Workshop Organisers

John Carroll  
University of Sussex, UK

Nelleke Oostdijk  
University of Nijmegen, The Netherlands

Richard Sutcliffe  
University of Limerick, Ireland

Workshop Programme Committee

Rens Bod  
Universities of Leeds and Amsterdam

Ted Briscoe  
University of Cambridge

John Carroll  
University of Sussex

Anette Frank  
DFKI, Saarbrücken

Gregory Grefenstette  
Clairvoyance, Pittsburgh

Claire Grover  
University of Edinburgh

Sadao Kurohashi  
University of Tokyo

Stephan Oepen  
CSLI, Stanford University

Nelleke Oostdijk  
University of Nijmegen

Richard Sutcliffe  
University of Limerick

Atro Voutilainen  
Conexor oy
The Workshop Programme

08:45–09:00 Welcome and Introduction

09:00–10:30 Session I: Grammar Development I

The Parallel Grammar Project
Miriam Butt, Helge Dyvik, Tracy Holloway King, Hiroshi Masuichi, Christian Rohrer
The Grammar Matrix: An open-source starter-kit for the rapid development of cross-linguistically consistent broad-coverage precision grammars
Emily M. Bender, Dan Flickinger, Stephan Oepen
Parallel distributed grammar engineering for practical applications
Stephan Oepen, Emily M. Bender, Uli Callmeier, Dan Flickinger, Melanie Siegel

10:30–11:00 Break

11:00–12:30 Session II: Grammar Development II

A development environment for large-scale multi-lingual parsing systems
Hisami Suzuki
Adapting existing grammars: The XLE approach
Ronald M. Kaplan, Tracy Holloway King, John T. Maxwell III
Coping with problems in grammars automatically extracted from treebanks
Carlos A. Prolo

12:30–13:30 Lunch

13:30–14:30 Session III: Formalisms and Approaches

A classification of grammar development strategies
Alexandra Kinyon, Carlos A. Prolo
Encoding and reusing linguistic information expressed by Linguistic Properties
Caroline Hagège, Gabriel G. Bès

14:30–15:30 Panel Session

How does a formalism influence grammar engineering? HPSG, LFG, LTAG & The Rest

15:30–16:00 Break

16:00–17:00 Session IV: Evaluation

Grammar and lexicon in the robust parsing of Italian: Towards a non-naïve interplay
Roberto Bartolini, Alessandro Lenci, Simonetta Montemagni, Vito Pirrelli
Machine translation as a testbed for multilingual analysis
Richard Campbell, Carmen Lozano, Jessie Pinkham, Martine Smets

17:00–17:30 Discussion and Closing Remarks
### Table of Contents

**The Parallel Grammar Project**  
Miriam Butt, Helge Dyvik, Tracy Holloway King, Hiroshi Masuichi, Christian Rohrer . . . . 1

**The Grammar Matrix: An open-source starter-kit for the rapid development of cross-linguistically consistent broad-coverage precision grammars**  
Emily M. Bender, Dan Flickinger, Stephan Oepen .......................... 8

**Parallel distributed grammar engineering for practical applications**  
Stephan Oepen, Emily M. Bender, Uli Callmeier, Dan Flickinger, Melanie Siegel 15

**A development environment for large-scale multi-lingual parsing systems**  
Hisami Suzuki ............................................................... 22

**Adapting existing grammars: The XLE approach**  
Ronald M. Kaplan, Tracy Holloway King, John T. Maxwell III .................. 29

**Coping with problems in grammars automatically extracted from treebanks**  
Carlos A. Prolo ............................................................ 36

**A classification of grammar development strategies**  
Alexandra Kinyon, Carlos A. Prolo ........................................ 43

**Encoding and reusing linguistic information expressed by Linguistic Properties**  
Caroline Hagège, Gabriel G. Bès .............................................. 50

**Grammar and lexicon in the robust parsing of Italian: Towards a non-naïve interplay**  
Roberto Bartolini, Alessandro Lenci, Simonetta Montemagni, Vito Pirrelli ........... 57

**Machine translation as a testbed for multilingual analysis**  
Richard Campbell, Carmen Lozano, Jessie Pinkham, Martine Smets .............. 64
# Author Index

Bès, Gabriel G. .................................................................................................................. 50  
Bartolini, Roberto ............................................................................................................. 57  
Bender, Emily M. ............................................................................................................. 8, 15  
Butt, Miriam ..................................................................................................................... 1  
Callmeier, Uli .................................................................................................................... 15  
Campbell, Richard ............................................................................................................ 64  
Dyvik, Helge ...................................................................................................................... 1  
Flickinger, Dan .................................................................................................................. 8, 15  
Hagège, Caroline ............................................................................................................ 50  
Holloway King, Tracy ...................................................................................................... 1, 29  
Kaplan, Ronald M. .......................................................................................................... 29  
Kinyon, Alexandra ........................................................................................................... 43  
Lenci, Alessandro ............................................................................................................ 57  
Lozano, Carmen ............................................................................................................... 64  
Masuichi, Hiroshi ............................................................................................................ 1  
Maxwell, John T. III ........................................................................................................ 29  
Montemagni, Simonetta ................................................................................................. 57  
Oepen, Stephan ................................................................................................................. 8, 15  
Pinkham, Jessie ................................................................................................................ 64  
Pirrelli, Vito ....................................................................................................................... 57  
Prolo, Carlos A. ............................................................................................................... 36, 43  
Rohrer, Christian ............................................................................................................. 1  
Siegel, Melanie .................................................................................................................. 15  
Smets, Martine .................................................................................................................. 64  
Suzuki, Hisami .................................................................................................................. 22
Abstract

We report on the Parallel Grammar (ParGram) project which uses the XLE parser and grammar development platform for six languages: English, French, German, Japanese, Norwegian, and Urdu.¹

1 Introduction

Large-scale grammar development platforms are expensive and time consuming to produce. As such, a desideratum for the platforms is a broad utilization scope. A grammar development platform should be able to be used to write grammars for a wide variety of languages and a broad range of purposes. In this paper, we report on the Parallel Grammar (ParGram) project (Butt et al., 1999) which uses the XLE parser and grammar development platform (Maxwell and Kaplan, 1993) for six languages: English, French, German, Japanese, Norwegian, and Urdu. All of the grammars use the Lexical-Functional Grammar (LFG) formalism which produces c(onstituent)-structures (trees) and f(unctional)-structures (AVMs) as the syntactic analysis.

LFG assumes a version of Chomsky’s Universal Grammar hypothesis, namely that all languages are structured by similar underlying principles. Within LFG, f-structures are meant to encode a language universal level of analysis, allowing for cross-linguistic parallelism at this level of abstraction. Although the construction of c-structures is governed by general wellformedness principles, this level of analysis encodes language particular differences in linear word order, surface morphological vs. syntactic structures, and constituency.

The ParGram project aims to test the LFG formalism for its universality and coverage limitations and to see how far parallelism can be maintained across languages. Where possible, the analyses produced by the grammars for similar constructions in each language are parallel. This has the computational advantage that the grammars can be used in similar applications and that machine translation (Frank, 1999) can be simplified.

The results of the project to date are encouraging. Despite differences between the languages involved and the aims and backgrounds of the project groups, the ParGram grammars achieve a high level of parallelism. This parallelism applies to the syntactic analyses produced, as well as to grammar development itself: the sharing of templates and feature declarations, the utilization of common techniques, and the transfer of knowledge and technology from one grammar to another. The ability to bundle grammar writing techniques, such as templates, into transferable technology means that new grammars can be bootstrapped in a relatively short amount of time.

There are a number of other large-scale grammar projects in existence which we mention briefly here. The LS-GRAM project (Schmidt et al., 1996), funded by the EU-Commission under LRE (Linguistic Research and Engineering), was concerned with the development of grammatical resources for nine European languages: Danish, Dutch, English, French, German, Greek, Italian, Portuguese, and Spanish. The project started in January 1994 and ended in July 1996. Development of grammatical resources was carried out in the framework of the

¹We would like to thank Emily Bender, Mary Dalrymple, and Ron Kaplan for help with this paper. In addition, we would like to acknowledge the other grammar writers in the ParGram project, both current: Stefanie Dipper, Jean-Philippe Marcotte, Tomoko Ohkuma, and Victoria Rosén; and past: Caroline Brun, Christian Fortmann, Anette Frank, Jonas Kuhn, Veronica Lux, Yukiko Morimoto, María-Eugenia Niño, and Frédérique Segond.
Advanced Language Engineering Platform (ALEP). The coverage of the grammars implemented in LS-GRAM was, however, much smaller than the coverage of the English (Riezler et al., 2002) or German grammar in ParGram. An effort which is closer in spirit to ParGram is the implementation of grammar development platforms for HPSG. In the Verbomobil project (Wahlster, 2000), HPSG grammars for English, German, and Japanese were developed on two platforms: LKB (Copestake, 2002) and PAGE. The PAGE system, developed and maintained in the Language Technology Lab of the German National Research Center on Artificial Intelligence DFKI GmbH, is an advanced NLP core engine that facilitates the development of grammatical and lexical resources, building on typed feature logics. To evaluate the HPSG platforms and to compare their merits with those of XLE and the ParGram projects, one would have to organize a special workshop, particularly as the HPSG grammars in Verbomobil were written for spoken language, characterized by short utterances, whereas the LFG grammars were developed for parsing technical manuals and/or newspaper texts. There are some indications that the German and English grammars in ParGram exceed the HPSG grammars in coverage (see (Crysmann et al., 2002) on the German HPSG grammar).

This paper is organized as follows. We first provide a history of the project. Then, we discuss how parallelism is maintained in the project. Finally, we provide a summary and discussion.

2 Project History

The ParGram project began in 1994 with three languages: English, French, and German. The grammar writers worked closely together to solidify the grammatical analyses and conventions. In addition, as XLE was still in development, its abilities grew as the size of the grammars and their needs grew.

After the initial stage of the project, more languages were added. Because Japanese is typologically very different from the initial three European languages of the project, it represented a challenging case. Despite this typological challenge, the Japanese grammar has achieved broad coverage and high performance within a year and a half. The South Asian language Urdu also provides a widely spoken, typologically distinct language. Although it is of Indo-European origin, it shares many characteristics with Japanese such as verb-finality, relatively free word order, complex predicates, and the ability to drop any argument (rampant pro-drop). Norwegian assumes a typological middle position between German and English, sharing different properties with each of them. Both the Urdu and the Norwegian grammars are still relatively small.

Each grammar project has different goals, and each site employs grammar writers with different backgrounds and skills. The English, German, and Japanese projects have pursued the goal of having broad coverage, industrial grammars. The Norwegian and Urdu grammars are smaller scale but are experimenting with incorporating different kinds of information into the grammar. The Norwegian grammar includes a semantic projection; their analyses produce not only c- and f-structures, but also semantic structures. The Urdu grammar has implemented a level of argument structure and is testing various theoretical linguistic ideas. However, even when the grammars are used for different purposes and have different additional features, they have maintained their basic parallelism in analysis and have profited from the shared grammar writing techniques and technology.

Table (1) shows the size of the grammars. The first figure is the number of left-hand side categories in phrase-structure rules which compile into a collection of finite-state machines with the listed number of states and arcs.

<table>
<thead>
<tr>
<th>Language</th>
<th>Rules</th>
<th>States</th>
<th>Arcs</th>
</tr>
</thead>
<tbody>
<tr>
<td>German</td>
<td>444</td>
<td>4883</td>
<td>15870</td>
</tr>
<tr>
<td>English</td>
<td>310</td>
<td>4935</td>
<td>13268</td>
</tr>
<tr>
<td>French</td>
<td>132</td>
<td>1116</td>
<td>2674</td>
</tr>
<tr>
<td>Japanese</td>
<td>50</td>
<td>333</td>
<td>1193</td>
</tr>
<tr>
<td>Norwegian</td>
<td>46</td>
<td>255</td>
<td>798</td>
</tr>
<tr>
<td>Urdu</td>
<td>25</td>
<td>106</td>
<td>169</td>
</tr>
</tbody>
</table>

3 Parallelism

Maintaining parallelism in grammars being developed at different sites on typologically distinct languages by grammar writers from different linguistic traditions has proven successful. At project meetings held twice a year, analyses of sample sentences are compared and any differences are discussed; the goal is to determine whether the differences are justified or whether the analyses should be changed to maintain parallelism. In addition, all of the f-structure features and their values are compared; this not only ensures that trivial differences in naming conventions do not arise, but also gives an overview of the constructions each language covers and how
they are analyzed. All changes are implemented before the next project meeting. Each meeting also involves discussion of constructions whose analysis has not yet been settled on, e.g., the analysis of partitives or proper names. If an analysis is agreed upon, all the grammars implement it; if only a tentative analysis is found, one grammar implements it and reports on its success. For extremely complicated or fundamental issues, e.g., how to represent predicate alternations, subcommittees examine the issue and report on it at the next meeting. The discussion of such issues may be reopened at successive meetings until a consensus is reached.

Even within a given linguistic formalism, LFG for ParGram, there is usually more than one way to analyze a construction. Moreover, the same theoretical analysis may have different possible implementations in XLE. These solutions often differ in efficiency or conceptual simplicity and one of the tasks within the ParGram project is to make design decisions which favor one theoretical analysis and concomitant implementation over another.

3.1 Parallel Analyses

Whenever possible, the ParGram grammars choose the same analysis and the same technical solution for equivalent constructions. This was done, for example, with imperatives. Imperatives are always assigned a null pronominal subject within the f-structure and a feature indicating that they are imperatives, as in (2).

(2) a. Jump! (French) Saute! (French) Spring! (German) Tobe! (Japanese) Hopp! (Norwegian) kuudoo! (Urdu)

b. \[
\begin{array}{c}
\text{PRED} \quad \text{\textquoteleft horse\textquoteright} \\
\text{SUBJ} \quad \text{\textquoteleft jump<SUBJ>\textquoteright} \\
\text{STMT-TYPE} \quad \text{imp}
\end{array}
\]

Another example of this type comes from the analysis of specifiers. Specifiers include many different types of information and hence can be analyzed in a number of ways. In the ParGram analysis, the c-structure analysis is left relatively free according to language particular needs and slightly varying theoretical assumptions. For instance, the Norwegian grammar, unlike the other grammars, implements the principles in (Bresnan, 2001) concerning the relationship between an X*-based c-structure and the f-structure. This allows Norwegian specifiers to be analyzed as functional heads of DPs etc., whereas they are constituents of NPs in the other grammars. However, at the level of f-structure, this information is part of a complex SPEC feature in all the grammars. Thus parallelism is maintained at the level of f-structure even across different theoretical preferences. An example is shown in (3) for Norwegian and English in which the SPEC consists of a QUANT(ifer) and a POSS(essive) (SPEC can also contain information about DETERMINERS and DEMONSTRATIVES).

(3) a. alle mine hester (Norwegian)
   all my horses
   ‘all my horses’

b. \[
\begin{array}{c}
\text{PRED} \quad \text{\textquoteleft all\textquoteright} \\
\text{QUANT} \quad \text{\textquoteleft pro\textquoteright} \\
\text{SPEC} \quad \text{\textquoteleft poss\textquoteright} \\
\text{PERS} \quad 1 \\
\text{NUM} \quad \text{sg}
\end{array}
\]

Interrogatives provide an interesting example because they differ significantly in the c-structures of the languages, but have the same basic f-structure. This contrast can be seen between the German example in (4) and the Urdu one in (5). In German, the interrogative word is in first position with the finite verb second; English and Norwegian pattern like German. In Urdu the verb is usually in final position, but the interrogative can appear in a number of positions, including following the verb (5c).

(4) Was hat John Maria gegeben? (German)
   what has John Maria give.PerfP
   ‘What did John give to Mary?’

b. \[
\begin{array}{c}
\text{PRED} \quad \text{\textquoteleft all\textquoteright} \\
\text{QUANT} \quad \text{\textquoteleft pro\textquoteright} \\
\text{SPEC} \quad \text{\textquoteleft poss\textquoteright} \\
\text{PERS} \quad 1 \\
\text{NUM} \quad \text{sg}
\end{array}
\]

(5) a. jon=nee marii=koo kyaa diiyaa? (Urdu)
   John=Erg Mary=Dat what gave
   ‘What did John give to Mary?’

b. jon=nee kyaa marii=koo diiyaa?

c. jon=nee marii=ko diiyaa kyaa?

Despite these differences in word order and hence in c-structure, the f-structures are parallel, with the interrogative being in a FOCUS-INT and the sentence having an interrogative STMT-TYPE, as in (6).
subject and the predicate adjective being an adjective, with the pronoun being the subject. As such, In the project grammars, many basic constructions are of this type. However, as we will see in the next section, there are times when parallelism is not possible and not desirable. Even in these cases, though, the grammars which can be parallel are; so, three of the languages might have one analysis, while three have another.

3.2 Justified Differences
Parallelism is not maintained at the cost of misrepresenting the language. This is reflected by the fact that the c-structures are not parallel because word order varies widely from language to language, although there are naming conventions for the nodes. Instead, the bulk of the parallelism is in the f-structure. However, in the f-structure, situations arise in which what seems to be the same construction in different languages do not have the same analysis. An example of this is predicate adjectives, as in (7).

(7) a. It is red.
   b. Sore wa akai. (Japanese)
      it    TOP red
      ‘It is red.’

In English, the copular verb is considered the syntactic head of the clause, with the pronoun being the subject and the predicate adjective being an XCOMP. However, in Japanese, the adjective is the main predicate, with the pronoun being the subject. As such, these receive the non-parallel analyses seen in (8a) for Japanese and (8b) for English.

(8) a. [PRED 'red<SUBJ>'][SUBJ ['pro']]

b. [PRED 'be<XCOMP>SUBJ'][SUBJ ['pro']]
   [XCOMP ['red<SUBJ>']]

Another situation that arises is when a feature or construction is syntactically encoded in one language, but not another. In such cases, the information is only encoded in the languages that need it. The equivalence captured by parallel analyses is not, for example, translational equivalence. Rather, parallelism involves equivalence with respect to grammatical properties, e.g. construction types. One consequence of this is that a typologically consistent use of grammatical terms, embodied in the feature names, is enforced. For example, even though there is a tradition for referring to the distinction between the pronouns he and she as a gender distinction in English, this is a different distinction from the one called gender in languages like German, French, Urdu, and Norwegian, where gender refers to nominal agreement classes. Parallelism leads to the situation where the feature GEND occurs in German, French, Urdu, and Norwegian, but not in English and Japanese. That is, parallelism does not mean finding the same features in all languages, but rather using the same features in the same way in all languages, to the extent that they are justified there. A French example of grammatical gender is shown in (9); note that determiner, adjective, and participle agreement is dependent on the gender of the noun. The f-structure for the nouns crayon and plume are as in (10) with an overt GEND feature.

(9) a. Le petit crayon est cassé. (French)
   the-M little-M pencil-M is broken-M.
   ‘The little pencil is broken.’

b. La petite plume est cassée. (French)
   the-F little-F pen-F is broken-F.
   ‘The little pen is broken.’

(10) [PRED 'crayon'][GEND masc]
     [PRED 'plume'][GEND fem]
     [PERS 3]
     [PERS 3]

F-structures for the equivalent words in English and Japanese will not have a GEND feature.

A similar example comes from Japanese discourse particles. It is well-known that Japanese has syntactic encodings for information such as honorification. The verb in the Japanese sentence (11a) encodes information that the subject is respected, while the verb in (11b) shows politeness from the writer (speaker) to the reader (hearer) of the sentence. The f-structures for the verbs in (11) are as in
(12) with RESPECT and POLITE features within the ADDRESS feature.

(11) a. sensei ga hon wo oyomininaru.
   teacher Nom book Acc read-Respect
   ‘The teacher read the book.’ (Japanese)

b. seito ga hon wo yomimasu.
   student Nom book Acc read-Polite
   ‘The student reads the book.’ (Japanese)

(12) a. [PRED 'yomu<OBJ,OBJ>']
    [ADDRESS  [RESPECT + ]]

b. [PRED 'yomu<OBJ,OBJ>']
   [ADDRESS  [POLITE + ]]

A final example comes from English progressives, as in (13). In order to distinguish these two forms, the English grammar uses a PROG feature within the tense/aspect system. (13b) shows the f-structure for (13a.ii).

(13) a. John hit Bill. i. He cried.
    ii. He was crying.

b. [PRED 'cry<OBJ>']
   [SUBJ  [PRED 'pro']]
   [TNS-ASP  [TENSE past  PROG + ]]

However, this distinction is not found in the other languages. For example, (14a) is used to express both (13a.i) and (13a.ii) in German.

(14) a. Er weinte. (German)
    he cried
    ‘He cried.’

b. [PRED 'weinen<OBJ>']
   [SUBJ  [PRED 'pro']]
   [TNS-ASP  [TENSE past  PROG + ]]

As seen in (14b), the German f-structure is left underspecified for PROG because there is no syntactic reflex of it. If such a feature were posited, rampant ambiguity would be introduced for all past tense forms in German. Instead, the semantics will determine whether such forms are progressive.

Thus, there are a number of situations where having parallel analyses would result in an incorrect analysis for one of the languages.

3.3 One Language Shows the Way

Another type of situation arises when one language provides evidence for a certain feature space or type of analysis that is neither explicitly mirrored nor explicitly contradicted by another language. In theoretical linguistics, it is commonly acknowledged that what one language codes overtly may be harder to detect for another language. This situation has arisen in the ParGram project. Case features fall under this topic. German, Japanese, and Urdu mark NPs with overt case morphology. In comparison, English, French, and Norwegian make relatively little use of case except as part of the pronominal system. Nevertheless, the f-structure analyses for all the languages contain a case feature in the specification of noun phrases.

This “overspecification” of information expresses deeper linguistic generalizations and keeps the f-structural analyses as parallel as possible. In addition, the features can be put to use for the isolated phenomena in which they do play a role. For example, English does not mark animacy grammatically in most situations. However, providing a ANIM + feature to known animates, such as people’s names and pronouns, allows the grammar to encode information that is relevant for interpretation. Consider the relative pronoun who in (15).

(15) a. the girl[ANIM + ] who[ANIM + ] left

b. the box[ANIM + ] who[ANIM + ] left

The relative pronoun has a ANIM + feature that is assigned to the noun it modifies by the relative clause rules. As such, a noun modified by a relative clause headed by who is interpreted as animate. In the case of canonical inanimates, as in (15b), this will result in a pragmatically odd interpretation, which is encoded in the f-structure.

Teasing apart these different phenomena crosslinguistically poses a challenge that the ParGram members are continually engaged in. As such, we have developed several methods to help maintain parallelism.

3.4 Mechanics of Maintaining Parallelism

The parallelism among the grammars is maintained in a number of ways. Most of the work is done during two week-long project meetings held each year.
Three main activities occur during these meetings: comparison of sample f-structures, comparison of features and their values, and discussions of new or problematic constructions.

A month before each meeting, the host site chooses around fifteen sentences whose analysis is to be compared at the meeting. These can be a random selection or be thematic, e.g., all dealing with predicatives or with interrogatives. The sentences are then parsed by each grammar and the output is compared. For the more recent grammars, this may mean adding the relevant rules to the grammars, resulting in growth of the grammar; for the older grammars, this may mean updating a construction that has not been examined in many years. Another approach that was taken at the beginning of the project was to have a common corpus of about 1,000 sentences that all of the grammars were to parse. For the English, French, and German grammars, this was an aligned tractor manual. The corpus sentences were used for the initial f-structure comparisons. Having a common corpus ensured that the grammars would have roughly the same coverage. For example, they all parsed declarative and imperative sentences. However, the nature of the corpus can leave major gaps in coverage; in this case, the manual contained no interrogatives.

The XLE platform requires that a grammar declare all the features it uses and their possible values. Part of the Urdu feature table is shown in (16) (the notation has been simplified for expository purposes). As seen in (16) for QUANT, attributes which take other attributes as their values must also be declared. An example of such a feature was seen in (3b) for SPEC which takes QUANT and POSS features, among others, as its values.

(16) PRON-TYPE: \{ pers poss null \},

PROPER: \{ date location name title \},

PSEM: \{ locational directional \},

PTYPE: \{ sem nosem \},

QUANT: \{ PRED QUANT-TYPE QUANT-FORM \}.

The feature declarations of all of the languages are compared feature by feature to ensure parallelism. The most obvious use of this is to ensure that the grammars encode the same features in the same way. For example, at a basic level, one feature declaration might have specified GEN for gender while the others had chosen the name GEND; this divergence in naming is regularized. More interesting cases arise when one language uses a feature and another does not for analyzing the same phenomena. When this is noticed via the feature-table comparison, it is determined why one grammar needs the feature and the other does not, and thus it may be possible to eliminate the feature in one grammar or to add it to another.

On a deeper level, the feature comparison is useful for conducting a survey of what constructions each grammar has and how they are implemented. For example, if a language does not have an ADEGREE (adjective degree) feature, the question will arise as to whether the grammar analyzes comparative and superlative adjectives. If they do not, then they should be added and should use the ADEGREE feature; if they do, then the question arises as to why they do not have this feature as part of their analysis.

Finally, there is the discussion of problematic constructions. These may be constructions that already have analyses which had been agreed upon in the past but which are not working properly now that more data has been considered. More frequently, they are new constructions that one of the grammars is considering adding. Possible analyses for the construction are discussed and then one of the grammars will incorporate the analysis to see whether it works. If the analysis works, then the other grammars will incorporate the analysis. Constructions that have been discussed in past ParGram meetings include predicative adjectives, quantifiers, partitives, and clefts. Even if not all of the languages have the construction in question, as was the case with clefts, the grammar writers for that language may have interesting ideas on how to analyze it. These group discussions have proven particularly useful in extending grammar coverage in a parallel fashion.

