about bills and members. The reasoning about plans and domain knowledge is performed integratedly in a unform manner.
In general, pre– and postconditions can also be specified in first order predicate logic. Let us consider the STRIPS example in (Yang, 1997) p. 17. For instance, the plan return–brush with PRE = {have–brush(?b)} and POST = {¬ have–brush(?b)} is rewritten by the feature specifications PRE = {((b have–brush) +)} and POST = {((b have–brush) –)}. For reasons of space we cannot go into more details here (cf. (Otto *et al.*, 1998)).
The implementation of the above described transformation component has just begun using general parser generator concepts in the same way as for the TAG–to–STAG transformation.

References

A. ABEILLÉ, T. BECKER, O. RAMBOW, G. SATTA & K. VIJAY-SHANKER, Eds. (1998). *Procs. of the 4th International TAG Workshop*, IRCS-Report 98–12.

APPELT D. E. (1985). *Planning English Sentences*. Cambridge University Press.

BECKER T., FINKLER W., KILGER A. & POLLER P. (1998). An efficient kernel for multilingual generation in speech–to–speech dialogue translation. In (Isabelle, 1998), p. 110–116.

DORAN C., EGEDI D., HOCKEY B. A., SRINIVAS B. & ZAIDEL M. (1994). XTAG system — a wide coverage grammar for english. In M. NAGAO, Ed., *Procs of the 15th COLING*, Kyoto, Japan.

HARBUSCH K. (2000). Natural-language processing with Schema–TAGs. In A. ABEILLÉ & O. RAMBOW, Eds., *Tree Adjoining Grammars: Formal Properties, Linguistic Theory and Applications*. CLSI Lecture Notes.

HARBUSCH K., WIDMANN F. & WOCH J. (1998). Towards a workbench for Schema-TAGs. In (Abeillé *et al.*, 1998), p. 56–61. IRCS-Report 98–12.

HARBUSCH K. & WOCH J. (2000). Direct parsing of Schema–TAGs. In H. C. BUNT, Ed., *Procs. of the Sixth International Workshop on Parsing Technologies (IWPT)*, p. 305–306, Trento, Italy.

P. ISABELLE, Ed. (1998). *Procs. of the 36th ACL and 17th COLING*. Université de Montréal, Canada, Morgan Kaufmann.

METEER M. W. (1990). *The Generation Gap – The Problem of Expressibility in Text-Planning*. PhD thesis, University of Massachusetts, Amherst, MA, USA.

NICOLOV N. (1998). Memoisation in sentence generation with lexicalized grammars. In (Abeillé *et al.*, 1998), p. 124–127. IRCS-Report 98–12.

OTTO F., NARENDRAN P. & DOUGHERTY D. J. (1998). Equational unification, word unification, and 2nd-order equational unification. *Theoretical Computer Science*, 198 (1–2 p.), 1–47.

SLADE S. (1994). *Goal–Based Decision Making: An Interpersonal Model*. Hillsdale, NJ, USA: Lawrence Erlbaum Associates Inc.

STONE M. & DORAN C. (1997). Sentence planning as description using Tree Adjoining Grammar. In *Procs. of the 35th ACL and 8th EACL*, Madrid, Spain.

WEBBER B. L. & JOSHI A. K. (1998). Anchoring a Lexicalized Tree–Adjoining Grammar for discourse. In (Isabelle, 1998).

YANG Q. (1997). *Intelligent Planning*. Berlin, Germany: Springer-Verlag.

CDL–TAGs: A grammar formalism for flexible and efficient syntactic generation

Anne Kilger and Peter Poller

DFKI GmbH, Stuhlsatzenhausweg 3
D-66123 Saarbrücken, Germany

Abstract

During the last decade we developed and continuously improved CDL–TAGs, *an extension of* TAGs *for incremental syntactic generation. This paper presents the current state of development and gives details of the definition of context dependent linearization rules.*

1. Introduction

This paper presents *Tree Adjoining Grammars with Context-Dependent Disjunctive Linearization Rules (CDL–TAG)* that have been developed for incremental syntactic generation in the system WIP (WAF+93). CDL–TAGs were successfully used in the projects PERFECTION (Fin96), EFFENDI (PH 96), PRACMA (JKN+94), and VERBMOBIL (Wah93).

A fully incremental system is characterized by realizing interleaved input consumption, processing and output production, so that first output elements may even be produced before the input is complete. So, decisions based on the data at hand impose assumptions about the outstanding input, thereby reducing the set of input increments that can be consistently integrated into processing. For syntactic generation, two different processing levels can be distinguished. First, for each new element the hierarchical structure of the sentence under construction has to be expanded. Second, elements have to be positioned in the final utterance thereby constraining any further positioning. According to that, it is essential to choose a syntactic representation formalism that facilitates the dynamic construction of the hierarchical structure and the stepwise linearization and utterance production for its substructures. The grammar formalism must be flexible enough to preserve word order variations as long as possible during generation. Thereby, it should be easy to handle the prefix of the sentence already uttered as constraining the set of applicable linearization rules. Additionally, the grammar formalism should support linearization rules that describe situational factors (e.g., time or space restrictions).

The separation of a grammar into **H**ierarchical and **P**ositional constraints (in the following called *H/P paradigm*) fulfills these requirements. Such a grammar (e.g., LD/LP–TAGs (Jos87)) consists of two distinct sets of rules, one merely describing mother–daughter relations only hierarchically, while the other describes positional constraints by referring to elements of the hierarchical structures.

This paper presents *CDL–TAGs*, that almost perfectly reflect the required different levels of processing for incremental syntactic generation and thereby strongly facilitate the implementation of the incrementality effects on syntactic generation (FS92).

2. Definition of CDL–TAGs

TAG with Context–Dependent Disjunctive Linearization Rules (CDL–TAG) is an extension of Tree Adjoining Grammar (JLT75) that helps to design an extremely compact grammar by avoiding redundant descriptions without extending the power of the formalism.

2.1. *The Standard TAG Formalism*

Standard TAG combines elementary (initial and auxiliary) trees by *adjoining*, an operation which makes the grammar mildly context–sensitive and adequate for the representation of natural language. The TAG formalism has been extended by a second combination operation called *substitution* which has only context–free power (SAJ88). In order to allow compact representations of complex syntactic dependencies, TAG has been extended furthermore by feature structures (*TAGs with unification*, (Kil92), (Kil94), or *Feature Structure Based TAG*, (VJ88)).
The H/P–paradigm was applied to TAG by (Jos87). He defined *Local Dominance/Linear Precedence–TAG (LD/LP–TAG)* by "taking the elementary trees as domination structures over which linear precedences can be defined." The descriptive power of LP–rules in LD/LP–TAGs is not sufficient to describe all linearization alternatives of one hierarchical structure locally, i.e., without duplicating the hierarchical structure (e.g., for German verbal phrases subject–verb–object, object–verb–subject, ...). Furthermore, there is no means to associate different LP–rules with contextual (semantic and pragmatic) constraints. To get more flexible linearization, we developed a new extension of TAG on the basis of LD/LP–TAGs.

2.2. *CDL–TAG*

CDL–TAG is defined according to the H/P paradigm, i.e., domination structures are used as elementary structures instead of trees. The possible orderings of sister nodes are restricted by *linearization rules* which are associated with the mother node. They have the form: "(<" {"("context lin-rule* ")"}* ")". The rules are initiated by the key "<". Each alternative starts with the name of a *context* in which the rule is valid. The value of *context* is matched with a feature *lin-context* of the feature structure associated with the respective node.
The left part of Figure 1 illustrates a VP–node whose subtree represents a German verbal phrase.

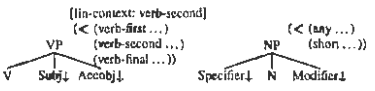


Figure 1: Examples for German Linearization Rules

Its linearization rules include statements about verb–first, verb–second and verb–final word order while `verb-second' is the actual `lin-context' inside its feature structure. Other contexts (like `any' or `short' at the NP–node in the right part of the figure) distinguish word order rules that differ with respect to their suitability for specific situational — non–syntactic — factors which is useful for a generation system with globally set `parameters'. E.g., the value `short' [1] is used for word orders that permit to save space and time in the final utterance.
Each $lin - rule$ is encoded as a list that contains linearization elements $lin - el$. The order of the list elements defines the order of the elements of the TAG tree they refer to. A symbol sym is a $lin - el$ and refers to a daughter of the node the linearization rule is associated with.
In order to describe constraints on sister nodes which include complements as well as optional elements, we extended the formalism by a combination operation that allows to add sister nodes without introducing additional depth into the tree. The operation of *furcation* has been defined by (DK88) as the unification of two root nodes of structures to one root node with two substructures. We adapted it to CDL–TAGs by defining an new kind of elementary tree, namely a *furcation auxiliary tree* whose foot node is leftmost or rightmost daughter of the root node, as a structure leaving away the foot node[2].

