IMPROVED PORTABILITY AND PARSING THROUGH INTERACTIVE ACQUISITION OF SEMANTIC INFORMATION[†]

François-Michel Lang and Lynette Hirschman

Paoli Research Center, UNISYS
P. O. Box 517, Paoli, PA 19301

ABSTRACT

This paper presents SPQR (Selectional Pattern Queries and Responses), a module of the PUNDIT text-processing system designed to facilitate the acquisition of domain-specific semantic information, and to improve the accuracy and efficiency of the parser. SPQR operates by interactively and incrementally collecting information about the semantic acceptability of certain lexical co-occurrence patterns (e.g., subject-verb-object) found in partially constructed parses. The module has proved to be a valuable tool for porting PUNDIT to new domains and acquiring essential semantic information about the domains. Preliminary results also indicate that SPQR causes a threefold reduction in the number of parses found, and about a 40% reduction in total parsing time.

1. INTRODUCTION

A major concern in designing a natural-language system is portability: It is advantageous to design a system in such a way that it can be ported to new domains with a minimum of effort. The level of effort required for such a port is considerably simplified if the system features a high degree of modularity. For example, if the domain-independent and domain-specific components of a system are clearly factored, only the domain-specific knowledge bases need be changed when porting to a new domain. Even if a system demonstrates such separation, however, the problem remains of ***acquiring*** this domain-specific knowledge.

[†]This work has been supported in part by DARPA under contract N00014-85-C-0012, administered by the Office of Naval Research, and in part by National Science Foundation contract DCR-85-02205, as well as by Independent R&D funding from System Development Corporation, now part of Unisys Corporation.

One obvious benefit of acquiring domain-specific semantic information is rejecting parses generated by the syntactic component which are semantically anomalous. Using domain knowledge to rule out semantically anomalous parses is especially important when parsing with large, broad-coverage grammars such as ours: Our Prolog implementation of Restriction Grammar [Hirschman1982,Hirschman1985] includes about 100 grammar rules and 75 restrictions, and is based on Sager's Linguistic String Grammar [Sager1981]. It also includes a full treatment of sentential fragments and telegraphic message style. As a result of this extended coverage, many sentences receive numerous syntactic analyses. A majority of these analyses, however, are incorrect because they violate some semantic constraint.

Let us take as an example the sentence ***High lube oil temperature believed contributor to unit failure.*** Two of the parses for this sentence could be paraphrased as:

(1) The high lube oil temperature believed the contributor to the unit failure.

(2) The high lube oil temperature was believed to be a contributor to the unit failure.

but our knowledge of the domain (and common sense) tells us that the first parse is wrong, since temperatures cannot hold beliefs.

It is only because of this semantic information that we know that parse (2) is correct, and that parse (1) is not, since we cannot rule out parse (1) on syntactic grounds alone. In fact, our grammar generates the incorrect parse before the correct one, since it produces full assertion parses before fragment parses. If the syntactic component has access to semantic knowledge, however, many incorrect parses such as (1) will never be generated.

How then can we collect the necessary semantic information about a domain? One traditional approach involves analysing a corpus of texts by hand, or perhaps even simply relying on one's intuitive knowledge of the domain in order to gather information about what relations can hold among domain entities. Several obvious drawbacks to these approaches are that they are time-consuming, error-prone, and incomplete. A more robust approach would be to use (semi-) automated tools designed to collect such information by cataloguing selectional patterns found in correct parses of sentences.

However, our reasoning appears circular: The desired domain-specific information can only be obtained from analyses of correctly parsed sentences, but our goal is to restrict the parser to these correct analyses precisely by using this domain knowledge. In the example above, we need the semantic knowledge to rule out the first parse; but it is only by knowing that this parse is semantically anomalous that we can obtain the selectional information about the domain.

One way to avoid this circularity is to bootstrap into a state of increasingly complete domain knowledge. We have implemented in SPQR such a bootstrapping process by incrementally collecting and storing domain-specific data gathered through interaction with the user. The data are in the form of selectional constraints expressed as allowable and unallowable syntactic co-occurrence patterns. All the data collected while parsing a set of sentences can then be used to help guide the parser to correct analyses and to decrease the search space traversed during future parsing. As the system's semantic knowledge becomes increasingly rich, we can expect it to demonstrate some measure of learning, since it will produce fewer incorrect analyses and present fewer queries to the user about the validity of syntactic patterns.