Once a consensus is reached, it is the responsibility of each grammar to make sure that its analyses match the new standard. As such, after each meeting, the grammar writers will rename features, change analyses, and implement new constructions into their grammars. Most of the basic work has now been accomplished. However, as the grammars expand coverage, more constructions need to be integrated into the grammars, and these constructions tend to be ones for which there is no standard analysis in the linguistic literature; so, differences can easily arise in these areas.
4 Conclusion

The experiences of the ParGram grammar writers has shown that the parallelism of analysis and implementation in the ParGram project aids further grammar development efforts. Many of the basic decisions about analyses and formalism have already been made in the project. Thus, the grammar writer for a new language can use existing technology to bootstrap a grammar for the new language and can parse equivalent constructions in the existing languages to see how to analyze a construction. This allows the grammar writer to focus on more difficult constructions not yet encountered in the existing grammars.

Consider first the Japanese grammar which was started in the beginning of 2001. At the initial stage, the work of grammar development involved implementing the basic constructions already analyzed in the other grammars. It was found that the grammar writing techniques and guidelines to maintain parallelism shared in the ParGram project could be efficiently applied to the Japanese grammar. During the next stage, LFG rules needed for grammatical issues specific to Japanese have been gradually incorporated, and at the same time, the biannual ParGram meetings have helped significantly to keep the grammars parallel. Given this system, in a year and a half, using two grammar writers, the Japanese grammar has attained coverage of 99% for 500 sentences of a copier manual and 95% for 10,000 sentences of an eCRM (Voice-of-Customer) corpus.

Next consider the Norwegian grammar which joined the ParGram group in 1999 and also emphasized slightly different goals from the other groups. Rather than prioritizing large textual coverage from the outset, the Norwegian group gave priority to the development of a core grammar covering all major construction types in a principled way based on the proposals in (Bresnan, 2001) and the inclusion of a semantic projection in addition to the f-structure. In addition, time was spent on improving existing lexical resources (> 80,000 lemmas) and adapting them to the XLE format. Roughly two man-years has been spent on the grammar itself. The ParGram cooperation on parallelism has ensured that the derived f-structures are interesting in a multilingual context, and the grammar will now serve as a basis for grammar development in other closely related Scandinavian languages.

Thus, the ParGram project has shown that it is possible to use a single grammar development platform and a unified methodology of grammar writing to develop large-scale grammars for typologically different languages. The grammars’ analyses show a large degree of parallelism, despite being developed at different sites. This is achieved by intensive meetings twice a year. The parallelism can be exploited in applications using the grammars: the fewer the differences, the simpler a multilingual application can be (see (Frank, 1999) on a machine-translation prototype using ParGram).

References


Emily M. Bender and Dan Flickinger and Stephan Oepen
Center for the Study of Language and Information
Stanford University
{bender|dan|oe}@csli.stanford.edu

Abstract
The grammar matrix is an open-source starter-kit for the development of broad-coverage HPSGs. By using a type hierarchy to represent cross-linguistic generalizations and providing compatibility with other open-source tools for grammar engineering, evaluation, parsing and generation, it facilitates not only quick start-up but also rapid growth towards the wide coverage necessary for robust natural language processing and the precision parses and semantic representations necessary for natural language understanding.

1 Introduction
The past decade has seen the development of wide-coverage implemented grammars representing deep linguistic analysis of several languages in several frameworks, including Head-Driven Phrase Structure Grammar (HPSG), Lexical-Functional Grammar (LFG), and Lexicalized Tree Adjoining Grammar (LTAG). In HPSG, the most extensive grammars are those of English (Flickinger, 2000), German (Müller & Kasper, 2000), and Japanese (Siegel, 2000; Siegel & Bender, 2002). Despite being couched in the same general framework and in some cases being written in the same formalism and consequently being compatible with the same parsing and generation software, these grammars were developed more or less independently of each other. They each represent between 5 and 15 person years of research efforts, and comprise 35–70,000 lines of code. Unfortunately, most of that research is undocumented and the accumulated analyses, best practices for grammar engineering, and tricks of the trade are only available through painstaking inspection of the grammars and/or consultation with their authors. This lack of documentation holds across frameworks, with certain notable exceptions, including Alshawi (1992), Müller (1999), and Butt, King, Niño, & Segond (1999).

Grammars which have been under development for many years tend to be very difficult to mine for information, as they contain layers upon layers of interacting analyses and decisions made in light of various intermediate stages of the grammar. As a result, when embarking on the creation of a new grammar for another language, it seems almost easier to start from scratch than to try to model it on an existing grammar. This is unfortunate—being able to leverage the knowledge and infrastructure embedded in existing grammars would greatly accelerate the process of developing new ones. At the same time, these grammars represent an untapped resource for the bottom-up exploration of language universals.

As part of the LinGO consortium’s multi-lingual grammar engineering effort, we are developing a ‘grammar matrix’ or starter-kit, distilling the wisdom of existing grammars and codifying and documenting it in a form that can be used as the basis for new grammars.

In the following sections, we outline the inventory of a first, preliminary version of the grammar matrix, discuss the interaction of basic construction types and semantic composition in unification grammars by means of a detailed example, and consider extensions to the core inventory that we foresee and an evaluation methodology for the matrix proper.

2 Preliminary Development of Matrix
We have produced a preliminary version of the grammar matrix relying heavily on the LinGO project’s English Resource Grammar, and to a lesser extent on the Japanese grammar developed jointly between DFKI Saarbrücken (Germany) and YY Technologies (Mountain View, CA). This early version of the matrix comprises the following com-
ponents:

- Types defining the basic feature geometry and technical devices (e.g., for list manipulation).
- Types associated with Minimal Recursion Semantics (see, e.g., Copestake, Lascarides, & Flickinger, 2001), a meaning representation language which has been shown to be well-suited for semantic composition in typed feature structure grammars. This portion of the grammar matrix includes a hierarchy of relation types, types and constraints for the propagation of semantic information through the phrase structure tree, a representation of illocutionary force, and provisions for grammar rules which make semantic contributions.
- General classes of rules, including derivational and inflectional (lexical) rules, unary and binary phrase structure rules, headed and non-headed rules, and head-initial and head-final rules. These rule classes include implementations of general principles of HPSG, like, for example, the Head Feature and Non-Local Feature Principles.
- Types for basic constructions such as head-complement, head-specifier, head-subject, head-filler, and head-modifier rules, coordination, as well as more specialized classes of constructions, such as relative clauses and noun-noun compounding. Unlike in specific grammars, these types do not impose any ordering on their daughters in the grammar matrix.

Included with the matrix are configuration and parameter files for the LKB grammar engineering environment (Copestake, 2002).

Although small, this preliminary version of the matrix already reflects the main goals of the project: (i) Consistent with other work in HPSG, semantic representations and in particular the syntax-semantics interface are developed in detail; (ii) the types of the matrix are each representations of generalizations across linguistic objects and across languages; and (iii) the richness of the matrix and the incorporation of files which connect it with the LKB allow for extremely quick start-up as the matrix is applied to new languages.

Since February 2002, this preliminary version of the matrix has been in use at two Norwegian universities, one working towards a broad-coverage reference implementation of Norwegian (NTNU), the other—for the time being—focused on specific aspects of clause structure and lexical description (Oslo University). In the first experiment with the matrix, at NTNU, basic Norwegian sentences were parsing and producing reasonable semantics within two hours of downloading the matrix files. Linguistic coverage should scale up quickly, since the foundation supplied by the matrix is designed not only to provide a quick start, but also to support long-term development of broad-coverage grammars. Both initiatives have confirmed the utility of the matrix starter kit and already have contributed to a series of discussions on cross-lingual HPSG design aspects, specifically in the areas of argument structure representations in the lexicon and basic assumptions about constituent structure (in one view, Norwegian exhibits a VSO topology in the main clause). The user groups have suggested refinements and extensions of the basic inventory, and it is expected that general solutions, as they are identified jointly, will propagate into the existing grammars too.

3 A Detailed Example

As an example of the level of detail involved in the grammar matrix, in this section we consider the analysis of intersective and scopal modification. The matrix is built to give Minimal Recursion Semantics (MRS; Copestake et al., 2001; Copestake, Flickinger, Sag, & Pollard, 1999; Copestake, Flickinger, Malouf, Riehemann, & Sag, 1995) representations. The two English examples in (1) exemplify the difference between intersective and scopal modification:

1 a. Keanu studied Kung Fu on a spaceship.
   Keanu probably studied Kung Fu.

The MRSs for (1a-b) (abstracting away from agreement information) are given in (2) and (3). The MRSs are ordered tuples consisting of a top handle (h1 in both cases), an instance or event variable (e in both cases), a bag of elementary predicators (eps), and a bag of scope constraints (in these cases, QEQ constraints or ‘equal modulo quantifiers’). In a well-formed MRS, the handles can be

1These examples also differ in that probably is a pre-head modifier while on a spaceship is a post-head modifier. This word-order distinction cross-cuts the semantic distinction, and our focus is on the latter, so we won’t consider the word-order aspects of these examples here.
identified in one or more ways respecting the scope constraints such that the dependencies between the eps form a tree. For a detailed description of MRS, see the works cited above. Here, we will focus on the difference between the intersective modifier \textit{on (a spaceship)} and the scopal modifier \textit{probably}.

In (2), the ep contributed by \textit{on} (‘on-rel’) shares its handle ($h7$) with the ep contributed by the verb it is modifying (‘study-rel’). As such, the two will always have the same scope; no quantifier can intervene. Further, the second argument of the on-rel ($e$) is the event variable of the study-rel. The first argument, \textit{e’}, is the event variable of the on-rel and the third argument, $z$, is the instance variable of the spaceship-rel.

(2) $h1, e,$

\begin{verbatim}
{ h1:prpstn-rel(h2), h3:def-np-rel(x, h4, h5),
    h6:named-rel(x, ‘Keanu’), h7:study-rel(e, x, y),
    h8:def-np-rel(y, h9, h10),
    h11:named-rel(y, ‘Kung Fu’), h7:on-rel(e’, e, z),
    h12:a-quant-rel(z, h13, h14),
    h15:spaceship-rel(z) },
{ h2 QEQ h7, h4 QEQ h6, h19 QEQ h11, h13 QEQ h15 }
\end{verbatim}

In (3), the ep contributed by the scopal modifier \textit{probably} (‘probably-rel’) has its own handle ($h7$) which is not shared by anything. Furthermore, it takes a handle ($h8$) rather than the event variable of the study-rel as its argument. $h8$ is equal modulo quantifiers (QEQ) to the handle of the study-rel ($h9$), and $h7$ is equal modulo quantifiers to the argument of the prpstn-rel ($h2$). The prpstn-rel is the ep representing the illocutionary force of the whole expression. This means that quantifiers associated with the NPs \textit{Keanu} and \textit{Kung Fu} can scope inside or outside \textit{probably}.

(3) $h1, e,$

\begin{verbatim}
{ h1:prpstn-rel(h2), h3:def-np-rel(x, h4, h5),
    h6:named-rel(x, ‘Keanu’),
    h7:probably-rel(h8), h9:study-rel(e, x, y),
    h10:def-np-rel(y, h11, h12),
    h13:named-rel(y, ‘Kung Fu’) },
{ h2 QEQ h7, h4 QEQ h6, h19 QEQ h9,
    h11 QEQ h13 }
\end{verbatim}

While the details of modifier placement, which parts of speech can modify which kinds of phrases, etc., differ across languages, we believe that all languages display a distinction between scopal and intersective modification. Accordingly, the types

\begin{verbatim}
isect-mod-phrase := head-mod-phr-simple &
    { HEAD-DTR.SYNSEM.LOCAL
      [ CONT [ TOP #hand,
        INDEX #index ],
      KEYS.MESSAGE 0-dlist ],
    NON-HEAD-DTR.SYNSEM.LOCAL
      [ CAT.HEAD.MOD <| LOCAL isect-mod |>,
        CONT.TOP #hand ],
    C-CONT.INDEX #index ].
\end{verbatim}

\begin{verbatim}
scopal-mod-phrase := head-mod-phr-simple &
    { NON-HEAD-DTR.SYNSEM.LOCAL
      [ CAT.HEAD.MOD <| LOCAL scopal-mod |>,
        CONT.INDEX #index ],
    C-CONT.INDEX #index ].
\end{verbatim}

\begin{figure}
\centering
\includegraphics[width=\textwidth]{isect-mod-phrase.png}
\caption{TDL description of \textit{isect-mod-phrase}}
\end{figure}

\begin{figure}
\centering
\includegraphics[width=\textwidth]{scopal-mod-phrase.png}
\caption{TDL description of \textit{scopol-mod-phrase}}
\end{figure}

necessary for describing these two kinds of modification are included in the matrix. The types \textit{isect-mod-phrase} and \textit{scopal-mod-phrase} (shown in Figures 1 and 2) encode the information necessary to build up in a compositional manner the modifier portions of the MRSs in (2) and (3).

These types are embedded in the type hierarchy of the matrix. Through their supertype \textit{head-mod-phr-simple} they inherit information common to many types of phrases, including the basic feature geometry, head feature and non-local feature passing, and semantic compositionality. These types also have subtypes in the matrix specifying the two word-order possibilities (pre- or post-head modifiers), giving a total of four subtypes.\footnote{\textit{All four subtypes are provided on the theory that most languages will make use of all or most of them.}}

The most important difference between these types is in the treatment of the handle of the head daughter’s semantics, to distinguish intersective and scopal modification. In \textit{isect-mod-phrase}, the top handles (TOP) of the head and non-head (i.e., modifier) daughters are identified (#hand). This allows for MRSs like (2) where the eps contributed by the head (‘study-rel’) and the modifier (‘on-rel’) take the same scope. The type \textit{scopal-mod-phrase} bears no such constraint. This allows for MRSs like (3) where the modifier’s semantic contribution (‘probably-rel’) takes the handle of the head’s semantics (‘study-rel’) as its argument, so that the modifier outscopes the head. In both types of mod-
ifier phrase, a constraint inherited from the supertype ensures that the handle of the modifier is also the handle of the whole phrase.

The constraints on the LOCAL value inside the modifier’s MOD value regulate which lexical items can appear in which kind of phrase. Intersective modifiers specify lexically that they are \([\text{MOD} \{ [\text{LOCAL isect-mod}] \}]\) and scopal modifiers specify lexically that they are \([\text{MOD} \{ [\text{LOCAL scopal-mod}] \}]\). These constraints exemplify the kind of information that will be developed in the lexical hierarchy of the matrix.

It is characteristic of broad-coverage grammars that every particular analysis interacts with many other analyses. Modularization is an on-going concern, both for maintainability of individual grammars, and for providing the right level of abstraction in the matrix. For the same reasons, we have only been able to touch on the highlights of the semantic analysis of modification here, but hope that this quick tour will suffice to illustrate the extent of the jump-start the matrix can give in the development of new grammars.

4 Future Extensions

The initial version of the matrix, while sufficient to support some useful grammar work, will require substantial further development on several fronts, including lexical representation, syntactic generalization, sociolinguistic variation, processing issues, and evaluation. This first version drew most heavily from the implementation of the English grammar, with some further insights drawn from the grammar of Japanese. Extensions to the matrix will be based on careful study of existing implemented grammars for other languages, notably German, Spanish and Japanese, as well as feedback from those using the first version of the matrix.

For lexical representation, one of the most urgent needs is to provide a language-independent type hierarchy for the lexicon, at least for major parts of speech, establishing the mechanisms used for linking syntactic subcategorization to semantic predicate-argument structure. Lexical rules provide a second mechanism for expressing generalizations within the lexicon, and offer ready opportunities for cross-linguistic abstractions for both inflectional and derivational regularities. Work is also progressing on establishing a standard relational database (using PostgreSQL) for storing information for the lexical entries themselves, improving both scalability and clarity compared to the current simple text file representation. Form-based tools will be provided both for constructing lexical entries and for viewing the contents of the lexicon.

The primary focus of work on syntactic generalization in the matrix is to support more freedom in word order, for both complements and modifiers. The first step will be a relatively conservative extension along the lines of Netter (1996), allowing the grammar writer more control over how a head combines with complements of different types, and their interleaving with modifier phrases. Other areas of immediate cross-linguistic interest include the hierarchy of head types, control phenomena, clitics, auxiliary verbs, noun-noun compounds, and more generally, phenomena that involve the word/phrase distinction, such as noun incorporation. A study of the existing grammars for English, German, Japanese, and Spanish reveals a high degree of language-specificity for several of these phenomena, but also suggests promise of reusable abstractions.

Several kinds of sociolinguistic variation require extensions to the matrix, including grammaticized aspects of pragmatics such as politeness and empathy, as well as dialect and register alternations. The grammar of Japanese provides a starting point for representations of both empathy and politeness. Implementations of familiar vs. formal verb forms in German and Spanish provide further instances of politeness to help build the cross-linguistic abstractions. Extensions for dialect variation will build on some exploratory work in adapting the English grammar to support American, British, and Australian regionalisms, both lexical and syntactic, while restricting dialect mixture in generation and associated spurious ambiguity in parsing.

While the development of the matrix will be built largely on the LKB platform, support will also be needed for using the emerging grammars on other processing platforms, and for linking to other packages for pre-processing the linguistic input. Several other platforms exist which can efficiently parse text using the existing grammars, includ-

---

3 Note that there are no further subtypes of LOCAL values beyond isect-mod and scopal-mod. Since these grammars do not make extensive use of subtypes of LOCAL values, they were available for encoding this distinction. Alternative solutions include positing a new feature.
ing the PET system developed in C++ at Saarland University (Germany) and the DFKI (Callmeier, 2000); the PAGE system developed in Lisp at the DFKI (Uszkoreit et al., 1994); the LiLFeS system developed at Tokyo University (Makino, Yoshida, Torisawa, & Tsujii, 1998), and a parallel processing system developed in Objective C at Delft University (The Netherlands; van Lohuizen, 2002). As part of the matrix package, sample configuration files and documentation will be provided for at least some of these additional platforms.

Existing pre-processing packages can also significantly reduce the effort required to develop a new grammar, particularly for coping with the morphology/syntax interface. For example, the ChaSen package for segmenting Japanese input into words and morphemes (Asahara & Matsumoto, 2000) has been linked to at least the LKB and PET systems. Support for connecting implementations of language-specific pre-processing packages of this kind will be preserved and extended as the matrix develops. Likewise, configuration files are included to support generation, at least within the LKB, provided that the grammar conforms to certain assumptions about semantic representation using the Minimal Recursion Semantics framework.

Finally, a methodology is under development for constructing and using test suites organized around a typology of linguistic phenomena, using the implementation platform of the [incr tsdbi] profiling package (Oepen & Flickinger, 1998; Oepen & Callmeier, 2000). These test suites will enable better communication about current coverage of a given grammar built using the matrix, and serve as the basis for identifying additional phenomena that need to be addressed cross-linguistically within the matrix. Of course, the development of the typology of phenomena is itself a major undertaking for which a systematic cross-linguistic approach will be needed, a discussion of which is outside the scope of this report. But the intent is to seed this classification scheme with a set of relatively coarse-grained phenomenon classes drawn from the existing grammars, then refine the typology as it is applied to these and new grammars built using the matrix.

5 Case Studies

One important part of the matrix package will be a library of phenomenon-based analyses drawn from the existing grammars and over time from users of the matrix, to provide working examples of how the matrix can be applied and extended. Each case study will be a set of grammar files, simplified for relevance, along with documentation of the analysis, and a test suite of sample sentences which define the range of data covered by the analysis. This library, too, will be organized around the typology of phenomena introduced above, but will also make explicit reference to language families, since both similarities and differences among related languages will be of interest in these case studies. Examples to be included in the first release of this library include numeral classifiers in Japanese, subject pro drop in Spanish, partial-VP fronting in German, and verb diathesis in Norwegian.

6 Evaluation and Evolution

The matrix itself is not a grammar but a collection of generalizations across grammars. As such, it cannot be tested directly on corpora from particular languages, and we must find other means of evaluation. We envision overall evaluation of the matrix based on case studies of its performance in helping grammar engineers quickly start new grammars and in helping them scale those grammars up. Evaluation in detail will be based on automatable deletion/substitution metrics, i.e., tools that determine which types from the matrix get used as is, which get used with modifications, and which get ignored in various matrix-derived grammars. Furthermore, if the matrix evolves to include defeasible constraints, these tools will check which constraints get overridden and whether the value chosen is indeed common enough to be motivated as a default value. This evaluation in detail should be paired with feedback from the grammar engineers to determine why changes were made.

The main goal of evaluation is, of course, to improve the matrix over time. This raises the question of how to propagate changes in the matrix to grammars based on earlier versions. The following three strategies (meant to be used in combination) seem promising: (i) segregate changes that are important to sync to (e.g., changes that affect MRS outputs, fundamental changes to important analyses), (ii) develop a methodology for communicating changes in the matrix, their motivation and their implementation to the user community, and (iii) develop tools for semi-automating resynching
of existing grammars to upgrades of the matrix. These tools could use the type hierarchy to predict where conflicts are likely to arise and bring these to the engineer’s attention, possibly inspired by the approach under development at CSLI for the dynamic maintenance of the LinGO Redwoods treebank (Oepen et al., 2002).

Finally, while initial development of the matrix has been and will continue to be highly centralized, we hope to provide support for proposed matrix improvements from the user community. User feedback will already come in the form of case studies for the library as discussed in Section 5 above, but also potentially in proposals for modification of the matrix drawing on experiences in grammar development. In order to provide users with some cross-linguistic context in which to develop and evaluate such proposals themselves, we intend to provide some sample matrix-derived grammars and corresponding testsuites with the matrix. A user could thus make a proposed change to the matrix, run the testsuites for several languages using the supplied grammars which draw from that changed matrix, and use \[\text{incr tsdb()}\] to determine which phenomena have been affected by the change. It is clear that full automation of this evaluation process will be difficult, but at least some classes of changes to the matrix will permit this kind of quick cross-linguistic feedback to users with only a modest amount of additional infrastructure.

7 Conclusion

This project carries linguistic, computational, and practical interest. The linguistic interest lies in the HPSG community’s general bottom-up approach to language universals, which involves aiming for good coverage of a variety of languages first, and leaving the task of what they have in common for later. (Of course, theory building is never purely data-driven, and there are substantive hypotheses within HPSG about language universals.) Now that we have implementations with fairly extensive coverage for a somewhat typologically diverse set of languages, it is a good time to take the next step in this program, working to extract and generalize what is similar across these existing wide-coverage grammars. Moreover, the central role of types in the representation of linguistic generalizations enables the kind of underspecification which is useful for expressing what is common among related languages while allowing for the further specialization which necessarily distinguishes one language from another.

The computational interest is threefold. First there is the question of what formal devices the grammar matrix will require. Should it include defaults? What about domain union (linearization theory)? The selection and deployment of formal devices should be informed by on-going research on processing schemes, and here the crosslinguistic perspective can be particularly helpful. Where there are several equivalent analyses of the same linguistic phenomena (e.g., morphosyntactic ambiguity or optionality), the choice of analysis can have processing implications that aren’t necessarily apparent in a single grammar. Second, having a set of wide-coverage HPSGs with fairly standardized fundamentals could prove interesting for research on stochastic processing and disambiguation, especially if the languages differ in gross typological features such as word order. Finally, there are also computational issues involved in how the grammar matrix would evolve over time as it is used in new grammars. The matrix enables the developer of a grammar for a new language to get a quick start on producing a system that parses and generates with non-trivial semantics, while also building the foundation for a wide-coverage grammar of the language. But the matrix itself may well change in parallel with the development of the grammar for a particular language, so appropriate mechanisms must be developed to support the merging of enhancements to both.

There is also practical industrial benefit to this project. Companies that are consumers of these grammars benefit when grammars of multiple languages work with the same parsing and generation algorithms and produce standardized semantic representations derived from a rich, linguistically motivated syntax-semantics interface. More importantly, the grammar matrix will help to remove one of the primary remaining obstacles to commercial deployment of grammars of this type and indeed of the commercial use of deep linguistic analysis: the immense cost of developing the resource.

Acknowledgements

Since the grammar matrix draws on prior research and existing grammars, it necessarily reflects contributions from many people. Rob Malouf, Jeff Smith, John Beavers, and Kathryn
Campbell-Kibler have contributed to the LinGO ERG; Melanie Siegel is the original developer for the Japanese grammar. Tim Baldwin, Ann Copestake, Ivan Sag, Tom Wasow, and other members of the LinGO Laboratory at CSLI have had a great deal of influence on the design of the grammatical analyses and corresponding MRS representations. Warmest thanks to Lars Hellan and his colleagues at NTNU and Jan Tore Lønning and his students at Oslo University for their cooperation, patience, and tolerance.

References
Parallel Distributed Grammar Engineering for Practical Applications

Stephan Oepen*, Emily M. Bender*, Uli Callmeier†, Dan Flickinger*, Melanie Siegel*

*CSLI Stanford
Stanford (CA)

†YY Technologies
Mountain View (CA)

DFKI GmbH
Saarbrücken (Germany)

oebender dan @csli.stanford.edu
uc dan @yy.com siegel@dfki.de

Abstract

Based on a detailed case study of parallel grammar development distributed across two sites, we review some of the requirements for regression testing in grammar engineering, summarize our approach to systematic competence and performance profiling, and discuss our experience with grammar development for a commercial application. If possible, the workshop presentation will be organized around a software demonstration.

1 Background

The production of large-scale constraint-based grammars and suitable processing environments is a labour- and time-intensive process that, maybe, has become somewhat of a growth industry over the past few years, as companies explore products that incorporate grammar-based language processing. Many broad-coverage grammars have been developed over several years, sometimes decades, typically coordinated by a single grammarian who would often draw on additional contributors (e.g. the three HPSG implementations developed as part of the VerbMobil effort, see Flickinger, Copes-take, & Sag, 2000, Müller & Kasper, 2000, and Siegel, 2000; or the LFG implementations developed within the ParGram consortium, Butt, King, Niño, & Segond, 1999).

More recently, we also find genuinely shared and distributed development of broad-coverage grammars, and we will use one such initiative as an example—viz. an open-source HPSG implementation for Japanese jointly developed between DFKI Saarbrücken (Germany) and YY Technologies (Mountain View, CA)—to demonstrate the technological and methodological challenges present in distributed grammar and system engineering.