[1] The key `short' is meant in the sense of saving space and time in the final utterance when using this alternative. This may be meaningful under time pressure or when the space for the written text is restricted.

[2] This is comparable to the *modifier auxiliary tree* in contrast to the *predicative auxiliary tree* introduced by

Some symbols sym in $lin - rule$ denote adjuncts. They have to be detailed enough to express all aspects that influence word positions. For English, e.g., different adverb classes have to be defined according to their different linearization constraints. Symbols referring to adjuncts always appear inside disjunctions with `regular–like' expressions. They permit to describe exactly one occurrence of one element of a list by $((sym_1...sym_n)^1)$, one or zero occurrences by $((sym_1...sym_n)^{1/0})$, at least one occurrence of elements by $((sym_1...sym_n)^+)$ or an arbitrary (or zero) number of elements by $((sym_1...sym_n)^*)$.
The following expression is a possible linearization rule of the VP–node of Figure 1:

```
(verb-second
   (subj v ...(advp)* ...accobj ...)        ((advp)1 v subj ...(advp)* ...accobj ...)
...)
```

It shows two alternatives to fill the first position of a verbal phrase in the linearization context *verb–second*, namely a complement (`subj' refers to the subject), or exactly one optional element (`advp' refers to an optional adverbial phrase). After the first element, the finite part of the inflected verb (referred to by `v' in the linearization rule) has to follow. The second expression prescribes that the subject directly follows the verb in case of a topicalized adverbial phrase.
Furthermore, the selection of adequate linearization rules may be restricted by features of the subtree to be linearized. CDL–TAGs use *child-info* that is inherited from the daughters of the node the linearization rule is associated with. The resulting structure for LP–rules is "(<" {"("context child-info lin-rule* ")"}* ")". The entry *child–info* realizes a specific test (identified by the key `test') for feature–value–combinations which have to hold for some of the daughters of the actual node. The LP–rule

```
(short (test (mod (cat) name))
      (...mod ...(adjp)* ...n ...)
...)
```

might be associated with the NP–node on the right in Figure 1. It describes a possible linearization of a Specifier–Noun–Modifier construction in German: Instead of "Die Werke Goethes" (the works of Goethe) it is also possible to say "Goethes Werke" (Goethe's works). The presupposition for choosing this `brief' linearization alternative is that the modifier is realized as a proper name which is tested by referring to the third daughter of NP (the Modifier↓ node, referred to in the test above by `mod') and then checking the equality of feature–value of `cat' and the atomic value `name'.
The generative power of CDL–TAGs is equivalent to standard TAG (with constraints) because the only addition to standard TAG is the combination operation "furcation" which has only context–free power. So, CDL–TAGs are not sufficient to describe all linearization phenomena that include adjuncts. E.g., there is no easy way to describe scrambling without mixing hierarchical and positional information. Nevertheless, we use it as a promising starting point, concentrating on its usefulness for (incremental) syntactic generation.

3. Conclusions and Future Work

In this paper we presented CDL–TAGs, a highly compact grammar formalism, that is especially well–suited for the representation of grammar sources for (incremental) natural language generation. Furthermore, the lexicalization allows the grammar to consider a subset of word class specific elementary trees (tree families) for each lexical entry.
The TAG–GEN generator (Kil94) makes use of the CDL–rules by preferring linearization alternatives that reflect the order of input elements so that the output can start as early as possible,

(SS92). In this sense, furcation auxiliary trees are the CDL–TAG variant of sister adjunction in, e.g., DTG (RVW95) and furcation in, e.g., TFG (Cav98).

e.g., by fronting elements which are given early in the input. It also sorts linearization alternatives according to some generation parameters such as time pressure and style.
Although the formalism has been successfully used in several different application systems, there is no grammar developing tool yet. So, the most important task for future work is the development and implementation of a CDL–TAG parser, e.g., as an extension of the work described in (Pol94).

References

M. Cavazza. An integrated parser for TFG wich Explicit Tree Typing In *TAG+4'98*, Institute for Research in Cognitive Science (IRCS), University of Pennsylvania, Philadelphia, PA, 1998.

K. De Smedt and G. Kempen. The representation of grammatical knowledge in a model for incremental sentence generation. In *INLG'88*, Santa Catalina Island, CA, 1988.

W. Finkler. Automatische Selbstkorrektur bei der inkrementellen Generierung gesprochener Sprache unter Realzeitbedingungen. Dissertation. Universität des Saarlandes, Saarbrücken, 1996.

W. Finkler and A. Schauder. Effects of Incremental Output on Incremental Natural Language Generation. In B. Neumann, editor, *ECAI'92*, p. 505–507, Vienna, Austria, August 1992.

A. Jameson, B. Kipper, A. Ndiaye, B. Schäfer, J. Simons, T. Weis, D. Zimmermann. Cooperating to be Noncooperative: The Dialog System PRACMA In B. Nebel and L. Dreschler-Fischer (Ed.), *Proceedings of the 18th German Annual Conference on Artificial Intelligence : KI-94: Advances in Artificial Intelligence*, Saarbrücken, 1994. LNAI 861. Springer, Berlin (1994). 106-117.

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

A. K. Joshi. Word-order variation in natural language generation. In *AAAI'87*, p. 550–555, Seattle, USA, 1987.

A. Kilger. Realization of tree adjoining grammars with unification. DFKI Technical Memo TM-92–08, German Research Center for Artificial Intelligence – DFKI GmbH, 1992.

A. Kilger. Using utags for incremental and parallel generation. *Computational Intelligence*, 10(4):591–603, November 1994.

P. Poller. Incremental parsing with LD/TLP-TAGs. *Computational Intelligence*, 10(4):549–562, November.

P. Poller and P. Heisterkamp. EFFENDI – Effizientes Formulieren von Dialogbeiträgen – Bericht zum Projektende – Handbuch zur SIL–Schnittstelle. Technischer Bericht Nr. F3–96–014, Daimler–Benz Forschung und Technik, Ulm, 1996.

O. Rambow, K. Vijay–Shanker, and D. Weir D–Tree Grammars Proceeding of *ACL'95*, MIT, Cambridge, MA, 1995.

Y. Schabes, A. Abeillé, and A. K. Joshi. Parsing strategies with lexicalized grammars: Application to tree adjoining grammar. In *COLING'88*, p. 578–583, Budapest, Hungary, 1988.

Y. Schabes and S. M. Shieber. An alternative conception of tree–adjoining derivation. In *ACL'92*, p. 167–176, Newark, DW, 1992.

K. Vijay-Shanker and A.K. Joshi. Feature Structure Based Tree Adjoining Grammars. In *COLING'88*, Budapest, Hungary, 1988.

W. Wahlster, E. André, W. Finkler, H.-J. Profitlich, and T. Rist. Plan-based integration of natural language and graphics generation. *Artificial Intelligence*, 63:387–427, 1993.

W. Wahlster. Verbmobil: Translation of face–to–face dialogs. Research Report RR-93-34, DFKI GmbH, Saarbrücken, FRG, 1993.

Predicative LTAG grammars for Term Analysis

Patrice Lopez* and David Roussel†

*DFKI GmbH
Stuhlsatzenhausweg 3
D-66123 Saarbrücken, Germany
lopez@dfki.de

†Aerospatiale Matra
Centre Commun de Recherches
Suresnes, France
David.Roussel@aeromatra.com

Abstract

In a restricted domain and task, we propose that the elementary tree backbones represent statically the predicative level and the possible distribution of arguments while the syntactic categories and constraints would be only processed dynamically by the way of features. The resulting grammar can be viewed as an intermediate level between the surface syntax of a sentence and its conceptual representation. In addition to possible speed efficiency and robustness relevance, an interesting property is that such a grammar could be tested in a straightforward way to integrate contraints provided by additional trees and to inject progressively semantic and pragmatic constraints during the analysis.