A number of systems have been developed to assist the user in acquiring domain-specific knowledge, including TELI [Ballard1986], TEAM [Gross1983], KLAUS [Hendrix1980], ASK [Thompson1983] and [Thompson1985], TQA [Damerau1985] and IRACQ [Moser1984,Ayuso1987]. Related work has also been reported in [Tomita1984], as well as in [Grishman1986] and [Hirschman1986a]. The work described here differs from these previous efforts in several ways:[1]

- Since PUNDIT is not a natural-language interface or a database front end, but rather a full text-processing system, sentences analyzed by PUNDIT are taken from corpuses of naturally-occurring texts. The semantic information gathered is therefore empirically or statistically based, and not derived from sentences generated by a user.
- The elicitation of information from the user follows a highly structured, data-driven approach, yielding results which should be more reproducible and consistent among users.
- Many systems have a clearly defined knowledge-acquisition phase which must be completed before the system can be effectively used or tested. We have chosen instead to adopt a paradigm of incremental knowledge acquisition.

Our incremental approach is based on the assumption that gathering complete knowledge about domain is an unattainable ideal, especially for a system which performs in-depth analysis of texts written in technical sublanguages: Even if one could somehow be assured of acquiring all conceivable knowledge about a domain, the system's omniscience would be transient, since the technical fields themselves are constantly changing, and thus require modifications to one's knowledge base. An incremental acquisition method therefore allows us to start from an essentially empty knowledge base. Each sentence parsed can add information about the domain, and the system thereby effectively bootstraps itself until its knowledge about the selectional patterns in a domain approaches completeness.

In this paper we present SPQR, the component of the PUNDIT[2] text-understanding system which is designed to acquire domain-specific selectional information [Lang1987]. We present in Section 2 the methodology we have adopted to collect and use selectional patterns, and then give in Section 3 some examples of the operation of our

[1]See [Ballard1986] for a detailed and informative comparison of TELI, TEAM, IRACQ, TQA, and ASK.

[2]PUNDIT (Prolog UNDerstands Integrated Text) is implemented in Quintus Prolog, and has been described in [Hirschman1985] and [Hirschman1986b] (syntax), [Palmer1986] (semantics), [Dahl1986] (discourse), and [Passonneau1986] (temporal analysis).

module. We conclude by presenting some experimental results and discussing some future plans to extend the module.

SPQR has been used in analysing texts in three domains: casualty reports (CASREPs) dealing with mechanical failures of ***starting air compressors*** (SACs are a component of a ship's engine), queries to a Navy ships database, and Navy sighting messages (RAINFORMs).

2. METHODOLOGY

The essential feature of our parser which facilitates the collecting of syntactic patterns is the INTERMEDIATE SYNTACTIC REPRESENTATION (ISR) produced by the syntactic analyser. The ISR is the result of regularising the surface syntactic structure into a canonical form of operators and arguments. Since there are only a limited number of structures which can appear in an ISR, we have been able to write a program to analyse the ISR and examine the syntactic patterns as they are generated.

```
[past,repair,
    [tpos(the),
        [nvar([engineer,singular,_])]],
    [tpos(the),
        [nvar([sac,singular,_])],
        adj([pastpart,break])]]
```

Figure 1: ISR for the sentence
The engineer repaired the broken sac

A brief note about the implementation: Since the ISR is represented as a Prolog list, the program which analyses it was written as a definite-clause grammar and has the flavor of a small parser. As a sample ISR, we present in Figure 1 the regularised representation of the obvious parse for the sentence ***The engineer repaired the broken sac*** (pretty-printed for clarity). At the top level, the ISR consists of the main verb (preceded by its tense operators), followed by its subject and object. The ISR of a noun phrase contains first the determiner (labelled TPOS), then the head noun, (the label NVAR stands for "noun or variant"), and finally any nominal modifiers. Note that part of the regularisation performed by the ISR is morphological, since the actual lexical items appearing in the ISR are represented by their root forms. Hence ***broken*** in the input sentence is regularised to ***break*** in the ISR, and ***repaired*** in the input sentence appears in the ISR simply as ***repair***.

SPQR is invoked by two restrictions which are called after the BNF grammar has assembled a complete NP (and constructed the ISR for that NP), and after it has assembled a complete sentence (and constructed its ISR). The program operates by presenting to the user a syntactic pattern (either a head-modifier pattern or a predicate-argument pattern) found in the ISR, and querying him/her about the acceptability of that pattern.