2 Parallel Distributed Grammar Development—A Case Study

The Japanese grammar builds on earlier work performed jointly between DFKI and the Computational Linguistics Department at Saarland University (Germany) within VerbMobil; much like for the German VerbMobil grammar, two people were contributing to the grammar in parallel, one building out syntactic analyses, the other charged with integrating semantic composition into the syntax. This relatively strict separation of responsibilities mostly enabled grammarians to serialize incremental development of the resource: the syntactician would supply a grammar with extended coverage to the semanticist and, at the onset of the following iteration, start subsequent work on syntax from the revised grammar.

In the DFKI–YY cooperation the situation was quite different. Over a period of eight months, both partners had a grammarian working on syntax and semantics simultaneously on a day-to-day basis; both grammarians were submitting changes to a joint, version-controlled source repository and usually would start the work day by retrieving the most recent revisions. At the same time, product building and the development of so-called ‘domain libraries’ (structured collections of knowledge about a specific domain that is instantiated from semantic representations delivered from grammatical analysis) at YY already incorporated the grammar and depended on it for actual, customer-specific contracts. Due to a continuous demand for improvements in coverage and analysis accuracy, the grammar used in the main product line would be updated from the current development version about once or twice a week. Parallel to work on the Japanese grammar (and simultaneous work on grammars for English and Spanish), both the grammar development environment (the
open-source LKB system; Copestake, 2002) and the HPSG run-time component powering the YY linguistic analysis engine (the open-source PET parser; Callmeier, 2002) continued to evolve, as did the YY-proprietary mapping of meaning representations extracted from the HPSG grammars into domain knowledge—all central parts of a complex system of interacting components and constraints.

As has been argued before (see, for example, Oepen & Flickinger, 1998), the nature of a large-scale constraint-based grammar and the subtle interactions of lexical and constructional constraints make it virtually impossible to predict how a change in one part of the grammar affects overall system behaviour. A relatively minor repair in one lexical class, numeral adjectives as in ‘three books were ordered’ for instance, will have the potential of breaking the interaction of that class with the construction deriving named (numeric) entities from a numeral (e.g. as in ‘three is my favourite number’) or the partitive construction (e.g. as in ‘three have arrived already’). A ripple effect of a single change can thus corrupt the semantics produced for any of these cases and in the consequence cause failure or incorrect behaviour in the back-end system. In addition to these quality assurance requirements on grammatical coverage and correctness, the YY application (like most applications for grammar-based linguistic analysis) utilizes a set of hand-constructed parse ranking heuristics that enables the parser to operate in best-first search mode and to return only one reading, i.e. the analysis that is ranked best by the heuristic component. The parse ranking machinery builds on preferences that are associated with individual or classes of lexical items and constructions. The set of preferences is maintained in parallel to the grammar, in a sense providing a layer of performance-oriented annotations over the basic building blocks of the core competence grammar. Without discussing the details of the parse ranking approach, it creates an additional element of uncertainty in assessing grammar changes: since the preference for a specific analysis results implicitly from a series of local preferences (of lexical items and constructions contributing to the complete derivation), introducing additional elements (i.e. new local or global ambiguity) into the search space and subjecting them to the partial ordering can quickly skew the overall result.

Summing up, the grammar and application engineering example presented here illustrates a number of highly typical requirements on the engineering environment. First, all grammarians and system engineers participating in the development process need to keep frequent, detailed, and accurate records of a large number of relevant parameters, including but not limited to grammatical coverage, correctness of syntactic analyses and corresponding semantic forms, parse selection accuracy, and overall system performance. Second, as modifications to the system as a whole are made daily—and sometimes several times each day—all developers must be able to assess the impact of recent changes and track their effects on all relevant parameters; gathering the data and analyzing it must be simple, fast, and automated as much as possible. Third, not all modifications (to the grammar or underlying software) will result in ‘monotonic’ or backwards-compatible effects. A change in the treatment of optional nominal complements, for example, may affect virtually all derivation trees and render a comparison of results at this level uninformative. At the same time, a primarily syntactic change of this nature will not cause an effect in associated meaning representations, so that a semantic equivalence test over analyses should be expected to yield an exact match to earlier results. Hence, the machinery for representation and comparison of relevant parameters needs to facilitate user-level specification of informative tests and evolution criteria. Finally, the metrics used in tracking grammar development cannot be isolated from measurements of system resource consumption and overall performance (specific properties of a grammar may trigger idiosyncrasies or software bugs in a particular version of the processing system); therefore, and to enable exchange of reference points and comparability of experiments, grammarians and system developers alike should use the same, homogenous set of relevant parameters.

3 Integrated Competence and Performance Profiling

The integrated competence and performance profiling methodology and associated engineering platform, dubbed [iner tsdb] (Oepen & Callmeier, 2000)\(^1\) and reviewed in the remainder of this sec-

\(^1\)See ‘http://www.coli.uni-sb.de/itsdb/’ for the (draft) [iner tsdb] user manual, pronunciation rules, and instructions on obtaining and installing the package.
tion, was designed to meet all of the requirements identified in the DFKI–YY case study. Generally speaking, the [incr tsdb()] environment is an integrated package for diagnostics, evaluation, and benchmarking in practical grammar and system engineering. The toolkit implements an approach to grammar development and system optimization that builds on precise empirical data and systematic experimentation, as it has been advocated by, among others, Erbach & Uszkoreit (1990), Erbach (1991), and Carroll (1994). [incr tsdb()] has been integrated with, as of June 2002, nine different constraint-based grammar development and parsing systems (including both environments in use at YY, i.e. the LKB and PET), thus providing a pre-standard reference point for a relatively large (and growing) community of NLP developers. The [incr tsdb()] environment builds on the following components and modules:

- test and reference data stored with annotations in a structured database; annotations can range from minimal information (unique test item identifier, item origin, length et al.) to fine-grained linguistic classifications (e.g. regarding grammaticality and linguistic phenomena presented in an item), as they are represented in the TSLNP test suites, for example (Oepen, Netter, & Klein, 1997);
- tools to browse the available data, identify suitable subsets and feed them through the analysis component of processing systems like the LKB and PET, LiLFes (Makino, Yoshida, Torisawa, & Tsujii, 1998), TRALE (Penn, 2000), PAGE (Uszkoreit et al., 1994), and others;
- the ability to gather a multitude of precise and fine-grained (grammar) competence and (system) performance measures—like the number of readings obtained per test item, various time and memory usage statistics, ambiguity and non-determinism metrics, and salient properties of the result structures—and store them in a uniform, platform-independence data format as a competence and performance profile; and
- graphical facilities to inspect the resulting profiles, analyze system competence (i.e. grammatical coverage and overgeneration) and performance (e.g. cpu time and memory usage, parser search space, constraint solver workload, and others) at variable granularities, aggregate, correlate, and visualize the data, and compare among profiles obtained from previous grammar or system versions or other processing environments.

As it is depicted in Figure 1, the [incr tsdb()] architecture can be broken down into three major parts: (i) the underlying database management system (DBMS), (ii) the batch control and statistics kernel (providing a C and Lisp application program interface to client systems that can be distributed across the network), and (iii) the graphical user interface (GUI). Although, historically, the DBMS was developed independently and the kernel can be operated without the GUI, the full functionality of the integrated competence and performance laboratory—as demonstrated below—only emerges from the combination of all three components. Likewise, the flexibility of a clearly defined API to client systems and its ability to parallelize batch processing and distribute test runs across the network have greatly contributed to the success of the package. The following paragraphs review some of the fundamental aspects in more detail, sketch essential functionality, and comment on how they have been exploited in the DFKI–YY cooperation.

**Abstraction over Processors** The [incr tsdb()] environment, by virtue of its generalized profile format, abstracts over specific processing environments. While grammar engineers in the DFKI–YY collaboration regularly use both the LKB (primarily for interactive development) and PET (mostly for batch testing and the assessment
of results obtained in the YY production environment), usage of the \([\text{incr tsdb}]()\) profile analysis routines in most aspects hides the specifics of the token processor used in obtaining a profile. Both platforms interpret the same typed feature structure formalism, load the same set of grammar source files, and (unless malfunctioning) produce equivalent results. Using \([\text{incr tsdb}]()\), grammarians can obtain summary views of grammatical coverage and overgeneration, inspect relevant subsets of the available data, break down analysis views according to various aggregation schemes, and zoom in on specific aggregates or individual test items as appropriate. Moreover, processing results obtained from the (far more efficient) PET parser (that has no visualization or debugging support built in), once recorded as an \([\text{incr tsdb}]()\) profile, can be used in conjunction with the LKB (contingent on the use of identical grammars), thereby facilitating graphical inspection of parse trees and semantic formulae.

**Parallelization of Test Runs** The \([\text{incr tsdb}]()\) architecture (see Figure 1) separates the batch control and statistics kernel from what is referred to as client processors (i.e. parsing systems like the LKB or PET) through an application program interface (API) and the Parallel Virtual Machine (PVM; Geist, Bequelin, Dongarra, Manchek, & Sunderam, 1994) message-passing protocol layer. The use of PVM—in connection with task scheduling, error recovery, and roll-over facilities in the \([\text{incr tsdb}]()\) kernel—enables developers to transparently parallelize and distribute execution of batch processing. At YY, grammarians had a cluster of networked Linux compute servers configured as a single PVM instance, so that execution of a test run—using the efficient PET run-time engine—could be completed as a matter of a few seconds. The combination of near-instantaneous profile creation and \([\text{incr tsdb}]()\) facilities for quick, semi-automated assessment of relevant changes (see below) enabled developers to pursue a strongly empiricist style of grammar engineering, assessing changes and their effects on actual system behavior in small increments (often many times per hour).

**Structured Comparison** One of the facilities that has proven particularly useful in the distributed grammar engineering setup outlined in Section 2 above is the flexible comparison of competence and performance profiles. The \([\text{incr tsdb}]()\) package eases comparison of results on a per-item basis, using an approach similar to Unix \texttt{diff()}1, but generalized for structured data sets. By selection of a set of parameters for intersection (and optionally a comparison predicate), the user interface allows browsing the subset of test items (and associated results) that fail to match in the selected properties. One dimension that grammarians found especially useful in intersecting profiles is on the number of readings assigned per item—detecting where coverage was lost or added—and on derivation trees (bracketed structures labeled with rule names and identifiers of lexical items) associated with each parser analysis—assessing where analyses have changed. Additionally, using a user-supplied equivalence predicate, the same technique was regularly used at YY to track the evolution of meaning representations (as they form the interface from linguistic analysis into the back-end knowledge processing engine), both for all readings and the analysis ranked best by the parse selection heuristics.

**Zooming and Interactive Debugging** In analysing a new competence and performance profile, grammarians typically start from summary views (overall grammatical coverage, say), then single out relevant (or suspicious) subsets of profile data, and often end up zooming in to the level of individual test items. For most \([\text{incr tsdb}]()\) analysis views the ‘success’ criteria can be varied according to user decisions: in assessing grammatical coverage, for example, the scoring function can refer to virtually arbitrary profile elements—ranging from the most basic coverage measure (assigning at least one reading) to more refined or application-specific metrics, the production of a well-formed meaning representation, say. Although the general approach allows output annotations on the test data (full or partial constituent structure descriptions, for example), developers so far have found the incremental, semi-automated comparison against earlier results a more adequate means of regression testing. It would appear that, especially in an application-driven and tightly scheduled engineering situation like the DFKI—YY partnership, the pace of evolution and general lack of locality in changes (see the examples discussed in Section 2) precludes the construction of a static, ‘gold-standard’ target for comparison. Instead, the structured comparison facilities of \([\text{incr tsdb}]()\) enable developers to incrementally approximate target results and, even
in a highly dynamic environment where grammar and processing environment evolve in parallel, track changes and identify regression with great confidence.

4 Looking Back—Quantifying Evolution

Over time, the \texttt{[incr tsdb[]]} profile storage accumulates precise data on the grammar development process. Figure 2 summarizes two aspects of grammatical evolution compiled over a five-month period (and representing some 130 profiles that grammarians put aside for future reference): grammatical coverage over two representative samples of customer data—one for an on-line banking application, the other from an electronic stock trading domain—is contrasted with the development of global ambiguity (i.e. the average number of analyses assigned to each test item). As should be expected, grammatical coverage on both data sets increases significantly as grammar development focuses on these domains ('banking' for the first three months, 'trading' from there on). While the collection of available profiles, apparently, includes a number of data points corresponding to ‘failed’ experiments (fairly dramatic losses in coverage), the larger picture shows mostly monotonic improvement in coverage. As a control experiment, the coverage graph includes another data point for the 'banking' data towards the end of the reporting period. Two months of focussed development on the 'trading' domain have not negatively affected grammatical coverage on the data set used earlier. Corresponding to the (desirable) increase in coverage, the graph on the right of Figure 2 depicts the evolution of grammatical ambiguity. As hand-built linguistic grammars put great emphasis on the precision of grammatical analysis and the exclusion of ungrammatical input, the overall average of readings assigned to each sentence varies around relatively small numbers. For the moderately complex email data the grammar often assigns less than ten analyses, rarely more than a few dozens. However, not surprisingly the addition of grammatical coverage comes with a sharp increase in ambiguity (which may indicate overgeneration); the graphs in Figure 2 clearly show that, once coverage on the 'trading' data was above eighty per cent, grammarians shifted their engineering focus on ‘tightening’ the grammar, i.e. the elimination of spurious ambiguity and overgeneration (see Siegel & Bender, 2002, for details on the grammar).

Another view on grammar evolution is presented in Figure 3, depicting the ‘size’ of the Japanese grammar over the same five-month development cycle. Although measuring the size of

\footnote{Quantifying input complexity for Japanese is a non-trivial task, as the count of the number of input words would depend on the approach to string segmentation used in a specific system (the fairly aggressive tokenizer of ChaSen, Asahara & Matsumoto, 2000, in our case); to avoid potential for confusion, we report input complexity in the (overly system-specific) number of lexical items stipulated by the grammar instead: around 50 and 80, on average, for the 'banking' and 'trading' data sets, respectively (as of February 2002).}
computational grammars is a difficult challenge, for the HPSG framework two metrics suggest themselves: the number of types (i.e. the size of the grammatical ontology) and the number of grammar rules (i.e. the inventory of construction types). As would be expected, both numbers increase more or less monotonically over the reporting period, where the shift of focus from the ‘banking’ into the ‘trading’ domain is marked with a sharp increase in (primarily lexical) types. Contrasted to the significant gains in grammatical coverage (a relative improvement of more than seventy per cent on the ‘banking’ data), the increase in grammar size is moderate, though: around fifteen and twenty per cent in the number of types and rules, respectively.

5 Conclusions
At YY and cooperating partners (primarily DFKI Saarbrücken and CSLI Stanford), grammarians (for all languages) as well as developers of both the grammar development tools and of the production system all used the competence and performance profiling environment as part of their daily engineering toolbox. The combination of [incr tsdb()] facilities to parallelize test run processing and a break-through in client system efficiency (using the PET parser; Callmeier, 2002) has created an experimental development environment where grammarians can obtain near-instantaneous feedback on the effects of changes they explore.

For the Japanese grammar specifically, the grammar developers at both ends would typically spend the first ten to twenty minutes of the day obtaining fresh profiles for a number of shared test sets and diagnostic corpora, thereby assessing the most recent set of changes through empirical analysis of their effects. In conjunction with a certain rigor in documentation and communication, it was the ability of both partners to regularly, quickly, and semi-automatically monitor the evolution of the joint resource with great confidence that has enabled truly parallel development of a single, shared HPSG grammar across continents. Within a relatively short time, the partners succeeded in adapting an existing grammar to a new genre (email rather than spoken language) and domain (customer service requests rather than appointment scheduling), greatly extending grammatical coverage (from initially around forty to above ninety per cent on representative customer corpora), and incorporating the grammar-based analysis engine into a commercial product. And even though in February 2002, for business reasons, YY decided to reorganize grammar development for Japanese, the distributed, parallel grammar development effort positively demonstrates that methodological and technological advances in constraint-based grammar engineering have enabled commercial development and deployment of broad-coverage HPSG implementations, a paradigm that until recently was often believed to still lack the maturity for real-world applications.

Acknowledgements
The DFKI – YY partnership involved a large group of people at both sites. We would like to thank Kirk Oatman, co-founder of YY and first CEO, and Hans Uszkoreit, Scientific Director at DFKI, for their initiative and whole-hearted support to the project; it takes vision for both corporate and academic types to jointly develop an open-source resource. Atsuko Shimada (from Saarland University), as part of a two-month internship at YY, has greatly contributed to the preparation of representative data samples, development of robust preprocessing rules, and extensions to lexical coverage. Our colleague and friend Asahara-san (of the Nara Advanced Institute of Technology, Japan), co-developer of the open-source ChaSen tokenizer and morphological analyzer for Japanese, was instrumental in the integration of ChaSen into the YY product and also helped a lot in adapting and (sometimes) fixing tokenization and morphology.
References


A Development Environment for Large-scale Multi-lingual Parsing Systems

Hisami Suzuki
Microsoft Research
One Microsoft Way
Redmond WA 98052 USA
hisamis@microsoft.com

Abstract
We describe the development environment available to linguistic developers in our lab in writing large-scale grammars for multiple languages. The environment consists of the tools that assist writing linguistic rules and running regression testing against large corpora, both of which are indispensable for realistic development of large-scale parsing systems. We also emphasize the importance of parser efficiency as an integral part of efficient parser development. The tools and methods described in this paper are actively used in the daily development of broad-coverage natural language understanding systems in seven languages (Chinese, English, French, German, Japanese, Korean and Spanish).

1 Introduction
The goal of the grammar development at Microsoft Research is to build robust, broad-coverage analysis and generation systems for multiple languages. The runtime system is referred to as NLPWin, which provides the grammar development environment described in this paper. The graphical user interface of NLPWin is shown in Figure A in the Appendix. The system is modular in that the linguistic code is separate from non-linguistic code. All languages share the same parsing engine, which is a bottom-up chart parser and is fully Unicode-enabled. Linguistic code itself is also modular, in that it can be specific to a particular language (e.g., syntax rules) or can be largely shared across languages (e.g., semantic mapping rules). Linguistic rules are written in a proprietary language called G; a sample syntax rule written in G is given in Figure B in the Appendix. G-rules are translated into C, yet they are more convenient for a linguist to use than C, as it gives special notational support to attribute-value data structures used within the system. The rule and data structure formalisms are shared by all languages; for details, see Jensen et al. (1993) and Heidorn (2000).

In this paper, we describe the tools and methods for the cross-linguistic development of analysis components of our system, which consists of three major modules: (i) the tokenization component, which performs word segmentation (in the case of Chinese and Japanese) and morphological analysis; (ii) the parsing component, which performs phrase-structure analysis and creates parse tree(s); (iii) the Logical Form (LF) component, which computes the basic predicate-argument structure from parse tree(s). In this paper, we focus on the tools and the methods for the development of parsing and LF components, which are essentially the same.

For an efficient development of a computational grammar of these modules, we find it necessary to have a development environment that can provide immediate feedback to the grammar-writer of the changes he or she has made. We have three types of tools in our system to meet these requirements:

- Tools for linguistic rule writing: these include the tools that let linguists navigate through the final and intermediate parse trees, and trace rule application (Section 2).
- Tools for grammar testing: these tools allow linguists to compare results of two versions of

---

1 This component is further divided into the Sketch component, which produces trees with default attachment of constituents, and the Portrait component, which finds the best attachment sites (Heidorn, 2000).
2 LF is computed from a surface syntax tree via a level of representation called Language-Neutral Syntax (LNS), which serves as an interface to various semantic representations including predicate-argument structure. For a more detailed description of LNS, see Campbell and Suzuki (2002).
3 Similar tools and methods are also available for the development of sentence realization component.
grammar, and update the database of desired output structures (called regression suites, Section 3).

* A very fast processing environment (Section 4).

These tools are described in the following three sections of this paper. Section 5 gives a summary and suggests directions for future research.

2 Tools for linguistic rule writing

In this section we present the tools available for the development of the parsing component. The output structure of parsing is graphically represented as a phrase-structure tree, as in Figure 1 above. Various functionalities are available to navigate through this tree as well as intermediate (or failed) representations, by simple operations such as double-clicking the node in the user interface, or by typing in commands in the Command window, which can be invoked by the Command menu in the user interface (see Figure A). Below is a selected set of examples of tree navigation functionalities which are essential to the fast development of linguistic rules:
Accessible records
At any point, a linguist can access the records underlying the parse tree by double-clicking the node. The record for a node is comprised of lexical and morphological information, syntactic and functional features and attributes, as well as pointers to the sub-constituents and parent of the node. For example, double-clicking on the VERB1 node in Figure 1 will display the record structure in Figure 2.

Derivational tree
We can also display the history of rule application in graphical form, as in Figure 3. Any node in the history tree (called the derivational tree) can also be double-clicked in order to access the record underlying it.

Apply Rule and Rule Explain
Rule Explain shows the application of the rule underlying the formation of a node in the tree. The rule application is displayed using a color-coded display to highlight successful conditions (green), failed conditions (red) and the actions performed on the resulting record (purple) on the rules such as the one displayed in Figure B. The display is available for both successful and failed rule applications: we can access the Rule Explain display by double-clicking the resulting node, or we can manually apply any rule to any constituent to bring up Rule Explain.

Compare
Parsed trees can be quite large and it may be difficult to determine exactly where two trees differ from each other. In such a case, trees and nodes can be easily compared to detect subtle differences in composition or rule history by the Compare function.

Display trees
This command is particularly useful in checking the edges of possible, intermediate constituents. It displays all the partial trees with a certain label that includes a particular node or spans over specified nodes. The following are some examples of possible variations in the query:
(a) display trees VP 1 5
(b) display trees NP NOUN
(c) display trees AJP

(a) displays all VPs that span from position 1 to 5;
(b) displays all NPs that include the node NOUN;
and (c) displays all possible subtrees whose nodetype (label) is AJP.

Tree filters
This functionality does not directly assist the grammarian in writing rules, yet is extremely useful in collecting and examining particular linguistic constructions of interest that are output by the parser. The linguistic developer can write custom-made tree filters in G, which traverses the parse trees or LF structures and exports only the information needed for a particular purpose, or only those sentences with particular linguistic configurations. Tree filters are also convenient in creating a linguistic annotation for external applications.

The tools described in this section enable linguists to inspect the effect of grammar changes in detail, with the information of how exactly a particular rule applied or failed. These tools are used in the context of daily grammar development, which we describe in the next section.

3 Process of grammar development
3.1 Incremental grammar testing and creation of regression suites
The standard practice of parser development within our group is schematically shown in Figure 4. The grammarian for each language processes a text file with input sentences and adds only the sentences with desired parses to what we call a master file, which contains the sentences and their target structure. A collection of master files is called a regression suite. A regression suite thus contains the target structures given a particular version of the grammar. When new grammar changes are made in order to accommodate a new sentence or construction, the linguist runs the new grammar against the regression suite (called regression testing) to examine the consequences of the changes to the grammar. When differences are found, they are kept in *.diff files and are displayed in two colors, highlighting the differences. Figure C in the Appendix is an example display of a difference (unfortunately in black and white): the highlights in green (here the first three lines) correspond to the analysis in the master file, while those in red (the next three lines) indicate the new
analysis introduced by the new grammar rules. The lines that did not change are grayed out. If the change is an improvement, the developer can choose to update the master file by double-clicking on the sentence number (in purple), adding the sentence or construction that is newly accommodated by the parser to the regression suite. If the change is evaluated as negative, the linguistic developer reworks the rules that caused the regression.

3.2 Testing against relative standards
As is described above, we run regression tests against the machine-created master files rather than against an independent set of hand-annotated target corpora. The test is therefore incremental and relative, in that new sentences and their target structures are constantly added as the grammar develops, and what it measures is not the coverage against an absolute standard, but the coverage improvements relative to the output of an old version of the grammar.

The incremental and relative testing method has proven to facilitate the development of a broad-coverage parsing system in some important respects. First, it ensures that the desired structures in the master files are always current. Because the master files are constantly incremented and updated using the most recent version of the grammar, they will never become obsolete should the target structures change. The ease of maintenance of the regression suite is one of the key features contributing to the usefulness of the regression suite in our daily development work.

Secondly, because the master files are created automatically rather than by hand, the resultant annotation is guaranteed to be consistent. Creating a test corpus for parser evaluation by hand is known to be an extremely laborious, inconsistency-prone task, especially when the tagging is performed on real-life data. In addition, a broad-coverage grammar must also work with input strings that are not necessarily well-formed, including sentence fragments, ungrammatical sentences and extreme colloquialisms. Hand-annotating these structures may either be impossible or extremely error-prone. In contrast, by annotating them automatically using the output of the parser, these structures can be added to our regression suite easily and consistently. Effects of later grammar changes can easily be detected by running the regression testing as part of the regular development process.

Finally, incremental and relative testing makes the parser development data-driven rather than being dictated by a theory. This is an important feature for a large-scale system. Though it is eventually up to the grammarian to accept or reject a particular analysis, the system always provides a candidate analysis for any input string, which facilitates the rapid creation of the master files. It also allows linguistic developers to experiment on the grammar code in the following sense: assume that there is a sentence or a construction that allows multiple linguistically valid analyses, and that there is no obvious reason to prefer one to the other, a situation that arises often in the development of a broad-coverage grammar. In this case, the grammarian can temporarily choose one of the structures as a target, and add it to the regression suite. If the target structure the grammarian has selected is inconsistent with the rest of the grammar, it will constantly come back as a regression (difference) when further changes to the other parts of the grammar are made, because the assumption implicit in the tentative target structure is not consistent with the rest of the grammar. Once the change is made to the target structure that is consistent with the remainder of the grammar, it typically stops appearing as a difference in regression tests. The data-driven nature of development therefore helps the grammarians to proceed with grammar development even when there is indeterminacy in the target structure. Regular regression testing over large corpora

---

4 One piece of evidence for this statement is that the bracketing guidelines for Penn Treebank project (Bies et al. 1995) consist of over 300 pages of documentation for annotating relatively homogeneous text.
ensures that any outlandish analyses have only a short life span in our regression suites.