1. Introduction

Considering applications such as spoken annotation of elements into a specific virtual environment, the most important task is first to identify referred objects or terms among several speech hypotheses. Given these expectations, how can be used the assets of a LTAG grammar with robustness? To address this question, we propose a Feature-Based LTAG grammar focusing on the semantic and predicative level while the pure syntactic processing is achieved by the two-step unification mechanism. Before introducing this predicative LTAG grammar, we define the applicative framework.

2. From Terms extraction to spoken annotations

The research project under consideration is based on a virtual platform (which represents an architecture of aeronautical components and a terminological model obtained from technical documents (example : cautions to set on the manipulation of components).

Let us clarify that first the virtual platform (i.e. a 3D scene) is used as an interface between the desing and assembly tasks. The aim of this interface is to let people easily move in a complex architecture, to display or mask related annotations, and to gather vocal synthetic annotations that overlap one or several elements of a scene (example : recommendations for people of a related trade).

Secondly, the terminology of the technical documents is ideally subjected to editorial constraints and is getting close to a controlled language. A terms extraction and clusterization based on statistical criteria supply classes of elements. Then, an expert is efficient to grab the terms in a knowledge base containing ontological and conceptual relationships. Tools of the market are helpful for these tasks (Fig. 1, Fig. 3). The aim of this step is twofold:

- Build a model used to check the cohesion from various technical documents or versions.
- Identify the stable terms and build up various terminological resources (authoring memory, multilingual thesaurus needed for automatic language processing). For example, the

knowledge base designed by the experts is used to categorize various technical documents within an Information Retrieval System. For the spoken annotation purpose, we derived constraints from the knowledge base in order to restrict the combination between technical properties (ex: float valve, needle valve), functionalities (ex: drain valve, directional valve) and the system in which a unit is used (ex: water valve, bleed valve). By this way, terms like *water drain valve*, *electrical drain valve* are well recognized, but some other complex terms are rejected.

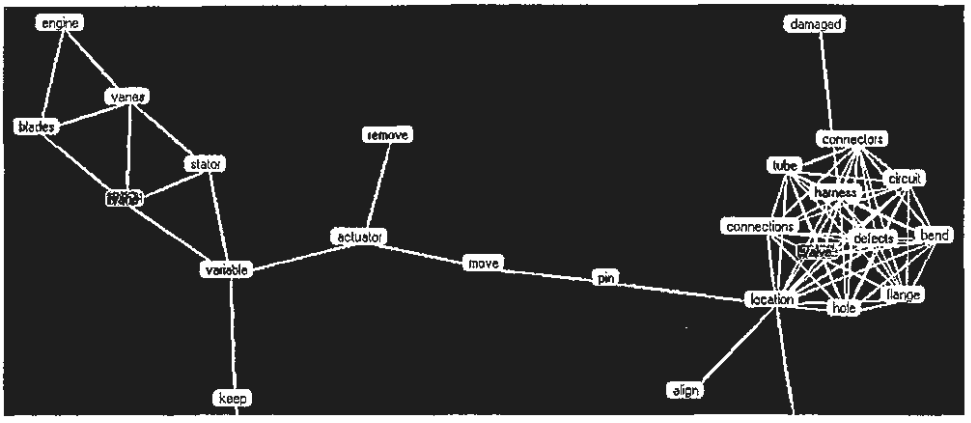


Figure 1: Cluster of words computed for technical documentation extracts. Note that the word *valve* covers at least three notions expressed in French by the terms *valve*, *soupape* and *vanne*

<C.2616> MAKE SURE YOU WILL NOT CAUSE UNWANTED CHANGES TO OTHER SYSTEMS BEFORE YOU PUSH THE ENG 1 (2, 3 OR 4). WHEN YOU PUSH THE ENG 1 (2, 3 OR 4) FIRE PUSHBUTTON SWITCH THESE VALVES CLOSE: THE LP FUEL VALVE. THE HYDRAULIC FIRE VALVE. THE BLEED AIR VALVES. THE ANTI-ICE VALVES. THE AIR CONDITIONING PACK VALVES.

Figure 2: Example of caution integrated in a structured technical documentation.

Taking advantage of lexical resources obtained from technical procedures called "warnings and cautions" (see Fig. 3), the MRTERESA project (Multilocutor speech Recognition, TERms Extraction and Spoken Annotation) consists in the customization of a speech recognizer for vocal annotations, a robust term analysis of speech recognition hypotheses and vocal annotations indexing with regard to components existing in a virtual scene. If necessary, the indexing has to be confirmed by the users. The robust terms analysis relies upon:

- A mapping between lexicalized elementary trees and technical terms. Some category labels in these trees are semantic types that belong to an ontology.
- A representation of the terms variability in the spoken annotations thanks to the TAG substitution and adjonction operations. This variability results from spatial relations between the displayed objects and the spontaneity of the verbalizations.
- Semantic labels compatibility constraints for modification and dependency relations
- If necessary, syntactic constraints are applied to filter out speech recognition hypotheses.

3. Syntactic vs. semantic Dependencies

The *semantic head* is the lexical unit that represents the semantic type of the interpretation of a given phrase structure. We consider that the *syntactic head* is the lexical unit that constraints the

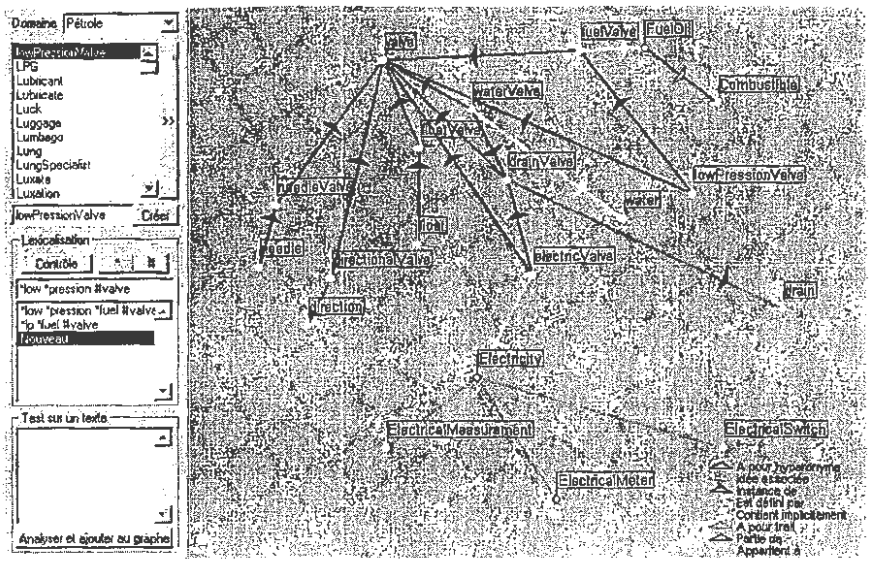


Figure 3: Example of knowledge base from few valves achieved with a tool of the market

morphosyntactic and mode features of the phrase structure it belongs to. The LTAG formalism is well suited to localize semantic dependencies, but is limited to represent syntactic dependencies for very frequent phenomena as object extraction with auxiliary.

When we use the term *localizing semantic dependencies* for a LTAG grammar, we suppose that the elementary trees have been designed properly to capture this kind of dependencies, i.e. that the elementary trees respect the Predicate-Argument (PA) and Semantic Consistency (SC) principles introduced in (Abeillé, 1991). These principles stipulate that a lexicalized elementary tree corresponds to an unique semantic unit (*semanteme*) and that we have a terminal node (substitution or foot node) per argument expected by the corresponding semanteme. In our approach we systemize the localization of semantic dependencies: we drop out from the elementary tree backbones all the aspects which traditionally refer to syntactic categories and replace them dynamically with semantic types.

4. A new definition for the elementary trees

The first point is to capture in an elementary tree a particular word distribution and the corresponding predictive structure under the form of semantic dependencies. Closely to the solution proposed in (Abeillé, 1992) for the representation of this level, we use the following *predicative categories* as node labels of elementary trees:

- Formula (**F**) or proposition representing the association of a relation and its arguments.
- Term (**T**) which corresponds to the non-relational semantic heads.
- Relation (**R**).
- Property (**P**).
- Null (**N**): used for semantically empty nodes (in general preterminal nodes of co-anchors, semantically empty prepositions or auxiliaries).

