PROCESSING ENGLISH WITH A GENERALIZED PHRASE STRUCTURE GRAMMAR

Jean Mark Gawron, Jonathan King, John Lamping, Egon Loebner,
E. Anne Paulson, Geoffrey K. Pullum, Ivan A. Sag, and Thomas Wasow

Computer Research Center
Hewlett Packard Company
1501 Page Mill Road
Palo Alto, CA 94304

ABSTRACT

This paper describes a natural language processing system implemented at Hewlett-Packard's Computer Research Center. The system's main components are: a Generalized Phrase Structure Grammar (GPSG); a top-down parser; a logic transducer that outputs a first-order logical representation; and a "disambiguator" that uses sortal information to convert "normal-form" first-order logical expressions into the query language for HIRE, a relational database hosted in the SPHERE system. We argue that theoretical developments in GPSG syntax and in Montague semantics have specific advantages to bring to this domain of computational linguistics. The syntax and semantics of the system are totally domain-independent, and thus, in principle, highly portable. We discuss the prospects for extending domain-independence to the lexical semantics as well, and thus to the logical semantic representations.

1. INTRODUCTION

This paper is an interim progress report on linguistic research carried out at Hewlett-Packard Laboratories since the summer of 1981. The research had three goals: (1) demonstrating the computational tractability of Generalized Phrase Structure Grammar (GPSG), (2) implementing a GPSG system covering a large fragment of English, and (3) establishing the feasibility of using GPSG for interactions with an inferencing knowledge base.

Section 2 describes the general architecture of the system. Section 3 discusses the grammar and the lexicon. A brief dicussion of the parsing technique used in found in Section 4. Section 5 discusses the semantics of the system, and Section 6 presents a detailed example of a parse-tree complete with semantics. Some typical examples that the system can handle are given in the Appendix.

The system is based on recent developments in syntax and semantics, reflecting a modular view in which grammatical structure and abstract logical structure have independent status. The understanding of a sentence occurs in a number of stages, distinct from each other and governed by different principles of organization. We are opposed to the idea that language understanding can be achieved without detailed syntactic analysis. There is, of course, a massive pragmatic component to human linguistic interaction. But we hold that pragmatic inference makes use of a logically prior grammatical and semantic analysis. This can be fruitfully modeled and exploited even in the complete absence of any modeling of pragmatic inferencing capability. However, this does not entail an incompatibility between our work and research on modeling discourse organization and conversational interaction directly. Ultimately, a successful language understanding system will require both kinds of research, combining the advantages of precise, grammar-driven analysis of utterance structure and pragmatic inferencing based on discourse structures and knowledge of the world. We stress, however, that our concerns at this stage do not extend beyond the specification of a system that can efficiently extract literal meaning from isolated sentences of arbitrarily complex grammatical structure. Future systems will exploit the literal meaning thus extracted in more ambitious applications that involve pragmatic reasoning and discourse manipulation.

The system embodies two features that simultaneously promote extensibility, facilitate modification, and increase efficiency. The first is that its grammar is context-free in the informal sense sometimes (rather misleadingly) used in discussions of the autonomy of grammar and pragmatics: the syntactic rules and the semantic translation rules are independent of the specific application domain. Our rules are not devised ad hoc with a particular application or type of interaction in mind. Instead, they are motivated by recent theoretical developments in natural language syntax, and evaluated by the usual linguistic canons of simplicity and generality. No changes in the knowledge base or other exigencies deriving from a particular context of application can introduce a problem for the grammar (as distinct, of course, from the lexicon).

The second relevant feature is that the grammar in the system is context-free in the sense of formal language theory. This makes the extensive mathematical literature on context-free phrase structure grammars (CF-PSG's) directly relevant to the enterprise, and permits utilization of all the well-known techniques for the computational implementation of context-free grammars. It might seem anachronistic to base a language understanding system on context-free

parsing. As Pratt (1975, 423) observes: "It is fashionable these days to want to avoid all reference to context-free grammars beyond warning students that they are unfit for computer consumption as far as computational linguistics is concerned." Moreover, widely accepted arguments have been given in the linguistics literature to the effect that some human languages are not even weakly context-free and thus cannot possibly be described by a CF-PSG. However, Gazdar and Pullum (1982) answer all of these arguments, showing that they are either formally invalid or empirically unsupported or both. It seems appropriate, therefore, to take a renewed interest in the possibility of CF-PSG description of human languages, both in computational linguistics and in linguistic research generally.

