Lexical Resource Reconciliation in the Xerox Linguistic Environment

Ronald M. Kaplan
Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA, 94304, USA
kaplan@parc.xerox.com

Paula S. Newman
Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA, 94304, USA
pnewman@parc.xerox.com

Abstract

This paper motivates and describes those aspects of the Xerox Linguistic Environment (XLE) that facilitate the construction of broad-coverage Lexical Functional grammars by incorporating morphological and lexical material from external resources. Because that material can be incorrect, incomplete, or otherwise incompatible with the grammar, mechanisms are provided to correct and augment the external material to suit the needs of the grammar developer. This can be accomplished without direct modification of the incorporated material, which is often infeasible or undesirable. Externally-developed finite-state morphological analyzers are reconciled with grammar requirements by run-time simulation of finite-state calculus operations for combining transducers. Lexical entries derived by automatic extraction from on-line dictionaries or via corpus-analysis tools are incorporated and reconciled by extending the LFG lexicon formalism to allow fine-tuned integration of information from difference sources.

1 Introduction

The LISP-based LFG Grammar Writers Workbench (GWB) (Kaplan and Maxwell, 1996) has long served both as a testbed for the development of parsing algorithms (Maxwell and Kaplan, 1991; Maxwell and Kaplan, 1993) and as a self-contained environment for work in syntactic theory. The C/Unix-based Xerox Linguistic Environment (XLE) further develops the GWB parsing algorithms, extends them to generation, and adapts the environment to a different set of requirements.

This paper motivates and describes the morphological and lexical adaptations of XLE. They evolved concurrently with PARGRAM, a multi-site XLE-based broad-coverage grammar writing effort aimed at creating parallel grammars for English, French, and German (see Butt et. al., forthcoming). The XLE adaptations help to reconcile separately constructed linguistic resources with the needs of the core grammars.

The paper is divided into three major sections. The next section sets the stage by providing a short overview of the overall environmental features of the original LFG GWB and its provisions for morphological and lexicon processing. The two following sections describe the XLE extensions in those areas.

2 The GWB Data Base

GWB provides a computational environment tailored especially for defining and testing grammars in the LFG formalism. Comprehensive editing facilities internal to the environment are used to construct and modify a data base of grammar elements of various types: morphological rules, lexical entries, syntactic rules, and "templates" allowing named abbreviations for combinations of constraints. (See Kaplan and Maxwell, 1996; Kaplan and Bresnan, 1982; and Kaplan, 1995 for descriptions of the LFG formalism.) Separate "configuration" specifications indicate how to select and assemble collections of these elements to make up a complete grammar, and alternative configurations make it easy to experiment with different linguistic analyses.

This paper focuses on the lexical mapping process, that is, the overall process of translating between the characters in an input string and the initial edges of the parse-chart. We divide this process into the typical stages of tokenization, morphological analysis, and LFG lexicon lookup. In GWB tokenizing is accomplished with a finite-state transducer compiled from a few simple rules according to the methods described by (Kaplan and Kay, 1994). It tokenizes the input string by inserting explicit token boundary symbols at appropriate character positions. This process can produce multiple outputs because of uncertainties in the interpretation of punctuation such as spaces and periods. For example, "I like Jan." results in two alternatives ("I@like@Jan@.@" and "I@like@Jan.@.@") because the period in "Jan." could optionally mark an abbreviation as well as a sentence end.

Morphological analysis is also implemented as a finite-state transducer again compiled from a set of rules. These rules are limited to describing only simple suffixing and inflectional morphology. The morphological transducer is arranged to apply to individual tokens produced by the tokenizer, not to strings of tokens. The result of applying the morphological rules to a token is a stem and one or more inflectional tags, each of which is the heading for an entry in the LFG lexicon. Morphological ambiguity can lead to alternative analyses for a single token, so this stage can add further possibilities to the alternatives coming from the tokenizer. The token "cooks" can be analyzed as "cook +NPL" or "cook +V3SG", for instance.

In the final phase of GWB lexical mapping, these stem-tag sequences are looked up in the LFG lexicon to discover the syntactic category (N, V, etc.) and constraints (e.g. $(\uparrow \text{NUM})=\text{PL}$) to be placed on a single edge in the initial parse chart. The category of that edge is determined by the particular combination of stems and tags, and the corresponding edge constraints are formed by conjoining the constraints found in the stem/tag lexical entries. Because of the ambiguities in tokenization, morphological analysis and also lexical lookup, the initial chart is a network rather than a simple sequence.