Top and bottom features are added on this backbone in order to check syntactical constraints at the end of the parsing. The figure 4 gives examples of Feature-Based predicative LTAG elementary trees. During the lexicalization process, semantic types are added to the LTAG tree backbone according to the semanteme that the elementary tree represents and an ontol-

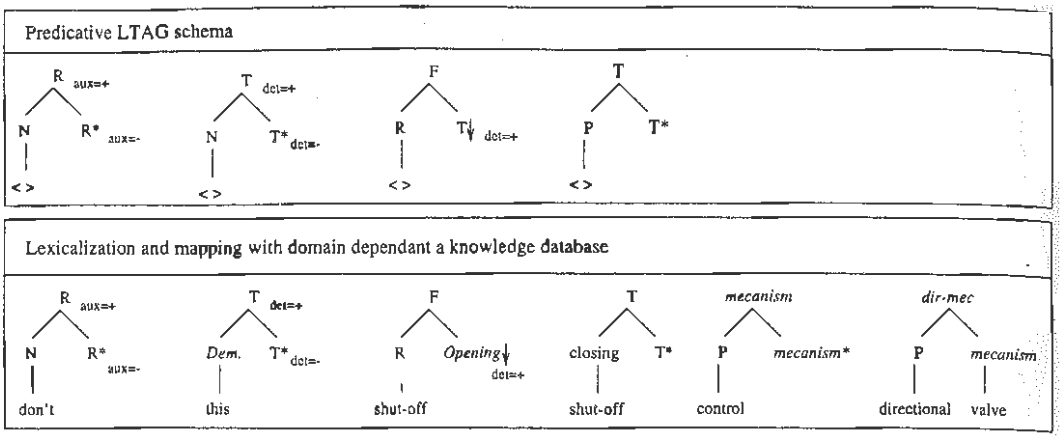


Figure 4: Examples of predicative LTAG elementary trees and their lexicalization

ogy obtained as explained in section 2. This ontology controls the adjunction and substitution operations between the semantic categories.
On the contrary to classical LTAG, the semantic basis for post-parsing processing is here the derived tree and not the derivation tree. For complex cases, semantic features may control the derivation with specific mecanisms as suggested in (Roussel, 1999).

5. Related works and conclusion

Previous works have shown that focusing parsing first on semantics can lead to superior speed efficiency than syntax-first approach, particularly on restricted domain as shown in (Lytinen, 1991), but also for large coverage grammar (Dowding *et al.*, 1994). The trees currently developped for our application and their lexicalization are closed from the semantic grammars paradigm (Seneff, 1992) and works on terminological variability (Jacquemin, 1999). We expect that such a LTAG grammar will allow, in our application, a stronger and an easier integration of different level of constraints. In terms of reusability, the same linguistic representation (the predicative LTAG grammar) could be mapped into concepts of various restricted domains with a domain-dependent semantic module.

References

ABEILLÉ A. (1991). *Une grammaire lexicalisée d'arbres adjoints pour le français.* PhD thesis, Paris 7.

ABEILLÉ A. (1992). Synchronous TAGs and French Pronominal Clitics. In *COLING*, Nantes, France.

DOWDING J., MOORE R., ANDRY F. & MORAN D. (1994). Interleaving syntax and semantics in an efficient bottom-up parser. In *ACL'94.*

JACQUEMIN C. (1999). Syntagmatic and paradigmatic representations of term variation. In *ACL'99, University of Maryland.*

LYTINEN S. (1991). Semantic-first natural language processing. In *AAAI'91, Anaheim, CA.*

ROUSSEL D. (1999). *Intégration de prédictions linguistiques issues d'applications a partir d'une grammaire d'arbres hors contexte. Contribution a l'analyse de la parole.* PhD thesis, Joseph Fourier University, Grenoble.

SENEFF S. (1992). Tina: A natural language system for spoken language applications. *Computational Linguistics*, **18** (1 p.), 61–86.

Reliability in Example-Based Parsing

Oliver Streiter

Academia Sinica, Institute of Information Science, Nankang, Taipei, Taiwan 115
oliver@hp.iis.sinica.edu.tw http://rockey.iis.sinica.edu.tw

Abstract

In this paper we introduce an example-based parser for Chinese. One strong point of the parsers is its high reliability. We propose a formal definition for reliability and derive from it K as a metric for the evaluation of parsers. In a row of experiments we try to identify some factors which support the reliability of the parser. It is suggested that these factors are independent of the parsing approach and can be realized in TAGs.

1. Introduction

Example-based parsers adhere to the *lazy learning algorithm* while converting tree-bank entries into a parser. So-called *treebank grammars*, (Bod, 1992; Charniak, 1996) are *eager learners*, i.e. they abstract knowledge structures or statistical information from the treebank and reason on the basis of these abstractions. *Explanation-based parsing* is a different eager learning approach aiming at the extraction of specialized grammars out of a general-purpose grammars on the bases of parsing examples (Rayner & Christer, 1994; Srivinas & Joshi, 1995).

Lazy learners keep all training data (e.g. all trees in the treebank) available in their original form. They may operate on similar abstractions as *eager learners* do, e.g. parse from partial trees with category labels, but dispose in addition of the original encoding which can be referred to if generalizations become ambiguous (Daelemans *et al.*, 1999). The learning set is not filtered or modified and contains among regular phenomena redundancies, syntactic and semantic exceptions, phraseologies including lexical functions (Mel'čuk, 1974), pronouns with their antecedents, markers of text-coherence (e.g. *fire, cigarette, match*), and pieces of common sense knowledge (*he sees the sparrow with the spyglass*), all pieces of information which are necessary, or at least helpful for high-quality parsing (Doi & Maraki, 1992; Bod, 1999).

All words and categories are of equal importance to the parser unless special weights are assigned to them. It might be argued that this equal distribution of weights is not sense-less and that, for example, the linguistic notion of head as pivot should and can be dispensed with. Giving preference to specific matches (e.g. verbs) might produce a bias which endangers the reliability, i.e. a good match is not chosen, just because another match contains more verbs. Linguistic support may come from observations in verb-last languages where speakers are contradicted/approved before the final main verb has been pronounced. The list of actants, circumstances, lexical functions as magnifiers etc are often sufficient in order to identify the verb or its syntactic or semantic type.

2. An Example-Based Parser

An example-based parser is currently developed at the Academia Sinica of Taiwan (Streiter, 1999; Streiter & Hsueh, 2000), based on a Chinese treebank of about 30.000 trees (Chen *et al.*, 1999). The annotation scheme comprises 200 lexical labels, 45 phrasal categories and 46 semantic roles. The parser retrieves trees from a treebank via a fuzzy match of the sentence to be

parsed and the terminals of the all trees in the treebank. The 20 best matching trees are further processed and aligned with the sentence in case the tree is smaller than the sentence. The best aligned tree is selected. Mainly through re-parsing awkward subtrees, badly matched trees are corrected and unmatched words are inserted. The parser is fast and by means of the fuzzy match extremely robust. The complexity of other parsing approaches is avoided, as parsing consists mainly of retrieving large chunks from a databank. The coverage, as evaluated in (Streiter & Chen, 2000) is not yet fully satisfying. Unchallenged, however, is the reliability of this parser.

3. What Reliability is about

Reliability is an important evaluation criterion for NLP which until now has failed to obtain a formal definition as well the attention it merits. The standard evaluation tests a parser on unlearned corpora, determining its coverage in terms of recall and precision. The pendant of the coverage is the *reliability*, which we define as a system property, i.e. as *performance on trained corpora*. Reliability is thus close the notion of *tunability*. However, the impact of reliability is more fare-reaching: A system which has a high reliability can always enlarge its coverage by learning new items. A system with low reliability cannot improve its coverage by learning new items: the system is quickly over-trained.

We define coverage ($\mathcal{C}$) and reliability ($\mathcal{R}$) as meta-scores which elaborate the values of recall and precision. As $\mathcal{R}$ is neither compatible with low precision (false alarm) nor with low recall (a silent system), we define $\mathcal{R}$ and $\mathcal{C}$ as f-score with learned respectively unlearned test corpora. With this definitions we formulate the *hypothesis of converging $\mathcal{C}$ and $\mathcal{R}$*: **1)** $\mathcal{R}$ is always higher than $\mathcal{C}$. **2)** $\mathcal{R}$ decreases with more training data (due to ambiguities which arise). **3)** $\mathcal{C}$ approaches $\mathcal{R}$ with more training data (more items are known or similar to known items). **4)** Before $\mathcal{C}$ and $\mathcal{R}$ converge $\mathcal{C}$ may decreases under the influence of decreasing $\mathcal{R}$.

