Syntactic Theory and Grammar Design for Machine Translation

Lori Levin
Center for Machine Translation
Carnegie Mellon University

1. Introduction

Recently, the field of machine translation has increasingly reflected the influence of linguistic theory. Some systems have been designed around principles or structures of specific theories (Dorr, 1989; Kaplan, Netter, Wedekind, & Zaenen, 1989; Sharp, 1986), whereas others, adhering less strongly to particular theories, nevertheless reflect great care in considering and adapting the work of linguists. Most notable in this respect is the EUROTRA project as described, for example, in Arnold & des Tombe (1987) and Durand et al. (to appear). There seems to be general agreement that linguistic theory can contribute to machine translation (Raskin, 1987a, 1987b; Wehrli, 1987), but at the same time there is skepticism about its potential to break the barriers to robust, high quality translation (Nagao et al., 1988). As a result, many questions remain about the place of linguistic theory in machine translation.

This paper focuses on syntactic theory in knowledge-based machine translation and argues that a productive interweaving of theory with practical concerns can be achieved. Much of the discussion is based on our experience with knowledge-based systems—primarily KBMT-89 (Goodman & Nirenburg, 1989)—at the Center for Machine Translation. The two most salient contributions of syntactic theory to these systems are cross-linguistically valid levels of representation and strategies for analyzing a number of basic constructions. The theory we follow most closely is Lexical Functional Grammar (LFG) (Kaplan & Bresnan, 1982). However, there are many constructions for which syntactic theory provides no guidance and there are many instances in which we follow a general strategy laid out by syntactic theory, but for reasons of efficiency or the structure of the system our rules and structures are not identical to those of theoretical syntax. Therefore, although we follow LFG in many ways, very little about the theory is actually enforced. In this way, we allow enough flexibility to address issues having to do with efficiency and the demands of other components of the system, while still taking advantage of the modularity and cross-linguistic generality of syntactic theory.

Section 4 of this paper argues for the importance of syntactic theory in the design of machine translation systems and describes the contribution of syntactic theory to the KBMT-89 project. However, before engaging in arguments about the importance of syntactic theory, it will be useful to establish some background on the purpose of syntax in a knowledge based system and what exactly is involved in building grammars. These issues are addressed in Sections 2 and 3.

2. The Role of Syntax in Knowledge-Based Systems

The goal of an interlingual knowledge-based system is to represent the content of an utterance in a language-independent way and to produce a translation based on content rather than form. This emphasis on content might lead us to question the role of syntax in knowledge-based systems. What is the importance of analyzing forms in a system that is concerned primarily with content?

Even in knowledge-based systems, syntactic processing is necessary because the content of an utterance is encoded in its syntactic constructions and these constructions must be analyzed in order to extract their content. To give a few examples: predicate-argument relationships are encoded by case marking, word order and agreement; aspect is encoded by auxiliary verbs, other aspectual verbs, adverbs and other modifiers; and temporal relationships between clauses are encoded by temporal sequencing of clauses, connective words and adverbs. Predicate-argument relations, aspect and temporal sequencing are not syntactic notions, but they can only be extracted from a text by paying attention to syntactic relationships among constituents.

The major role of a syntactic component in a machine translation system is, therefore, to handle the extraction and encoding of information in syntactic constituents and to make this information accessible to other components of the system. The next question that must be addressed is what it takes to do this job well. I will show that building a good syntactic component depends in part on a theoretically sound framework for writing grammars and in part on bending to the demands of other system components and practical concerns about efficiency and robustness.

3. Grammar Design

Syntax strikes many people as being easier than semantics and pragmatics in that, although we have not unraveled all the mysteries of human syntactic competence, there are techniques, categories and structures adequate for most applications. There is a sense that we know how to do syntax. But even if it is true that there are adequate parsers, generators and grammar formalisms for syntactic processing, standard off-the-shelf grammars do not exist. The reason for a lack of standard grammars is that grammars must be customized to suit the needs of individual systems.