The grammar writer enters morphological rules, syntactic rules, and lexical entries into a database. These are grouped by type into named collections. The collections may overlap in content in that different syntactic rule collections may contain alternative expansions for a particular category and different lexical collections may contain alternative definitions for a particular headword. A configuration contains an ordered list of collection names to indicate which alternatives to include in the active grammar.

This arrangement provides considerable support for experimentation. The grammar writer can investigate alternative hypotheses by switching among configurations with different inclusion lists. Also, the inclusion list order is significant, with collections mentioned later in the list having higher precedence than ones mentioned earlier. If a rule for the same syntactic category appears in more than one included rule collection, or an entry for the same headword appears in more than one included lexical collection, the instance from the collection of highest precedence is the one included in the grammar. Thus the grammar writer can tentatively replace a few rules or lexical entries by placing some very small collections containing the replacements later in the configuration list.

We constructed XLE around the same database-plus-configuration model but adapted it to operate in the C/Unix world and to meet an additional set of user requirements. GWB is implemented in a residential Lisp system where rules and definitions on text files are "loaded" into a memory-based database and then selected and manipulated. In C/Unix we treat the files themselves as the analog of the GWB database. Thus, the XLE user executes either a "create-parser" or "create-generator" command to select a file containing one or more configurations and to select one of those configurations to specify the current grammar. The selected configuration, in turn, names a list of files comprising the data base, and identifies the elements in those files to be used in the grammar.

This arrangement still supports alternative versions of lexical entries and rules, but the purpose is not just to permit easy and rapid exploration of these alternatives. The XLE database facilities also enable linguistic specifications from different sources and with different degrees of quality to be combined together in an orderly and coherent way. For XLE the goal is to produce efficient, robust, and broad coverage processors for parsing and generation, and this requires that we make use of large-scale, independently developed morphological analyzers and lexicons. Such comprehensive, well-engineered components exist for many languages, and can relieve the grammar developer of much of the effort and expense of accounting again for those aspects of language processing.

3 XLE Morphological Processing

One obvious obstacle to incorporating externally developed morphological analyzers, even ones based on finite-state technology, is that they may be supplied in a variety of data-structure formats. We overcame this obstacle by implementing special software interpreters for the different transducer formats. But, as we learned in an evolutionary process, a more fundamental problem in integrating these components with syntactic processing is to reconcile the analyses they produce with the needs of the grammar.

Externally-available morphological components are, at present, primarily aimed at relatively undemanding commercial applications such as information retrieval. As such, they may have mistakes or gaps that have gone unnoticed because they have no effect on the target applications. For example, function words are generally ignored in IR, so mistakes in those words might go undetected; but those words are crucial in syntactic processing. And even correct analyses generally deviate in some respects from the characterizations desired by the grammar writer.

These mistakes, gaps, and mismatches are often not easy to correct at the source. It may not be practical to refer problems back to the supplier as they are uncovered, because of inherent time lags, economic feasibility, or other factors. But the grammar writer may not have the tools, permissions, or skills, to modify the source specifications directly.

3.1 Basic Approach

It is often the case that repairs to an external transducer whose behavior is unsatisfactory can be described in terms of operations in the calculus of regular relations (Kaplan and Kay, 1994). Transducers that encode the necessary modifications can be combined with the given one by combinations of composition, union, concatenation, and other operators that preserve regularity.

For example, suppose an externally created finite-state morphological analyzer for English maps surface forms such as "email" into stem+inflectional-morpheme sequences such as "email +Nmass", but that certain other relatively recent usages (e.g. "email" as a verb) are not included. A more adequate transducer can be specified as **External** ∪ **Addition**, the union of the external transducer with a locally-defined modification transducer. XLE provides a facility for constructing simple **Addition** transducers from lists of the desired input/output pairs:

```
email : email  +Nsg
emails: email  +Npl
email : email  +Vpres
```

The relational composition operator is also quite useful, since it enables the output of the external transducer to be transformed into more suitable arrangements. For example, suppose the given transducer maps "buy" into "buy +VBase", where +VBase subsumes both the present-tense and infinitival interpretations. This is not a helpful result if these two interpretations are treated by separate entries in the LFG lexicon, but this problem can be repaired by the composition **External** ∘ **TagFst**, where **TagFst** is a transducer specified as

```
+Vbase : +Verb +Inf
+Vbase : +Verb +Not3Sg
```

Finally, in cases where the output of the external analyzer is simply wrong, corrections can be developed again in a locally specified way and combined with a "priority union" operator that blocks the incorrect analyses. The priority union of regular relations A and B is defined as $A \cup_p B = A \cup [Id(\overline{Dom(A)}) \circ B]$. The relation $A \cup_p B$ contains all the pairs of strings in A together with all the pairs in B except those whose domain string is also in the domain of A. With this definition the expression **Correction** $\cup_p$ **External** describes a transducer that implements the desired overrides.