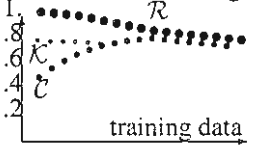


Figure 1: The hypothesis of converging $\mathcal{C}$ and $\mathcal{R}$ for hypothetical data. $\mathcal{K}$ is the estimated maximal coverage.

While most experimental data available support an asymptotic rise of $\mathcal{C}$, little is known about $\mathcal{R}$. Given the above (hypothetic) distribution, the maximal coverage a system can achieve as well as its current position are important data. We propose to estimated the *maximal coverage* $\mathcal{K}$ as $\mathcal{C} + \frac{(\mathcal{R}^2-\sqrt{\mathcal{C}})\cdot\sqrt{\mathcal{C}}}{(1-\mathcal{R}^2)+\sqrt{\mathcal{C}}}$. With $\mathcal{K} > \mathcal{C}$ further investment in more teaching is profitable, otherwise system properties have to be changed in order to enforce $\mathcal{R}$ and with it future grow.

4. Factors determining Reliability

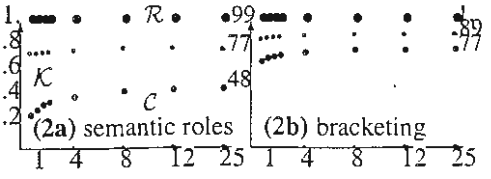
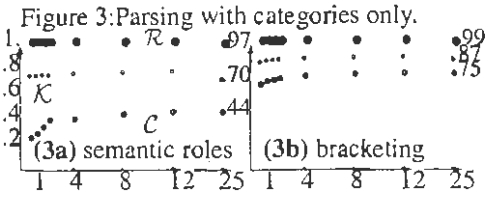


Figure 2: **Reference Data**. $\mathcal{C}$ and $\mathcal{R}$ for 500 to 25.000 training sentences, evaluating the recognition of semantic relations (agent, theme, goal, experiencer, time etc) between head and dependents (2a) and the bracketing (2b) for the example-based parser on a test-corpus of 727 sentences.

Experiment 1 In order to establish the effect of the string and lexeme encoding in addition to the category encoding we removed the string and lexeme encoding as done in all eager learners.

Figure 3:Parsing with categories only.

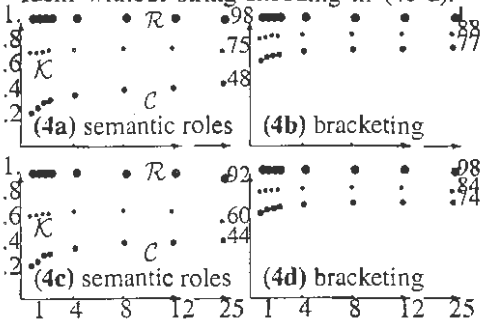


F.(3) shows a loss is $\mathcal{R}$, $\mathcal{C}$ and $\mathcal{K}$ compared to F.(2). We assume that the drop in $\mathcal{C}$ has been produced by the drop in $\mathcal{R}$, as unknown items are treated only in reference to learned items. It is the ambiguity in the learned items (the inverse of $\mathcal{R}$) which causes the drop of $\mathcal{C}$.

Experiment 2 In order to establish the effect of the context sensitivity we not only re-parsed awkward subtrees (see our description of the parser above), but re-parsed (artificially) all subtrees, thus breaking the links between sisters.

We observe a small loss of $\mathcal{R}$ compared to F.(2). If we test a context-free version with category encoding only (4c-d) and thus simulate standard parsing approaches, we observe an additional drop of $\mathcal{R}$ compared to F.(3). Thus context sensitivity is important for $\mathcal{R}$ but to a smaller extend than the encoding of lexemes and strings. Without string encoding the context-free grammar loses heavily in its $\mathcal{R}$. The drop of $\mathcal{K}$ shows that the loss cannot be compensated for by more training data.

Figure 4: Context-free parsing in (4a-b). Idem without string-encoding in (4c-d).



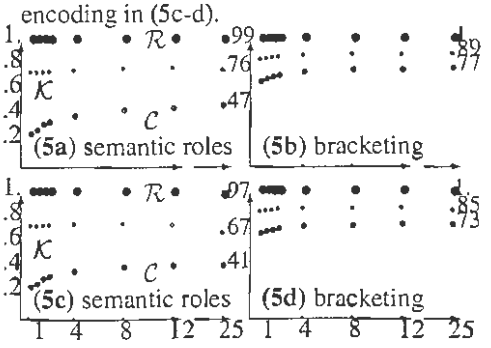
Experiment 3 To test the equal distribution of weights, we assigned 0.5 points for a matching verb, in addition to the 1 point for every match, assuming that in most cases the verb functions as head and a matching head is more important than a matching non-head.

F.(5) shows a small loss of $\mathcal{C}$ for semantic roles compared to F.(2). With category encoding only, we observe a drop of $\mathcal{C}$ for the bracketing compared to (3). The drop in the bracketing supports our claim that the bias is towards matching deeper branching structures by preference. This bias is unlikely to be produced by the specific additional score 0.5 we assigned:

score	+0	+0.25	+0.5	+1
K	.8660	.8585	.8548	.8393
Figure	3b		5d	

Figure 6: $\mathcal{K}$ for bracketing with additional scores to verbs when parsing without string-encoding (25.000 training sentences).

Figure 5:Additional scores to verb matches in (5a-b). Idem without string-encoding in (5c-d).



5. Summary

We have introduced, although shortly, an example-based parser. A formal grammar which bears most resemblance to this approach is TAG. Both approaches are based on collections of trees, atomic trees for TAGs and all trees and subtrees for example-based grammars. Parsing starts similarly by extracting trees via the indices formed by words. A distinguishing property of example-based grammars is that a tree preserves all terminal nodes per tree. The influence

of this strategy could not be tested, as this would require to leave the paradigm suggested. However, we could evaluate the effect of not necessarily distinguishing features, (i.e. the string-lemma encoding, the high degree of context-sensitivity and the non-preference of heads.)
The experiments have been preceded by a discussion of the notions of $\mathcal{R}$ and $\mathcal{C}$, for which a formal definition has been proposed. $\mathcal{K}$ has been proposed as evaluation measure which is less dependent on the size of the training corpus than $\mathcal{R}$ and $\mathcal{C}$ are.
In the experiments we could show that the string-lemma encoding is of utmost importance for $\mathcal{R}$ and $\mathcal{C}$, even though a very rich set of categories is employed. When the string encoding is renounced to, the grammar becomes more dependent on other features, such as a high degree of context-sensitivity and the correct assignment of weights.
The dominant role the head plays in formal grammars has been questioned as it has no priority in parsing relevant dimensions such as world knowledge, text coherence and idiomaticity.
Throughout 14 meaningful comparisons of test settings we observe 12 cases in which $\mathcal{R}$ and $\mathcal{C}$ decrease both. In two instance $\mathcal{C}$ improved with $\mathcal{R}$ remaining equal or decreasing, thus supporting our claim of a causal relation between declining $\mathcal{R}$ and declining $\mathcal{C}$.

6. Conclusion

Example-based grammars base their $\mathcal{R}$ mainly on the string-encoding. We hypothesize that TAGs with multiple terminals and a string-lemma encoding, if still be called TAG, could handle NLP task more reliable. In order to achieve this, automatic learning experiments should apply, unlike past experiments (Srivinas & Joshi, 1995; Xia, 1999), lazy learning approaches.

References

BOD R. (1992). Data oriented parsing (DOP). In *COLING*.

BOD R. (1999). Extracting stochastic grammars from treebanks. In *Journées ATALA sur les Corpus annotés pour la syntaxe*: Talana, Paris VII.

CHARNIAK E. (1996). Tree-bank grammars. In *13th National Conference on Artificial Intelligence*.

CHEN K.-J. ET AL. (1999). The CKIP Chinese Treebank. In *Journées ATALA sur les Corpus annotés pour la syntaxe*: Talana, Paris VII.

DAELEMANS W., BUCHHOLZ S. & VEENSTRA J. (1999). Memory-based shallow parsing. In *Proceedings of CoNLL-99*, Bergen, Norway. //ilk.kub.nl/papers.html.

DOI S. & MARAKI K. (1992). Translation ambiguity resolution based on text corpora of source and target language. In *COLING'92*.