A common misguided assumption among researchers in natural language processing is that, because we have a good set of grammar formalisms and parsers to choose from, there is simply nothing left to do with respect to syntax, and certainly nothing interesting to talk about that will affect the speed or quality of translation. An analogy between grammar writing and programming will help reveal the flaw in this reasoning. The attitude that we do not have to worry about syntax because we have good parsers and grammar formalisms is like saying that we don't have to worry about programming because we have good programming languages and compilers. In syntactic processing, a grammar formalism is analogous to a programming language and a parser is analogous to a compiler. An additional component, which seems to have been widely neglected, is the grammar, a large set of rules which is like a complex program. Grammars, like programs, differ greatly in efficiency and quality of design and there are issues to be addressed in grammar writing that can significantly affect the performance and maintainability of a system.

I would like to emphasize that this section is about the design of grammars, not about grammar formalisms or algorithms for parsing and generation. The concern here is how to write a set of grammar rules that are within the constraints imposed by the system architecture (including the algorithms and grammar formalism) and result in optimal performance. I will summarize some of the problems that arise in grammar design, giving examples from the KBMT-89 project.

The issues that influence the design of the grammar fall into several classes. First, there is a set of partially unsolved problems—including ambiguity, ill-formed input and robustness—whose solutions call for different strategies in different systems. Specific questions that arise are how much overgeneration is acceptable for analysis, which rules can be omitted for a sublanguage, which special rules must be added for domain-specific constructions and how much ambiguity the rules can allow.

In KBMT-89, the English grammar was tailored to the domain of personal computer manuals. It handled fairly complex sentences such as *When the expansion unit is installed, the fixed disk drive and fixed disk adapter from the system unit are installed in the expansion unit,* and it handled many domain specific sentence types having to do with numbered lists, instructions, names of keyboard keys and error messages and titles of chapters. However, the KBMT-89 grammar does not handle many syntactic constructions that are common in other types of text.

Our strategy was to keep the grammar to a manageable size by restricting it to what was necessary for our corpus. One strong motivation for this strategy is that many parsers slow down when grammars become large because they have to attempt to apply more rules in order to find the appropriate ones for each sentence that they parse. Another problem with comprehensive grammars is that, if semantic constraints are not strong enough, the parser can produce extra parses that are structurally well-formed, but semantically corrupt or produce parses for ungrammatical sentences, in effect losing the ability to distinguish between grammatical and ungrammatical sentences. It may, therefore, be beneficial to restrict the coverage of the grammar, but the decision of just how much to restrict it will have to be made on an individual basis by each system designer.

There are also reasons to allow the parsing grammar to accept sentences that are not grammatical. Adding many constraints to a grammar can make it unreadable, slow and hard to maintain and can be unreasonable in limiting the user to enter completely grammatical text. For these reasons, it might be desirable to eliminate some constraints and allow the grammar to accept some ill-formed input. However, in addition to ruling out ungrammatical sentences, constraints in the grammar also eliminate some bad paths in the parsing of grammatical sentences. Eliminating the constraints could slow down the parsing of grammatical sentences by allowing the parser to follow bad paths for too long. It can also result in extra parses for grammatical sentences. Of course, many approaches to ill-formed input are possible and these have impact on the design of the grammar as well as the parsing algorithm.

With respect to ambiguity, our approach has been to allow the grammar to produce multiple parses only when they correspond to legitimate alternate readings. However, in some cases, very clean, well-organized grammars produce extra parses that do not correspond to extra readings of a sentence. Reducing these spurious ambiguities often requires complicating the grammar or including "hacks" that block extra parses. Since extra parses placed more of a burden on our system than complications in the grammar rules, we chose to complicate the grammar in order to

reduce extra parses. Other approaches might be to eliminate multiple parses for even legitimate ambiguities or, on the other extreme, to favor simplicity of the grammar over the elimination of spurious parses.