In principle, these and other modifications to an externally supplied transducer can be built by using a full-scale regular expression compiler, such as the Xerox calculus described in (Karttunen et. al., 1997). The calculus can evaluate these corrective expressions in an off-line computation to create a single effective transducer for use in syntactic processing. However, in practice it is not always possible to perform these calculations. The external transducer might be supplied in a form (e.g. highly compressed) not susceptible to combination. Or (depending on the modifications) the resultant transducer may grow too large to be useful or take too long to create.

Therefore, our approach is to allow transducer combining operations to be specified in a "morphological configuration" (morph-config) referenced from the XLE grammar configuration. The effects of the finite-state operations specified in the morph-config are simulated at run time, so that we obtain the desired behavior without performing the off-line calculations. In the case of union, the analysis is the combination of results obtained by passing the input string through both transducers. For composition, the output of the first transducer is passed as input to the second. For priority union, the first transducer is used in an attempt to produce an output for a given input; the result from the second transducer is used only if there is no output from the first.

Specifying the transducer combining operations in the XLE morph-config rather than relying on a script for an offline computation has another important advantage: it gives the grammar writer a single place to look to understand exactly what behavior to expect from the morphological analyzer. It removes a serious audit-trail problem that would otherwise arise in large-scale grammar development.

3.2 The Role of Tokenization

The role of tokenization in a production-level parser is more than a matter of isolating substrings bounded by blanks or punctuation or, in generation, reinserting blanks in the appropriate places. The tokenizer must also undo various conventions of typographical expression to put the input into the canonical form expected by the morphological analyzer. Thus it should include the following capabilities in addition to substring isolation:

- Normalizing: removing extraneous white-space from the string.
- Editing: making prespecified types of modifications to the string, e.g., removing annotations or SGML tags in an annotated corpus.
- Accidental capitalization handling: analyzing capitalized words, at least in certain contexts such as sentence-initial, as, alternatively, their lower-case equivalents,
- Contraction handling: separating contracted forms such as:

  ```
  can't   --> can  _not
  l'homme --> le  homme
  du      --> de  le
  John's  --> John _poss
  ```

  These forms represent combinations of elements that would typically appear under different syntactic nodes. That is, the elements of "du" fall under both PREP and NP, and those of "John's" within an NP and a POSS attached to the NP. Separating these in the tokenizing process avoids what would otherwise be unnecessary grammatical complications.
- Compound word isolation: determining which space-separated words should be considered as single tokens for purposes of morphological analysis and parsing, at least as an alternative, This function can range from simply shielding blank-containing words from separation, e.g., "a priori", "United States" to recognizing and similarly shielding highly structured sequences such as date (Karttunen et. al., 1997) and time expressions.

Because a tokenizing transducer is not only language-specific but often specific to a particular application and perhaps also to grammar-writer preference, it may be necessary to build or modify the tokenizing transducer as part of the grammar-writing effort.

Logically, the relationship between the effective finite-state tokenizer, which carries out all of the above functions, and the morphological analyzer can be defined as:

$$\texttt{tokenizer} \circ [\texttt{morph\_analyzer} \ @]^*$$

That is, the tokenizer, which inserts token boundaries (@) between tokens, is composed with a cyclic transducer that expects a @ after each word. However, the feasibility considerations applying to the offline compilation of morphological transducers apply as well to the pre-compilation of their composition with the effective tokenizer. So the XLE morph-config file provides for the specification of a separate tokenizer, and the cycle and composition in the expression above are simulated at run-time.