MEL'ČUK I. A. (1974). *Opyt teorii lingvističeskix modelej Smysl⇔Tekst. Semantika, sintaksis*. Moskva

RAYNER M. & CHRISTER S. (1994). Corpus-based grammar specification for fast analysis. In A. ET AL., Ed., *Spoken Language Translator: First Year Report*, SRI Technical Report CRC-043.

SRIVINAS B. & JOSHI A. K. (1995). Some novel applications of explanation-based learning to parsing lexicalized tree-adjoining grammars. cmp-lg archive 9505023.

STREITER O. (1999). Parsing Chinese with randomly generalized examples. In *NLPRS'99 Workshop on Multi-lingual Information Processing and Asian Language Processing*, Beijing.

STREITER O. & CHEN K.-J. (2000). Experiments in example-based parsing. In *Dialogue 2000, International Seminar in Computational Linguistics and Applications*, Tarusa, Russia.

STREITER O. & HSUEH P.-Y. (2000). A case-study on example-based parsing. In *International Conference on Chinese Language Computing 2000*, Chicago.

XIA F. (1999). Extracting TAGs from bracketed corpora. In *Proceedings NLPRS'99*. Beijing.

LFG-DOT: a Probabilistic, Constraint-Based Model for Machine Translation

Andy Way

School of Computer Applications,
Dublin City University,
Dnblin 9, Ireland.

Abstract

We develop novel models for Machine Translation (MT) based on Data-Oriented Parsing (DOP: Bod, 1995; 1998) allied to the syntactic representations of Lexical Functional Grammar (LFG: Kaplan & Bresnan, 1982).

Introduction

It is accepted that the main paradigmatic approaches to MT—transfer, interlingua, and statistical—do not at present produce the quality of translation reqnired. There have, however, been a number of attempts at combining elements of these different approaches in an attenipt to increase overall translation performance (cf. Carbonnel *et al.*, 1992; Grishman & Kosaka, 1992). Onr efforts to bring abont a better solution to the problems of MT can be viewed in this new hybrid spirit.

DOP has produced interesting results for a range of NLP problems. DOP language models consider past experiences of language to be significant in both perception and prodnction. DOP prefers performance models over competence grammars: models based on large collections of previously occnrring fragments of language are preferred to abstract grammar rules. New language fragments are handled with respect to existing fragments from the corpus, which are combined using statistical techniques to determine the most probable analysis for the new fragment.

DOP Translation Models

DOP has been used already as a basis for MT–Data-Oriented Translation (DOT: Poutsma, 1998). DOP models typically use surface PS-trees as the chosen representation for strings. The DOT translation model relates tree-fragments between two (or more) languages with an accompanying probability, linking source-target translations at all possible nodes in accordance with the principle of Compositionality of Meaning. Once the most likely parse of the source language sentence has been produced, the tree structure of the target is assembled, from which the string is (trivially) derived. Nevertheless, there are usually many different derivations for the source sentence, so many different translations may be available. As is the case when DOP is used monolingually, Poutsma shows that the most probable translation can be computed using Monte-Carlo disambiguation.

DOT is an interesting model, but it is not guaranteed to produce the correct translation when this is non-compositional and considerably less probable than the default, compositional alternative. An example is *commit suicide* $\longleftrightarrow$ *se suicider*, where *John commits suicide* is wrongly translated by DOT as **John commet le suicide*. DOT's adherence to left-most substitution in the target given *a priori* left-most substitution in

the source is too strictly linked to the linear order of words. As soon as this deviates to any significant degree between languages, DOT has a significant bias in favour of the incorrect translation (assuming the corpus to be representative). Another example is the *like* $\longleftrightarrow$ *plaire* case, where the arguments need to be 'switched' between English and French. Even if the correct, non-compositional translation is achievable, DOT derives other wrong alternatives with higher probabilities. In such cases, the correct translation will be dismissed, unless all possible translations are inspected manually.

This is not at all surprising: being based on STSG, DOT is necessarily limited to those contextual dependencies actually occurring in the corpus, a reflection of surface phenomena only. It is well known that models based solely on CFGs are insufficiently powerful to deal with all natural language problems. In this regard, DOP models have been augmented (van den Berg *et al.*, 1994; Tugwell 1995) to deal with richer representations, but such models have remained context-free.

LFG, however, is known to be beyond context-free. It can capture and provide representations of linguistic phenomena other than those occurring at surface structure. Given this, the functional structures of LFG have been harnessed to the techniques of DOP to create a new model, LFG-DOP (Bod & Kaplan, 1998). LFG-DOP permits (via the *Discard* operator) the relaxation of certain constraints on LFG representations, thereby creating generalised fragments against which new input can be compared, and the best analysis constructed.

LFG-DOP Translation Models

We propose that LFG-DOP has the potential to be used as the basis for an innovative MT model, LFG-DOT. We have designed two LFG-DOT models:

1. a simple, linear model which builds a target f-structure from a source c-structure and f-structure, the mapping between them ϕ, and the τ-equations. This model leaves the task of generating the target string from the target f-structure to the standard LFG generation algorithms (e.g. Wedekind, 1988);

2. a more complex model, containing explicit links between both surface constituents and f-structure units in both languages, unlike the previous model which relates the languages just at the level of f-structure (via τ).

Probability models have been constructed for both translation models, and small experiments have been performed for particular cases of 'hard' translation problems. Being able to link exactly those source-target elements which are translations of each other using LFG's τ-equations, LFG-DOT overcomes some of the problems specific to the DOT system. For example, the LFG-MT solution to the *like* $\longleftrightarrow$ *plaire* case is (1):

(1) *like*:
$(\tau\uparrow \text{PRED FN}) = \text{plaire}$
$\tau(\uparrow \text{SUBJ}) = (\tau\uparrow \text{OBL})$
$\tau(\uparrow \text{OBJ}) = (\tau\uparrow \text{SUBJ})$

That is, the subject of *like* is translated as the oblique argument of *plaire*, while the object of *like* is translated as the subject of *plaire*. The solution to the *commit suicide* $\longleftrightarrow$ *se suicider* problem is (2):

(2) *commit:*
$(\tau\uparrow$ PRED FN) = se suicider
$\tau(\uparrow$ SUBJ) = $(\tau\uparrow$ SUBJ)
$(\uparrow$ OBJ PRED) $=_c$ suicide

Where the PRED value of the OBJ of *commit* is constrained ($=_c$) to *suicide*, then the collocational units '*commit + suicide*' are translated as a whole to *se suicider*. DOP's statistical model gives a 'level of correctness' figure to alternative translations. This is useful in cases like these where the default translation in LFG-MT (and in many other systems) cannot be suppressed when the specific translation is required. We have conducted small experiments which show that for a treebank constructed from 10 sentences, despite 7 instances of *commit* $\longleftrightarrow$ *commettre* compared to just one *commits suicide* $\longleftrightarrow$ *se suicide* example, the correct translation *Marie commits suicide* $\longleftrightarrow$ *Marie se suicide* is preferred by both LFG-DOT models over the wrong, compositional alternative by a factor of between 3 and 6 times, depending on which LFG-DOP definition of competition set is selected.

Furthermore, LFG-DOT promises to improve upon the correspondence-based LFG-MT model (Kaplan *et al.*, 1989), particularly where robustness is concerned, as LFG-DOP's *Discard* function enables both nnseen and ill-formed input to he dealt with. For example, Bod & Kaplan (1998) show that given a treebank for the sentences *People walked* and *John fell*, probability models can be constructed where for the 'unseen' sentences *John walked* and *People fell*, the nnmarked interpretation is less likely that the two specific interpretations, and of these the intnitively correct ones are selected for each corresponding verb.

Problems and Future Work

The major problem with any models based on LFG-DOP is the explosion of fragments caused by *Discard*. Allowing *Discard* to operate in the unconstrained manner of Bod & Kaplan's (1998) model results in an exponential number of fragments in which the non-*Discard* fragments are overwhelmed, resulting in the probabilities of derivations via *Root* and *Frontier* being vastly outnumbered by the 'ungrammatical' alternatives. While there is a large increase in the number of fragments produced via *Discard* in LFG-DOT models, compared to the monolingnal LFG-DOP corpora from which they are derived, the explosion of fragments is nowhere near as severe. Notwithstanding this, we propose to restrict the scope of the *Discard* operator by creating two different bags of fragments: the well-formed ones (derived via *Root* and *Frontier*) and the *Discard* ones. Using Good-Tnring (cf. Bod, 2000), we can allocate a fixed, *small* probability mass to the fragments generated by *Discard* to ensure that the derivations using the 'good' non-*Discard* fragments will still be favoured.