A second set of issues in grammar design has to do with the integration of syntax with other system components. These issues include (1) designing the input to, and the output from syntax, so that it is compatible with other system components, including morphological, lexical, semantic and pragmatic processors, (2) determining which linguistic phenomena will be handled by syntax and which by some other component of the system and (3) evaluating the desirability of bidirectional grammars for analysis and generation.

In KBMT, the most fundamental decision about syntactic structures was the decision to use LFG-like feature structures with grammatical functions in the output of parsing and the input to generation. In addition to the major grammatical functions, our choice of syntactic features for representing determiners, modality, relationships between clauses and tense were determined by the design of the interlingua text that the syntactic structure must ultimately map onto. Other constraints were imposed by the KBMT programs. For example, a constraint that the syntactic head of a phrase correspond to its semantic head in the interlingua text was imposed by the program that mapped functional structures onto instantiated frames.

In KBMT-89, decisions about what to handle in each component are were often guided by the principles of LFG. For example, the similarity in meaning between active and passive sentences is not captured at the level of phrase structure or grammatical functions, but in the mapping from grammatical functions to semantic roles. (See Section 4 for more detail) In contrast, other decisions about the division of labor between components were based on efficiency, even though in many cases the most efficient approach to a syntactic phenomenon was not the approach that syntactic theory called for. For example, traditional analyses of noun phrases based on X-bar theory distinguish between arguments and adjuncts of the head noun, attaching the arguments at a lower projection closer to the head. However, in order to avoid producing extra phrase structure trees when it could not be determined whether something is an argument or an adjunct, we decided not to distinguish between arguments and adjuncts of nouns at the level of phrase structure.

With respect to bidirectionality of grammars for analysis and generation, the English and Japanese teams in KBMT-89 chose different approaches. The Japanese grammar was designed to be bidirectional so that only one grammar needed to be written and maintained. The English grammars for analysis and generation, in contrast, were kept separate in order to optimize each one for its own purpose.

A third set of issues in grammar design concerns the ease of grammar writing and debugging. If these things were not given high priority, more projects would use context-free grammars instead of various augmented grammar formalisms (such as unification-based formalisms) that are more convenient to write but slower to parse. One reason for the large amount of energy devoted to the development of grammar formalisms in computational linguistics is that grammars are large rule systems that must be maintained, updated and adapted to new applications, and a good grammar formalism can greatly facilitate these tasks.[1] Our experience has been that good grammar

[1] Computational linguists are also interested in developing and testing theories of grammar.

formalisms and grammar-debugging tools are at least as important as good programming languages and program-debugging tools.

The design of grammars is important because even the best parser won't run well without a good grammar. The point I hope to have made in this section is that designing a grammar involves juggling many priorities and decisions in combinations that will be different for each application and each system architecture. There are, indeed, important issues in syntactic processing that will affect the speed and quality of translation. The next question is how syntactic theory can help.

4. The Role of Syntactic Theory in Grammar Design

A false expectation about the role of syntactic theory in designing grammars has lead to disappointment about what it has actually been able to contribute to natural language processing. The false expectation is that the role of syntactic theory is to bring about comprehensive coverage of a corpus. The disappointment in what syntactic theory has been able to contribute is expressed in the following common criticisms. (1) Syntactic theory does not cover enough. Syntacticians have concerned themselves with a small core of interesting phenomena which comprise only a tiny fraction of the phenomena in real text (Nagao et al., 1988). (2) Syntactic theories provide analyses that are too detailed to be useful. They deal with sentences that are contrived to test subtle predictions of principles of grammar, but these sentences are not likely to occur with any significant frequency in real text. Furthermore, since there is often disagreement among native speakers about whether the contrived sentences are grammatical or not, they should probably all be treated as grammatical for the sake of robustness. (3) There are exceptions to many analyses that come out of syntactic theory. This would also be a barrier to robustness if syntactic theories were implemented in natural language processing systems.