Further, it is advantageous to allow the effective tokenizer to also be specified in a modular way, as regular combinations of separate transducers interpreted by run-time simulation. The need for modularity in this case is based on both potential transducer size, as combining some of the tokenizer functions can lead to unacceptably large transducers, and on the desirability of using the same morph-config specifications for both parsing and generation. Most of the tokenizer functions are appropriate to both of these processes but some, such as white-space normalization, are not. Our modular specification of the effective tokenizer allows the grammar writer to mark those components that are used in only one of the processes.

3.3 XLE Morph-Config

The current XLE morph-config provides for these capabilities in a syntactically-sugared form so that a grammar writer not interested in the finite-state-calculus can nevertheless combine externally supplied transducers with locally-developed ones, and can easily comprehend the expected behavior of the effective transducer.

The general structure of the morph-config is illustrated by the following:

```
# English Morph Config
TOKENIZE:
P!blank-normalizer  tokenizer

ANALYZE USEFIRST:
morph-override
main-morph
P!accent-fix main-morph

ANALYZE USEALL:
morph-add1
morph-add2
```

The transducers named in the TOKENIZE section are applied in sequence to each input string, with the result being the same as that of applying their composition. However those prefixed by P! or G! are applied only in parsing or generation respectively.

The individual output tokens are submitted to the transducers specified in the ANALYZE sections. There may be two such sections. The ANALYZE USEFIRST section is borrowed directly from the Rank Xerox transducer "lookup" facility (Karttunen et. al., 1997). Each line in this section specifies an effective transducer equivalent to the result of composing all the transducers mentioned on that line. The effective transducer resulting from the entire section is equivalent to the combination, by priority union, of the effective transducers specified by the individual lines.

So, in the above example of an ANALYZE USEFIRST section, the large **main-morph**, assumed to be externally supplied, handles most inputs, but its incorrect analyses are blocked by the preemptive results of the **morph-override** transducer. And if neither the override transducer nor the main transducer obtains an analysis during parsing, the transduction specified as the composition of an accent fixer (e.g., for correcting erroneous accents) with the main transducer is applied. The accent fixer is not used in generation, so that only properly accented strings are produced.

In the ANALYZE USEALL section, transducers on a single line are again interpreted as composed, but the resultant transducers from separate lines are understood as combined by simple union with each other and with the (single) effective ANALYZE USEFIRST transducer. In this way the grammar writer can specify one or more small transducers delivering alternative analyses where the results of the external analyzer are correct, but additional analyses are needed.

The meaning of all the specifications in a morph-config file can be represented as an expression in the finite-state calculus that describes a single effective transducer. This transducer maps from the characters of an input sentence to a finite-state machine that represents all possible sequences of morpheme/tag combinations. The network structure of that machine is isomorphic to the initial parse-chart. For our simple morph-config example, the effective parsing transducer is

$$\begin{array}{c}\texttt{blank-normalizer}\\ \circ \\ \texttt{tokenizer} \\ \circ \\ \left[\left[\begin{array}{c}\left[\begin{array}{ll} & \texttt{morph-override} \\ \cup_p & \texttt{main-morph} \\ \cup_p & \texttt{accent-fix} \circ \texttt{main-morph}\end{array}\right] \\ \cup \\ \texttt{morph-add1} \\ \cup \\ \texttt{morph-add2}\end{array}\right] @ \right]^*\end{array}$$

4 XLE Lexicon Structures

The lexical extensions XLE makes to the GWB database setup are also aimed at obtaining broad coverage via the use of independently developed resources. External lexical resources may be derived from machine readable dictionaries or large corpora by means of corpus analysis tools that can automatically produce LFG lexical entries (for example, the tools described by Eckle and Heid, 1996). However, the results of using such repositories and tools are relatively error-prone and cannot be used without correction. In this section we describe the conventions for combining lexical information from different sources.

As in GWB, an entry in the XLE lexicon contains one or more subentries associated with different parts of speech, and separate entries for stems and affixes, e.g.,

```
cook  N  BASE (^ PRED)= 'cook'
              (^ NTYPE) =c COUNT;
      V  BASE (^ PRED)='cook<(^ SUBJ)
                              (^ OBJ)>'.

 +Npl  N  SFX (^ NTYPE) = COUNT
              (^ NUMBER) = PL.
```