2. COMPONENTS OF THE SYSTEM

The linguistic basis of the GPSG linguistic system resides in the work reported in Gazdar (1981, 1982) and Gazdar, Pullum, and Sag (1981).[1] These papers argue on empirical and theoretical grounds that context-freeness is a desirable constraint on grammars. It clearly would not be so desirable, however, if (1) it led to lost generalizations or (2) it resulted in an unmanageable number of rules in the grammar. Gazdar (1982) proposes a way of simultaneously avoiding these two problems. Linguistic generalizations can be captured in a context-free grammar with a *metagrammar*, i.e. a higher-level grammar that generates the actual grammar as its language. The metagrammar has two kinds of statements:

(1) Rule schemata. These are basically like ordinary rules, except that they contain variables ranging over categories and features.

(2) Metarules. These are implicational statements, written in the form $\alpha ==> \beta$, which capture relations between rules. A metarule $\alpha ==> \beta$ is interpreted as saying, "for every rule that is an instantiation of the schema α, there is a corresponding rule of form β." Here β will be $\theta(\alpha)$, where θ is some mapping specified partly by the general theory of grammar and partly in the metarule formulation. For instance, it is taken to be part of the theory of grammar that θ preserves unchanged the subcategorization (rule name) features of rules (cf. below).

The GPSG system also assumes the Rule-to-Rule Hypothesis, first advanced by Richard Montague, which requires that each syntactic rule be associated with a single semantic translation rule. The syntax-semantics match is realized as follows: each rule is a triple consisting of a rule name, a syntactic statement (formally a local condition on node admissibility), and a semantic translation, specifying how the higher-order logic representations of the daughter nodes combine to yield the correct translation for the mother.[2]

The present GPSG system has five components:

1. Grammar
   a. Lexicon
   b. Rules and Metarules
2. Parser and Grammar Compiler
3. Semantics Handler
4. Disambiguator
5. HIRE database

3. GRAMMAR AND LEXICON

The grammar that has been implemented thus far is only a subset of a much larger GPSG grammar that we have defined on paper. It nevertheless describes a broad sampling of the basic constructions of English, including a variety of prepositional phrase constructions, noun-noun compounds, the auxiliary system, genitives, questions and relative clauses, passives, and existential sentences.

Each entry in the lexicon contains two kinds of information about a lexical item, syntactic and semantic. The syntactic part of an entry consists of a syntactic feature specification; this includes, *inter alia*, information about any irregular morphology the item may have, and what is known in the linguistic literature as *strict subcategorization* information. In our terms the latter is information linking lexical items of a particular category to specific environments in which that category is introduced by phrase structure rules. Presence in the lexical entry for an item *I* of the feature *R* (where *R* is the name of a rule) indicates that *I* may appear in structures admitted by *R*, and absence indicates that it may not.

The semantic information in a lexical entry is sometimes simple, directly linking a lexical item with some HIRE predicate or relation. With verbs or prepositions, there is also a specification of what case roles to associate with particular arguments (cf. below for discussion of case roles). Expressions that make a complex logical contribution to the sentence in which they appear will in general have complicated translations. Thus *every* has the translation:

---

1. See also Gazdar, Pullum, Sag, and Wasow (1982) for some further discussion and comparison with other work in the linguistic literature.

2. There is a theoretical issue here about whether semantic translation rules need to be stipulated for each syntactic rule or whether there is a general way of predicting their form. See Klein and Sag (1981) for an attempt to develop the latter view, which is not at present implemented in our system.

(LAMBDA P (LAMBDA Q ((FORALL X (P X))

--> (Q X)))),

This indicates that it denotes a function which takes as argument a set P, and returns the set of properties that are true of all members of that set (cf. below for slightly more detailed discussion).