For each of the basic types of patterns which the program currently generates, the chart in Table 1 shows that pattern's components, an example of that pattern, and a sentence in which the pattern occurs. When presented with a syntactic pattern such as those in the chart in Table 1, the user can respond to the query in one of two ways, depending on the semantic compatibility of the predicate and arguments (e.g., in the case of an SVO pattern) or of the head and modifiers (e.g., for an ADJ pattern) contained in the pattern. If the pattern describes a relationship that can be said to hold among domain entities (i.e., if the pattern occurs in the sublanguage), the user accepts the pattern, thereby classifying it as good. The analysis of the ISR and the parsing of the sentence are then allowed to continue. If, however, the pattern describes a relationship among domain entities that is not consistent with the user's domain knowledge or with his/her pragmatic knowledge (i.e., if the pattern cannot or does not occur in the sublanguage) the user rejects it, thereby classifying it as bad, and signalling an incorrect parse. This response causes the restriction which checks selection to fail, and as a result, the parse under construction is immediately failed, and the parser backtracks.

As the user classifies these co-occurrence patterns into ***good patterns*** and ***bad patterns***, they are stored in a pattern database which is consulted before any query to the user is made. Thus, once a pattern has been classified as good or bad, the user is not asked to classify it again. If a pattern previously classified as bad by the user is encountered in the course of analyzing the ISR, SPQR consults the database, recognises that the pattern is bad, and automatically fails the parse being assembled. Similarly, if a pattern previously

TABLE 1: Selectional Patterns

PATTERN	COMPONENTS	EXAMPLE
(1) SVO	subject, main verb, object	inspection reveal particle
INSPECTION of lube oil filter REVEALED metal PARTICLES.		
(2) ADJ	adjective, head*	normal pressure
Troubleshooting revealed NORMAL sac lube oil PRESSURE.		
(3) ADV	head, adverb	decrease rapidly
Sac air pressure DECREASED RAPIDLY to 5.7¼ psi.		
(4) CONJ	conjunct$_1$, conjunction, conjunct$_2$	pressure and temperature
Troubleshooting revealed normal PRESSURE AND TEMPERATURE.		
(5) NOUN-NOUN	noun modifier, head	valve part
VALVE PARTS excessively corroded.		
(6) PREP	head, prep, object	disengage after alarm
DISENGAGED immediately AFTER ALARM.		
(7) PREDN	noun, predicate nominal	capability necessity
Alarm CAPABILITY is a NECESSITY.		

*We use "head" throughout the chart to denote the head of a construction in which a modifier appears. The head can simply be thought of as that word which the modifier modifies.

recorded as good is encountered, SPQR will recognise that the pattern is good simply by consulting the database, and allow the parsing to proceed.

The selectional mechanism as described so far deals only with lexical patterns (i.e., patterns involving specific lexical items appearing in the lexicon). However, we have implemented a method of generalising these patterns by using information taken from the domain ***isa*** (generalisation/specialisation) hierarchy to construct semantic class patterns from the lexical patterns. After deciding whether a given pattern is good or bad, the user is asked if the relation described by the pattern can be generalized. In presenting this second query, SPQR shows the user all the super-concepts of each word appearing in the pattern, and asks for the most general super-concept(s), if any, for which the relation holds.

Let us take as an example the noun-noun pattern generated by the compound nominal ***oil pressure***. While parsing a sentence containing this expression, the user would accept the noun-noun pattern [***oil, pressure***]. The program will then show the user in hierarchically ascending order all the generalizations for ***oil*** (***fluid, physical_object***, and ***root_concept***), and all the generalisations for ***pressure*** (***scalar_quantity, object_property, abstract_object***, and again ***root_concept***). The user can then identify which of those super-concepts of ***oil*** and ***pressure*** can form a semantically acceptable compound nominal. In this case, the correct generalization would be [***fluid, scalar_quantity***], because

- The fluids in the domain are oil, air, and water; the scalar quantities are pressure and temperature; and it is consistent with the domain to speak of the pressure and the temperature of oil, air, and water.
- We cannot generalise higher than ***fluid*** since it would be semantically anomalous to speak of "physical_object pressure" for every ***physical_object*** in the domain (e.g., one

would not speak of ***connecting_pin pressure*** or ***gearbox pressure***).

- We cannot generalize higher than ***pressure*** since ***shape*** is also an ***object_property***, and it would be infelicitous to speak of ***oil shape***.

As with the lexical-level patterns, the user's generalizations are stored for reference in evaluating patterns generated by other sentences. The obvious advantage of storing not just lexical patterns but also semantic patterns is the broader coverage of the latter: Knowing that the semantic class pattern [***fluid, pressure***] is semantically acceptable provides much more information than knowing only that the lexical pattern [***oil, pressure***] is good.