Possible disadvantages of testing against a relative standard include: (i) it is difficult to get a feel for how mature the grammar is in general; (ii) it makes the comparison across different systems difficult. The first problem is addressed partially by running evaluation testing against blind benchmark corpora, which consists of sentences never used in the grammar development. The parser coverage is automatically measured in terms of the number of sentences that received at least one spanning parse, versus those that failed to receive any spanning analysis.

Testing and comparing parser performance across different systems is an extremely difficult task, given different aims and grammatical foundations. One possibility, which is currently pursued in our group, is to develop a metric that enables comparison with manually created golden standards, as they have become more widely available for various languages, such as the Penn Treebank for Chinese and English, NEGRA corpus for German, and Kyoto Corpus for Japanese.

Ultimately, the parser output must be compared and evaluated at the level of an application that uses the result of linguistic analysis. Campbell et al. (2002) is an attempt to use machine translation as a test bed for a multi-lingual parsing system.

4 Parser efficiency as part of efficient parser development

For a development of a truly broad-coverage parser, it is critical that grammar changes are constantly verified against a very large set of sentences, and that the time for feedback is minimal. The efficiency of the parsing engine is thus inseparable from efficient grammar development.

Our parsing engine is already quite fast: for example, our English system currently parses Section 21 of Penn Wall Street Journal (WSJ) Treebank (1,671 sentences) in 110 seconds (or about 15 sentences/sec) on a standard machine (993MHz Pentium III with 512MB RAM); this performance is comparable across languages.

Speed improvements are usually performed by non-linguistic developers following standard optimization techniques. We use internal profiling tools to identify performance bottlenecks, and make a special effort to ensure that the G-to-C translator generates efficient C-code. Because the linguistic code is independent of the non-linguistic code of the system, the parsers for any language can immediately benefit from performance improvements made at the system level.

For regression testing, we also have a means to distribute the processing onto multiple CPUs: the processing cluster currently consists of 19 machines with 2 CPUs each (500MHz, 128-512MB RAM), which parses the entire WSJ section of Penn Treebank (49,208 sentences) in 3 minutes and 10 seconds (or 259 sentences/sec), and a one million-sentence Nikkei newspaper corpus of Japanese in about 30 minutes (550+ sentences/sec).

In daily grammar development, each grammarian typically works with a regression suite consisting of 10,000 to 30,000 sentences at various levels of analyses; the time required for processing a regression suite is 2 to 6 minutes. In addition, automatic regression testing is run nightly against relevant regression suites using the most recent builds of the system, ensuring that no negative impact is made by any changes introduced during the day.

In this section, we have discussed the issue of parser efficiency from the perspective of grammar development. Our processing environment enables immediate feedback to grammar changes over very large corpora, and is thus an essential part of the development environment for a broad-coverage parser.

5 Conclusion

In this paper we have described the tools and methods for a development of large-scale parsing systems. We have argued that constant testing of the grammar against a large regression suite is the central part of the daily grammar development, and that the tools and methods described in this paper are indispensable for maximizing the productivity of linguistic developers. Though the tools are specific to NLPWin, we believe that the general practice of grammar development presented in this paper is of interest to anyone engaged in grammar development under any grammar formalism.

As a cross-linguistic development environment for analysis and generation components, some of

---

5 We use standard version control software to manage both linguistic and non-linguistic source code.
the properties of NLPWin discussed in this paper are shared with such projects as ParGram (Butt et al., 1999). One of the main differences between ParGram and NLPWin is that the latter has so far been developed and used at one site. As there are more parsers available in many languages, it would be interesting to see if externally developed components can be plugged into NLPWin at the level of LNS. Such research is left for the future as a possible extension to the modularity and cross-linguistic aspect of NLPWin.

Acknowledgements

This paper presents the work that has been designed and implemented by many people in the NLP Group at Microsoft Research, particularly George Heidorn and Karen Jensen.

References


Appendix

Figure A: Graphical user interface of NLPWin
Figure B: Example of a phrase-structure rule

```
AVPwAVP1:

AVP#1 ("Comma & "Conj & "NoAdv & "Top & NodeType(Head)="IJ" &
       (NodeType "in? set(AVPNP AVPVP) | Compr | Intens | Time | NTimes) &
       (Advzr -> ("Adv(Lex) & Intens)) &
       (ModalAdvz -> Intens) &
       (Nconj -> ("Conj(Lex) & Lemma"in? set(同)))
AVP#2 ("AVPcoord & "Conj & "Kakari & "Nconj & "NoAdv & "Wh &
       NodeType "in? set(AVPNP AVPVP) & NodeType (Head)="IJ" &
       toktest ("ADV", Ft(AVP#1), Lt, []) &
       toktest (-1, Ft(AVP#1), Lt (first (Factrecs)), []) &
       (Compr (AVP#1) -> (Advzr | Quant)) &
       (Demo (AVP#1) -> _j_state_zyoo) &
       (Intens -> Intens (AVP#1)) &
       (Time -> (Quant | _every_mail)) &
       (Time (AVP#1) -> (Intens (AVP#1)) | Compr (AVP#1))) &
       Lem="hoka" &
       Lemma="in? set(早)
       --> AVP { %AVP#2; Temp-segrec(%AVP#1; -Quant;
               if (Quant) NodeType="QUANP";
               if (Comp & "Mai5 (AVP#2) ~Min; );
               Prmmod=Temp+Prmmod; Degree=Degree (AVP); -Temp;
               if (Compr (AVP#1) | (Lem (AVP)="mou" & Quant)) +Comp;
               }
```