The answer to these criticisms is that covering a corpus is simply not what should be expected from syntactic theory. In fact, syntacticians have often pointed out that it is not their goal to provide comprehensive sets of rules. Instead, what they are looking for are common organizing principles that underlie apparently diverse sets of rules and that are cross-linguistically valid. Syntacticians also seek ways to divide complex sets of principles into manageable subsystems. Proponents of the major syntactic theories have come up with different principles and different ways of dividing the principles into subsystems, but as a group they have made significant progress in these areas, and, as a group, this is what they have contributed to natural language processing.

What we should look for as contributions of syntactic theory are, therefore, not rules but general principles and strategies. The principles and strategies may not be comprehensive and may need to be stretched and augmented to adapt to the needs of real systems, but they provide a general architecture for grammar design that results in efficient and easily maintained grammars.[2]

[2] This sort of flexible adaptation of syntactic theory is also part of the methodology of the EUROTRA project. However, the most salient reason that is cited for not strictly following any particular syntactic theory that the linguistic traditions of different language groups must be respected (Kirchmeier-Anderson, to appear), and it is not yet clear that syntactic theories provide optimal analyses of all languages.

The remainder of this section illustrates the role of syntactic theory in the KBMT-89 system, concentrating on levels of representation and general strategies for various syntactic phenomena.

Levels of representation in syntactic theory are subsystems of grammar that interact with other subsystems, but differ from them in the way that linguistic objects are represented and in the principles that govern them. Their significance from a practical point of view is that they divide complex syntactic phenomena into manageable subsystems. Furthermore, since syntactic theory is the study of human language competence, levels of representation are intended to be applicable to all human languages.[3]

The KBMT-89 system adopts three levels of representation (constituent structure, functional structure and predicate-argument structure)[4] and two inter-level mappings (grammatical encoding and lexical linking) from Lexical Functional Grammar. These levels of representation and inter-level mappings are described extensively elsewhere from the point of view of LFG (Kaplan & Bresnan, 1982; Bresnan, 1982a, 1982b; Bresnan & Kanerva, 1989; Levin, 1987) and from the point of view of KBMT-89 (Nirenburg & Levin, 1989; Levin (forthcoming)). Therefore, they will be summarized briefly here, emphasizing how constructs that are well-motivated in syntactic theory have turned out to be beneficial for machine translation as well.

Perhaps the best way to understand the levels of representation in LFG is to think of the problem of associating phrases with their semantic roles. How is it determined that, in the sentence *The user removed the diskette from the drive, the user* is the agent of the removing event, *the diskette* is the thing that got removed and *the drive* was the location of the diskette before it was removed? In KBMT-89, as in LFG, this is accomplished in two steps—assignment of grammatical functions (mapping constituent structure to functional structure) and linking of grammatical functions with semantic roles (mapping functional structure to predicate-argument structure).

Another way to understand the levels of representation of LFG is in terms of how similarities between related sentences are captured. Each distinct sentence has a distinct constituent structure. Sentences with different constituent structures but the same grammatical relations (such as *The user removed the diskette* and *Did the user remove the diskette)* will map onto similar functional structures via grammatical encoding. Sentences with different grammatical relations (such as *The user removed the diskette* and *The diskette was removed by the user)* will map onto different functional structures, but their similarity in meaning will be captured by the lexical linking rules which will associate them with similar predicate-argument structures.

In what follows, I will illustrate the assignment of grammatical functions and semantic roles with the three related sentences shown in (1). Sentences (l)a and (l)b are to be taken as representative of sentences that are related in meaning, but have different grammatical relations. Sentences (l)a and (l)c are to be taken as representative of related sentences with the same grammatical relations but different word order and constituency.

[3] There are some doubts about the cross-linguistic validity of levels of representation in syntactic theory. Government and Binding Theory uses configurational tree structures that are not obviously suitable for free word order languages, and Lexical Functional Grammar and Relational Grammar use grammatical functions like subject and object, which are motivated more strongly in some languages and less strongly in others.