3. SOME (SIMPLIFIED) EXAMPLES

As we mentioned earlier, multiple syntactic analyses which can only be disambiguated by using semantic information abound in our corpuses because of the telegraphic and fragmentary nature of our texts. This ambiguity has two principal causes:

(1) A sentence which parses correctly as a fragment can often be parsed as a full assertion as well.

(2) Determiners are often omitted from our sentences, thus making it difficult to establish NP boundaries.

Since such syntactically degenerate sentences will generally contain fewer syntactic markers than full, non-telegraphic English sentences, they are characterised by correspondingly greater ambiguity. We now present an example of the use of selection to rule out a semantically anomalous assertion parse in favor of a correct fragment reading.[3] Consider the sentence ***Loss of second installed sac.*** In the correct analysis, the sentence is parsed is a noun string fragment; however, another reading is available in which the sentence is analyzed as a full assertion, with ***loss of second*** as the subject, ***installed*** as main verb, and ***sac*** as direct object. A paraphrase of this parse might be ***The loss of a second installed the sac.*** But this analysis is semantically completely anomalous for several reasons, but most notably because it makes no sense to say that the loss of a second can cause a sac (or anything else) to be installed. Since our parser tries assertion parses before fragment parses, the incorrect reading of this sentence is produced first. In generating the assertion parse, the parser encounters the SVO pattern [***loss, install, sac***], and queries the user as follows:

```
<SVO> pattern : loss install sac
```

This query asks if a loss can install a sac in this domain, or if a domain expert would ever speak of a loss installing a sac. Since it is nonsensical to speak of a loss installing a sac, the correct response to SPQR's query in this case is to reject the pattern, causing the assertion parse to fail after the module elicits the appropriate generalizations of the pattern.

In order to generalize the pattern, the user is shown all the the super-ordinates of ***loss*** and ***sac***, and asked to generalize the anomalous SVO pattern [***loss, install, sac***]. The super-concepts of ***loss*** are ***failure***, ***problem***, ***event***, ***abstract_object***, and ***root_concept***. The super-concepts of ***sac*** are ***unit***, ***mechanical_device***, ***system_component***, ***physical_object***, and ***root_concept***. Since nothing that is an abstract object can install anything at all, the correct generalization would be [***abstract_object, install, root_concept***]. Since the user's response to the original prompt labelled the pattern as bad, the assertion parse under construction then fails, and the parser backtracks.

An especially convoluted example of assertion-fragment ambiguity is found in the sentence ***Experienced frequent losses of pressure following clutch engage command.*** In the correct reading (which is again a fragment), the subject is elided, the main verb is ***experienced***, and the direct object is the ***frequent losses of pressure*** (in this parse, ***following clutch engage command*** functions as a sentence adjunct, with ***clutch engage command*** as a compound nominal). However, in another reading generated by our parser, the subject is ***experienced frequent losses of pressure following clutch***, the main verb is ***engage***, and ***command*** is the direct object. This reading would fail selection at the SVO level (if not sooner) because the SVO pattern [***loss, engage,***

[3]In this simplified explanation, we present only the SVO pattern. In actual parsing of this sentence, however, additional patterns would be generated from the NP level.

TABLE 2: Statistical Summary of 31 Sentences

PARSING INFORMATION	WITH SPQR	W/OUT SPQR
# of sentences receiving a correct parse	31	29
# of sentences receiving a correct FIRST parse	30	17
# of sentences receiving more than one parse	8	22
average # of parses found per sentence	1.45	4.66
average correct parse number*	1.10	2.45
average search focus to reach correct parse	19.60	24.48
average search focus in generating all parses	38.67	51.74
average time taken (seconds) to reach correct parse†	35.92	56.18
average time taken (seconds) in generating all parses†	81.63	125.94

SEARCH FOCUS RATIO TO CORRECT PARSE = 0.80
(with SPQR / without SPQR)

SEARCH FOCUS RATIO TO ALL PARSES = 0.75
(with SPQR / without SPQR)

TIMING RATIO TO CORRECT PARSE = 0.64

TIMING RATIO TO COMPLETION = 0.65

NEW CORRECT PARSES FOUND USING SPQR = 2

NEW CORRECT FIRST PARSES FOUND USING SPQR = 13

*That is, which parse, on the average, was the correct one.
†SPQR has not yet been optimized.

command] is anomalous for two reasons: The subject of ***engage*** cannot be an abstract concept such as ***loss***, and the object of ***engage*** must be a machine part.