A typical rule looks like this:

<VP109: V! -> V N!! N!!2: ((V N!!2) N!!)>

The exclamation marks here are our notation for the bars in an X-bar category system. (See Jackendoff (1977) for a theory of this type—though one which differs on points of detail from ours.) The rule has the form <a: b: c>. Here *a* is the name 'VP109'; *b* is a condition that will admit a node labeled 'V!' if it has three daughter nodes labeled respectively 'V' (verb), 'N!!' (noun phrase at the second bar level), and 'N!!' (the numeral 2 being merely an index to permit reference to a specific symbol in the semantics, the metarules, and the rule compiler, and is not a part of the category label); and *c* is a semantic translation rule stating that the V constituent translates as a function expression taking as its argument the translation of the second N!!, the result being a function expression to be applied to the translation of the first N!!.

By a general convention in the theory of grammar, the rule name is one of the feature values marked on the lexical head of any rule that introduces a lexical category (as this one introduces V). Only verbs marked with that feature value satisfy this rule. For example, if we include in the lexicon the word *give* and assign to it the feature *VP109*, then this rule would generate the verb phrase *gave Anne a job*.

A typical metarule is the passive metarule, which looks like this (ignoring semantics):

<PAS: <V!-> V N!! W> => <V! -> V[PAS] W>>

W is a string variable ranging over zero or more category symbols. The metarule has the form <N: <A> => <B>>, where *N* is a name and <A> and <B> are schemata that have rules as their instantiations when appropriate substitutions are made for the free variables. This metarule says that for every rule that expands a verb phrase as verb followed by noun phrase followed by anything else (including nothing else), there is another rule that expands verb phrase as verb with passive morphology followed by whatever followed the noun phrase in the given rule. The metarule PAS would apply to grammar rule VP109 given above, yielding the rule:

<VP109: V! -> V[PAS] N!!>

As we noted above, the rule number feature is preserved here, so we get *Anne was given a job*, where the passive verb phrase is *given a job*, but not **Anne was hired a job*.[3]

Passive sentences are thus analyzed directly, and not reduced to the form of active sentences in the course of being analyzed, in the way that is familiar from work on transformational grammars and on ATN's. However, this does not mean that no relation between passives and their active counterparts is expressed in the system, because the rules for analyzing passives are in a sense derivatively defined on the basis of rules for analyzing actives.

More difficult than treating passives and the like, and often cited as literally impossible within a context-free grammar,[4] is treating constructions like questions and relative clauses. The apparent difficulty resides in the fact that in a question like *Which employee has Personnel reported that Anne thinks has performed outstandingly?*, the portion beginning with the third word must constitute a string analyzable as a sentence except that at some point it must lack a third person singular noun phrase in a position where such a noun phrase could otherwise have occurred. If it lacks no noun phrase, we get ungrammatical strings of the type **Which employee has Personnel reported that Anne thinks Montague has performed outstandingly?*. If it lacks a noun phrase at a position where the verb agreement indicates something other than a singular one is required, we get ungrammaticalities like **Which employee has Personnel reported that Anne thinks have performed outstandingly?*. The problem is thus one of guaranteeing a grammatical dependency across a context that may be arbitrarily wide, while keeping the grammar context-free. The technique used is introduced into the linguistic literature by Gazdar (1981). It involves an augmentation of the nonterminal vocabulary of the grammar that permits constituents with "gaps" to be treated as not belonging to the same category as similar constituents without gaps. This would be an unwelcome and inelegant enlargement of the grammar if it had to be done by means of case-by-case stipulation, but again the use of a metagrammar avoids this. Gazdar (1981) proposes a new set of syntactic categories of the form α/β, where α and β are categories from the basic nonterminal vocabulary of the grammar. These are called *slash categories*. A slash category α/β may be thought of as representing a constituent of category α with a missing internal occurrence of β. We employ a method of introducing slash categories that was suggested by Sag (1982): a metarule stating that for every rule introducing some β under α there is a parallel rule introducing β/γ under α/γ. In other words, any constituent can have a gap of type γ if one of its daughter constituents does too. Wherever this would lead to a daughter constituent with the label γ/γ in some

---

3. We regard *was given a job* not as a passive verb phrase itself but as a verb phrase containing the verb *be* plus a passive verb phrase containing *given* and *a job*.

4. See Pullum and Gazdar (1982) for references.

rule, another metarule allows a parallel rule without the γ/γ, and therefore defines rules that allow for actual gaps—i.e., missing constituents. In this way, complete sets of rules for describing the unbounded dependencies found in interrogative and relative clauses can readily be written. Even long-distance agreement facts can be (and are) captured, since the morphosyntactic features relevant to a specific case of agreement are present in the feature composition of any given γ.