[4] The EUROTRA project uses similar levels of representation (Arnold & des Tombe, 1987; Durand et al., to appear).

a. The user removed the diskette from the drive.

b. The diskette was removed from the drive by the user.

c. What did the user remove from the drive?

The first level of representation, *constituent structure or c-structure,* represents syntactic constituents. At this level, it is determined that *the user, the diskette* and *the drive* are noun phrases; *from the drive* and *by the user* are prepositional phrases; and *removed the diskette from the drive, remove from the drive* and *was removed from the drive* are verb phrases. The grammar describes c-structures via a set of context-free phrase structure rules.

C-structures are associated *with functional structures* (f-structures) by a process *of grammatical encoding.* Grammatical encoding assigns grammatical functions to c-structure phrases. For sentence (l)a, grammatical encoding will assign the function SUBJ (subject) to *the user,* OBJ (object) to *the diskette* and OBL (oblique)to *from the drive.* The verb *remove* will be identified as the head of the sentence.

Grammatical encoding is carried out by unification of features associated with phrase structure rules and lexical entries. Problems in verb forms, agreement and case marking, along with many other problems will be detected during grammatical encoding as unification failures. Various syntactic features of the sentence will also be extracted at this stage. For example, it will be determined that the verb in (l)a is past tense and active and that the sentence is declarative.

The wh-question, (l)c, will be similar to a(l) in grammatical encoding. SUBJ will be assigned to *the user* and OBL will be assigned to *from the drive.* Slash-category rules (Gazdar et al., 1985) will determine that there is no noun phrase in the position that would normally be occupied by the OBJ of *remove,* but various unifications will result in the OBJ function being assigned to *what.*[5] Grammatical Encoding for the passive sentence (l)b differs more from the active (l)a. In this case, *the diskette* is the SUBJ and *from the drive* and *by the user* are OBL.

The output of grammatical encoding is an f-structure. F-structures are sets of pairs of features and values.[6] They store syntactic features (like TENSE, MOOD, CASE, NUMBER and PERSON) and they represent the grammatical functions that result from grammatical encoding.

F-structures are mapped *onto predicate-argument structures* by lexical linking rules. Predicate-argument structures in KBMT-89 take the form of ontological frames and slots (Nirenburg & Levin, 1989). The verb *remove,* for example, is mapped via lexical mapping rules (Gates et al., 1989) onto a REMOVE frame with slot labels agent, theme, source and goal, among others. In this particular example, the slot names happen to coincide with thematic role labels that are widely accepted by syntacticians. However, ontological frames and slots do not always correspond

[5] We have not implemented the mechanism for functional uncertainty described by Kaplan & Zaenen (1989) and Kaplan & Maxwell (1987).

[6] In the KBMT-89 system, feature structures cannot be *re-entrant.* That is, two features cannot share the same instance of the same value, although they can have values that look identical. This is a result of the pseudo-unification algorithm. See Tomita and Knight (1988) for details.

so closely with predicate-argument structures from syntactic theory. Predicate-argument structures represent the lexical semantics of predicates and their arguments. These are often similar, but are not always identical, to the predicate-argument structures of translational equivalents in other languages. They are, therefore, not always suitable as the basis of a language-independent interlingua.

The rules that map the contents of f-structures to the slot-fillers of frames were called *structural mapping rules* in KBMT-89 (Gates et al., 1989). The structural mapping rules for the active verb *remove* in (l)a and (l)c indicate that the SUBJ of *remove* maps onto the agent slot, the OBJ maps onto the theme slot and an OBL marked with the preposition *from* maps onto the source slot. In (l)a, this will result in *the user* filling the agent slot, *the diskette* filling the theme slot and *the drive* filling the source slot. In (l)c, this will result in *the user* filling the agent slot, *what* filling the theme slot and *the drive* filling the source slot.