These entries specify the syntactic categories for the different senses of the morphemes together with the LFG constraints that are appropriate to each sense. The syntactic category for a sense, the one that matches the preterminal nodes of the phrase-structure grammar, is created by packing together two components that are given in the entry, the major category indicator (N and V above) and a category-modifier (BASE and SFX). The noun and verb senses of "cook" are thus categorized as N-BASE and V-BASE, and the category of the plural morpheme is N-SFX. These categories are distinct from the N and V that are normally referenced by higher-level syntactic rules, and this distinction permits the grammar to contain a set of "sublexical" rules that determine what morphotactic combinations are allowed and how the constraints for an allowable combination are composed from the constraints of the individual morphemes. The sublexical rule for combining English nouns with their inflectional suffixes is

```
N --> N-BASE  N-SFX
```

Here the constraints for the N are simply the conjunction of the constraints for the base and suffix, and this rule represents the interpretation that is built in to the GWB system. By distinguishing the morphological categories from the syntactic ones and providing the full power of LFG rules to describe morphological compositions, the XLE system allows for much richer and more complicated patterns of word formation.

Also as in GWB, the lexicon for a grammar can be specified in the configuration by an ordered list of identifiers for lexical entry collections such as:

```
LEXENTRIES (CORPUS ENGLISH)
           (CORRECTIONS ENGLISH).
```

To find the effective definition for a given headword, the identified lexicons are scanned in order and the *last* entry for the headword is used. Thus the order of lexicon identifiers in an XLE configuration allows hand-created accurate entries to be identified later in the list and override all earlier (errorful) entries.

This is a practical approach to correction; the alternative approach of manually correcting erroneous entries *in situ* requires the edits to be redone whenever new versions of the external lexicon are created. But more can be done, as discussed in the next section.

4.1 Lexicon Edit Entries

We found that an erroneous entry in an externally provided lexicon often contains many subentries that are completely acceptable and should not be thrown away. However, the complete overrides afforded by the configuration priority order do not allow these desirable subentries to be preserved.

For this reason, we defined a new kind of lexical entry, called an "edit entry", that allows more finely tuned integration of multiple entries for a headword. The category-specific subentries of an edit entry are prefixed by operators specifying their effects on the subentries collected from earlier definitions in the priority sequence. For example, the external lexicon might supply the following preposition and directional-adverb subentries for the word "down":

```
down P   BASE @PREP;
     ADV BASE @DIRADV.
```

where @PREP and @DIRADV invoke the LFG templates that carry the constraints common to all prepositions and directional adverbs. The following edit entry specified later in the lexicon sequence can be used to add the less likely transitive verb reading (e.g. "He downed the beer"):

```
down +V  BASE @TRANS;
     ETC.
```

The + operator indicates that this V subentry is to be disjoined with any previous verb subentries (none in this case). The ETC flag specifies that previous subentries with categories not mentioned explicitly in the edit entry (P and ADV in this case) are to be retained.

We allow three other operators on a category C besides +. !C indicates that all previous category-C subentries for the headword should be replaced by the new one. -C prefixes otherwise empty subentries and specifies that all previous category-C subentries for the headword should be deleted. For example, if the above verbal meaning of "down" actually appeared in the external lexicon and was not desired, it could be deleted by an edit entry:

```
down -V;
     ETC.
```

Finally, =C indicates that all previous category-C subentries should be retained. This is useful when the flag ONLY appears instead of ETC. ONLY sets the default that the subentries for all categories not mentioned in the current entry are to be discarded; =C can be used to override that default just for the category C..

Combinations of these operators and flags allow quite precise control over the assembly of a final effective entry from information coming from different sources. To give one final example, if the constraints provided for the adverbial reading were insufficiently refined, the correcting entry might be:

```
down +V    BASE @TRANS;
     !ADV  BASE @DIRADV
                (^ ADV-TYPE)=VPADV-FINAL;
      ETC.
```

This adds the verbal entry as before, but replaces the previous adverb entry with a new set of constraints.

Edit entries have been especially valuable in the German branch of the PARGRAM effort, which uses lexicons automatically derived from many sources. Subentries in the sequence of automatic lexicons are generally prefixed by + so that each lexicon can make its own contributions. Then manually-coded lexicons are specified with highest precedence to make the necessary final adjustments.

4.2 Default definitions and unknown words

Morphological analysis coupled with sublexical rules determines which categories a word belongs to and what suffixes it can take. Subentries in the LFG lexicons specify additional syntactic properties which for many words are not predictable from their morphological decompositions. There are large sets of words in each category, however, that have exactly the same syntactic properties; most common nouns, for example, do not take arguments. XLE provides for default specifications that permit many redundant individual entries to be removed from all lexicons.