Figure C: Example of the difference display with master file

```
<table>
<thead>
<tr>
<th>DECLI</th>
<th>NP1</th>
<th>RELCLI</th>
<th>NP2</th>
<th>NOUN1*</th>
<th>&quot;クック 語島&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>PP1</td>
<td>POSP1*</td>
<td>&quot;で&quot;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PP1</td>
<td>POSP1*</td>
<td>&quot;もっとも&quot;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOUN1*</td>
<td>&quot;産業&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PP2</td>
<td>POSP2*</td>
<td>&quot;は&quot;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOUN1*</td>
<td>&quot;観光業&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VERB*</td>
<td>&quot;で&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AUXP1</td>
<td>VERB2*</td>
<td>&quot;ある&quot;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHAR!</td>
<td>&quot;&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>-----</td>
<td>--------</td>
<td>-----</td>
<td>--------</td>
<td>-------------</td>
</tr>
</tbody>
</table>

---

This rule, taken from the NLPWin Japanese grammar, is read as "AVP with AVP to the left", which takes two adjacent nodes, whose categories are both AVP (adverbial phrase), and creates a new node that spans both of the input nodes, also labeled as AVP, whose head is the second AVP of the left-hand side of the rule (indicated by %AVP#2 in the right-hand side of the rule). A rule can be as small as this one, or can be very large (up to hundreds of lines of code). Each language in NLPWin has about 100 to 150 phrase-structure rules, in 10 to 20 files that are language-specific. LF rules are also written in G and have a similar format, but the files are shared by all languages, as are most rules, to ensure the output of the LF component is consistent across languages.
Adapting Existing Grammars: The XLE Experience

Ronald M. Kaplan and Tracy Holloway King and John T. Maxwell III
Palo Alto Research Center
Palo Alto, CA 94304 USA
{kaplan, thking, maxwell}@parc.com

Abstract
We report on the XLE parser and grammar development platform (Maxwell and Kaplan, 1993) and describe how a basic Lexical Functional Grammar for English has been adapted to two different corpora (newspaper text and copier repair tips).

1 Introduction
Large-scale grammar development platforms should be able to be used to develop grammars for a wide variety of purposes. In this paper, we report on the XLE system (Maxwell and Kaplan, 1993), a parser and grammar development platform for Lexical Functional Grammars. We describe some of the strategies and notational devices that enable the basic English grammar developed for the ParGram project (Butt et al., 1999; Butt et al., 2002) to be adapted to two corpora with different properties.

1.1 The Corpora
The STANDARD ParGram English grammar covers the core phenomena of English (e.g., main and subordinate clauses, noun phrases, adjectives and adverbs, prepositional phrases, coordination; see (Butt et al., 1999)). We have built two different specialized grammars on top of this: the EUREKA grammar and the WSJ grammar.

The EUREKA grammar parses the Eureka corpus of copier repair tips, a collection of documents offering suggestions for how to diagnose and fix particular copier malfunctions. These informal and unedited documents were contributed by copier repair technicians, and the corpus is characterized by a significant amount of ungrammatical input (e.g., typos, incorrect punctuation, telegraphic sentences) and much technical terminology (1). The goal of parsing this corpus is to provide input to a semantics and world-knowledge reasoning application (Ev-erett et al., 2001).

(1) a. (SOLUTION 27032 70) If exhibiting 10-132 faults replace the pre-fuser transport sensor (Q10-130).
   b. (SOLUTION 27240 80) 4. Enter into the machine log, the changes that have been made.

The WSJ grammar covers the UPenn Wall Street Journal (WSJ) treebank sentences (Marcus et al., 1994). This corpus is characterized by long sentences with many direct quotes and proper names, (2a). In addition, for evaluation and training purposes we also parsed a version of this corpus marked up with labeled brackets and part-of-speech tags, as in (2b). Riezler et al. (2002) report on our WSJ parsing experiments.

(2) a. But since 1981, Kirk Horse Insurance Inc. of Lexington, Ky. has grabbed a 20% stake of the market.
   b. But since 1981, [NP-SBJ Kirk Horse Insurance Inc. of Lexington, Ky.] has/VBZ grabbed/VBN [NP a 20% stake of the market].

2 Priority-based Grammar Specialization
The XLE system is designed so that the grammar writer can build specialized grammars by both extending and restricting another grammar (in our case the base grammar is the STANDARD ParGram English grammar). An LFG grammar is presented to the XLE system in a priority-ordered sequence of files containing phrase-structure rules, lexical entries, abbreviatory macros and templates, feature declarations, and finite-state transducers for tokenization and morphological analysis. XLE is applied to a single root file holding a CONFIGURATION that identifies all the other files containing relevant linguistic specifications, that indicates how
those components are to be assembled into a complete grammar, and that specifies certain parameters that control how that grammar is to be interpreted. A key idea is that there can be only one definition of an item of a given type with a particular name (e.g., there can be only one NP rule although that single rule can have many alternative expansions), and items in a higher priority file override lower priority items of the same type with the same name. This set up is similar to the priority-override scheme of the earlier LFG Grammar Writer's Workbench (Kaplan and Maxwell, 1996).

This arrangement makes it relatively easy to construct a specialized grammar from a pre-existing standard. The specialized grammar is defined by a CONFIGURATION in its own root file that specifies the relevant STANDARD grammar files as well as the new files for the specialized grammar. The files for the specialized grammar can also contain items of different types (phrase-structure rules, lexical entries, templates, etc.), and they are ordered with higher priority than the STANDARD files.

Consider the configuration for the EUREKA grammar. It specifies all of the STANDARD grammar files as well as its own rule, template, lexicon, and morphology files. A part of this configuration is shown in (3) (the notationtemplates.lfg are shared by all the languages’ grammars, not just English).

\[
(3) \text{FILES } \ldots /\text{standard/english-lexicons.lfg} \\
/\text{standard/english-rules.lfg} \\
/\text{standard/english-templates.lfg} \\
/\text{./common/notationtemplates.lfg} \\
\text{english-eureka-morphconfig} \\
\text{eureka-lexicons.lfg} \\
\text{eureka-rules.lfg} \\
\text{eureka-templates.lfg} \\
\]

This configuration specifies that the EUREKA rules, templates, and lexical entries are given priority over the STANDARD items by putting the special EUREKA files at the end of the list. Thus, if the \(./\text{standard/english-rules.lfg} \) and \(\text{eureka-rules.lfg} \) files both contain a rule expanding the NP category, the one from the STANDARD file will be discarded in favor of the EUREKA rule.

In the following subsections, we provide several illustrations of how simple overriding has been used for the EUREKA and WSJ grammar extensions.

2.1 Rules

The override convention makes it possible to: add rules (e.g., for new or idiosyncratic constructions); delete rules (e.g., to block constructions not found in the new corpus); and modify rules to allow different daughter sequences.

Rules may need to be added to allow for corpus-specific constructions. This is illustrated in the EUREKA corpus by the identifier information that precedes each sentence, as in (1). In order to parse this substring, a new category (FIELD) was defined with an expansion that covers the identifier information followed by the usual ROOT category of the STANDARD grammar. The top-level category is one of the parameters of a configuration, and the EUREKA CONFIGURATION specifies that FIELD instead of the STANDARD ROOT is the start-symbol of the grammar. Thus the EUREKA grammar produces the tree in (4) and functional-structure in (5) for (1a).

\[
(4) \text{FIELD} \\
( \text{LP SOLUTION 27032 70 ) …} \\
\text{RP} \\
\text{ROOT} \\
\]

\[
(5) \begin{bmatrix}
\text{PRED} & \text{’replace<SUBJ, OBJ>}' \\
\text{SUBJ} & [\ldots ] \\
\text{OBJ} & [\ldots ] \\
\ldots & \\
\text{FIELD} & \text{solution} \\
\text{TIP-ID} & 27032 \\
\text{SUB-TIP-ID} & 70 \\
\end{bmatrix}
\]

It is unusual in practice to need to delete a rule, i.e., to eliminate completely the possibility of expanding a given category of the STANDARD grammar. This is generally only motivated when the specialized grammar applies to a domain where certain constructions are rarely encountered, if at all. Although there has been no need to delete rules for the EUREKA and WSJ corpora, the override convention also provides a natural way of achieving this effect. For example, topicalization is extremely rare in the Eureka corpus and the STANDARD topicalization rule sometimes introduces parsing inefficiency. This can be avoided by having the high priority EUREKA file replace the STANDARD rule with the one in (6).

\[
(6) \text{CPtop} \longrightarrow . \\
\]

This vacuous rule expands the CPtop category to the empty language, the language containing no strings;
so, this category is effectively removed from the grammar.

Perhaps the most common change is to make modifications to the behavior of existing rules. The most direct way of doing this is simply to define a new, higher priority expansion of the same left-hand category. Since XLE only allows a single rule for a given category, the old rule is discarded and the new one comes into play. The new rule can be arbitrarily different from the STANDARD one, but this is not typically the case. It is much more common that the specialized version incorporates most of the behaviors of the original, with minor extensions or restrictions. One way of producing the modified behavior is to create a new rule that includes a copy of some or all of the STANDARD rule’s right side along with new material, and to give the new definition higher priority than the old. For example, plurals in the Eu-

perhaps reasonable for relatively stable and simple grammars. The Nbody category can be suppressed in the tree structure by invoking this rule as a macro (notationally indicated as @Nbody).

(11) \text{ROOT} \rightarrow \{\ \text{@DECL-BODY} \text{@DECL-PUNCT} \mid \text{@INT-BODY} \text{@INT-PUNCT} \mid \text{@HEADER} \}.

In the STANDARD grammar, the DECL-PUNCT macro is defined as in (12a). However, this must be modified in the EUREKA grammar because the punctuation is much sloppier and often does not occur at all; the EUREKA version is shown in (12b).

(12) a. DECL-PUNCT = \{\ \text{PERIOD} \mid \text{EXCL-POINT} \}.

b. DECL-PUNCT = \{\ \text{PERIOD} \mid \text{EXCL-POINT} \mid \text{COLON} \mid \text{SEMI-COLON} \}.

The modular specifications that macros and templates provide allow rule behavior to be modified without having to copy the parts of the rule that do not change.

XLE also has a mechanism for systematically modifying the behavior of all rules: the METARULEMACRO. For example, in order to parse labeled bracketed input, as in (2b), the WSJ grammar was altered so that constituents could optionally be surrounded by the appropriately labeled brackets. The METARULEMACRO is applied to each rule in the grammar and produces as output a modified version of that rule. This is used in the STANDARD grammar for coordination and to allow quote marks to surround any constituent. The METARULEMACRO is rede
defined for the WSJ to add the labeled bracketing possibilities for each rule, as shown in (13).

(13) METARULEMACRO(\text{CAT} \text{BASECAT} \text{RHS}) = \{\ \text{LSB LABEL} \text{[BASECAT]} \text{CAT RSB} \mid \text{copy of STANDARD coordination} \mid \text{copy of STANDARD surrounding quote} \}.

Often the necessary modification can be made simply by redefining a macro that existing rules already invoke. Consider the ROOT rule, in (11).
NP for the NP rule). XLE also allows for complex category symbols to specialize the expansion of particular categories in particular contexts. For example, the VP rule is parameterized for the form of its complement and its own form, so that VP[perf,fin] is one of the complex VP categories. When the METARULEMACRO applies to rules with complex left-side categories, _CAT refers to the category including the parameters and the _BASECAT refers to the category without the parameters. For the VP example, _CAT is VP[perf,fin] and _BASECAT is VP.

In the definition in (13), LSB and RSB parse the brackets themselves, while the LABEL[_BASECAT] parses the label in the bracketing and matches it to the label in the tree (NP in (2b)); the constituent itself is the _CAT. Thus, a label-bracketed NP is assigned the structure in (14).

(14) \[
\text{LSB} \quad \text{LABEL[NP]} \quad \text{NP} \quad \text{RSB} \\
\text{[ NP-SBJ Kirk Horse \ldots ]}
\]

These examples illustrate how the prioritized re-definition of rules and macros has enabled us to incorporate the STANDARD rules in grammars that are tuned to the special properties of the EUREKA and WSJ corpora.

2.2 Lexical Entries

Just as for rules, XLE’s override conventions make it possible to: add new lexical items or new part-of-speech subentries for existing lexical items; delete lexical items; and modify lexical items. In addition to the basic priority overrides, XLE provides for “edit lexical entries” (Kaplan and Newman, 1997) that give finer control over the construction of the lexicon. Edit entries were introduced as a way of reconciling information from lexical databases of varying degrees of quality, but they are also helpful in tailoring a STANDARD lexicon to a specialized corpus. When working on specialized corpora, such as the Eureka corpus, modifications to the lexicon are extremely important for correctly handling technical terminology and eliminating word senses that are not appropriate for the domain.

Higher-priority edit lexical entries provide for operators that modify the definitions found in lower-priority entries. The operators can: add a subentry (+); delete a subentry (−); replace a subentry (!); or retain existing subentries (=). For example, the STANDARD grammar might have an entry for button as in (15).

(15) button !V @(V-SUBJ-OBJ %stem); 
    !N @(NOUN %stem); 
    ETC.

However, the EUREKA grammar might not need the V entry but might require a special partname N entry. Assuming that the EUREKA lexicons are given priority over the STANDARD lexicons, the entry in (16) would accomplish this.

(16) button −V ; 
    +N @(PARTNAME %stem); 
    ETC.

Note that the lexical entries in (15) and (16) end with ETC. This is also part of the edit lexical entry system. It indicates that other lower-priority definitions of that lexical item will be retained in addition to the new entries. For example, if in another EUREKA lexicon there was an adjective entry for button with ETC, the V, N, and A entries would all be used. The alternative to ETC is ONLY which indicates that only the new entry is to be used. In our button example, if an adjective entry was added with ONLY, the V and N entries would be removed, assuming that the adjective entry occurred in the highest priority lexicon. This machinery provides a powerful tool for building specialized lexicons without having to alter the STANDARD lexicons.

The EUREKA corpus contains a large number of names of copier parts. Due to their particular syntax and to post-syntactic processing requirements, a special lexical entry is added for each part name. In addition, the regular noun parse of these entries is deleted because whenever they occur in the corpus they are part names. A sample lexical is shown in (17); the ‘ ’ is the escape character for the space.

(17) separator‘ finger 
    !PART-NAME @(PART-NAME %stem); 
    −N; 
    ETC.

The first line in (17) states that separator finger can be a PART-NAME and when it is, it calls a template PART-NAME that provides relevant information for the functional-structure. The second line removes the N entry, if any, as signalled by the − before the category name.
Because of the non-context free nature of Lexical Functional Grammar, it sometimes happens that extensions in one part of the grammar require a corresponding adjustment in other rules or lexical entries. Consider again the EUREKA’s plurals. The part-name UDH is singular when it appears without the’s and thus the morphological tag +Sg is appended to it. In the STANDARD grammar, the tag +Sg has a lexical entry as in (18a) which states that +Sg is of category NNUM and assigns sg to its NUM. However, if this is used in the EUREKA grammar, the sg NUM specification will clash with the pl NUM specification when UDH appears with’s, as seen in (7). Thus, a new entry for +Sg is needed which has sg as a default value, as in (18b). The first line of (18b) states that NUM must exist but does not specify a value, while the second line optionally supplies a sg value to NUM; when the’s is used, this option does not apply since the form already has a pl NUM value.

(18) a. +Sg NNUM (↑ NUM)=sg
    b. +Sg NNUM (↑ NUM)
       ((↑ NUM)=sg)

3 Tokenizing and Morphological Analysis

Tokenization and morphological analysis in XLE are carried out by means of finite state transduction. The STANDARD tokenizing transducer encodes the punctuation conventions of normal English text, which is adequate for many applications. However, the Eureka and WSJ corpora include strings that must be tokenized in non-standard ways. The Eureka part identifiers have internal punctuation that would normally cause a string to be broken up (e.g. the hyphen in PL1-B7), and the WSJ corpus is marked up with labeled brackets and part-of-speech tags that must also receive special treatment. An example of the WSJ mark-up is seen in (19).

(19) [NP-SBJ Lloyd’s, once a pillar of the world insurance market,] is/VBZ being/VBG shaken/VBN to its very foundation.

Part-of-speech tags appear in a distinctive format, beginning with a / and ending with a _ with the intervening material indicating the content of the tag (VBZ for finite 3rd singular verb, VBG for a progressive, VBN for a passive, etc.). The tokenizing transducer must recognize this pattern and split the tags off as separate tokens. The tag-tokens must be available to filter the output of the morphological analyzer so that only verbal forms are compatible with the tags in this example and the adjectival reading of shaken is therefore blocked.

XLE tokenizing transducers are compiled from specifications expressed in the sophisticated Xerox finite state calculus (Beesley and Karttunen, 2002). The Xerox calculus includes the composition, ignore, and substitution operator discussed by Kaplan and Kay (1994) and the priority-union operator of Kaplan and Newman (1997). The specialized tokenizers are constructed by using these operators to combine the STANDARD specification with expressions that extend or restrict the standard behavior. For example, the ignore operator is applied to allow the part-of-speech information to be passed through to the morphology without interrupting the standard patterns of English punctuation.

XLE also allows separately compiled transducers to be combined at run-time by the operations of priority-union, composition, and union. Priority-union was used to supplement the standard morphology with specialized “guessing” transducers that apply only to tokens that would otherwise be unrecognized. Thus, a finite-state guesser was added to identify Eureka fault numbers (09-425), adjustment numbers (12-23), part numbers (606K2100), part list numbers (PL1-B7), repair numbers (2.4), tag numbers (P-102), and diagnostic code numbers (dC131). Composition was used to apply the part-of-speech filtering transducer to the output of the morphological analyzer, and union provided an easy way of adding new, corpus-specific terminology.

4 Optimality Marks

XLE supports a version of Optimality Theory (OT) (Prince and Smolensky, 1993) which is used to rank an analysis relative to other possible analyses (Frank et al., 2001). In general, this is used within a specific grammar to prefer or disprefer a construction. However, it can also be used in grammar extensions to delete or include rules or parts of rules.

The XLE implementation of OT works as follows. OT marks are placed in the grammar and are associated with particular rules, parts of rules, or lexical entries. These marks are then ranked in the grammar CONFIGURATION. In addition to a simple ranking of constraints which states that a construction with a given OT mark is (dis)preferred to one

1The actual XLE OT implementation is more complicated than this, allowing for UNGRAMMATICAL and STOPPOINT marks as well. Only OT marks that are associated with NO-GOOD are of interest here. For a full description, see (Frank et al., 2001).
without it, XLE allows the marks to be specified as NOGOOD. A rule or rule disjunct which has a NOGOOD OT mark associated with it will be ignored by XLE. This can be used for grammar extensions in that it allows a standard grammar to anticipate the variations required by special corpora without using them in normal circumstances.

Consider the example of the eureka’s plurals discussed in section 2.1. Instead of rewriting the N rule in the eureka grammar, it would be possible to modify it in the STANDARD grammar and include an OT mark, as in (20).

(20) \[ N \rightarrow \text{original STANDARD N rules} \]
\[ (\text{PL: } @(@\text{OT-MARK EUR-PLURAL}).) \]

The configuration files of the STANDARD and eureka grammars would differ in that the STANDARD grammar would rank the EUR-PLURAL OT mark as NOGOOD, as in (21a), while the eureka grammar would simply not rank the mark, as in (21b).

(21) a. STANDARD optimality order:
EUR-PLURAL NOGOOD …

b. eureka optimality order:
NOGOOD …

Given the OT marks, it would be possible to have one large grammar that is specialized by different OT rankings to produce the STANDARD, eureka, and WSJ variants. However, from a grammar writing perspective this is not a desirable solution because it becomes difficult to keep track of which constructions belong to standard English and are shared among all the specializations and which are corpus-specific. In addition, it does not distinguish a core set of slowly changing linguistic specifications for the basic patterns of the language, and thus does not provide a stable foundation that the writers of more specialized grammars can rely on.

5 Maintenance with Grammar Extensions

Maintenance is a serious issue for any large-scale grammar development activity, and the maintenance problems are compounded when multiple versions are being created perhaps by several different grammar writers. Our STANDARD grammar is now quite mature and covers all the linguistically significant constructions and most other constructions that we have encountered in previous corpus analysis. However, every now and then, a new corpus, even a specialized one, will evidence a standard construction that has not previously been accounted for. If specialized grammars were written by copying all the STANDARD files and then modifying them, the implementation of new standard constructions would tend to appear only in the specialized grammar. Our techniques for minimizing the amount of copying encourages us to implement new constructions in the STANDARD grammar and this makes them available to all other specializations.

If a new version of a rule for a specialized grammar is created by copying the corresponding STANDARD rule, changes later made to the special rule will not automatically be reflected in the STANDARD grammar, and vice versa. This is the desired behavior when adding unusual, corpus-specific constructions. However, if the non-corpus specific parts of the new rule are modified, these modifications will not migrate to the STANDARD grammar. To avoid this problem, the smallest rule possible should be modified in the specialized grammar, e.g., modifying the N head rule instead of the entire NP. For this reason, having highly modularized rules and using macros and templates helps in grammar maintenance both within a grammar and across specialized grammar extensions.

As seen above, the XLE grammar development platform provides a number of mechanisms to allow for grammar extensions without altering the core (STANDARD) grammar. However, there are still areas that could use improvement. For example, as mentioned in section 2, the configuration file states which other files the grammar includes and how they are prioritized. The configuration contains other information such as declarations of the governable grammatical functions, the distributive features, etc. As this information rarely changes with grammar extensions, it would be helpful for an extension configuration to incorporate by reference such additional parameters of the STANDARD configuration. Currently these declarations must be copied into each configuration.

6 Discussion and Conclusion

As a result of the strategies and notational devices outlined above, our specialized grammars share substantial portions of the pre-existing STANDARD grammar. The statistics in table (22) give an indication of the size of the STANDARD grammar and of
the additional material required for the EUREKA and
WSJ specializations. As can be seen from this table,
the specialized grammars require a relatively small
number of rules compared to the rules in the STAN-
DARD grammar. The number of lines that the rules
and lexical entries take up also provides a measure
of the relative size of the specifications. The WSJ lex-
icons include many titles and proper nouns that may
ultimately be moved to the STANDARD files. The ta-
ble also shows the number of files called by the CON-
FIGURATION, as another indication of the size of the
specifications. This number is somewhat arbitrary as
separate files can be combined into a single multi-
sectioned file, although this is likely to reduce main-
tainability and readability.

(22)

<table>
<thead>
<tr>
<th></th>
<th>STANDARD</th>
<th>EUREKA</th>
<th>WSJ</th>
</tr>
</thead>
<tbody>
<tr>
<td>rules lines:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>310</td>
<td>32</td>
<td>14</td>
</tr>
<tr>
<td>rules</td>
<td>6,539</td>
<td>425</td>
<td>894</td>
</tr>
<tr>
<td>lexicons</td>
<td>44,879</td>
<td>5,565</td>
<td>15,135</td>
</tr>
<tr>
<td>files</td>
<td>14</td>
<td>5</td>
<td>8</td>
</tr>
</tbody>
</table>

The grammars compile into a collection of finite-
state machines with the number of states and arcs
listed in table (23). The WSJ grammar compiles into
the largest data structures, mainly because of its abil-
ity to parse labeled bracketed strings and part-of-
speech tags, (2b). This size increase is the result of
adding one disjunct in the METARULEMACRO and
hence reflects only a minor grammar change.

(23)

<table>
<thead>
<tr>
<th></th>
<th>STANDARD</th>
<th>EUREKA</th>
<th>WSJ</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4,935</td>
<td>5,132</td>
<td>8,759</td>
</tr>
<tr>
<td></td>
<td>13,268</td>
<td>13,639</td>
<td>19,695</td>
</tr>
</tbody>
</table>

In sum, the grammar specialization system used
in XLE has been quite sucessful in developing cor-
pus specific grammars using the STANDARD English
grammar as a basis. A significant benefit comes from
being able to distinguish truly unusual constructions
that exist only in the specialized grammar from those
that are (or should be) in the STANDARD grammar.
This allows idiosyncratic information to remain in a
specialized grammar while all the specialized gram-
mars benefit from and contribute to the continuing
development of the STANDARD grammar.

References
Morphology: Xerox Tools and Techniques*. Cambridge
University Press. To Appear.
M. Butt, T.H. King, M.-E. Niño, and F. Segond.
Publications, Stanford, CA.
M. Butt, H. Dyvik, T.H. King, H. Masuichi,
In *Proceedings of COLING 2002*, Workshop on
Grammar Engineering and Evaluation.
J. Everett, D. Bobrow, R. Stolle, R. Crouch,
V. de Paiva, C. Condoravdi, M. van den Berg,
and L. Polanyi. 2001. Making ontologies work
for resolving redundancies across documents.
A. Frank, T.H. King, J. Kuhn, and J. T. Maxwell III.
2001. Optimality theory style constraint rank-
ing in large-scale LFG grammars. In Peter Sells,
editor, *Formal and Empirical Issues in Optimal-
ity Theoretic Syntax*. CSLI Publications, Stanford,
CA.
R. Kaplan and M. Kay. 1994. Regular models of
phonological rule systems. *Computational Lin-
guistics*, 20:331–378.
R. Kaplan and J. Maxwell. 1996. LFG Gram-
mar Writer’s Workbench. System documentation
manual; available on-line at PARC.
conciliation in the Xerox Linguistic Environment.
In *Proceedings of the ACL Workshop on Com-
putational Environments for Grammar Develop-
ment and Engineering*.
M. Marcus, G. Kim, M. A. Marcinkiewicz, R. Mac-
Intyre, A. Bies, M. Ferguson, K. Katz,
and B. Schasberger. 1994. The Penn treebank: An-
otiative predicate argument structure. In *ARPA
Human Language Technology Workshop*.
J. Maxwell and R. Kaplan. 1993. The interface be-
 tween phrasal and functional constraints. *Computa-
A. Prince and P. Smolensky. 1993. Optimality the-
ory: Constraint interaction in generative gram-
mar. RuCCS Technical Report #2, Rutgers Uni-
versity.
S. Riezler, T.H. King, R. Kaplan, D. Crouch, J. T.
the Wall Street Journal using a lexical-functional
grammar and discriminative estimation tech-
niques. In *Proceedings of the Annual Meeting of
the Association for Computational Linguistics,
University of Pennsylvania*. 35
Coping with problems in grammars automatically extracted from treebanks

Carlos A. Prolo
Computer and Information Science Department
University of Pennsylvania
Suite 400A, 3401 Walnut Street
Philadelphia, PA, USA, 19104-6228
prolo@linc.cis.upenn.edu

Abstract
We report in this paper on an experiment on automatic extraction of a Tree Adjoining Grammar from the WSJ corpus of the Penn Treebank. We use an automatic tool developed by (Xia, 2001) properly adapted to our particular need. Rather than addressing general aspects of the automatic extraction we focus on the problems we have found to extract a linguistically (and computationally) sound grammar and approaches to handle them.

1 Introduction
Much linguistic research is oriented to finding general principles for natural language, classifying linguistic phenomena, building regular models (e.g., grammars) for the well-behaved (or well-understood) part of languages and studying remaining “interesting” problems in a compartmentalized way. With the availability of large natural language corpora annotated for syntactic structure, the treebanks, e.g., (Marcus et al., 1993), automatic grammar extraction became possible (Chen and Vijay-Shanker, 2000; Xia, 1999). Suddenly, grammars started being extracted with an attempt to have “full” coverage of the constructions in a certain language (of course, to the extent that the used corpora represents the language) and that immediately poses a question: If we do not know how to model many phenomena grammatically how can that be that we are extracting such a wide-coverage grammar?

To answer that question we have to start a new thread at the edge of linguistics and computational linguistics. More than numbers to express coverage, we have to start analyzing the quality of automatically generated grammars, identifying extraction problems and uncovering whatever solutions are being given for them, however interesting or ugly they might be, challenging the current paradigms of linguistic research to provide answers for the problems on a “by-need” basis.

In this paper we report on a particular experience of automatic extraction of an English grammar from the WSJ corpus of the Penn Treebank (PTB) (Marcus et al., 1994)\(^1\) using Tree Adjoining Grammar (TAGs, (Joshi and Schabes, 1997)). We use an automatic tool developed by (Xia, 2001) properly adapted to our particular needs and focus on some problems we have found to extract a linguistically (and computationally) sound grammar and the solutions we gave to them. The list of problems is a sample, far from being exhaustive\(^2\) Likewise, the solutions will not always be satisfactory.

In Section 2 we introduce the method of grammar extraction employed. The problems are discussed in Section 3. We conclude in Section 4.

2 The extracted grammar

2.1 TAGs
A TAG is a set of lexicalized elementary trees that can be combined, through the operations of tree adjunction and tree substitution, to derive syntactic structures for sentences. We follow a common approach to grammar development for natural language using TAGs, under which, driven by locality principles, each elementary tree for a given lexical head is expected to contain its projection, and slots for its arguments (e.g., (Frank, 2002)). Figure 1 shows typical grammar template trees that can be selected by lexical items and combined to generate the structure in Figure 2. The derivation tree, to the right, contains the history of the tree grafting process that generated the derived tree, to the left.\(^3\)

---
\(^1\) We assume some familiarity with the basic notations in the PTB as in (Marcus et al., 1994).
\(^2\) (Prolo, 2002) includes a more comprehensive and detailed discussion of grammar extraction alternatives and problems.
\(^3\) For a more comprehensive introduction to TAGs and Lexicalized TAGs we refer the reader to (Joshi and Schabes, 1997).
2.2 LexTract

Given an annotated sentence from the PTB as input Xia’s LexTract tool (Xia, 1999; Xia, 2001) first executes a rebracketing. More precisely, additional nodes are inserted to separate arguments and modifiers and to structure the modifying process as binary branching. A typical rebracketed PTB tree is shown in Figure 3, in which we have distinguished the tree nodes inserted by LexTract.

The second stage is the extraction of the grammar trees proper shown in Figure 4. In particular, recursive modifier structures have to be detected and factored out of the derived tree to compose the auxiliary trees, the rest becoming an initial tree. The process is recursive also in the sense that factored subtree structures still undergo the spinning off process until we have all modifiers with their own trees, all the arguments of a head as substitution nodes of the tree containing their head, and the material under the argument nodes defining additional initial trees for themselves. Auxiliary trees are extracted from parent-child pairs with matching labels if the child is elected the parent’s head and the child’s sibling is marked as modifier: the parent is mapped into a root of an auxiliary tree, the head-child into its

foot, with the sibling subtree (after being recursively processed) being carried together into the auxiliary tree. Notice that the auxiliary trees are therefore either strictly right or left branching, the foot always immediately under the root node. Other kinds of auxiliary trees are therefore not allowed.

3 Extraction Problems

Extraction problems arise from several sources, including: (1) lack of proper linguistic account, (2) the (Penn Treebank) annotation style, (3) the (LexTract) extraction tool, (4) possible unsuitability of the (TAG) model, and (5) annotation errors. We refrained from making a rigid classification of the problems we present according to these sources. In particular it is often difficult to decide whether to blame sources (1), (3), or (5) for a certain problem. We will not discuss in this paper problems due to annotation errors. As for the PTB style problems we only discuss one, the first listed below.

Footnote: Figures 3 and 4 are thanks to Fei Xia. We are also grateful to her for allowing us to use LexTract and make changes to its source code to customize to our needs.
Figure 4: LexTract extraction stage

3.1 Free Relatives

Free relatives are annotated in the Penn Treebank as sentential complements as in Figure 5.a. The extracted tree corresponding to the occurrence of “make” would be of a verb that takes a sentential complement (SBAR). This does not seem to be correct, as the proper subcategorization of the verb occurrence is transitive.

In fact, free relatives may occur wherever an NP argument may occur. So, the only reasonable extraction account consistent with maintaining them as SBARs would be one in which every NP substitution node in an extracted tree would admit the existence of a counterpart tree, identical to the first, except that the NP argument label is replaced with an SBAR. Instead we opted to reflect the NP character of the free relatives by pre-processing the corpus (using the Head-analysis, for practical convenience). The annotated example is then automatically replaced with the one in Figure 5.b. Other cases of free-relatives (non-NP) are rare and not likely to interfere with verb subcategorization.