4. PARSING

The system is initialized by expanding out the grammar. That is, the metarules are applied to the rules to produce the full rule set, which is then compiled and used by the parser. Metarules are not consulted during the process of parsing. One might well wonder about the possible benefits of the other alternative: a parser that made the metarule-derived rules to order each time they were needed, instead of consulting a precompiled list. This possibility has been explored by Kay (1982). Kay draws an analogy between metarules and phonological rules, modeling both by means of finite state transducers. We believe that this line is worth pursuing; however, the GPSG system currently operates off a precompiled set of rules.

Application of ten metarules to forty basic rules yielded 283 grammar rules in the 1/1/82 version of the GPSG system. Since then the grammar has been expanded somewhat, though the current version is still undergoing some debugging, and the number of rules is unstable. The size of the grammar-plus-metarules system grows by a factor of five or six through the rule compilation. The great practical advantage of using a metarule-induced grammar is, therefore, that the work of designing and revising the system of linguistic rules can proceed on a body of statements that is under twenty percent of the size it would be if it were formulated as a simple list of context-free rules.

The system uses a standard type of top-down parser with no lookahead, augmented slightly to prevent it from looking for a given constituent starting in a given spot more than once. It produces, in parallel, all legal parse trees for a sentence, with semantic translations associated with each node.

5. SEMANTICS

The semantics handler uses the translation rule associated with a node to construct its semantics from the semantics of its daughters. This construction makes crucial use of a procedure that we call Cooper storage (after Robin Cooper; see below). In the spirit of current research in formal semantics, each syntactic constituent is associated directly with a single logic expression (modulo Cooper Storage), rather than any program or procedure for producing such an expression. Our semantic analysis thus embraces the principle of "surface compositionality." The semantic representations derived at each node are referred to as the Logical Representation (LR).

The disambiguator provides the crucial transition from LR to HIRE queries; the disambiguator uses information about the *sort*, or *domain of definition*, of various terms in the logical representation. One of the most important functions of the disambiguator is to eliminate parses that do not make sense in the conceptual scheme of HIRE.

HIRE is a relational database with a certain amount of inferencing capability. It is implemented in SPHERE, a database system which is a descendant of FOL (described in Weyhrauch (1980)). Many of the relation-names output by the disambiguator are derived relations defined by axioms in SPHERE. The SPHERE environment was important for this application, since it was essential to have something that could process first-order logical output, and SPHERE does just that. A noticeable recent trend in database theory has been a move toward an interdisciplinary comingling of mathematical logic and relational database technology (see especially Gallaire and Minker (1978) and Gallaire, Minker and Nicolas (1981)). We regard it as an important fact about the GPSG system that links computational linguistics to first-order logical representation just as the work referred to above has linked first-order logic to relational database theory. We believe that SPHERE offers promising prospects for a knowledge representation system that is principled and general in the way that we have tried to exemplify in our syntactic and semantic rule system. Filman, Lamping and Montalvo (1982) present details of some capabilities of SPHERE that we have not as yet exploited in our work, involving the use of multiple contexts to represent viewpoints, beliefs, and modalities, which are generally regarded as insuperable stumbling-blocks to first-order logic approaches.

Thus far the linguistic work we have described has been in keeping with GPSG presented in the papers cited above. However two semantic innovations have been introduced to facilitate the disambiguator's translation from LR to a HIRE query. As a result the linguistic system version of LR has two new properties:

(1) The intensional logic of the published work was set aside and LR was designed to be an extensional first-order language. Although constituent translations built up on the way to a root node may be second-order, the system maintains first-order reducibility. This reducibility is illustrated by the following analysis of noun phrases as second-order properties (essentially the analysis of Montague (1970)). For example, the proper name *Egon* and the quantified noun phrase *every applicant* are both translated as sets of properties:

```
Egon = LAMBDA P (P EGON)
Every applicant = LAMBDA P (FORALL X
                    ((APPLICANT X) --> (P X)))
```

Egon is translated as the set of properties true of Egon, and *every applicant*, as the set of properties true of all applicants. Since basic predicates in the logic are first-order, neither of the above expressions can be made the direct argument of any basic predicate; instead the argument is some unique entity-level variable which is later bound to the quantifier-expression by quantifying in. This technique is essentially the storage device proposed in Cooper (1975). One advantage of this method of "deferring" the introduction into the interpretation process of phrases with quantifier meanings is that it allows for a natural, nonsyntactic treatment of scope ambiguities. Another is that with a logic limited to first-order predicates, there is still a natural treatment for coordinated noun phrases of apparently heterogeneous semantics, such as *Egon and every applicant*.