The passive verb *removed* in (l)b triggers different mapping rules. The passive mapping rules for *remove* associate the SUBJ with the theme slot, an OBL with the preposition *from* with the source slot and an OBL with the preposition *by* with the agent slot. The result is that, *the user* fills the agent slot, *the diskette* fills the theme slot and *the drive* fills the source slot. Thus the active and passive sentences are identical at the level of instantiated frames even though their c-structures and f-structures are different.

Structural mapping rules are considered to be lexical information because they reflect the subcategorization patterns of lexical items. Each language has a *mapping rule hierarchy* whose nodes represent classes of words with similar structural mapping rules. In this way, lexical items can inherit mapping rules by virtue of membership in a class (Mitamura, 1989).

The methods just described for associating phrases with semantic slots and for capturing the relationships between related sentences are well-motivated in the theory of Lexical Functional Grammar, but we have found that they are also well-motivated for machine translation. In the remarks that follow, I will concentrate on the motivation for f-structure as an intervening level between c-structure and instantiated frames.

The importance of f-structures in syntactic theory is that they allow us to talk about a grammatical function such as SUBJ without referring to its various possible case markings, agreement patterns and structural positions. As a result, many cross-linguistic generalizations can be described more easily in terms of f-structures than in terms of c-structures, and even language-specific rules are more easily stated over f-structures in languages with free word order. Phenomena that are more cleanly treated in terms of f-structures include control (Simpson & Bresnan 1982; Bresnan 1982a; Mohanan, 1983), syntactic constraints on pronoun reference (Bresnan, Halvorsen & Maling, 1987; Dalrymple, 1989) and resolution of some aspects of ellipsis (Levin, 1982). This argument for f-structures in syntactic theory is equally valid in machine translation. The rules for handling these phenomena would be less elegant and would vary more from language to language if they were formulated in terms of c-structure or ontological frames.

F-structure also plays a significant role as the intermediate stage in the two-step mapping from surface constituents to predicate-argument relations. This is important in linguistic theory because there are salient cross-linguistic generalizations in the mapping from c-structure to grammatical functions (Bresnan 1982a, 1982b; Mohanan, 1982) and salient cross-linguistic generalizations in the mapping from grammatical functions to argument positions (Bresnan and Kanerva, 1989; Levin, 1987). But the mapping directly from c-structure positions to argument positions does not reveal significant insights into the nature of human language.

In KBMT-89, f-structures constitute the input to mapping rules that produce instantiated frames. It is more convenient to map to the interlingua texts from f-structure than from c-structure because it is sometimes the case that more than one c-structure maps onto essentially the same f-structure. If we mapped from c-structures, we would have to write more mapping rules. Also, if we map from f-structures, our mapping rules for different languages will be more alike. This will allow us to take advantage of principles of linguistic theory and lexical semantics in the organization of mapping rules (Mitamura, 1989). In short, a messy problem, the mapping from c-structure to instantiated frames, is divided into two manageable modules, a mapping from c-structure to f-structure and a mapping from f-structure to frames.

A third role of f-structure in syntactic theory as well as machine translation is to store features of words (e.g., number, person and tense) to be matched against features of other words in order to handle agreement, case marking and verbal morphology. We currently do not require our grammars to conform to any particular theory of these phenomena, but we find the unification-based formalism for grammatical encoding to be ideal for implementing them.

The discussion of levels of representation has, so far, alluded to specific analyses of syntactic constructions such as passive sentences and wh-questions. In adapting these analyses to a machine translation system, we have found that the rules and principles of syntactic theory cannot be implemented verbatim because of the practical demands of building an efficient and robust working system with many interacting components. However, in most cases, the general strategy proposed by syntactic theory is applicable and useful in a working machine translation program.