3.2 Wh percolation up

In the Penn Treebank the same constituent is annotated with different syntactic categories depending on whether it possesses or not the wh feature. For instance, a regular noun phrase has the syntactic category NP, whereas when the constituent is wh-marked, and is in the landing site of wh-movement, it carries the label WHNP. While that might look appealing since the two constituents seem to have distinct distributional properties, it poses a design problem. While regular constituents inherit their syntactic categorial feature (i.e. their label) from their heads, wh projections are often formed by inheritance from their modifiers. For instance: “the father” is an NP, but modified by a wh expression (“the father of whom”, “whose father”, “which father”), it becomes a WHNP. The only solution we see is to allow for nouns and NPs to freely project up to WHNPs during extraction. On the other hand, in

---

6In both standard accounts for free relatives, the Head Account (e.g., (Bresnan and Grimshaw, 1978)) and the Comp Account (e.g., (Groos and von Riemsdijk, 1979)), commonly discussed in the literature, the presence of the NP (or DP) is clear.

7When the constituent is not wh-moved, it is correctly preserved as an NP, as “what” in “Who ate what?”.

8Of course another simple solution would be merging the wh constituents with their non-wh counterparts.
cases when the wh constituent is in a non-wh position, we need the opposite effect: a WHNP (or wh-noun POS tag) is allowed to project up to an NP.

### 3.3 Unlike Coordinated Phrases (UCP)

This is the expression used in the PTB to denote coordinated phrases in which the coordinated constituents are not of the same syntactic category. The rationale for the existence of such constructions is that the coordinated constituents are alternative realizations of the same grammatical function with respect to a lexical head. In Figure 6.a, both a noun and an adjective are allowed to modify another noun, and therefore they can be conjoined while realizing that function. Two other common cases are: coordination of predicates in copular constructions (Figure 6.b) and adverbial modification (Figure 6.c).

We deal with the problem as follows. First, we allow for a UCP to be extracted as an argument when the head is a verb and the UCP is marked predicative (PRD function tag) in the training example; or whenever the head is seen to have an obligatory argument requirement (e.g., prepositions: “They come from ((NP the house) and (PP behind the tree))”). Second, a UCP is allowed to modify (adjoin to) most of the nodes, according to evidence in the corpus and common sense (in the first and third examples above we had NP and VP modification). With respect to the host tree, when attached as an argument they are treated like any other non-terminal: a substitution node. The left tree in Figure 7 shows the case where the UCP is treated as a modifier. In fact the trees are both for the example in Figure 6.a. Notice that the tree is non-lexicalized to avoid effects of sparseness. The UCP is then expanded as in the right tree in Figure 7: an initial tree anchored by the conjunction (the tree attaches either to a tree like the one in the left or as a true argument – the latter would be the case for the example in Figure 6.b).

Now, the caveats. First, we are giving the UCP the status of an independent non-terminal, as if it had some intrinsic categorial significance (as a syntactic projection). The assumption of independence of expansion, that for context-free grammars is inherent to each non-terminal, in TAGs is further restricted to the substitution nodes. For example, when an NP appears as substitution node, in a subject or object position, or as an argument of a preposition or a genitive marker, we are stating that any possible expansion for the NP is licensed there. The same happens for other labels in argument positions as well. While that is an overgenerating assumption (e.g. the expletive “there” cannot be the realization of an NP in object position), it is generally true. For the UCP, however, we know that its expansion is in fact strongly dependent on where the substitution node is, as we have argued before. In fact it is lexically dependent (cf. “I know ((the problem) and (that there is no solution to it))”, where the conjuncts are licensed by the subcategorizations of the verb “know”). On the other hand, it does not seem reasonable to expand the UCP node at the hosting tree – a cross product explosion. A possible way of alleviating this effect could be to expand only the auxiliary trees (a UCP modifying a VP is distinct from a UCP modifying an NP, and moreover they are independent of lexical items). But for true argument positions there seems to be no clear solution.

Second, the oddity of the UCP as a label becomes apparent once again when there are multiple conjuncts, as in Figure 8: it is enough for one of them to be distinct to turn the entire constituent into a UCP. Recursive decomposition in the grammar in these situations clearly leads to some non-standard trees.

Finally, and more crucially, we have omitted one case in our discussion: the case in which the UCP
Figure 8: UCP with multiple conjuncts

Figure 9: UCP involving VP argument of the copula

is the natural head-child of some node. Under some accounts of grammar development this never happens: we have observed that UCP does not appear as head child in the account where the head is the syntactic head of a node. We have not always followed this rule. With respect to the VP head, so far we have followed one major tendency in the computational implementation of lexicalized grammars, according to which lexical verbs are preferred to auxiliary verbs to head the VP. Now, consider the pair of sentences in Figure 9.

Under the lexical verb paradigm, in the first sentence the derivation would start with an initial tree anchored by the past participle verb (“rated”). But then we have an interesting problem in the second sentence, for which we do not currently have a neat solution. Following Xia’s sample settings of LexTract parameters, in these cases the extraction is rescued by switching to the other paradigm: the initial tree is extracted anchored by the auxiliary verb with a UCP argument, and the VP is accepted as a possible conjunct. A systematic move to the syntactic head paradigm, which we may indeed try, would have important consequences in the locality assumptions for the grammar development.

3.4 VP topicalization

Another problem with the lexical verb paradigm (see also discussion under UCP above) is the VP topicalization as in the sentence in Figure 10. The solution currently adopted (again, inherited from Xia’s sample settings) is as above: the paradigm is switched and the auxiliary verb (“be”) is chosen as the anchor of the initial tree.

3.5 Extraposition and Verb Subcategorization

One of the key design principles that have been guiding grammar development with TAGs is to keep verb arguments as substitution slots local to the tree anchored by the verb. It is widely known that the Penn Treebank does not distinguish verb objects from adjuncts. So some sorts of heuristics are needed to decide, among the candidates, which are to be taken as arguments (Kinyon and Prolo, 2002); the rest is extracted as separate VP modifier trees. However, this step is not enough for the trees to correctly reflect verb subcategorizations. The occurrence of discontinuous arguments, frequently explained as argument extraposition (the argument is raised past the adjunct) creates a problem. In the sentence in Figure 11 the verb “pass” should anchor a tree with one NP object.

Figure 11: The extraposition problem
a) A parenthetical NP attached to another NP

\[
\begin{array}{l}
\text{(NP (NP the 3 billion New Zealand dollars))} \\
\text{(PRN (-LRB- -LRB-)} \\
\text{(NP US\$ 1.76 billion *U*)} \\
\text{(-RRB- -RRB-)))}
\end{array}
\]

b) A parenthetical S between subject and verb

\[
\begin{array}{l}
\text{(S (NP-SBJ The total relationship))} \\
\text{(PRN (, ,)} \\
\text{(SBAR-ADV as Mr. Lee sees it)} \\
\text{((, ,))} \\
\text{(VP (VBZ is) . . .))}
\end{array}
\]

3.6 Parentheticals

Parenthetical expressions are ubiquitous in language: they may appear almost everywhere in a sentence and can be of almost any category (Fig. 12).

We model them as adjoining, either to the left or right of the constituent they are dominated by, depending on whether they are to the left or right of the head child of the parent’s node. Occasionally such trees can also be initial. The respective trees for the examples of Figure 12 are drawn in Figure 13. It is always the case that the label PRN dominates a single substitution node. Whenever this was not the case in the training corpus, heuristics based on observation were used to enforce that, by inserting an appropriate missing node.

3.7 Projection labels

LexTract extracts trees with no concern for the appropriate projective structure of constituents when not explicitly marked in the PTB. Figure 14 shows two examples of NP modification where the modifiers are single lexical items. The extracted modifier trees, shown on the right, do not have the projection for the modifiers JJR “stronger” and the NNP “October” (which should be, respectively, an ADJP and an NP). That is so, because those nodes are not found in the annotation.

However, if the modifiers are complex, that is, if the modifiers are themselves modified, the PTB inserts their respective projections, and therefore they appear in the extracted trees, as shown in Figure 15.

There seems to be no reason for the two pairs of extracted trees to be different. Much of this is caused by the acknowledged flatness in the Penn Treebank annotation. That said, the trees like those in the second pair should be preferred. The projection node (ADJP or NP) is understood to be dominating its head even when there is no further modification, and it should be a concern of a good extraction process to insert the missing node into the grammar. Since LexTract do not allow us to spec-
Figure 15: Complex modification annotation and extracted trees

ify for the insertion of “obligatory” projections we had to accomplish this through a somewhat complicated post-processing step using a projection table. Some of our current projections are: nouns, personal pronouns and the existential expletive to NP; adjectives to ADJP; adverbs to ADVP; sentences either to SBAR (S, SINV) or to SBARQ (SQ); Cardinals (CD) to Quantifier Phrases (QP) which themselves project to NP. Notice that not all categories are forcefully projected. For instance, verbs are not, allowing for simple auxiliary extraction. IN is also not projected due to its double role as PP head (true preposition) and subordinate conjunction, which should project onto SBARs.

4 Conclusion

We discussed an experiment in grammar extraction from corpora with focus on problems arising while trying to give an adequate account for naturally occurring phenomena. Without being exhaustive in our list, we expect to have brought some attention to the need to discuss solutions for them which are as reasonable as possible given the current state-of-the-art of the linguistic research, computational grammar development and automatic extraction, and given the current corpus resources at our disposition.

ACKNOWLEDGEMENTS: Thanks to Tonia Bleam, Erwin Chan, Alexandra Kinyon, Rashmi Prasad, Beatrice Santorini, Fei Xia and the XTAG Group for valuable discussions along the realization of this work and/or comments on this paper or related material.

References


A Classification of Grammar Development Strategies

Alexandra Kinyon Carlos A. Prolo
Computer and Information Science Department
University of Pennsylvania
Suite 400A, 3401 Walnut Street
Philadelphia, PA, USA, 19104-6228
{kinyon,prolo}@linc.cis.upenn.edu

Abstract

In this paper, we propose a classification of grammar development strategies according to two criteria: hand-written versus automatically acquired grammars, and grammars based on a low versus high level of syntactic abstraction. Our classification yields four types of grammars. For each type, we discuss implementation and evaluation issues.

1 Introduction: Four grammar development strategies

There are several potential strategies to build wide-coverage grammars, therefore there is a need for classifying these various strategies. In this paper, we propose a classification of grammar development strategies according to two criteria:

- Hand-crafted versus Automatically acquired grammars
- Grammars based on a low versus high level of syntactic abstraction.

As summarized in table 1, our classification yields four types of grammars, which we call respectively type A, B, C and D.

Of these four types, three have already been implemented to develop wide-coverage grammars for English within the Xtag project, and an implementation of the fourth type is underway¹. Most of our examples are based on the development of wide coverage Tree Adjoining Grammars (TAG), but it is important to note that the classification is relevant within other linguistic frameworks as well (HPSG, GPSG, LFG etc.) and is helpful to discuss portability among several syntactic frameworks.

We devote a section for each type of grammar in our classification. We discuss the advantages and drawbacks of each approach, and especially focus on how each type performs w.r.t. grammar coverage, linguistic adequacy, maintenance, over- and under- generation as well as to portability to other syntactic frameworks. We discuss grammar replication as a mean to compare these approaches. Finally, we argue that the fourth type, which is currently being implemented, exhibits better development properties.

2 TYPE A Grammars: hand-crafted

The limitations of Type A grammars (hand-crafted) are well known: although linguistically motivated, developing and maintaining a totally hand-crafted grammar is a challenging (perhaps unrealistic?) task. Such a large hand-crafted grammar for TAGs is described for English in (XTAG Research Group, 2001). Smaller hand-crafted grammars for TAGs have been developed for other languages (e.g. French (Abeille, 1991)), with similar problems. Of course, the limitations of hand-crafted grammars are not specific to the TAG framework (see e.g. (Clement and Kinyon, 2001) for LFG).

2.1 Coverage issues

The Xtag grammar for English, which is freely downloadable from the project homepage ² along with tools such as a parser and an extensive documentation), has been under constant development for approximately 15 years. It consists of more than 1200 elementary trees (1000 for verbs) and has been tested on real text and test suites. For instance, (Do-ran et al., 1994) report that 61% of 1367 grammatical sentences from the TSNLP test-suite (Lehman and al, 1996) were parsed with an early version of the grammar. More recently, (Prasad and Sarkar, 2000) evaluated the coverage of the grammar on "the weather corpus", which contained rather complex sentences with an average length of 20 words per sentence, as well as on the "CSLI LKB test suite" (Copestake, 1999). In addition, in order to

¹We do not discuss here shallow-parsing approaches, but only full grammar development. Due to space limitations, we do not introduce the TAG formalism and refer to (Joshi, 1987) for an introduction.

²http://www.cis.upenn.edu/ xtag/
Table 1: A classification of grammars

<table>
<thead>
<tr>
<th></th>
<th>High level of syntactic abstraction</th>
<th>Low level of syntactic abstraction</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hand-crafted</strong></td>
<td><strong>Type A:</strong> Traditional hand-crafted grammars</td>
<td><strong>Type C:</strong> Hand-crafted level of syntactic abstraction Automatically generated grammars</td>
</tr>
<tr>
<td><strong>Automatically acquired</strong></td>
<td><strong>Type B:</strong> Traditional treebank extracted grammars</td>
<td><strong>Type D:</strong> Automatically acquired level of syntactic abstraction Automatically generated grammar</td>
</tr>
</tbody>
</table>

2.3 Are hand-crafted grammars useful?

Some degree of automation in grammar development is unavoidable for any real world application: small and even medium-size hand-crafted grammars are not useful for practical applications because of their limited coverage, but larger grammars give way to maintenance issues. However, despite the problems of coverage and maintenance encountered with hand-crafted grammars, such experiments are invaluable from a linguistic point of view. In particular, the Xtag grammar for English comes with a very detailed documentation, which has proved extremely helpful to devise increasingly automated approaches to grammar development (see sections below) 4.

3 TYPE B Grammars: Automatically extracted

To remedy some of these problems, Type B grammars (i.e. automatically acquired, mostly from annotated corpora) have been developed. For instance (Chiang, 2000), (Xia, 2001) (Chen, 2001) all automatically acquire large TAGs for English from the Penn Treebank (Marcus et al., 1993). However, despite an improvement in coverage, new problems arise with this type of grammars: availability of annotated data which is large enough to avoid sparse data problems, possible lack of linguistic adequacy, extraction of potentially unreasonably large grammars (slows down parsing and increases ambiguity),

---

3For instance, at first, Xtag parsed only 20% of the sentences in the weather corpus because this corpus contained frequent free relative constructions not handled by the grammar. After augmenting the grammar, 89.6% of the sentences did get a parse.

4Perhaps fully hand-crafted grammars can be used in practice on limited domains, e.g. the weather corpus. However, a degree of automation is useful even in those cases, if only to insure consistency and avoid some maintenance problems.
lack of domain and framework independence (e.g. a grammar extracted from the Penn Treebank will reflect the linguistic choices and the annotation errors made when annotating the treebank).

We give two examples of problems encountered when automatically extracting TAG grammars: The extraction of a wrong domain of locality; And The problem of sparse-data regarding the integration of the lexicon with the grammar.

3.1 Wrong domain of locality

Long distance dependencies are difficult to detect accurately in annotated corpora, even when such dependencies can be adequately modeled by the grammar framework used for extraction (which is the case for TAGs, but not for instance for Context Free Grammars). For example, (Xia, 2001) extracts two elementary trees from a sentence such as Which dog does Hillary Clinton think that Chelsea prefers. These trees are shown on figure 1. Unfortunately, because of the potentially unbounded dependency, the two trees exhibit an incorrect domain of locality: the Wh-extracted element ends up in the wrong elementary tree, as an argument of "think", instead of as an argument of "prefer".

Figure 1: Extraction of the wrong domain of locality

This problem is not specific to TAGs, and would translate in other frameworks into the extraction of the "wrong" dependency structure.

3.2 Sparse data for lexicon-grammar integration

Existing extraction algorithms for TAGs acquire a fully lexicalized grammar. A TAG grammar may be viewed as consisting of two components: on the one hand “tree templates” and on the other hand a lexicon which indicates which tree template(s) should be associated to each lexical item.

Suppose the following three sentences are encountered in the training data:

1. Peter watches the stars
2. Mary eats the apple
3. What does Peter watch?

From these three sentences, two tree templates will be correctly acquired, as shown on figure 2: The first one covers the canonical order of the realization of arguments for sentences 1 and 2, the second covers the case of a Wh-extracted object for sentence 3. Concerning the interaction between the lexicon and the grammar rules, the fact that “watch” should select both trees will be accurately detected. However, the fact that “eat” should also select both trees will be missed since “eat” has not been encountered in a Wh-extractedObject construction.

Figure 2: Correct templates, but incomplete lexicon-grammar interface

A level of syntactic abstraction is missing: in this case, the notion of subcategory frame. This is especially noticeable within the TAG framework from the fact that in a TAG hand-crafted grammar the grammar rules are grouped into “tree families”, with one family for each subcategorization frame (transitive, intransitive, ditransitive, etc.), whereas automatically extracted TAGs do not currently group trees into families.

4 TYPE C Grammars

To remedy the lack of coverage and maintenance problems linked to hand-crafted grammars, as well as the lack of generalization and linguistic adequacy of automatically extracted grammars, new syntactic levels of abstraction are defined. In the context of TAGs, one can cite the notion of MetaRules (Becker, 2000), (Prolo, 2002), and the notion of MetaGrammar (Candito, 1996), (Xia, 2001).

Some extraction algorithms such as those of (Chen, 2001) or (Chiang, 2000) do retrieve the right the right domain of locality for this specific example, but do extract a domain of locality which is incorrect in some other cases.

One can argue that the problem does not appear when using simple CFGs, and/or that this problem is only of interest to linguists. A counter-argument is that linguistic adequacy of a grammar, whether extracted or not, DOES matter. An extreme caricature to illustrate this point: the context free grammar $S \rightarrow S \text{ word} \text{ word}$ allows one to robustly and unambiguously parse any text, but is not very useful for any further NLP.
4.1 MetaRules
A MetaRule works as a pattern-matching tool on trees. It takes as input an elementary tree and outputs a new, generally more complex, elementary tree. Therefore, in order to create a TAG, one can start from one canonical elementary tree for each subcategorization frame and a finite number of MetaRules which model syntactic transformations (e.g. passive, wh-questions etc) and automatically generate a full-size grammar. (Prolo, 2002) started from 57 elementary trees and 21 hand-crafted MetaRules, and re-generated the verb trees of the hand-crafted Xtag grammar for English described in the previous section.

The replication of the hand-crafted grammar for English, using a MetaRule tool, presents interesting aspects: it allows to directly compare the two approaches. Some trees generated by (Prolo, 2002) were not in the hand-crafted grammar (e.g. various orderings of “by phrase passives”) while some others were in the hand-crafted grammar were not generated by the MetaRules9. This replication process makes it possible, with detailed scrutiny of the results, to:

- Identify what should be considered as under- or over-generation of the MetaRule tool.
- Identify what should be considered as under- or over-generation of the hand-crafted grammar.

Thus, grammar replication tasks make it possible to improve both the hand-crafted and the MetaRule generated grammars.

4.2 MetaGrammars
Another possible approach for compact and abstract grammar encoding is the MetaGrammar (MG), initially developed by (Candito, 1996). The idea is to compact linguistic information thanks to an additional layer of linguistic description, which imposes a general organization for syntactic information in a three-dimensional hierarchy: Dimension 1: initial subcategorization Dimension 2: valency alternations and redistribution of functions Dimension 3: surface realization of arguments.

Each terminal class in dimension 1 describes a possible initial subcategorization (i.e. a TAG 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, to erase one for passive with no agents ...). Each terminal class in dimension 3 represents the surface realization of a surface function (ex: declares if a direct-object is pronounlized, wh-extracted, etc.). Each class in the hierarchy corresponds to the partial description of a tree (Rogers and Vijay-Shanker, 1994). A TAG elementary tree is generated by inheriting from exactly one terminal class from dimension 1, one terminal class from dimension 2, and n terminal classes from dimension 3 (where n is the number of arguments of the elementary tree being generated). For instance the elementary tree for "Par qui sera accompagnée Marie" (By whom will Mary be accompanied) is generated by inheriting from transitive in dimension 1, from impersonal-passive in dimension 2 and subject-nominal-inverted for its subject and questioned-object for its object in dimension 3. This compact representation allows one to generate a 5000 tree grammar from a hand-crafted hierarchy of a few dozens of nodes, esp. since nodes are explicitly defined only for simple syntactic phenomena.

The MG was used to develop a wide-coverage grammar for French (Abeille et al., 1999). It was also used to develop a medium-size grammar for Italian, as well as a generation grammar for German (Gerdes, 2002) using the newly available implementation described in (Gaiffe et al., 2002). A similar MetaGrammar approach has been described in (Xia, 2001) for English.

4.3 MetaGrammars versus MetaRules: which is best?
It would be desirable to have a way of comparing the results of the MetaGrammar approach with that of the MetaRule approach. Unfortunately, this is not possible because so far none of the two approaches have been used within the same project(s). Therefore, in order to have a better comparison between these two approaches, we have started a second replication of the Xtag grammar for English, this time using a MG. This replication should allow us to make a direct comparison between the hand-crafted grammar, the grammar generated with MetaRules and the grammar generated with a MG.

For this replication task, we use the more recent implementation presented in (Gaiffe et al., 2002) because their tool:

9Due to space limitations, we refer to (Prolo, 2002) for a detailed discussion.
10Nodes for complex syntactic phenomena are generated by automatic crossings of nodes for simple phenomena
11but that particular work did not attempt to replicate the Xtag grammar, and thus the generated grammar is not directly comparable to the hand-crafted version of the grammar.
- Is freely available\(^{12}\), portable (java), well maintained and includes a Graphical User Interface.
- Outputs a standardized XML format\(^{13}\)
- Is flexible (one can have more than 3 dimensions in the hierarchy) and strictly monotonic w.r.t. the trees built
- Supports “Hypertags”, i.e. each elementary tree in the grammar is associated with a feature structure which describes its salient linguistic properties\(^{14}\).

In the (Gaiffe et al., 2002) implementation, each class in the MG hierarchy can specify:
- Its SuperClass(es)
- A Feature structure (i.e. Hypertag) which captures the salient linguistic characteristics of that class.
- What the class needs and provides
- A set a quasi-nodes
- Constraints between quasi-nodes (father, dominates, precedes, equals)
- traditional feature equations for agreement.

The MG tool automatically crosses the nodes in the hierarchy, looking to create “balanced” classes, that is classes that do not need nor provide anything. From these balanced terminal classes, elementary trees are generated. Figure 3 shows how a canonical transitive tree is automatically generated from 3 hand-written classes and the quasi-trees associated to these classes\(^{15}\).

### 4.4 Advantages and drawbacks of TYPE C grammars

It is often assumed that Metarule and MetaGrammar approaches exhibit some of the advantages of hand-crafted grammars (linguistic relevance) as well as some of the advantages of automatically extracted grammars (wide-coverage), as well as easier maintenance and better coherence. However, as is pointed out in (Barrier et al., 2000), grammar development based on hand-crafted levels of abstraction give rise to new problems while not necessarily solving all the old problems: Although the automatic generation of the grammar insures some level of consistency, problems arise if mistakes are made while hand-crafting the abstract level (hierarchy or MetaRules) from which the grammar is automatically generated. This problem is actually more serious than with simple hand-crafted grammars, since an error in one node will affect ALL trees that inherit from this node. Furthermore, a large portion of the generated grammar covers rare syntactic phenomena that are not encountered in practice, which unnecessarily augments the size of the resulting grammars, increases ambiguity while not significantly improving coverage\(^{16}\). One crucial problem is that despite the automatic generation of the grammar (which eases maintenance), the interface between lexicon and grammar is still mainly man-

\(^{12}\)http://www.loria.fr/equipes/led/outs/mgc/mgc.html

\(^{13}\)See http://atoll.inria.fr/lerger/tag20.dtd,xml for more details on format standardization efforts for TAG related tools.

\(^{14}\)The idea of “featurization” is very useful for applications such as text generation, supertagging (Kinyon, 2002), and is especially relevant for the automatic acquisition of a MG (see section 5)

\(^{15}\)This example is of course a simplification: for sake of clarity it does not reflect the complex structure of our real "hierarchy".

\(^{16}\)For instance, the 5000 tree grammar for French parses 80% of (simple) TSNLP sentences, and does not parse newspaper text, whereas the 1200 tree hand-crafted Xtag grammar for English does. Basically, instead of solving both under-generation and over-generation problems, a hand-crafted abstract level of syntactic encoding runs the risk of increasing both
ually maintained (and of course one of the major sources of parsing failures is due to missing or erroneous lexical entries).

5 TYPE D Grammars

However, the main potential advantage of such an abstract level of syntactic representation is framework independence. We argue that the main drawbacks of an abstract level of syntactic representation (over-generation, propagation of manual errors to generated trees, interface with the lexicon) may be solved if this abstract level is acquired automatically instead of being hand-crafted. Other problems such as sparse data problems are also handled by such a level of abstraction. This corresponds to type D in our classification. A preliminary description of this work, which consist in automatically extracting the hierarchy nodes of a MetaGrammar from the Penn Treebank (i.e. a high level of syntactic abstraction) may be found in (Kinyon and Prolo, 2002). The underlying idea is that a lot of abstract framework independent syntactic information is implicitly present in the treebank, and has to be retrieved. This includes: subcategorization information, potential valency alternations (e.g. passives are detected by a morphological marker on the POS of the verb, by the presence of an NP-Object "trace", and possibly by the presence of a Prepositional phrase introduced by "by", and marked as "logical-subject"), and realization of arguments (e.g. Wh-extractions are noticed by the presence of a Wh constituent, co-indexed with a trace). In order to retrieve this information, we have examined all the possible tag combinations of the Penn Treebank 2 annotation style, and have determined for each combination, depending on its location in the annotated tree whether it was an argument (optional or compulsory) or a modifier. We mapped each argument to a syntactic function. This allowed us to extract fine-grained subcategorization frames for each verb in the treebank. Each subcategorization frame is stored as a finite number of final classes using the (Gaiffe et al., 2002) MG tool: one class for each subcategorization frame (dimension 1 in Candito's terminology), and one class for each function realization (dimension 3 in Candito’s terminology). The same technique is used to acquire the valency alternation for each verb, and non-canonical syntactic realizations of verb arguments (Wh extractions etc...). This amounts to extracting "hypertags" (Kinyon, 2000) from the treebank, transforming these Hypertags into a MetaGrammar, and automatically generating a TAG from the MG. An example of extraction may be seen on figure 4: expose appears here in a reduced-relative construction. However, from the trace occupying the canonical position of a direct object, the program retrieves the correct subcategorization frame (i.e. tree family) for this verb. Hence, just this occurrence of expose correctly extracts the MG nodes from which both the "canonical tree" and the "Reduced relative tree" will be generated. If one was extracting a simple type B grammar, the canonical tree would not be retrieved in this example.

Input Sentence:
------------
(NP (NP (DT a) (NN group) )
 (PP (IN of) (NP (NP (NNS workers) )
  (RRC (VP (VBN exposed) (NP (-NONE- *) )
  (PP-CLR (TO to) (NP (PRP it) )))
 (ADVP-TMP (NP (QP (RBR more) (IN than) (CD 30) )
  (NNS years) ) (IN ago) )))))

Extracted Output:
#######
#VB: exposed
#Subj: NP-SBJ
#Arguments: NP#DirObj//PP-CLR#PrepObj(to)
#######

Figure 4: An example of extraction from the Penn Treebank

This work is still underway. From the abstract level of syntactic generalization, a TAG will be automatically generated. It is interesting to note that the resulting grammar does not have to closely reflect the linguistic choices of the annotated data from which it was extracted (contrary to type B grammars). Moreover, from the same abstract syntactic data, one could also generate a grammar in another framework (ex. LFG). Hence, this abstract...
level may be viewed as a syntactic interlingua which can solve some portability issues.

6 Conclusion
We have proposed a classification of grammar development strategies and have examined the advantages and drawbacks of each of the four approaches. We have explained how “grammar replication” may prove an interesting task to compare different development strategies, and have described how grammar replication is currently being used in the Xtag project at the University of Pennsylvania in order to compare hand-crafted grammars, grammars generated with MetaRules, and grammars generated with a MetaGrammar. We have reached the conclusion that of the four grammar development strategies proposed, the most promising one consists in automatically acquiring an abstract level of syntactic representation (such as the MetaGrammar). Future work will consist in pursuing this automatic acquisition effort on the Penn Treebank. In parallel, we are investigating how the abstract level we acquire can be used to generate formalisms other than TAGs (e.g. LFG).

Acknowledgements:
We thank the Xtag group, and more particularly W. Schuler and R. Prasad for helpful comments on earlier versions of this work. We also thank B. Crabbé and B. Gaiffe for their help with the LORIA MetaGrammar compiler.

References
T. Becker. 2000. Patterns in metarules for TAG. In Abeille Rambow, editor, Tree Adjoining Grammars, CSLI.

The notion of “syntactic interlingua” was used in other papers as an analogy to the terminology used for Machine translation: “simple” grammar extraction algorithms could be seen as “transfer approaches” (i.e. low level of abstraction) whereas MetaGrammar extraction could be seen as “interlingua” approaches, in the sense that a higher level of abstraction is needed (the “lingua” being a syntactic framework such as TAGs, LFG etc.)

D. Chiang. 2000. Statistical parsing with an automatically-extracted TAG. In ACL-00, Hong-Kong.
Encoding and reusing linguistic information expressed by Linguistic Properties

Caroline Hagège
Xerox Research Centre Europe (XRCE)
6 Chemin de Maupertuis
38240 Meylan - France
Caroline.Hagege@xrce.xerox.com

Gabriel G. Bès
GRIL, Université Blaise Pascal
34 Avenue Carnot
63000 Clermont Ferrand - France
Gabriel.Bes@univ-bpclermont.fr

Abstract

This paper presents a way to express linguistic knowledge independently of any algorithmic machinery and of any particular grammatical formalism. This is performed through Linguistic Properties, that will be presented. First, the status of linguistic knowledge in grammars is discussed, then the Linguistic Properties are presented and two experiments are mentioned. They illustrate the reusability of the linguistic information enclosed in these Properties.