Using different LFG-DOP probability models (in terms of which LFG grammaticality checks are enforced, and at which points in the translation process) results in different probabilities with respect to the corpus, bnt does not result in different rankings of alternative candidate translations. A potential problem, however, is that LFG-DOT models, like DOT models, show a tendency to exclude many potentially useful fragments owing to the strictness of Poutsma's (1998) definition of linked fragments. This may resnlt in translations which are theoretically describable not being achievable in practice. Only experimentation on a mnch wider scale will confirm this.

Given the small corpora from which onr findings were derived, any resnlts must be treated with some equivocatiou. Given the (relative) scarcity of some of the linguistic

examples cited previously, and the subject of the tests thereon, we regret that it is nigh on impossible to derive 'representative' corpora for the examples in hand. The absence of large-scale LFG-DOP corpora currently prohibits these models from being tested more widely. Nevertheless, recent work on automatic construction of the LFG-DOP corpora (Van Genabith *et al.*, 1999; Sadler *et al.*, 2000) needed for further experimentation using these techniques seems promising in this regard.

References

VAN DEN BERG M., BOD R. & SCHA R. (1994). A Corpus-Based Approach to Semantic Interpretation. In *9th Amsterdam Colloquium*.

BOD R. (1995). *Enriching Linguistics with Statistics: Performance Models of Natural Language*. PhD thesis, University of Amsterdam.

BOD R. (1998). *Beyond Grammar: An Experience-Based Theory of Language*. Stanford, California: CSLI Publications.

BOD, R. (2000). An Empirical Evaluation of LFG-DOP. In *Proceedings of the 19th International Conference on Computational Linguistics* (to appear).

BOD R. & KAPLAN R. (1998). A Probabilistic Corpus-Driven Model for Lexical-Functional Analysis. In *Proceedings of the 17th International Conference on Computational Linguistics & 36th Conference of the Association for Computational Linguistics*, p. 145–151.

CARBONELL J., MITAMURA T. & NYBERG 3RD E. (1992). The KANT Perspective: A Critique of Pure Transfer (and Pure Interlingua, Pure Statistics,...). In *4th International Conference on Theoretical and Methodological Issues in Machine Translation*, p. 225–235.

GRISHMAN R. & KOSAKA M. (1992). Combining Rationalist and Empiricist Approaches to MT. In *4th International Conference on Theoretical and Methodological Issues in Machine Translation*, p. 263–274.

KAPLAN R. & BRESNAN J. (1982). *Lexical Functional Grammar: A Formal System for Grammatical for Grammatical Representation*, In *The Mental Representation of Grammatical Relations*, chapter 4. MIT Press.

POUTSMA A. (1998). Data-Oriented Translation. In *Ninth Conference of Computational Linguistics In the Netherlands*.

R.KAPLAN, NETTER K., WEDEKIND J. & ZAENEN A. (1989). Translation by Structural Correspondences. In *Fourth Conference of the European Chapter of the Association for Computational Linguistics*, p. 272–281.

SADLER L., VAN GENABITH J. & WAY A. (2000). Automatic f-structure annotation of CFGs extracted from treebank resources. In *Proceedings of LFG-2000* (to appear).

TUGWELL D. (1995). A State-Transition grammar for Data-Oriented Parsing. In *Seventh European Conference on Computational Linguistics*, p. 272–277.

VAN GENABITH J., WAY A. & SADLER L. (1999). Semi-Automatic Generation of F-Structures from Treebanks. In *Proceedings of LFG-99*.

WEDEKIND J. (1988). Generation as structure driven derivation. In *12th International Conference on Computational Linguistics*, p. 732–737.

Comparing and Integrating Tree Adjoining Grammars

Fei Xia, Martha Palmer

Department of Computer and Information Science
University of Pennsylvania
Philadelphia PA 19104, USA
{fxia,mpalmer}@linc.cis.upenn.edu

Abstract

Grammars are core elements of many NLP applications. Grammars can be developed in two ways: built by hand or extracted from corpora. In this paper, we compare a hand-crafted grammar with a Treebank grammar. We contend that recognizing substructures of the grammars' basic units is necessary not only because it allows grammars to be compared at a higher level, but also because it provides the building blocks for consistent and efficient integration of the grammars.

1. Introduction

A Lexicalized Tree Adjoining Grammar (LTAG) is a core element of many NLP applications. It often has hundreds of elementary trees (*etrees*), which can either be built by hand (hand-crafted grammars), or extracted from annotated corpora (Treebank grammars). Hand-crafted grammars have rich representations (such as feature structures), and tend to be more precise, but they take a long time to build and their coverage on naturally-occurring data is hard to determine. In additiou, they lack statistical information which is crucial for statistical parsers. Treebank grammars, on the other hand, require little hnman effort (Xia, 1999; Chen & Vijay-Shanker, 2000) to build, once the Treebank has been created. They have rich statistical information and will cover at least the corpora from which the grammars are extracted. However, Treebank grammars are noise-prone because of annotation errors in the corpora and they also lack features and semantic information which are rarely represented in the corpora. It would be ideal if we could combine the strengths of both types of grammar. As a first step towards addressing this issue, in this paper we compare a hand-crafted grammar with a Treebank grammar and propose a way of integrating them to produce new grammars.

2. Two grammars

The two LTAGs that we compare are the XTAG English grammar (XTAG-Group, 1995) and a grammar extracted from Penn English Treebank. The XTAG grammar has 1004 tree templates.[1] The Treebank grammar that we use in this paper is extracted from the Penn English Treebank II (Marcus *et al.*, 1994) using the extraction algorithm described in (Xia, 1999). The extracted grammar has 3072 templates.

For lack of space, we will not describe the extraction algorithm, other than pointing out that by design all the *etrees* extracted from the Treebank fall into one of three types according to the relations between the anchor of the *etree* and other nodes in the tree, as shown in Figure 1. Figure 2 shows a bracketed sentence from the Penn Treebank. From that sentence, five *etrees* are extracted by the algorithm, as shown in Figure 3.

[1] If we remove the anchor(s) from *etrees*, we get *tree* templates. Each template indicates where the anchor(s) of that *etree* will be instantiated.

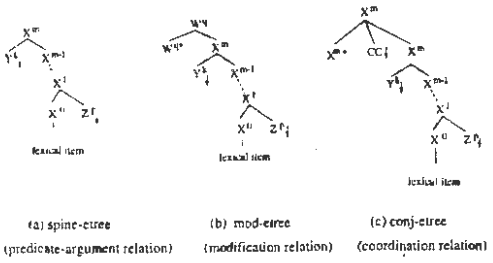


Figure 1: Forms of extracted *etrees*

```
(SBAR (WHNP-1 (WP who) )
      (S (NP-SBJ (-NONE- *T*-1) )
         (VP (VBD worried)
            (PP-CLR (IN about)
                    (NP (DT the)
                        (NN flood) ))))))
```

Figure 2: An example from the Treebank

3. Comparing two grammars

To compare the grammars, we need to find out how many trees in one grammar *match* trees in the other grammar. We define two types of matching : t-match and c-match. From now on, we use XTAG and Ext-G to stand for the XTAG grammar and the extracted grammar respectively.

3.1. t-match

We call two trees *t-match* (*t* for *tree*) if they are identical barring the type of information present only in one grammar, such as feature structures and subscripts[2] in XTAG and frequency information in Ext-G. In Figure 4, XTAG tree 4(a) and 4(b) t-match Ext-G tree 4(c).

XTAG also differs from Ext-G in that XTAG includes multi-anchor trees to handle idioms (Figure 5(a)), light verbs (Figure 5(b)) and so on. In each of these cases,

[2]The subscripts on the nodes mark the same semantic arguments in related subcategorization frames.