(2) HIRE represents events as objects. All objects in the knowledge base, including events, belong to various *sorts*. For our purposes, a *sort* is a set. HIRE relations are declared as properties of entities within particular sorts. For example, there is an *employment* sort, consisting of various particular employment events, and an *employment.employee* relation as well as *employment.organization* and *employment.manager* relations. More conventional relations, like *employee.manager* are defined as joins of the basic event relations. This allows the semantics to make some fairly obvious connections between verbs and events (between, say, the verb *work* and events of employment), and to represent different relations between a verb and its arguments as different first-order relations between an event and its participants. Although the lexical treatment sketched here is clearly domain dependent (the English verb *work* doesn't necessarily involve employment events), it was chosen primarily to simplify the ontology of a first implementation. As an alternative, one might consider associating *work* with events of a sort *labor*, one of whose subsorts was an employment event, defining employments as those labors associated with an organization.

Whichever choice one makes about the basic event-types of verbs, the mapping from verbs to HIRE relations cannot be direct. Consider a sentence like *Anne works for Egon*. The HIRE representation will predicate the *employment.manager* relation of a particular employment event and a particular manager, and the *employment.employee* relation of that same event and Anne. Yet where Egon in this example is picked out with the *employment.manager* relation, the sentence *Anne works for HP* will need to pick out HP with the *employment.organization* relation. In order to accomodate this many-to-many mapping between a verb and particular relations in a knowledge base, the lexicon stipulates special relations that link a verb to its eventual arguments. Following Fillmore (1968), these mediating relations are called case roles.

The disambiguator narrows the case roles down to specific knowledge base relations. To take a simple example, *Anne works for HP* has a logical representation reducible to:

```
(EXISTS SIGMA (AND (EMPLOYMENT SIGMA)
                   (AG SIGMA ANNE)
                   (LOC SIGMA HP)))
```

Here SIGMA is a variable over situations or event instantiations.[5] The formula may be read, "There is an employment-situation whose Agent is Anne and whose Location is HP." The lexical entry for *work* supplies the information that its subject is an Agent and its complement a Location. The disambiguator now needs to further specify the case roles as HIRE relations. It does this by treating each atomic formula in the expression locally, using the fact that Anne is a person in order to interpret AG, and the fact that HP is an organization in order to interpret LOC. In this case, it interprets the AG role as *employment.employee* and the LOC role as *employment.organization*.

The advantages of using the roles in Logical Representation, rather than going directly to predicates in a knowledge base, include (1) the ability to interpret at least some prepositional phrases, those known as adjuncts, without subcategorizing verbs specially for them, since the case role may be supplied either by a verb or a preposition. (2) the option of interpreting "vague" verbs such as *have* and *give* using case roles without event types. These verbs, then, become "purely" relational. For example, the representation of *Egon gave Montague a job* would be:

```
(EXISTS SIGMA (AND ((SO EGON) SIGMA)
                   ((POS MONTAGUE) SIGMA)
                   (EMPLOYMENT SIGMA)))
```

Here SO 'source' will pick out the same *employment.manager* relation it did in the example above; and POS 'possession' is the same relation as that associated with *have*. Here the situation-type is supplied by the translation of the noun *job*. It is important to realize that this representation is derived without giving the noun phrase *a job* any special treatment. The lexical entry for *give* contains the information that the subject is the source of the direct object, and the direct object the possession of the indirect object. If there were lamps in our knowledge base, the derived representation of *Egon gave Montague a lamp* would simply be the above formula with the predicate *lamp* replacing *employment*. The possession relation would hold between Montague and some

---

5. Our work in this domain has been influenced by the recent papers of Barwise and Perry on "situation semantics"; see e.g. Barwise and Perry (1982)).

lamp, and the disambiguator would retrieve whatever knowledge-base relation kept track of such matters.