1 Grammar and reusability of linguistic knowledge

One of the central points in linguistically-motivated natural language processing is the notion of grammar. It is commonly accepted that a grammar (see (TM90)) is intended, among other things, to be both a precise tool of natural languages' description and the declarative data source which must be interpreted by a computer.

This means that the same metalanguage (the grammatical formalism) is used to encode the declarative linguistic informations and the rules that will feed a parser.

We consider that this double function of a grammar is a disadvantage from the point of view of the reusability of linguistic knowledge. Indeed, the same device (the grammar) contains both linguistic information and some information adapted to a specific goal (parsing (shallow or not), generation, etc.) and to a specific algorithmic machinery. In order to be reusable and suitable for different goals, we hold that linguistic knowledge must be free from any specific requirements, while being, nevertheless, formally expressed. Furthermore, it must be modularly organised within different levels of explicit granularity in order to offer a tailorable access. Linguistic Properties, which we distinguished from Processes - i.e. effective computational procedures on strings of NL expressions - are a possible way to fulfill these requirements.

2 Linguistic Properties

Linguistic Properties (or simply Properties) were originally developed in the late 90’s within the 5P Paradigm. The 5 Ps stand for Protocole, systematic observations on sentences, Properties, linguistic declarative knowledge, Projections, generalizations on Properties of some natural language, Principles, cross-linguistic constraints on Projections or Principles and Processes that are effective computational procedures. We present in this section the kernel of Properties

The following example introduces intuitively Properties and their potential of modularity.

Given the following French expression in (i), it is possible to distinguish, in its metalanguage, different layers or aspects, illustrated by (i-a) to (i-c).

i les trois fleurs
i-a \{ les-[art, def], trois-[card], fleurs-[n] \}

i-b ( les-[art, def]1 trois-[card]2 fleurs-[n]3 Nn ,
<i-a>, \{<1, 3>, <2, 3>, <3, 3>\} )

(i) is a string of expressions; (i-a) is a set, each member of which is an expression of (i) associated to a category (or cat, see below), e.g. [art, def]; (i-b) is a parenthetised list obtained applying order relations to (i-a), and indexing it with the identifier Nn (for Nominal nuclear phrase); (i-c) is associated (i-b) to a set of pairs < p, q >, where p and q are positions in (i-b); the pair links the element in position p to the element in position q.

We will say that (i-b) is a basic model reduced to its model string, namely (i-b), that (i-a) is a pack

1For a complete and more formalised version, see (BH01).

The 5P Paradigm was presented as such in (BB99) and (BBH01) though antecedents, cited in (BH01), go back to a 89 report to the ESPRIT 393 ACORD European Project, where the notion of descriptive metalanguage, today's Properties, were explicitly introduced. For published work, see (BHC99), (BB01). For a different but related approach, see (BB99) and (BBH01) which remain “grammar oriented”.

2Another layer, expressing a semantic representation, can be added, but it is not discussed in this paper.
associated to (i-b), that pairs \(< p, q \rangle\) are **Arranging pairs**, that (i-c) is an **arranged model** incorporating the model string (i-b).

We distinguish three basic kinds of Properties: Existence Properties, Linearity Properties, Arranging Properties. Packs as in (i-a) are specified by Existence Properties; order relations, by Linearity properties; Arranging pairs, by Arranging Properties. Each identifier - e.g. \( N_n \) - has its associated set of Properties, e.g. Properties-\( N_n \), M-\( N_n \) being the set of all and only the models \( m-N_n \) satisfying Properties-\( N_n \). Properties are expressed on symbols which are cat's or identifiers. From hereafter we use \( Sm \) as a metavariable on identifiers, and \( Sy \) as a metavariable on cat's and \( Sm \)’s.

A cat is a set of label/value pairs, or, in reduced notation, a set of values (as in previous examples). A maximum categorie (mc) is a cat to which no other value can be added. The assumed Lexicon is a set of lexical entries, each one being an expression associated to one or more mc’s; cat’s subsumes cat’d, if cat’d \( \subseteq \) cat’d.

The whole system can be viewed as a modular axiomatic system in which models are the objects satisfying different kinds of Properties\(^2\). A basic model, as in (i-b), with its associated pack, as in (i-a), satisfies Existence and Linearity Properties; an arranged model, as in (i-c) satisfies also Arranging Properties. Furthermore, giving a set of Properties, a model can satisfy some, but not all of them. Properties can be expressed independently the ones from the others, and in any order. The set of features from which cat’s arc build can be more or less extended, and, consequently, the granularity of cat’s, and of Properties expressed on them, more or less refined.

The model substitution rule relates the identifiers \( Sm^1, \ldots, Sm^n \), each one with its associated M-\( Sm^1 \). In a \( m-Sm^1 \) with a \( Sm^2 \), it substitutes some \( m-Sm^2 \) for \( Sm^2 \). E.g., assuming \( \{\text{neg}, \text{adj}\}_1 \) \( \text{ADJ} \), as a French m-\( \text{ADJ} \) (underlying, e.g. the string *pas belles*), the model substitution rule obtains (2) from the following (1).

\[ \langle \text{art1 ADJ n2 n3 n4} \rangle_{\text{NN}}, \langle \text{<1, 3>, <2, 3>, <3, 3> } \rangle \]
\[ \langle \text{art1 neg2 adj3 n4 n5} \rangle_{\text{NN}}, \langle \text{<1, 4>, <2, 3>, <3, 4>, <4, 4> } \rangle \]

In an optimal situation, Properties associated to the \( Sm \)’s of some NL together with a Lexicon and the model substitution rule, specify the whole set of models required to describe the strings of expressions of the NL. We concentrate in the following in the intuitive presentation of Properties specifying models obtained without the model substitution rule. Given the different kinds of Properties, we will intuitively characterise the conditions that must be fulfilled by a model in order to satisfy each one them \(^4\).

Subsumption is the basic relation linking models and Properties. We already defined above subsumption between cat’s. As a shorthand, we say here that \( Sm^1 \) subsumes \( Sm^2 \) if \( Sm^1 = Sm^2 \). Furthermore, given sets \( S^1 \) and \( S^2 \) of \( Sy \) symbols, we say that \( S^1 \) subsumes \( S^2 \) if there is a bijection function between \( S^1 \) and \( S^2 \) such that each \( Sy^m \) in \( S^2 \) subsumes its corresponding \( Sy^m \) in \( S^1 \).

### 2.1 Existence Properties

Existence Properties associated to some M-\( Sm \) specify the set of packs from which any m-\( Sm \) is obtained. We distinguish five kinds: Vocabulary property, Unicity property, Nucleus Property, Exigency Property, Exclusion Property.

The Vocabulary Property, spelled by \( V_{Sm} = \{ Sy^1, \ldots, Sy^n \} \) says that each symbol in the pack associated to a \( m - Sm^1 \) is subsumed by some symbol in \( V_{Sm} \), and each symbol in \( V_{Sm} \) subsumes some symbol in the pack of some \( m - Sm^2 \). E.g. (singleton categories are spelled with their value) French \( V_{NN} = \{ \text{det}, \text{poss}, \text{card}, \text{noun} \ldots \} \) is the vocabulary for \( N_n \) (nominal nuclear) French phrases (roughly, nominal chunks), assuming mc’s: \{det, art, def\[\ldots\], [det, art, ind\[\ldots\], [det, dem\[\ldots\], which are associated in the Lexicon to, respectively, the expressions \{les, la, le\[\ldots\], \{un, une, des\[\ldots\], \{ce, ces, celles\[\ldots\].

The Unicity Property, spelled by \( U_{Sm} = \{ Sy^1, \ldots, Sy^n \} \) says that there are no two symbols in the pack associated to a \( m - Sm \) subsumed by one and the same symbol in \( U_{Sm} \). E.g. French \( U_{NN} = \{ \text{det}, \text{card}\[\ldots\} \) expresses that there are no two articles, or two demonstratives or an article and a demonstrative in a \( Nn \) phrase.

The Nucleus Property, spelled by \( Nu_{Sm} = \{ Sy^1, \ldots, Sy^n \} \) says that in each \( m - Sm \) there is one and only one position with a nucleus symbol - spelled \( Sy \) - subsumed by some symbol in \( Nu_{Sm} \). E.g. French \( Nu_{NN} = \{ \text{card}, \text{quant}, \text{noun} \ldots \} \) expresses that \( Nn \) phrases can have as a Nucleus either a cardinal (e.g. if a \( \text{au (tous)}_{NN} \) or a quantifier (e.g. if a \( \text{a} \) \( \text{au (tous)}_{NN} \)), or a noun (e.g. if a \( \text{a} \) \( \text{les fleurs)}_{NN} \)).

The Exigency Property, spelled by \( S^0 \rightarrow Sm \{ S^1, \ldots, S^n \} \)

\(^2\)The system benefits from the concept of factorizing relations of standard production rules of new traditional grammars. See in particular the LP statements of GPSG dissociated from dominance ID rule, and dependency grammar ([Tes69]), early HPSG in ([Pâ87]). The system tries to push this basic idea to its limits, dissolving thus the concept of production rules. An analog of what the system of Properties is expected to express compared to production rules, can be seen in regular expressions as compared to production grammars of type 3 in the Chomsky’s hierarchy.

\(^4\)For a more formal and complete presentation, see [BH01].
says (remember that S’s spell sets of Sy) that if in
the pack of a m – Sm there is included a set of
symbols S0 subsumed by S0 there must be also some
S included in the pack such that S is subsumed by
S0 21. E.g. French \{[n, c] \rightarrow Nn \{\{det\}, \{card\}\}\},
where [n, c] stands for common nouns, express that
common nouns require a determiner or a cardinal.
The Exclusion Property, spelled by
S0 2 sm \{S1, …, Sn\}
says that if in the pack of a m – Sm there is included a
set of symbols S0 subsumed by S0, then there is
not included a set S such that S is subsumed by
S0 21. E.g. French \{[quant, plf] \rightarrow Nn \{[art, indf]\}\} express that the quantifier tous cannot coexist with
an indefinite article in a Nn phrase.

2.2 Linearity Properties
Linearity Properties express order relations. A Linearity Property is spelled by Sy0 \rightarrow sm Sy1…Sy0n. It
says that if in a m – Sm there is a symbol subsumed by
Sy0 and a symbol subsumed by Sy0 1, the former
precedes the latter. E.g.: in French m-Nn’s, a quantifier tout(e-, s) precedes all other cat’s, which is expressed by quant
\rightarrow Nn n, det, poss, card...

2.3 Arrowing Properties
The basic role of Arrowing Properties is to specify the
graph - i.e. the set of Arrowing pairs - that is the
backbone from which the semantic representation is build. An arrowing pair (Ar) is a pair p,q, where p and q are positions in the model string, and which can be understood as "the Sy in position p
arrows to the Sy in position q". An Ar is thus an arc
between two Sy’s. Ar’s are expressed by arrowing
formulae, which, in their simplest formulation, are spelled Sy0 \rightarrow sm Sy01. It is also possible to spell disjunctive arrowing, expressing that some Sy arrows
to either Sy0 or to Sy01. By a general convention, a
nucleus SYb arrows to himself. General conditions
limit the expressive power of Arrowing formulae, as-
suring, among others, that the result graph must
be connected, and, with the exception of the reflexive
arrowing of SYa, acyclic.
E.g., among French Arrowing formulae, there is
quant \rightarrow Nn SY , where SYa is a variable on
the Nucleus and which express that the quantifier
tous arrows to any Nucleus in a Nn phrase:
<\{tous, tous\}/Nn, \{<1, >2, >3\}>,
<\{tous1, leu2, garon3\}/Nn, \{<1, >2, >3, >3\}>

3 Exploring properties
Two experiments have been carried out in the
exploration of Existence and Linearity properties. In
the first experiment, Linguistic Properties were used
to derive the linguistic data structures used by a
chunker and a NP extractor for Portuguese (see
(BHC99)). In the second experiment, Linguistic

Figure 1: The processing chain for NP extraction

Properties were used to structure lexical entries in
an HPSG-style grammar (see (BB98) and (Hag00)). In
both cases, the basic idea is the same: associate to
each category declared for a given model the combi-
natorial information attached to this category in a
certain grammatical context.
We describe here these two experiments in more
details

3.1 First experiment
3.1.1 Context
A fine-grained description of the Portuguese NP has
been accomplished with Linguistic Properties and
we wanted to use this linguistic description in or-
der to extract NPs from Portuguese running texts.
In a first step, the input text is tokenized and
morphologically analyzed (SMORPH (AM98)). Then,
the tokenized and morphologically analyzed text is
pre-processed, eliminating partially some ambiguity
and grouping or ungrouping some tokens previously
delimited (MPS). Then the text is chunked and fi-
ally, NPs (defined as regular expressions of chunks)
are extracted. Figure 1 summarizes the processing
chain for NP extraction.

Our chunker (called AF) consists in a very simple
algorithm (see (BHC99)) which uses linguistic struc-
tures (called leaves) associated to each token of the
text and tries to concatenate these structures from
left to right until the end of the text. Each concate-
nation introduces constraints for the next concate-
nation and, during parsing, part of the ambiguity is
solved as a side effect when concatenation fails.

To illustrate intuitively how our chunker works, as-
sume we want to analyze the following string with the
following leaves:

As danças
(The dances)

Leaf 1 This leaf is associated to As
- The lemma associated to As is o
- The category is a definite article
- The model where this category appears is nom-
inal chunk
• This category never starts a model of nominal chunk
• This category never ends a model of nominal chunk
• The set of categories that can follow this category in this model contains noun

Token dança is ambiguous plural-noun and verb (dances and dance) and have the following two associated leaves.

Leaf 2
• The lemma associated to dança is dança
• The category is noun
• The model where this category appears is nominal chunk
• This category can start a model of nominal chunk
• This category always ends a model of nominal chunk
• The set of categories that can follow this category in this model is empty

and

Leaf 3
• The lemma associated to dança is dançar
• The category is verb
• The model where this category appears is verbal chunk
• This category can start a model of verbal chunk
• This category can end a model of verbal chunk
• The set of categories that can follow this category in this model contains clitic pronouns.

After the concatenation of the leaf 1 associated to As, the only possibility is to concatenate leaf 2 because the model string on the right of As cannot be closed (leaf 1 never ends a nominal chunk) and leaf 3 is not a possible successor of leaf 1.

The process of chunking is reduced to perform all the possible concatenations of leaves from left to right, each concatenation being restricted by the previous concatenation.

Our chunker was used to process Portuguese text and was evaluated on the task of NP extraction with the results of 88% precision and 81.5% recall on the NP detection. (No exact match was required but the NP head detected in the reference corpus is extracted)

3.1.2 Leaves and Leaf Patterns
A leaf is thus a structure of the following form (We represent it as a Prolog predicate).
leaf(WF, L, Cat, ModId, BStat, EStat, Foll).

Where:
• WF (Word Form) is the token found in the text to analyze
• L (Lemma) is the corresponding lemma
• Cat (category) is the corresponding category
• ModId (Model Identifier) identifies the model in which this category can appear
• BStat (Begin Status) is the integer 0, 1 or 2 meaning respectively that this category never, always or sometimes starts the model identified by ModelIdentifier
• EStat (End Status) is the integer 0, 1 or 2 meaning respectively that this category never, always or sometimes ends the model identified by ModelIdentifier
• Foll (Followings) is the set of categories that can follow the category Cat in the model identified by ModId (The empty set when EndStatus is 1)

We call a Leaf Pattern a leaf structure in which the first argument (the word form) is not instantiated. Our problem here is to deduce, from the Properties, all the Leaves Patterns that are necessary to analyze one text.

3.1.3 Relations between categories appearing in a given model string
Given the vocabulary $V$ of some model identifier $Sm$, it is possible, using Existence Properties and and Linearity Properties to define the following relations in $V \times V^\circ$. $a$ and $b$ being elements of $V$.

precede1: $a$ precede $b$ if in any $m - Sm$ containing $a$ and $b$, $a$ always precedes $b$
order: $a$ order $b$ if there is at least a $m - Sm$ in which it is possible to say that $a$ precedes $b$ or that $b$ precedes $a$.

exige: $a$ exige $b$ if for each $m - Sm$ where $a$ appears, $b$ also appears.
exclu: $a$ exclu $b$ if there is no $m - Sm$ with $a$ and $b$.

It is also possible to define two subsets of $V$, $So$ and $S1$. $So$ consists of the elements of $V$ that are always alone in a model string and is defined the following way:

---

See [Hag00] for more details.
\[ S_0 = \{a \in V \mid \forall b \in V \text{ except}(a, b)\} \]

\( S_1 \) is the complementary of \( S_0 \) in \( V \).

For each category \( a \) of \( V \), it is also possible to define the set \( LP_a \) as the set of all categories that possibly follow \( a \) in at least one model string.

Having these relations and these sets, one can define the subsets of \( V \) that always, sometimes and never start a model string and the subsets of \( V \) that always, sometimes and never end a model string, which is precisely what is needed to define the leaves together with \( LP_a \).

We called these subsets \( AS \) (Always start), \( SS \) (Sometimes start), \( NS \) (Never start), \( AE \) (Always end), \( SE \) (Sometimes end) and \( NE \) (Never end).

With these definitions and considering the set of Properties that define the models identified by \( mSm \) we can then construct a set of leaf patterns the following way:

- The first argument is a variable (that will be then instantiate with a linguistic form present in the text)
- The second argument of the leaf predicate is instantiated to an element of \( V \)
- The third argument of the leaf predicate is instantiated to \( mSm \)
- The fourth argument of the leaf predicate is instantiated to 1, 2, 0 according to the fact that this element is member of \( AS, SS \) or \( NS \).
- The fifth argument of the leaf predicate is instantiated to 1, 2 or 0 according to the fact that this element is member of \( AE, SE \) of \( NE \).
- The sixth argument corresponds to the set \( LP_{cat} \) being \( cat \) the category that is present in the second argument.

3.2 Second experiment

In this second experiment, we want to use the Properties defined for the nominal chunk in order to construct lexical entries that can enable to analyze nominal chunks in an HPSG-style (see (CS94) and (SW99)). The HPSG grammar was then implemented in ALE (Attribute Logic Engine, developed by B. Carpenter and G. Penn). Only the syntactic part of the lexical entries is taken into account.

We decided that for our grammar a nominal chunk has to be a saturated sign with a nominal head. The analysis fails if:

- No analysis is produced
- A linguistic sign is obtained but it is not saturated

3.2.1 What we have to consider

We have to take into account the structuration of linguistic signs that HPSG formalism stipulates. That is:

**In the type hierarchy** A linguistic sign has in the path SYNSEM: SYN: LOC: CAT: HEAD (from now on the whole path is designed by \( H \)) a value of type \( head \) that has the following subtypes.

- head
  - subst
  - noun
  - verb
  - adj
  - func
  - det
  - mark

**In the structuration of lexical signs** If the value of \( HEAD \) is \( noun \) then there is a value for the path SYNSEM: SYN: LOC: CAT: VAL: SPR (from now on just VAL: SPR) which is of type list of linguistic signs.

If the value of \( HEAD \) is \( det \) then the value of the path SYNSEM: SYN: LOC: CAT: HEAD: SPEC (from now on just SPEC) is of type non-empty list of linguistic signs.

Finally, if the value of \( HEAD \) is \( adj \) then the value of VAL: SPR is the empty list and the value of SPEC is the empty list.

3.2.2 What we can infer from Linguistic Properties

**Definition of the set of categories that never can be alone in a nominal chunk model** Considering the set of Properties modelling nominal chunks, we can define the subset \( S2 \) of the vocabulary \( V \) consisting in the set of categories that never can be alone in a model.

\[ S_2 = \{a \in V \mid \exists b \in V \text{ except}(a, b)\} \]

**Rule 1** All the categories that are members of the above defined sets \( AE \cup SE \) must have the value \( noun \) for \( HEAD \). Nouns and nominalized adjectives that can be the head of a nominal chunks are concerned by this rule.

**Rule 2** All the categories that are member of the set \( S_0 \) (defined above) must be associated to a lexical entry with the value \( empty \) list for VAL: SPR. Plural nouns and pronouns that can be used alone in a nominal chunk are concerned by this rule. Note that Rule 2 applies to all the categories for which Rule 1 applies too as \( S_0 \) is included in \( AE \).
Rule 3 All the categories that are members of $(AS \cup SS) \cap S2$ and that are not considered traditionally as adjectives have the value *det* for HEAD and have for SPEC a value of type *sign* that is subsumed by SYNE:SYN:LOC:CAT:HEAD:noun. Determiners are concerned with this rule.

Rule 4 This rule handles with possible combination of determiners (or determiners and quantifiers) and gives one possibility to combine them together. It stipulates that if a category treated in Rule 3 can precede another category treated in Rule 3 (we know that through the relation order defined above), then it is necessary to provide either a complex determinant structure, or to add to the VAL:SPR value of all the categories treated in Rule 1 the whole list of determiners.

Rule 5 Any category of S2 that has not be considered by Rule 3 are taken as adjective and have the value *adj* for HEAD.

3.3 Extensions

It is well known that there are different kinds and different sources of ambiguity. We point here two of them and how they can be treated within our framework.

A linguistic expression can be associated in the Lexicon to more than one *me* : it is, e.g., the case for Leaf 2 and 3 of the first experiment in Section 3.1. The ambiguity is there resolved thanks to Leave 1. Suppose that, as in French, there is a string of expressions in a related pattern - as *le juge* - where both expressions are ambiguous (*le* being an article and a clitic, *juge* a noun or a verb). In this situation, the ambiguity is maintained, the system specyfing both *m-Nu* and *m-Vn* for the *le juge* (respectively, a nominal and a verbal chunk). This ambiguity will be resolved in a context - e.g. to the right of a preposition Leaf - in which the expression can follow if it is specified as *m-Nu* but not if it is as *m-Vn*.

As an important side-effect of the first experiment (Section 3.1), it is remarked in H00 (these) that applying the processing chain (see Figure 1) to previously and independently disambiguated expressions improves very little the final results. We think that observations as this one indicate that the incremental tactic of bottom-up parsing and that the requirement of a disambistuation layer before parsing is not the only possible way.

In the experiments presented in this paper we work on model strings build with *cat’s*, not with *Sm* symbols (identifiers). Two basic types of identifiers are recognize: the one related to nuclear phrases or chunks, which are spelled *Xn*, *X* being a variable on *N*, *V, ADJ*... and the ones related to not nuclear phrase, spelled with the bare *X* and its possible instantiations. In general, a *Xn* in the model string of some *Xn* are not ambiguously related. But attachment of *Xn* into the right of a pattern *Xn*... *Xn* can be ambiguous.

Properties here presented apply exactly the same on models strings with or without *Sm’s*. So the previously characterised ambiguity can be expressed by disjunctive arrowing in arrowing formula (see Section 2.3).

4 Conclusion

In current work on syntax (heuristics for robust parsing (see (AMCR01), (TJ97)) or unification-based grammatical formalisms), it is quite difficult to access pure linguistic information since the same syntax is used both for the linguistic description and the rules for the parsers. We believe that the expression of linguistic information by means of Linguistic Properties is a possible step in the direction of the centralization of linguistic knowledge with the following benefits:

- Syntacticians would spend less time rewriting rules carrying the same information for different formalisms or for different parsers.
- The construction of a grammatical reference, expressed in a formalized and non-ambiguous way.

The notion of a grammar as a source of linguistic knowledge is thus revisited in favor of a notion of linguistic knowledge base from which syntactic information could be extracted for one or another specific grammar or application. The two experiments that we described above seem to be a step in this direction.

References


[1]In Portuguese, combination of determiners and quantifiers.

---

[8]The idea of a linguistic knowledge base was originally mentioned by G. G. Bès in a project proposal (Cale) submitted in 1991.
Grammar and Lexicon in the Robust Parsing of Italian
Towards a Non-Naïve Interplay

Roberto BARTOLINI
Istituto di Linguistica Computazionale CNR
Area della Ricerca Via Moruzzi 1
56100 PISA (Italy)

Alessandro LENCI
Università di Pisa Via Santa Maria 36
56100 PISA (Italy)

Simonetta MONTEMAGNI
Istituto di Linguistica Computazionale CNR
Area della Ricerca Via Moruzzi 1
56100 PISA (Italy)

Vito PIRRELLI
Istituto di Linguistica Computazionale CNR
Area della Ricerca Via Moruzzi 1
56100 PISA (Italy)

{roberto.bartolini, alessandro.lenci, simonetta.montemagni, vito.pirrelli}@ilc.cnr.it

Abstract
In the paper we report a qualitative evaluation of the performance of a dependency analyser of Italian that runs in both a non-lexicalised and a lexicalised mode. Results shed light on the contribution of types of lexical information to parsing.

Introduction
It is widely assumed that rich computational lexicons form a fundamental component of reliable parsing architectures and that lexical information can only have beneficial effects on parsing. Since the beginning of work on broad-coverage parsing (Jensen 1988a, 1988b), the key issue has been how to make effective use of lexical information. In this paper we put these assumptions to the test by addressing the following questions: to what extent should a lexicon be trusted for parsing? What is the neat contribution of lexical information to overall parse success?

We present here the results of a preliminary evaluation of the interplay between lexical and grammatical information in parsing Italian using a robust parsing system based on an incremental approach to shallow syntactic analysis. The system can run in both a non-lexicalised and a lexicalised mode. Careful analysis of the results shows that contribution of lexical information to parse success is more selective than commonly assumed, thus raising the parallel issues of how to promote a more effective integration between parsers and lexicons and how to develop better lexicons for parsing.

1 Syntactic parsing lexicons
Syntactic lexical information generally feeds parsing systems distilled in subcategorization frames. Subcategorization is a formal specification of a phrase structure in terms of the type of arguments syntactically selected by the predicate entry (e.g., the verb hit selects for a subject NP and an object NP). Lexical frames commonly include: i.) number of selected arguments, ii.) syntactic categories of their possible realization (NP, PP, etc.), iii.) lexical constraints on the argument realization (e.g., the preposition heading a PP complement), and iv.) the argument functional role. Other types of syntactic information that are also found in syntactic lexicons are: argument optionality, verb control, auxiliary selection, order constraints, etc. On the other hand, collocation-based lexical information is only rarely provided by computational lexicons, a gap often lamented in robust parsing system development.

A number of syntactic computational lexicons are nowadays available to the NLP community. Important examples are LDOCE (Procter 1987), ComLex (Griswold et al. 1994), PAROLE (Rumi et al. 1998). These lexicons are basically hand-crafted by expert lexicographers, and their natural purpose is to provide general purpose, domain-independent syntactic information, covering the most frequent entries and frames. On the other hand, parsing systems
often complement general lexicons with corpus-driven, automatically harvested syntactic information (Federici et al. 1998b, Briscoe 2001, Korhonen 2002). Automatic acquisition of subcategorization frames allows systems to access highly context dependent constructions, to fill in possible lexical gaps and eventually rely on frequency information to tune the relative impact of specific frames (Carroll et al. 1998).

Lexicon coverage is usually regarded as the main parameter affecting use of lexical information for parsing. However, the real comparative impact of the type (rather than the mere quantity) of lexical information has been seldom discussed. Our results show that the contribution of various lexical information types to parse success is not uniform. The experiment focuses on a particular subset of the information available in syntactic lexicons - the representation of PP complements in lexical frames - tested on the task of PP-attachment. The reason for this choice is that this piece of information occupies a central and dominant position in existing lexicons. For instance in the Italian PAROLE lexicon, more than one third of verb frames contain positions realized by a PP, and this percentage raises up to the near totality noun-headed frames.

2 Robust Parsing of Italian

The general architecture of the Italian parsing system used for testing adheres to the following principles: 1) modular approach to parsing, 2) underspecified output (whenever required), 3) cautious use of lexical information, generally resorting to in order to refine and/or further specify analyses already produced on the basis of grammatical information. These principles underlie other typical robust parsing architectures (Chanod 2001, Briscoe and Carroll 2002).

The system consists of i.) CHUNK-IT (Federici et al. 1998a), a battery of finite state automata for non-recursive text segmentation (chunking), and ii.) IDEAL (Lenci et al. 2001), a dependency-based analyser of the full range of intra-sentential functional relations (e.g. subject, object, modifier, complement, etc.). CHUNK-IT requires a minimum of lexical knowledge: lemma, part of speech and morpho-syntactic features. IDEAL includes in turn two main components: (i.) a Core Dependency Grammar of Italian; (ii.) a syntactic lexicon of ~26,400 subcategorization frames for nouns, verbs and adjectives derived from the Italian PAROLE syntactic lexicon (Ruiny et al. 1998). The IDEAL Core Grammar is formed by ~100 rules (implemented as finite state automata) covering major syntactic phenomena,1 and organized into structurally-based rules and lexically-based rules. IDEAL adopts a slightly simplified version of the FAME annotation scheme (Lenci et al. 2000), where functional relations are head-based and hierarchically organised to make provision for underspecified representations of highly ambiguous functional analyses. This feature allows IDEAL to tackle cases where lexical information is incomplete, or where functional relations cannot be disambiguated conclusively (e.g. in the case of the argument vs. adjunct distinction). A “confidence score” is associated with some of the identified dependency relations to determine a plausibility ranking among different possible analyses.

In IDEAL, lexico-syntactic information intervenes only after possibly underspecified dependency relations have been identified on the basis of structural information only. At this second stage, the lexicon is accessed to provide extra conditions on parsing, so that the first stage parse can be non-monotonically altered in various ways (see section 3.3). This strategy minimises the impact of lexical gaps (whether at the level of lemma or of the associated subcategorization frames) on the system performance (in particular on its coverage).

3 The Experiment

3.1 The Test Corpus (TC)

The test corpus contains a selection of sentences extracted from the balanced partition of the Italian Syntactic Semantic Treebank (ISST, Montemagni et al. 2000), including articles from

1 Adjectival and adverbial modification; negation; (non-extraposed) sentence arguments (subject, object, indirect object); causative and modal constructions; predication constructions; PP complementation and modification; embedded finite and non-finite clauses; control of infinitival subjects; relative clauses (main cases); participial constructions; adjectival coordination; noun-noun coordination (main cases); PP-PP coordination (main cases); cliticization.
contemporary Italian newspapers and periodicals covering a high variety of topics (politics, economy, culture, science, health, sport, leisure, etc.). TC consists of 23,919 word tokens, corresponding to 721 sentences (with a mean sentence length of 33.18 words, including punctuation tokens). The mean number of grammatical relations per sentence is 18.

3.2 The Baseline Parser (BP)

The baseline parser is a non-lexicalised version of IDEAL including structurally-based rules only. The mean number of grammatical relations per sentence detected by BP in TC is 15.

The output of the baseline parser is shallow in different respects. First, it contains underspecified analyses, resorted to whenever available structural information does not allow for a more specific syntactic interpretation; e.g. at this level, no distinction is made between arguments and modifiers, which are all generically tagged as “complements”. Concerning attachment, the system tries all structurally-compatible attachment hypotheses and ranks them according to a confidence score. Strong preference is given to rightmost attachments; e.g. a prepositional complement is attached with the highest confidence score (50) to the closest, or rightmost, available lexical head. In the evaluation reported in section 4, we consider top-ranked dependents only, i.e. those enforcing rightmost attachment. Moreover, in matching the relations yielded by the parser with the ISST relations in TC we make allowance for one level of subsumption, i.e. a BP relation can be one level higher than its ISST counterpart in the hierarchy of dependency relations. Finally, the BP output is partial with respect to those dependencies (e.g. a that-clause or a direct object) that would be very difficult to identify with a sufficient degree of confidence through structurally-based rules only.

3.3 The Lexically-Augmented Parser (LAP)

The lexically-augmented version of IDEAL includes both structurally-based and lexically-based rules (using the PAROLE lexicon). In this lexically-augmented configuration, IDEAL first tries to identify as many dependencies as possible with structural information. Lexically-based rules intervene later to refine and/or complete structurally-based analyses. Those structurally-based hypotheses that find support in the lexicon are assigned the highest score (60). The contribution of lexically-based rules is non-monotonic: old relations can eventually be downgraded, as they happen to score, in the newly ranked list of possible relations, lower than their lexically-based alternatives. Furthermore, specification of a former underspecified relation is always accompanied by a re-ranking of the relations identified for a given sentence; from this re-ranking, restructuring (e.g. reattachment of complements) of the final output may follow.

LAP output thus includes:
a) fully specified dependency relations: e.g. an underspecified dependency relation such as “complement” (COMP), identified by a structurally-based rule, is rewritten, when lexically-supported, as “indirect object” (OBJ) and assigned a higher confidence value;
b) new dependency relations: this is the case, for instance, of that-clauses, direct objects and other relation types whose identification is taken to be too difficult and noisy without support of lexical evidence;
c) underspecified dependency relations, for those cases that find no lexical support.

The mean number of grammatical relations per sentence detected by LAP in TC is 16. In the evaluation of section 4, we consider top-ranked dependents only (confidence score ≥ 50), corresponding to either lexically-supported dependency relations or – in their absence – to rightmost attachments. Again, in matching the relations yielded by the parser with the ISST relations in TC we make allowance for one level of subsumption.

4 Analysis of Results

The parsing outputs of BP and LAP were compared and projected against ISST annotation to assess the contribution of lexical information to parse success. In this paper, we focus on the evaluation of how and to which extent lexico-syntactic information contributes to identification of the proper attachment of prepositional complements. For an assessment of the role and impact of lexical information in the analysis of dependency pairs headed by specific words, the interested reader is referred to Bartolini et al. (2002).
4.1 Quantitative Evaluation

Table 1 summarises the results obtained by the two different parsing configurations (BP and LAP) on the task of attaching prepositional complements (PC). Prepositional complements are classified with respect to the governing head: PC_{VNA} refers to all prepositional complements governed by V(eral), N(ominal) or A(djectival) heads. PC_{V} is the subset with a V(eral) head and PC_{N} the subset with a N(ominal) head. For each PC class, precision, recall and f-score figures are given for the different parsing configurations. Precision is defined as the ratio of correctly identified dependency relations over all relations found by the parser (prec = correctly identified relations / total number of identified relations); recall refers to the ratio of correctly identified dependency relations over all relations in ISST (recall = correctly identified relations / ISST relations). Finally, the overall performance of the parsing systems is described in terms of the f score, computed as follows: 2 prec recall / prec + recall.

<table>
<thead>
<tr>
<th>ISST</th>
<th>BP</th>
<th></th>
<th></th>
<th>LAP</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Prec</td>
<td>recall</td>
<td>F score</td>
<td>Prec</td>
<td>recall</td>
<td>f score</td>
</tr>
<tr>
<td>PC_{VNA}</td>
<td>3458</td>
<td>75.53</td>
<td>57.40</td>
<td>65.23</td>
<td>74.82</td>
<td>61.02</td>
</tr>
<tr>
<td>PC_{V}</td>
<td>1532</td>
<td>75.43</td>
<td>45.50</td>
<td>56.76</td>
<td>74.23</td>
<td>49.50</td>
</tr>
<tr>
<td>PC_{N}</td>
<td>1835</td>
<td>73.53</td>
<td>80.82</td>
<td>77.00</td>
<td>72.76</td>
<td>81.36</td>
</tr>
</tbody>
</table>

Table 1. Prepositional complement attachment in BP and LAP

<table>
<thead>
<tr>
<th>Lexicalised atts</th>
<th>Confirmed atts</th>
<th>Restructured atts</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>total</td>
<td>OK</td>
</tr>
<tr>
<td>PP_{VNA}</td>
<td>919</td>
<td>819</td>
</tr>
<tr>
<td>PP_{V}</td>
<td>289</td>
<td>244</td>
</tr>
<tr>
<td>PP_{N}</td>
<td>629</td>
<td>575</td>
</tr>
</tbody>
</table>

Table 2. Lexicalised attachments

To focus on the role of the lexicon in either confirming or revising structure-based dependencies, lexically-supported attachments are singled out for evaluation in Table 2. Their cumulative frequency counts are reported in the first three columns of Table 2 ("Lexicalised attachments"), together with their distribution per head categories. Lexicalised attachments include both those structure-based attachments that happen to be confirmed lexically ("Confirmed attachments"), and restructured attachments, i.e. when a prepositional complement previously attached to the closest available head to its left is eventually re-assigned as the dependent of a farther head, on the basis of lexicon look-up ("Restructured attachments"). Table 2 thus shows the impact of lexical information on the task of PP attachment. In most cases, 89% of the total of lexicalised attachments, LAP basically confirms dependency relations already assigned at the previous stage. Newly discovered attachments, which are detected thanks to lexicon look-up and re-ranking, amount to only 11% of all lexicalised attachments, less than 3% of all PP attachments yielded by LAP.

4.3 Discussion

4.3.1 Recall and precision on noun and verb heads

Let us consider the output of BP first. The striking difference in the recall of noun-headed vs verb-headed prepositional attachments (on comparable levels of precision, rows 2 and 3 of Table 1) prompts the suggestion that the typical context of use of a noun is more easily described in terms of local, order-contingent criteria (e.g. rightmost attachment) than a verb context is. We can give at least three reasons for that. First, frame bearing nouns tend to select fewer argu-
ments than verbs do. In our lexicon, 1693 verb-headed frames out of 6924 have more than one non subject argument (24.4%), while there being only 1950 noun-headed frames out of 15399 with more than one argument (12.6%). In TC, of 2300 head verb tokens, 328 exhibit more than one non subject argument (14%). Rightmost attachment trivially penalises such argument chains, where some arguments happen to be overtly realised in context one or more steps removed from their heads. The second reason is sensitive to language variation; verb arguments tend to be dislocated more easily than noun arguments, as dislocation heavily depends on sentence-level (hence main verb-level) phenomena such as shift of topic or emphasis. In Italian, topic-driven argument dislocation in preverbal position is comparatively frequent and represents a problem for the baseline parser, which works on a head-first assumption. Thirdly, verbs are typically modified by a wider set of syntactic satellites than nouns are, such as temporal and circumstantial modifiers (Di K 1989). For example, deverbal nouns do not inherit the possible temporal modifiers of their verb base (*I run the marathon in three hours*, but *the run of the marathon in three hours*). Modifiers of this sort tend to be distributed in the sentence much more freely than ordinary arguments.

4.3.2 Impact of the lexicon on recall
Of the three above mentioned factors, only the first one has an obvious lexical character. We can provide a rough estimate of the impact of lexical information on the performance of LAP. The lexicon filter contributes a 9% increase of recall on verb complements (4% over 45.5%), by correctly reattaching to the verbal head those arguments (61) that were wrongly attached to their immediately preceding constituent by BP. This leads to an overall 49.5% recall. All remaining false negatives (about 48%) are i) either verb modifiers or ii) proper verb arguments lying out of the reach of structure-based criteria, due to syntactic phenomena such as complement dislocation, complex coordination, parenthetic constructions and ellipsis. We shall return to a more detailed analysis of false negatives in section 4.3.4. In the case of noun complements, use of lexical information produces a negligible increase of recall: 0.6% (0.5% over 80.8%). This is not surprising, as our test corpus contains very few cases of noun-headed argument chains, fewer than we could expect if the probability of their occurrence reflected the (uniform) type distribution of noun frames in the lexicon. The vast majority of noun-headed false negatives, as we shall see in more detail in a moment, is represented by modifiers.

4.3.3 Impact of the lexicon on precision
Reattachment is enforced by LAP when the preposition introducing a candidate complement in context is found in the lexical frame of its head. Table 2 shows that ~37% of the 103 restructured attachments proposed by the lexicon are wrong. Even more interestingly, there is a strong asymmetry between nouns and verbs. With verb heads, precision of lexically-driven reattachments is fairly high (~70%), nonetheless lower than precision of rightmost attachment (~75%). In the case of noun heads, the number of lexically reattached dependencies is instead extremely low. The percentage of mistakes is high, with precision dropping to 26.6%.

The difference in the total number of restructured attachment may be again due to the richer complementation patterns exhibited by verbs in the lexicon. However, while in the case of verbs lexical information produces a significant improvement on restructured attachment precision, this contribution drops considerably for nouns. The main reason for this situation is that nouns tend to select semantically vacuous prepositions such as *of* much more often than verbs do. In our lexicon, out of 4157 frames headed by a noun, 4015 contain the preposition *di* as an argument introducer (96.6%). *Di* is in fact an extremely polysemous preposition, heading, among others, also possessive phrases and other kinds of modifiers. This trivially increases the number of cases of attachment ambiguity and eventually the possibility of getting false positives. Conversely, as shown by the number of confirmed attachments in Table 2, the role of lexical information in further specifying an attachment with no restructuring is almost uniform across nouns and verbs.

4.3.4 False negatives
The vast majority of undetected verb complements (80.6%) are modifiers of various kind. The remaining set of false negatives consists of
48 complements (7.7%), 30 indirect objects (4.8%) and 43 oblique arguments (6.9%). Most such complements are *by*-phrases in passive constructions which are not as such very difficult to detect but just happen to fall out of the current coverage of LAP. More interestingly, 2/3 of the remaining false negatives elude LAP because they are overtly realised far away from their verb head, often to its left. Most of these constructions involve argument dislocation and ellipsis. We can thus preliminarily conclude that argument dislocation and ellipsis accounts for about 14% of false negatives (7% over 50%). Finally, the number of false negatives due to attachment ambiguity is almost negligible in the case of verbal heads.

On the other hand, the impact of undetected modifiers of a verbal head on attachment recall is considerable. The most striking feature of this large subset is the comparative sparseness of modifiers introduced by *di* (of): 31 out of 504 (6.2%). At a closer scrutiny, the majority of these *di*-phrases are either phraseological adverbial modifiers (*di recente* ‘of late’, *del resto* ‘besides’ etc.) or quasi-arguments headed by participle forms. Notably, 227 undetected modifiers (45% of the total) are selected by semantically heavy and complex (possibly discontinuous) prepositions (*davanti a* ‘in front of’, *in mezzo a* ‘amid’, *verso* ‘towards’, *intorno a* ‘around’, *contro* ‘against’, *da ... a* ‘from ... to’ etc.). As to the remaining 241 undetected modifiers (48%), they are introduced by ‘light’ prepositions such as *a* ‘to’, *in* ‘in’ and *da* ‘from’. Although this 48% contains a number of difficult attachments, one can identify subsets of fairly reliable modifiers by focusing on the noun head introduced by the preposition, which usually gives a strong indication of the nature of the modifier, especially in the case of measure, temporal and locative expressions.

### 4.3.5 False positives

Table 2 shows a prominent asymmetry in the precision of confirmed and restructured attachments. Wrong restructured attachments are mainly due to a misleading match between the preposition introducing a PC and that introducing a slot in the lexical frame of its candidate head (~85%). This typically occurs with ‘light’ prepositions (e.g. *di, a*, etc.). Most notably, in a relevant subset of these mistakes, the verb or noun head belongs to an idiomatic multi-word expression. In the case of confirmed attachments, about one third of false positives (~5%) involve multi-word expressions, in particular compound terms such as *presidente del consiglio* ‘prime minister’, where the rightmost element of the compound is wrongly selected as the head of the immediately following PP. In both restructured and confirmed attachments, the remaining cases (on average ~4%) are due to complex syntactic structures (e.g. appositive constructions, complex coordination, ellipsis etc.) which are outside the coverage of the current grammar.

### Conclusion

Larger lexicons are not necessarily better for parsing. The issue of the interplay of lexicon and grammar, although fairly well understood at the level of linguistic theory, still remains to be fully investigated at the level of parsing. In this paper, we tried to scratch the surface of the problem through a careful analysis of the performance of an incremental dependency analyser of Italian, which can run in both a non-lexicalised and a lexicalised mode.

The contribution of lexical information to parse success is unevenly distributed over both part of speech categories and frame types. For reasons abundantly illustrated in section 4, the frames of noun heads are not quite as useful as those of verb heads, especially when available information is only syntactic. Moreover, while information on verb transitivity or clause embedding is crucial to filter out noisy attachments, information on the preposition introducing the oblique complement or the indirect object of a verb can be misleading, and should thus be used for parsing with greater care. The main reason is that failure to register in the lexicon all possible prepositions actually found in real texts may cause undesired over-filtering of genuine arguments (false negatives). In many cases, argument prepositions are actually selected by the lexical head of the subcategorised argument, rather than by its subcategorising verb. Similarly, while information about argument optionality vs obligatoriness is seldom confirmed in real language use, statistical preferences on the order of argument realisation can be very useful.
Most current lexicons say very little about temporal and circumstantial modifiers, but much more can be said about them that is useful to parsing. First, some prepositions only occur to introduce verb modifiers. These semantically heavy prepositions, often consisting of more than one lexical item, play a fundamental role in the organization of written texts, and certainly deserve a special place in a parsing-oriented lexicon. Availability of this type of lexical information could pave the way to the development of specialised “mini-parsers” of those satellite modifiers whose structural position in the sentence is subject to considerable variation. These mini-parsers could benefit from information about semantically-based classes of nouns, such as locations, measure terms, or temporal expressions, which should also contain indication of the preposition they are typically introduced by. Clearly, this move requires abandoning the prejudice that lexical information should only flow from the head to its dependents. Finally, availability of large repertoires of multi word units (both complex prepositions and compound terms) appears to have a large impact on improving parse precision.

There is no doubt that harvesting such a wide range of lexical information in the quantity needed for accurate parsing will require extensive recourse to bootstrapping methods of lexical knowledge acquisition from real texts.

References


Machine Translation as a testbed for multilingual analysis

Richard Campbell, Carmen Lozano, Jessie Pinkham and Martine Smets*

Microsoft Research
One Microsoft Way
Redmond, WA 98052 USA
{richcamp, clozano, jessiep, martines}@microsoft.com
*to whom all correspondence should be addressed

Abstract

We propose that machine translation (MT) is a useful application for evaluating and deriving the development of NL components, especially in a wide-coverage analysis system. Given the architecture of our MT system, which is a transfer system based on linguistic modules, correct analysis is expected to be a prerequisite for correct translation, suggesting a correlation between the two, given relatively mature transfer and generation components. We show through error analysis that there is indeed a strong correlation between the quality of the translated output and the subjectively determined goodness of the analysis. We use this correlation as a guide for development of a coordinated parallel analysis effort in 7 languages.

1 Introduction

The question of how to test natural language analysis systems has been central to all natural language work in the past two decades. It is a difficult question, for which researchers have found only partial answers. The most common answer is component testing, where the component is compared against a standard of goodness, usually the Penn Treebank for English (Marcus et al., 1993), allowing a numerical score of precision and recall (e.g. Collins, 1997).

Such methods have limitations, however, and need to be supplemented by additional methods. One limitation is the availability of annotated corpora, which do not exist for all languages. Secondly, comparison to an annotated corpus can only measure how well a system produces the kind of analysis for which the corpus is annotated, e.g. labeled bracketing of surface syntax. Evaluation of analysis of deeper, more semantically descriptive, levels requires additional annotated corpora, which may not exist. A more fundamental limitation of such methods is that they measure the goodness of a grammar without taking into account what the grammar is good for. This limitation is overcome, we claim, only by measuring the goodness of a grammar by its success in real-world applications.

We propose that machine translation (MT) is a good application to evaluate and drive the development of analysis components when the transfer component is based on linguistic modules. Multi-lingual applications such as MT allow evaluation of system components that overcomes the limitations mentioned above, and therefore serves as a useful complement to other evaluation techniques. Another significant advantage to using MT as a testbed for the analysis system is that it prioritizes analysis problems, highlighting those problems that have the greatest negative effect on translation output.

In this paper, we give an overview of NLWin, a multi-application natural language analysis and generation system under development at Microsoft Research (Jensen et al., 1993; Gamon et al., 1997; Heidorn 2000), incorporating analysis systems for 7 languages (Chinese, English, French, German, Japanese, Korean and Spanish). Our discussion focuses on a description of the three components of the analysis system (called sketch, portrait and logical form) with a particular emphasis on the logical form derived as the end-product, which serves as the medium for transfer in our MT system.

We also give an overview of the architecture of the MSR-MT system, and of the evaluation we use to measure correctness of the translations. We demonstrate the correlation between the scores
assigned to translation outputs and the correctness of the analysis, using as illustration two language-pairs at different stages of development: Spanish-English (SE) translation, as a testbed for the Spanish analysis system, and French-English (FE) translation, as a testbed for the French analysis system.

2 Overview of the analysis component of NLPWin

Analysis produces three representations for the input sentence: sketch, portrait and logical form. Sketch is the initial tree representation for the sentence, along with its associated attribute-value structure. An example of sketch is given in Figure 1, which shows the sketch tree for sentence (1).

(1) Ce format est pris en charge par Windows 2000

This format is taken in charge by Windows 2000.

Figure 1: Sketch analysis of (1)

Attachment sites for post-modifiers are not determined in sketch. In most cases, the information available as the syntactic tree is built is not sufficient to determine where e.g. prepositional phrases or relative clauses should be attached. Post-modifiers are thus systematically attached to the closest possible attachment site, and reattached, if necessary, by the reattachment module, a set of heuristic rules.

Reattachment rules apply to the sketch to produce the portrait; the portrait analysis of (1) is given in Figure 2, where the PP expressing the agent of the passive construction, originally attached to PP1 in sketch (see Figure 1) has been reattached at the sentence level.

Figure 2: Portrait analysis of (1)

The portrait is the input to the computation of the logical form (LF), a labeled directed unordered graph representing the deep syntactic relations among the content words of the sentence (i.e., basic predicate-argument structure), along with some semantic information, such as functional relations expressed by certain prepositions. At this level, the difference between active and passive constructions is normalized; control relations and long-distance dependencies, such as subjects of infinitives, arguments associated with gaps, etc., are resolved. The LF of (1) is shown in Figure 3. Note that the passive subject of the passive is rendered as the Dobj (deep object) in LF, and the par-phrase as the Dsubj (deep subject).

Figure 3: LF analysis of (1)

Modifications to any of the analysis components are tested using monolingual regression files containing thousands of analyzed sentences; differences caused by the modification are examined manually by the linguist responsible for the change (Suzuki, 2002). This process serves as an initial screening to ensure that modifications to the analysis have the desired effect.

3 MSR-MT

In this section we review the basics of the MSR-MT translation system and its evaluation. The reader is referred to Pinkham et al. (2001) and Richardson et al. (2001) for further details on the French and Spanish versions of the system. The overall architecture and basic component structure

---

1 The presentation of the analysis module is very simplified, but sufficient for our current discussion. More details can be found in the references.

2 LF as described here corresponds to the PAS representation of Campbell and Suzuki (2002).
are the same for both the FE and SE versions of the system.

3.1 Overview

MSR-MT uses the broad coverage analysis system described in Section 2, a large multi-purpose source-language dictionary, a learned bilingual dictionary, an application independent target-language generation component and a transfer component.

The transfer component consists of transfer patterns automatically acquired from sentence-aligned bilingual corpora (described below) using an alignment algorithm described in detail in Menezes and Richardson (2001). Training takes place on aligned sentences which have been analyzed by the source- and target-language analysis systems to yield logical forms. The logical form structures, when aligned, allow the extraction of lexical and structural translation correspondences which are stored for use at runtime in the transfer database. See Figure 4 for an overview of the training process. The transfer database is trained on 350,000 pairs of aligned sentences from computer manuals for SE, and 500,000 pairs of aligned Canadian parliamentary data (the Hansard corpus) for FE.

![Figure 4: MSR-MT training phase](image)

3.2 Evaluation of MSR-MT

Seven evaluators are asked to evaluate the same set of sentences. For each sentence, raters are presented with a reference sentence, the original English sentence from which the human French and Spanish translations were derived, and MSR-MT’s machine translation. In order to maintain consistency among raters who may have different levels of fluency in the source language, raters are not shown the original French or Spanish sentence (for similar methodologies, see Ringger et al., 2001; White et al., 1993).

All the raters enter scores reflecting the absolute quality of the translation as compared to the reference translation given. The overall score of a sentence is the average of the scores given by the seven raters. Scores range from 1 to 4, with 1 meaning unacceptable (not comprehensible), 2 meaning possibly acceptable (some information is transferred accurately), 3 meaning acceptable (not perfect, but accurate transfer of all important information), and 4 meaning ideal (grammatically correct and all the important information is transferred).

4 Examples from FE and SE

In this section we discuss specific examples to illustrate how results from MT evaluation help us to test and develop the analysis system.

4.1 FE translation: the Hansard corpus

The evaluation we are discussing in this section was performed in January 2002, at the beginning of our effort on the Hansard corpus. The evaluation was performed on a corpus of 250 sentences, of which 55.6% (139 sentences) were assigned a score of 2 or lower, 30.4% (76 sentences) were assigned a score greater than 2 but not greater than 3, and 14% (35 sentences) were assigned a score greater than 3.

Examination of French sentences receiving low-score translations led to the identification of some classes of analysis problems, such as the following:

- mis-identification of vocatives
- clefts not represented correctly
- mis-analysis of ce qui / ce que free relatives
- bad representation of complex inversion (pronoun-doubling of inverted subject)
- no treatment of reflexives
- fitted parses (i.e., not spanning the sentence)

Most of the problematic structures are characteristic of spoken language as opposed to more formal, written styles (vocatives, clefts, direct questions), and had not been encountered in our previous work, which had involved mostly translation of technical manuals. Other problems

---

3 Microsoft manuals are written in English and translated by hand into other languages. We use these translations as input to our system, and translate them back into English.
(free relatives, reflexives) are analysis issues that we had not yet addressed. Fitted parses are parses that do not span the whole sentence, but are pieced together by the parser from partial parses; fitted parses usually result in poor translations.

Examples of translations together with their score are given in Table I. The source sentences are the French sentences, the reference sentence is the human translation to which the translation is compared by the evaluators, and the translation is the output of MSR-MT. Each of the three categories considered above is illustrated by an example.

Sentence (2) (with a score of 1.5) is a direct question with complex inversion and the doubled subject typical of that construction. In the LF for (2), les ministres des finances is analyzed as a modifier, because the verb rucher already has a subject, the pronoun ils ‘they’. There are a couple of additional problems with this sentence: si is analyzed as the adverb meaning ‘so’ instead of as the conjunction meaning ‘if’, and a direct question is analyzed as a complement clause; the sketch and LF analyses of this sentence are given in the Appendix. The MSR-MT translation of this sentence has a very low score, reflecting the severity of the analysis problems.

The two other sentences, on the other hand, do not have analysis problems: the poor translation of (3) (score 2.16) is caused by bad alignment (droit translates as right instead of law), and the translation of (4) (score 3) is not completely fluent, but this is due to an English generation problem, rather than to a French analysis problem. This last sentence is the most correct with appropriate lexical items and has the highest score of the three.

Of the 139 sentences with score 2 or lower, 73% were due to analysis problems, and 24% to alignment problems. Most of the rest had bugs related to the learned dictionary. There were a few cases of very free translations, where the reference translation was very far from the French sentence, and our translation, based on the source sentence, was therefore penalized.

These figures show that, at this stage of development of our system, most of the problems in translation come from analysis. Translation can be improved by tackling analysis problems exhibited by the lowest scoring sentences, and, conversely, analysis issues can be discovered by looking at the sentences with the lowest translation score.

The next section gives examples of issues with the SE system, which is more mature than the FE system.

4.2 SE translation: Technical manuals

An evaluation of the Spanish-English MT system was also performed in January 2002, after work on the MT system had been progressing for approximately a year and a half. The SE system was developed and tested using a corpus of sentences from Microsoft technical manuals. A set of 600 unseen sentences was used for the evaluation.

Out of a total of 600 sentences, the number of sentences with a score from 3 to 4 was 251 (42%), the number of sentences with a score greater than 2 but less than 3 was 186 (31%), and the remaining 163 sentences, (27%) had a score of 2 or lower. Of these 163 sentences with the lowest scores, 50% (82 sentences) had analysis problems, and 17% of them (29 sentences) had fitted parses. A few of the fitted parses, 7 sentences out of 29, had faulty input, e.g. input that contained unusual characters or punctuation, typos, or sentence fragments.

Typical analysis problems that led to poor translation in the SE system include the following:
- incorrect analysis of arguments in relative clauses, especially those with a single argument (and a possible non-overt subject)
- failure to identify the referent of clitic le (i.e. usted ‘you’) in imperative sentences in LF
- mis-analysis of Spanish reflexive or se constructions in LF
- incorrect syntactic analysis of homographs
- incorrect analysis of coordination
- mis-identification of non-overt or controlled subjects
- fitted parses

Table II contains sample sentences from the SE evaluation. For each row, the second column displays the Spanish source sentence with the reference sentence in the next column, the translation produced by the MT system is in the fourth column, and the score for the translation assigned by the human evaluators in the last column.
Table I: Examples of FE translation

<table>
<thead>
<tr>
<th>#</th>
<th>Source</th>
<th>Reference</th>
<th>Translation</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Si tel n'était pas le cas, pourquoi les ministres des Finances des provinces se seraient-ils réunis hier pour essayer de s'entendre sur un programme commun à soumettre au ministre des Finances?</td>
<td>If that were not the case, why were the finance ministers of the provinces coalescing yesterday to try and come up with a joint program to bring to the finance minister?</td>
<td>Not was the case that they have the ministers met why yesterday Finances of the provinces trying to agree on a common program to bring Finances for the minister this so like?</td>
<td>1.5</td>
</tr>
<tr>
<td>3</td>
<td>Nous ne pouvons pas appuyer cette motion après que le Bloc québécois ait refusé de reconnaître la primauté du droit et de la démocratie pour tous.</td>
<td>We cannot support this motion after seeing the Bloc Quebeceois refuse to recognize the rule of law and the principle of democracy for all.</td>
<td>We cannot support this motion after the Bloc Quebecois has refused to recognize the rule of the right and democracy for all.</td>
<td>2.16</td>
</tr>
<tr>
<td>4</td>
<td>En tant que membre de l'opposition officielle, je continuerai d'exercer des pressions sur le gouvernement pour qu'il tienne ses promesses à cet égard.</td>
<td>As a member of the official opposition I will continue to pressure the government to fulfill its promises in this regard.</td>
<td>As member of the official opposition, I will continue to exercise pressures on the government for it to keep its promises in this regard.</td>
<td>3</td>
</tr>
</tbody>
</table>

Table II: Examples of SE translation

<table>
<thead>
<tr>
<th>#</th>
<th>Source</th>
<th>Reference</th>
<th>Translation</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Este procedimiento sólo es aplicable si está ejecutando una versión de idioma de Windows 2000 que no coincida con el idioma en el que desea escribir.</td>
<td>This procedure applies only if you are running a language version of Windows 2000 that doesn't match the language you want to type.</td>
<td>This procedure only applies if you are running a Windows 2000 language version that does not match the language that you want to type.</td>
<td>3.8</td>
</tr>
<tr>
<td>6</td>
<td>Repita este proceso hasta que haya eliminado todos los componentes de red desde las propiedades de Red, haga clic en Aceptar y, a continuación, haga clic en Sí cuando se le pregunte si desea reiniciar el equipo.</td>
<td>Repeat this process until you have deleted all of the network components from Network properties, click OK, and then click Yes when you are prompted to restart your computer.</td>
<td>Repeat this process until you have deleted all of the network components from the Network properties, you click OK, and you click Yes then when asking that to restart the computer is wanted for him.</td>
<td>2.0</td>
</tr>
<tr>
<td>7</td>
<td>En el siguiente ejemplo se muestra el nombre de la presentación que se está ejecutando en la ventana de presentación con diapositivas uno.</td>
<td>The following example displays the name of the presentation that's currently running in slide show window one.</td>
<td>In the following example, the display name that is being run in the slide show window is displayed I join.</td>
<td>1.4</td>
</tr>
</tbody>
</table>

In the evaluation process, human evaluators compared the MT translation to the reference sentence, in the manner described in Section 4.1.

Example (5), with a score of 3.8, illustrates the fact that human evaluators considered the translation 'a Windows 2000 language version' to be a slightly worse translation than 'a language version of Windows 2000' for una versión de idioma de Windows 2000; however the difference is so slight as to not be considered an analysis problem.

Example (6) illustrates the failure to identify usted 'you' (understood as the subject of the imperative) as the referent of the pronominal clitic le; as mentioned above, this is a common source of bad SE translations. The last example (7) is a sentence with a fitted parse due to misanalysis of a word as its homograph: uno is analyzed as the first person singular present form of the verb unir 'join' instead of as the noun uno 'one'; the LF of this sentence is given in the Appendix.

4.3 Discussion

The sentences discussed in this section are typical: The sentences for which MSR-MT produces better translations tend to be the ones with fewer analysis...
errors, while those which are misanalyzed tend to be mistranslated.

In this way, evaluation of MT output serves as one way to prioritize analysis problems; that is, to decide which among the many different analysis problems lead to the most serious problems. For example, the poor quality of the translation of (2) highlights the need for an improved analysis of complex inversion in the French grammar, which will need to be incorporated into the sketch and/or LF components. Similarly, the poor translation of (7) indicates the need to deal better with homographs in the Spanish morphological or sketch component.

More generally, the analysis of FE and SE translation problems has led to the lists of analysis problems given in Sections 4.1 and 4.2, respectively. Analysis problems identified in this way then become priorities for grammar/LF development.

5 Conclusion

We have outlined how the output of MT can be used as testbed for linguistic analysis in the source language, supplementing other methods. The main advantage of this approach, in our view, is that it helps to prioritize analysis problems, highlighting those which have the most direct bearing on the application(s), the correct functioning of which is the main goal of the system.

Acknowledgements

This paper represents the work of many people in the NLP group at MSR; we acknowledge their contributions.

References


Appendix

Figure 5: Sketch analysis of (2)

Figure 6: LF analysis of (2)

Figure 7: LF analysis of (7)