Analyses that we have adopted from syntactic theory in KBMT-89 include: alternate mappings between grammatical functions and semantic roles for passivization and other relation-changing rules; alternate grammatical encodings for non-relation-changing rules; slash categories for long-distance dependencies; restoring "understood" elements in the mapping from c-structure to f-structure; using f-structures to store and check grammatical features; checking subcategorization frames in f-structure rather than c-structure or predicate-argument structure; having heads of phrases constrain the grammatical features of their complements; treating auxiliary verbs as main verbs that constrain the features of their complements; and dividing verbs into classes with similar subcategorization patterns for the purpose of lexical linking. For the KBMT-89 project, we considered these things to be implementation-independent strategies that could be stretched to fit the needs of different applications. We have found in working on the successors of KBMT-89—DIANA and DIOGENES—that the same general strategies can be adapted to systems with different architectures and different implementational constraints.

5. Conclusion

Syntactic processing involves parsing and generation algorithms, grammar formalisms and sets of grammar rules. Much attention has been paid to algorithms and formalisms, but little attention has been paid to the grammars themselves. Nevertheless, I have shown that there are significant issues to be addressed in the design of grammars that can affect the speed and quality of translation. Since the resolution of these issues depends on the requirements of individual systems, grammar design is an important part of the process of overall system design. The role of syntactic theory in all of this is to lay out general strategies and principles that result in well-designed grammars and that are flexible enough to adapt to the varying demands of different implementations.

Our approach to the mixing of theory and engineering in KBMT-89 was to make the theory available for grammar writers to follow as closely as they could, but also to allow freedom to stray from pure theoretical implementations when it was necessary for reasons of efficiency or because of the demands of the system. It is a mistake to abandon syntactic theory on the grounds that it has not produced a set of rules capable of analyzing unrestricted text and it is equally misguided to enforce the theory rigidly, not allowing for the necessary deviations. Although syntactic theory alone will not solve all the problems of robust, high quality translation, it is at least part of the solution.

Acknowledgements:

I would like to thank Ken Goodman, Sergei Nirenburg and Hiroaki Kitano for helpful discussions. The KBMT-89 grammar writers, Donna Gates, Teruko Mitamura and Koichi Takeda also deserve special credit.

6. References

Arnold, D., and L. des Tombe. 1987. Basic Theory and Methodology in EUROTRA. In S. Nirenburg (ed.), **Machine Translation: Theoretical and Methodological Issues.** Cambridge: Cambridge University Press.

Bresnan, J. 1982a. Control and Complementation. In J. Bresnan (ed.), **The Mental Representation of Grammatical Relations.** Cambridge, MA: MIT Press.

Bresnan, J. 1982b. The Passive in Lexical Theory . In J. Bresnan (ed.), **The Mental Representation of Grammatical Relations.** Cambridge, MA: MIT Press.

Bresnan, J., P.-K. Halvorsen, and J. Maling. 1987. Logophoricity and Bound Anaphors. ms, Stanford University, Stanford, CA.

Bresnan, J., and J. Kanerva. 1989. Locative Inversion in Chichewa: A Case Study of Factorization in Grammar. *Linguistic Inquiry* **20:1,**

Dalrymple, M. 1989. Anaphoric Binding: Syntactic Constraints and Semantic Interpretation. Thesis Proposal, Stanford University, Stanford, CA.

Dorr, B. 1989. Lexical Conceptual Structure and Generation in Machine Translation. *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society.* Ann Arbor, MI.

Durand, J., P. Bennett, V. Allegranza, F. van Eynde, L. Humphreys, P. Schmidt, & E. Steiner. To Appear. The Eurotra Linguistic Specifications: An Overview. To appear in *Machine Translation.*

Gates, D., D. Haberlach, T. Kaufmann, M. Kee, R. McCardell, T. Mitamura, I. Monarch, S. Morrisson, S. Nirenburg, E. Nyberg, K. Takeda, and M. Zabludowski. 1989. Lexicons. *Machine Translation* **4:1.**

Gazdar, G., E. Klein, G. Pullum, and I. Sag. 1985. **Generalized Phrase Structure Grammar.** Cambridge, MA: Harvard University Press.