Two active research goals of the current project are to give all lexical entries domain independent representations, and to make all knowledge base-specific predicates and relations the exclusive province of the disambiguator. One important means to that end is case roles, which allow us a level of abstract, purely "linguistic" relations to mediate between logical representations and HIRE queries. Another is the use of general event types such as labor, to replace event-types specific to HIRE, such as employments. The case roles maintain a separation between the domain representation language and LR. Insofar as that separation is achieved, then absolute portability of the system, up to and including the lexicon, is an attainable goal.

Absolute portability obviously has immediate practical benefits for any system that expects to handle a large fragment of English, since the effort in moving from one application to another will be limited to "tuning" the disambiguator to a new ontology, and adding "specialized" vocabulary. The actual rules governing the production of first-order logical representations make no reference to the facts of HIRE. The question remains of just how portable the current lexicon is; the answer is that much of it is already domain independent. Quantifiers like *every* (as we saw in the discussion of NP semantics) are expressed as logical constants; verbs like *give* are expressed entirely in terms of the case relations that hold among their arguments. Verbs like *work* can be abstracted away from the domain by a simple extension. The obvious goal is to try to give domain independent representations to a core vocabulary of English that could be used in a variety of application domains.

6. AN EXAMPLE

We shall now give a slightly more detailed illustration of how the syntax and compositional semantics rules work. We are still simplifying considerably, since we have selected an example where role frames are not involved, and we are not employing features on nodes. Here we have the grammar of a trivial subset of English:

```
<S1: S -> NP VP: (NP VP)>
<NP1: NP -> DET N: (DET N)>
<VP1: VP -> V NP: (V NP)>
<VP2: VP -> V A: A>
```

Suppose that the lexicon associated with the above rules is:

```
<every:DET: (LAMBDA P (LAMBDA Q
               (FORALL X ((P X)
                  IMPLIES (Q X)))))>
<applicant: N: APPLICANT>
<interviewed: V[(RULE VP1)]: INTERVIEW>
<Bill: NP: (LAMBDA P (P BILL))>
<is: V[(RULE VP2)]: (BE)>
<competent: A: (LAMBDA Y
               (EXPERT.LEVEL HIGH Y))>
```

The syntax of a lexical entry is <*L*: *C*: *T*>, where *L* is the spelling of the item, *C* is its grammatical category and feature specification (if other than the default set) and *T* is its translation into LR.

Consider how we assign an LR to a sentence like *Every applicant is competent*. The translation of *every* supplies most of the structure of the universal quantification needed in LR. It represents a function from properties to functions from properties to truth values, so when applied to *applicant* it yields a constituent, namely *every applicant*, which has one of the property slots filled, and represents a function from properties to truth-values; it is:

```
(LAMBDA P (FORALL X
    ((APPLICANT X) IMPLIES (P X))))
```

This function can now be applied to the function denoted by *competent*, i.e.

```
(LAMBDA Y
     (EXPERT.LEVEL HIGH Y))
```

This yields:

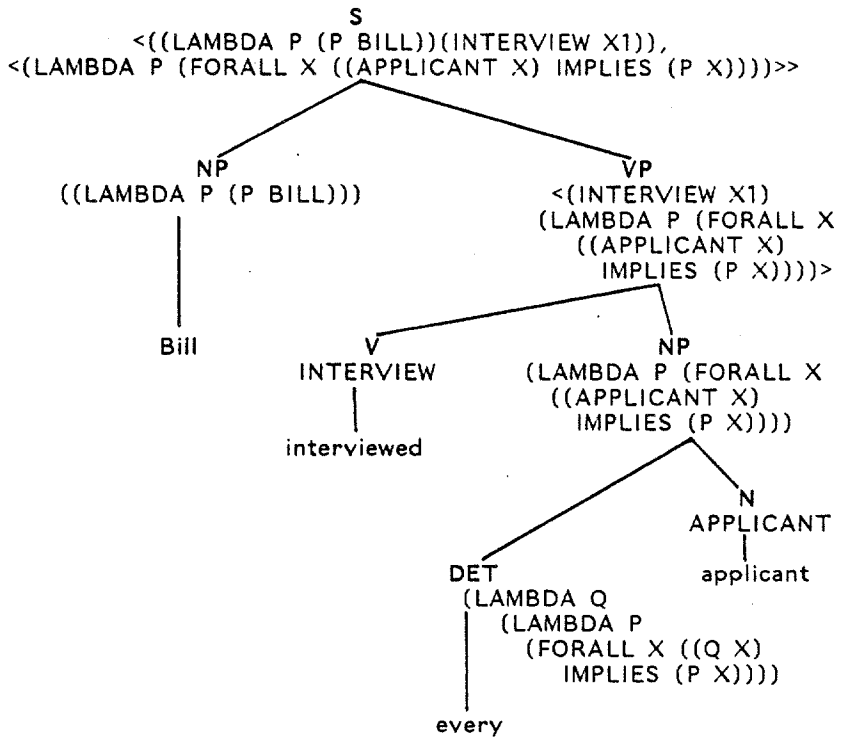
```
(FORALL X
   ((APPLICANT X)
   IMPLIES
   (LAMBDA Y
    (EXPERT.LEVEL HIGH Y)) X))
```