The lexical entry for the special headword **-Lunknown** contains subentries giving the default syntactic properties for each category. The entry

```
-Lunknown  N BASE @CN;
           V BASE @TRANS.
```

indicates that the default properties of morphological nouns are provided by the common-noun template **@CN** and that verbs are transitive by default. **-Lunknown** is taken as the definition for each lower-case word at the beginning of the scan through the ordered list of lexicons, and these subentries will prevail unless they are overridden by more specific information about the word. Another special headword **-LUnknown** gives the default definition for upper-case words (e.g. for proper nouns in English).

To illustrate, suppose that the morphological analyzer produces the following analysis

```
doors:    door -Npl
```

but that there is no explicit lexical entry for "door". Edges for both **N-SC** and **V-SC**, with constraints drawn from the -Lunknown entry would be added to the initial chart, along with edges for the tags. But only the sublexical rule

```
N --> N-BASE N-SFX
```

would succeed, because the morphological analyzer supplied the **-Npl** tag and no verbal inflection. Thus there is no need to have a separate entry for "door" in the lexicon. This feature contributes to the robustness of XLE processing since it will produce the most common definition for words known to the morphology but not known to the lexicon.

5 Summary

While GWB provides a self-contained environment for work in syntactic theory exploration, XLE is intended for production grammar development for parsing and generation. It must therefore provide for the careful integration of externally-created morphological and lexical resources. These resources cannot be used directly because they typically contain errors or are otherwise inconsistent with the needs of grammar developers.

XLE contains several mechanisms for reconciling external resources with grammar-specific lexical and syntactic requirements. XLE allows the grammar writer to assemble effective tokenizers and morphological analyzers by combining externally-provided components with locally-developed ones. These combinations are expressed with the finite-state calculus, many of whose operations are simulated at run-time. XLE also allows the grammar writer to obtain effective LFG lexical definitions by means of edit entries that combine information from various external sources with manually coded and default definitions. These extensions to the original GWB specifications have proven extremely helpful in our large-scale grammar development efforts.

6 Acknowledgements

We would like to thank the participants of the Pargram Parallel Grammar project for raising the issues motivating the work described in this paper, in particular Miriam Butt and Christian Rohrer for identifying the lexicon-related problems, and Tracy Holloway King and María-Eugenia Niño for bringing morphological problems to our attention. We also thank John Maxwell for his contribution towards formulating one of the approaches described, and Max Copperman for his help in implementing the facilities. And we thank Max Copperman, Mary Dalrymple, and John Maxwell for their editorial assistance.

References

Miriam Butt, Tracy Holloway King, María-Eugenia Niño, and Frédérique Segond. Forthcoming. *A grammar writer's handbook*, CSLI, Stanford, CA.

Judith Eckle and Ulrich Heid. 1996. Extracting raw material for a German subcategorization lexicon from newspaper text. In *International Conference on Computational Lexicography (COMPLEX-96)*, Budapest, September.

Ronald M. Kaplan. 1995. The formal architecture of lexical-functional grammar. In M. Dalrymple, R. M. Kaplan, J. T. Maxwell, A. Zaenen editors, *Formal Isssues in Lexical-Functional Grammar*, CSLI Publications, Stanford, CA.

Ronald M. Kaplan and Joan Bresnan. 1982. Lexical-functional grammar: a formal system for grammar representation. In J. Bresnan, editor, *The Mental Represenation of Grammatical Relations*, MIT Press, Cambridge, MA.

Ronald M. Kaplan and John T. Maxwell 1996 LFG grammar writers workbench. Available at ftp://ftp.parc.xerox.com/pub/lfg/

1994. Ronald M. Kaplan and Martin Kay. Regular models of phonological rule systems. *Computational Linguistics*, 20:3, pages 331-378.

L. Karttunen, J-P. Chanod, G. Grefenstette, A. Schiller 1997. Regular expressions for language engineering. In *Natural Language Engineering* Cambrige University Press.

John T. Maxwell, and Ronald M. Kaplan. 1991. A method for disjunctive constraint satsifaction. In M. Tomita, editor, *Current Issues in Parsing Technology.* Kluwer Academic Publications.

John T. Maxwell, and Ronald M. Kaplan. 1993. The interface between phrasal and functional constraints. *Computational Linguistics*, 19:4, pages 571-590.