Figure 3: The extracted *Etrees*

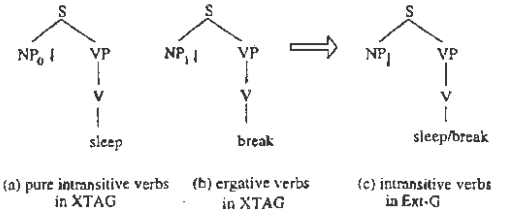


Figure 4: An example of *t-match*

the multi-anchors form the predicate. These trees are the same as the spine-etree in Figure 1(a) except that some nodes of the XTAG trees (e.g. NP_1 in Figure 5(a) and its counterpart Z_p in Figure 1) are expanded. By having multi-anchors, each tree can be associated with semantic representations directly (as shown in in Figure 5), which is an advantage of LTAG formalism. Ext-G does not have multi-anchor trees because semantics is not marked in the Treebank and consequentially the extraction algorithm can not distinguish idiomatic meanings from literal meanings. Two trees are called *t-match without expansions* if they *t-match* after the expanded part is removed from the XTAG trees. Figure 5 is such an example.

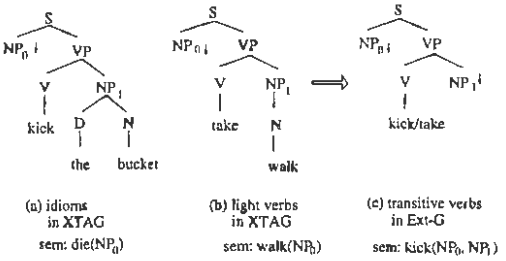


Figure 5: *t-match w/o expansion*

3.2. c-match

t-match requires two trees to have exactly the same structure, therefore, it does not tolerate minor differences between the trees. For instance, in XTAG, relative pronouns such as *which* and the complementizer *that* occupy distinct positions in the *etree* for relative clauses, whereas the Penn Treebank treats both as pronouns and therefore they occupy the same position in Ext-G, as shown in Figure 6. Because the circled

subtrees will occur in every tree for relative clauses and wh-movement, all these trees will not *t-match* their counterparts in the other grammar. Nevertheless, the two trees share the same subcategorization frame *(NP V NP)*, the same subcategorization chain[3] $S \rightarrow V\ P \rightarrow V$ and the same modification pair (NP, S). To capture this kind of similarity, we decompose a mod-etree into a tuple of (subcat frame, subcat chain, modification pair). Similarly, a spine-etree is decomposed into a (subcat frame, subcat chain) pair, and a conj-etree into (subcat frame, subcat chain, coordination sequence). Two *etrees* are said to *c-match* (*c* for *component*) if they are decomposed into the same tuples. According to this definition, the two trees in Figure 6 *c-match*.

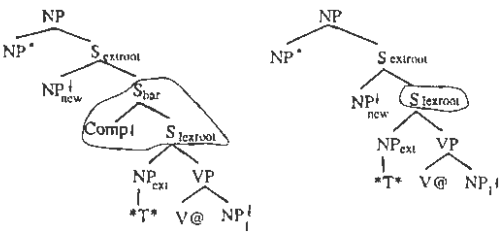


Figure 6: Relative clause trees

3.3. Comparison results

So far, we have defined several types of matching. Table 1 lists the numbers of tree templates[4] in one grammar that match some tree templates in the other grammar.[5] The last row lists the frequencies of the matched Ext-G templates. For instance, the fourth column says 496 templates in XTAG *match* 189 templates in Ext-G, and these 189 templates account for 57.1% of the template tokens in the Penn Treebank. If we decompose templates into components as mentioned in Section 3.2, the components that are shared by both grammars will cover 82.9% of all the component occurrences, as shown in Table 2. Templates in Ext-G are missing from the XTAG grammar for one or more of the following reasons:

T1: incorrect templates in Ext-G These templates result from Treebank annotation errors. Our extraction algorithm has a filter that detects implausible templates in Ext-G by decomposing a template into parts and checking each part against several small hand-crafted tables. The filter marks 2299 templates in Ext-G as implausible and they account for 5.2% of the template tokens in the Treebank.

T2: conj-etrees in XTAG Most conj-etrees in XTAG are generated on-the-fly while parsing (Sarkar & Joshi, 1996), and are not part of the 1004 templates. Therefore, many of the *conj-etrees* in Ext-G, which account for 2.8% of the template tokens in the Treebank, do not match any templates in XTAG.

T3: different analyses XTAG and Ext-G often choose different analyses for the same phenomenon. For example, the two grammars treat reduced relative clauses differently. [6]

T4: missing constructions in XTAG Some constructions such as the unlike coordination phrase (UCP) in the Treebank are not covered in XTAG.[7]

[3] A *subcategorization chain* is a subsequence of the spine in a spine-etree where each node on the chain is a parent of some argument(s) in the subcategorization frame. The nodes on a subcategorization chain roughly correspond to various lexical projections in GB-theory.

[4] We compare tree templates, not trees, in the two grammars because we are focusing on general syntactic structure.

[5] If a template in one grammar matches several templates in the other grammar and the match types are different, we label it with the strongest match type.

[6] Also, in XTAG, adjectives and nouns directly modify nouns, whereas in Ext-G, they modify noun phrases. These two pairs – (N, NP) and (A, NP) – account for 26.6% of the modification pairs in the Treebank, explaining XTAG's lack of coverage (53.1%) of the modification pair occurrences in the Treebank.

[7] The difference between matched templates (58.0%) and matched components (82.9%) imply that some combinations of components are missing from XTAG, The problem is very common for hand-crafted grammars because the the redundancy among trees in the grammar makes it very hard

	t-match	t-match w/o expansion	e-match	subtotal	conj-etree templates	no-match	total
XTAG	73	107	316	496(49.4%)	39	469	1004
Ext-G	59	5	125	189(6.15%)	411	2472	3072
frequency	53.9%	0.5%	2.7%	57.1%	2.8%	40.1%	100%

Table 1: Numbers of templates that match and their frequencies

	subcat chains	subcat frames	modification pairs	coordination pairs	total
in XTAG	44	115	72	25	256
in Ext-G	471	507	309	53	1340
matched types	35	45	31	10	121
matched tokens	977,218	954,776	357,563	22,937	2,312,494
frequency	93.7%	91.6%	53.1%	77.7%	82.9%

Table 2: Numbers of components in the two grammars

3.4. Integrating the two grammars

Simply taking the union of the two template sets will only yield a more noisy and inconsistent grammar. Our method has several steps: First, starting from Table 2, nse the plausibility filter to automatically rule out all of the implausible components in XTAG and Ext-G, then integrate the remaining plausible components into a new set, one for each type of component (such as subcat frames, subcat chains, etc.). Next, generate a new grammar from the component sets using various grammar development tools such as Metarules(Becker, 1994) or LexOrg(Xia *et al.*, 1998). The new grammar will be of high quality and have good coverage of the Treebank.

4. Conclusion

In this paper, we compare the XTAG grammar with the Penn Treebank grammar and propose a way of integrating them in order to derive a new grammar which has the strength of both. We believe that recognizing components of elementary trees in the two grammars is necessary because it not only allows the grammars to be compared at a more fine-grained level, but also provides the building blocks for integrating the grammars in a consistent and efficient way.

to maintain the grammar by hand. Various tools to semi-automatically generate templates (Becker, 1994; Candito, 1996; Xia *et al.*, 1998) could alleviate the problem.

References

BECKER T. (1994). Patterns in metarules. In *Proceedings of the 3rd International Workshop on TAG and Related Frameworks(TAG+3)*, Paris, France.

CANDITO M.-H. (1996). A principle-based hierarchical representation of ltags. In *Proceedings of COLING-96*, Copenhagen, Denmark.

CHEN J. & VIJAY-SHANKER K. (2000). Automated extraction of tags from the penn treebank. In *6th International Workshop on Parsing Technologies (IWPT 2000), Italy.*

MARCUS M., KIM G., MARCINKIEWICZ M. A. *et al.* (1994). The Penn Treebank: annotating predicate argument structure. In *Proc of ARPA speech and Natural language workshop.*

SARKAR A. & JOSHI A. (1996). Coordination in Tree Adjoining Grammars: Formalization and Implementation. In *Proceedings of the 18th COLING*, Copenhagen, Denmark.

XIA F. (1999). Extracting tree adjoining grammars from bracketed corpora. In *Proc. of NLPRS-99*, Beijing, China.

XIA F., PALMER M., VIJAY-SHANKER K. & ROSENZWEIG J. (1998). Consistent Grammer Development Using Partial-tree Descriptions for Lexicalized Tree-Adjoining Grammar. In *Proc. of tag+4.*

XTAG-GROUP T. (1995). *A Lexicalized Tree Adjoining Grammar for English.* Technical Report IRCS 95-03, University of Pennsylvania.