And after one more lambda-conversion, we have:

```
(FORALL X
   ((APPLICANT X)
   IMPLIES
    (EXPERT.LEVEL HIGH X)))
```

Fig. 1 shows one parse tree that would be generated by the above rules, together with its logical translation. The sentence is *Bill interviewed every applicant*. The complicated translation of the VP is necessary because INTERVIEW is a one-place predicate that takes an entity-type argument, not the type of function that *every applicant* denotes. We thus defer combining the NP translation with the verb by using Cooper storage. A translation with a stored NP is represented above in angle-brackets. Notice that at the S node the NP *every applicant* is still stored, but the subject is not stored. It has directly combined with the VP, by taking the VP as an *argument*. INTERVIEW is itself a two-place predicate, but one of its argument places has been filled by a place-holding variable, X1. There is thus only one slot left. The translation can now be completed via the operations of Storage Retrieval and lambda conversion. First, we simplify the part of the semantics that isn't in storage:

Fig. 1. A typical parse tree



((LAMBDA P (P BILL))(INTERVIEW X1)) =>
((INTERVIEW X1) BILL)

The function (LAMBDA P (P BILL)) has been evaluated with P set to the value (INTERVIEW X1); this is a conventional lambda-conversion. The rule for storage retrieval is to make a one-place predicate of the sentence translation by lambda-binding the placeholding variable, and then to apply the NP translation as a function to the result. The S-node translation above becomes:

```
((LAMBDA P
   (FORALL X
     ((APPLICANT X) IMPLIES (P X))))
      (LAMBDA X1 ((INTERVIEW X1) BILL)))

[lambda-conversion]      ==>

(FORALL X ((APPLICANT X) IMPLIES
   ((LAMBDA X1
       ((INTERVIEW X1) BILL)) X)))

[lambda-conversion]      ==>

(FORALL X ((APPLICANT X) IMPLIES
            (((INTERVIEW X) BILL))))
```

This is the desired final result.

7. CONCLUSION

What we have outlined is a natural language system that is a direct implementation of a linguistic theory. We have argued that in this case the linguistic theory has the special appeal of computational tractability (promoted by its context-freeness), and that the system as a whole offers the hope of a happy marriage of linguistic theory, mathematical logic, and advanced computer applications. The system's theoretical underpinnings give it compatibility with current research in Generalized Phrase Structure Grammar, and its augmented first order logic gives it compatibility with a whole body of ongoing research in the field of model-theoretic semantics.

The work done thus far is only the first step on the road to a robust and practical natural language processor, but the guiding principle throughout has been extensibility, both of the grammar, and of the applicability to various spheres of computation.

ACKNOWLEDGEMENT

Grateful acknowledgement is given to two brave souls, Steve Gadol and Bob Kanefsky, who helped give this system some of its credibility by implementing the actual hook-up with HIRE. Thanks are also due Robert Filman and Bert Raphael for helpful comments on an early version of this paper. And a special thanks is due Richard Weyhrauch, for encouragement, wise advice, and comfort in times of debugging.

APPENDIX

This appendix lists some sentences that are actually translated into HIRE and answered by the current system. Declarative sentences presented to the system are evaluated with respect with their truth value in the usual way, and thus also function as queries.