Goodman, K., and S. Nirenburg. 1989. **KBMT-89.** CMU CMT Technical Report.

Kaplan, R., and J. Bresnan. 1982. Lexical Functional Grammar: A Formal System for Grammatical Representation. In J. Bresnan (ed.), **The Mental Representation of Grammatical Relations.** Cambridge, MA: MIT Press.

Kaplan, R., K. Netter, J. Wedekind, and A. Zaenen. 1989. Translation by Structural Correspondences. *Proceedings of the European Association for Computational Linguistics.*

Kaplan, R., and J. Maxwell. 1987. An Algorithm for Functional Uncertainty. Manuscript, Xerox PARC.

Kaplan, R., and A. Zaenen. 1989. Long Distance Dependencies, Constituent Structure, and Functional Uncertainty. In M. Baltin and A. Kroch, (eds.), **Alternative Conceptions of Phrase Structure.** Chicago: University of Chicago Press.

Kirchmeier-Andersen, S. To Appear. The Implementation of the Language Modules. To appear in *Machine Translation.*

Levin, L. 1982. Sluicing: A Lexical Interpretation Procedure. In J. Bresnan (ed.), **The Mental Representation of Grammatical Relations.** Cambridge, MA: MIT Press.

Levin, L. 1987. Toward a Linking Theory of Relation Changing Rules in LFG. *CSLI Report No. CSLI-87-115,* Center for the Study of Language and Information, Stanford, CA.

Levin, L. (forthcoming) Grammar Formalisms. To appear in K. Goodman and S. Nirenburg, (eds.) **Knowledge Based Machine Translation: A Case Study.** Morgan-Kaufmann.

Mitamura, T. 1989. **Organization of Predicate Conceptual Frames and Mapping Rules for Natural Language Processing.** Ph.D. Thesis, Department of Linguistics, University of Pittsburgh, Pittsburgh, PA.

Mohanan, K. P. 1982. Grammatical Relations and Clause Structure in Malayalam. In J. Bresnan (ed.), **The Mental Representation of Grammatical Relations.** Cambridge, MA: MIT Press.

Mohanan, K. P. 1983. Functional and Anaphoric Control. *Linguistic Inquiry* **14:4.**

Nagao, M., K. Jensen, D. Roesner, E. Hajicova, J. Tsujii, and M. Tomita. 1988. Panel on: Language Engineering: The Real Bottle Neck of Natural Language Processing. *Proceedings of the International Conference on Computational Linguistics (COLING 88).* Budapest.

Nirenburg, S., and L. Levin. 1989. Knowledge Representation Support. *Machine Translation.* **4:1.**

Raskin, V. 1987a. Linguistics and Natural Language Processing. In S. Nirenburg (ed.), **Machine Translation: Theoretical and Methodological Issues.** Cambridge: Cambridge University Press.

Raskin, V. 1987b. What is there in Linguistic Semantics for Natural Language Processing? *Presentations from the 1987 Natural Language Planning Workshop.* Northeast Artificial Intelligence Consortium. New York.

Sharp, R. 1986. A Parametric NL Translator. *Proceedings of the Eleventh International Conference on Computational Linguistics (COLING '86).* Bonn.

Simpson, J., and J. Bresnan. 1982. Control and Obviation in Warlpiri. *Proceedings of the first Annual West Coast Conference on Formal Linguistics.*

Tomita, M., and K. Knight. 1988. Pseudo-Unification and Full Unification. Technical Memo, Center for Machine Translation, Carnegie Mellon University, Pittsburgh, PA.

Wehrli, E. 1987. Recent Developments in Theoretical Linguistics and Implications for Machine Translation. In M. King (ed.), **Machine Translation Today: The State of the Art.** Edinburgh: Edinburgh University Press.