SIMPLE SENTENCES

1. HP employs Egon.
2. Egon works for HP.
3. HP offered Montague the position.
4. HP gave Montague a job.
5. Montague got a job from HP.
6. Montague's job is at HP
7. HP's offer was to Capulet.
8. Montague had a meeting with Capulet.
9. Capulet has an offer from Xerox.
10. Capulet is competent.

IMPERATIVES AND QUESTIONS

11. Find the programmers in CRC
    who attended the meeting.
12. How many applicants for the
    position are there?
13. Which manager interviewed Capulet?
14. Whose job did Capulet accept?
15. Who is a department manager?
16. Is there a LISP programmer
    who Xerox hired?
17. Whose job does Montague have?
18. How many applicants
    did Capulet interview?

RELATIVE CLAUSES

19. The professor whose student Xerox
    hired visited HP.
20. The manager Montague met with hired
    the student who attended Berkeley.

NOUN-NOUN COMPOUNDS

21. Some Xerox programmers visited HP.
22. Montague interviewed a job applicant.
23. Who are the department managers?
24. How many applicants have a LISP
    programming background?

COORDINATION

25. Who did Montague interview and visit?
26. Which department's position did
    every programmer and a manager
    from Xerox apply for?

PASSIVE AND EXISTENTIAL SENTENCES

27. Egon was interviewed by Montague.
28. There is a programmer
    who knows LISP in CRC.

INFINITIVAL COMPLEMENTS

29. Montague managed to get a job at HP.
30. HP failed to hire a programmer
    with Lisp programming background.

REFERENCES

Barwise, Jon, and John Perry. 1981. "Situations and attitudes." *Journal of Philosophy* 78, 668-692.

Cooper, Robin. 1975. *Montague's Semantic Theory and Transformational Syntax.* Doctoral dissertation, University of Massachusetts, Amherst.

Fillmore, Charles. 1968. "The Case for Case." In Bach, Emmon and Robert Harms. *Universals in Linguistic Theory.* New York: Holt, Rinehart and Winston.

Filman, Robert E., John Lamping, and Fanya Montalvo. 1982. "Metalanguage and Metareasoning." Submitted for presentation at the AAAI National Conference on Artificial Intelligence, Carnegie-Mellon University, Pittsburgh, Pennsylvania.

Gallaire, Hervé, and Jack Minker, eds. 1978. *Logic and Data Bases.* New York: Plenum Press.

Gallaire, Hervé, Jack Minker, and Jean Marie Nicolas, eds. 1981. *Advances in Data Base Theory.* New York: Plenum Press.

Gazdar, Gerald. 1981. "Unbounded Dependencies and Coordinate Structure." *Linguistic Inquiry* 12, 155-184.

Gazdar, Gerald. 1982. "Phrase Structure Grammar." In Pauline Jacobson and Geoffrey K. Pullum, eds. *The Nature of Syntactic Representation.* Dordrecht: D. Reidel.

Gazdar, Gerald, Geoffrey K. Pullum, and Ivan A. Sag. In press. "Auxiliaries and Related Phenomena." *Language.*

Gazdar, Gerald, Geoffrey K. Pullum, Ivan A. Sag, and Thomas Wasow. 1982. "Coordination and Transformational Grammar". *Linguistic Inquiry* 13.

Jackendoff, Ray. 1977. $\overline{X}$ *Syntax.* Cambridge: MIT Press.

Kay, Martin. 1982. "When Metarules are not Metarules." Ms. Xerox Palo Alto Research Center.

Montague, Richard. 1970. "The Proper Treatment of Quantification in English." in Richmond Thomason, ed. 1974. *Formal Philosophy.* New Haven: Yale University Press.

Pratt, Vaughan R. 1975. "LINGOL - a progress report." *Advance Papers of the Fourth International Joint Conference on Artificial Intelligence, Tbilisi, Georgia, USSR, 3-8 September 1975.* Cambridge, MA: Artificial Intelligence Laboratory. 422-428.

Pullum, Geoffrey K. and Gerald Gazdar. 1982. Natural languages and context-free languages. *Linguistics and philosophy* 4.

Sag, Ivan A. 1982. "Coordination, Extraction, and Generalized Phrase Structure Grammar." *Linguistic Inquiry 13.*

Weyhrauch, Richard W. 1980. "Prolegomena to a theory of mechanized formal reasoning." Artificial Intelligence, 1, pp. 133-170.