4. EXPERIMENTAL RESULTS

The experimental results we present here are based on a sample of 31 sentences from one of our CASREP corpuses, each of which was parsed with and without invoking SPQR. We compare results obtained without using the selectional module to results obtained with the parser set to query the user about selectional patterns (starting from an empty pattern database). The chart in Table 2 summarizes the results for the 31 sentences.

One of the statistics presented in Table 2 is the SEARCH FOCUS, which is a measure of the efficiency of the parser in either reaching the correct parse of a sentence, or generating all possible parses. It is equal to the ratio of the number of nodes attached to the parse tree in the course of parsing (and possibly detached upon backtracking),[4] to the number of nodes in the completed, correct parse tree. Thus a search focus of 1.0 in reaching the correct parse would indicate that for every (branching) grammar rule tried, the first option was the correct one, or, in other words, that the parser had never backtracked.

The first line of the Table 2 chart deserves some explanation. One might wonder how a mechanism designed in part to ***rule out*** parses can actually ***produce*** a correct analysis for a sentence where none had been available without the module. The explanation is the COMMITTED DISJUNCTION mechanism we have implemented in our parser in order to reduce the (often spurious) ambiguity caused by allowing both full sentential and fragmentary readings. This pruning of the search space is most apparent when the parser is turned loose and set to generate all possible parses, as it was when we gathered the statistics summarized above. Recall that our parser tries full assertion parses before fragment parses. The effect of the COMMITTED DISJUNCTION mechanism is to commit the parser to produce only assertion parses (and no fragment parses) if an assertion parse is found. Fragment parses are tried only if no assertion parse is available. Thus no fragment reading will ever be generated for a sentence which can be analysed as both an assertion and a fragment. This has proved to be the correct behavior in a majority of the texts we have analysed. However, a fragment which can also be analysed as an assertion will never receive a correct parse unless all assertion parses can be blocked using selection. Thus it is possible for selection to make available a correct syntactic analysis where none would be available without selection.

5. FUTURE PLANS

Our ultimate goal is to integrate SPQR with the domain model and the semantic component that maps syntactic constituents into predicates and associated thematic roles. At present, these components are developed independently. Our aim is to link these components in order to maintain consistency and facilitate updating the system. For example, if semantic rules exist to fill thematic roles of a given predicate, we should be able to ***derive*** a set of "surface" selectional patterns consistent with the underlying semantics. Similarly, given a set of selectional patterns, we should be able to suggest a (set of) semantic rule(s) consistent with the observed selection. In addition, if a word encountered in parsing is not represented in the domain model, it should be possible to suggest where the word should fit in the model, based on similarity to previously observed patterns. If, for example, in the CASREP domain, we encounter a sentence such as ***The widget broke***, but widgets do not appear in our domain model, the system would check for any patterns of the form [***X, break***]; if it finds such a pattern, e.g., [***machine_part, break***], the system can then suggest that ***widget*** be classified as a machine part in the domain model. If the user concurs, ***widget*** would then automatically be entered into the model.

In addition to the above work, which is already underway, we plan to improve the user interface, to measure the rate at which selectional patterns are acquired, and to investigate the use of selectional patterns in developing a weighting algorithm based on frequency of occurrence in the domain.

5.1. The User Interface

In the current implementation, the questions which the program asks the user are phrased in terms of grammatical categories, and are thus tailored to users who know what is meant by such terms as "SVO" and "noun-noun compounds". As a result, only linguists can be reasonably expected to make sense of the questions and provide meaningful answers. Our intended users, however, are not linguists, but rather domain experts who will know what can and cannot be said in the sublanguage, but who cannot be expected to reason in terms of grammatical categories. Deciding how to phrase questions designed to elicit the desired information is a difficult problem. Our first attempt will be to paraphrase the pattern. E.g., for the SVO pattern [***loss, install, sac***], the query to the user would be something like "Can a loss install a sac?".

[4]Another way to interpret this figure is that it represents the number of grammar rules tried.

We are also examining what kind of knowledge the user must draw upon in order to answer the system's questions. Users' answers are usually based on a combination of commonsense knowledge (e.g., losses cannot install things) and domain-specific information. In certain cases, however, the user can be called upon to make fine linguistic distinctions. For example, in the sentence ***Sac disengaged immediately after alarm***, does the adverb ***immediately*** modify the verb ***disengaged***, or the prepositional phrase ***after alarm***? Most users, and even trained linguists familiar with the domain, find it difficult to provide definitive answers to such questions, because there is often no definitively correct answer. In this case, the adverbial attachment would seem to be genuinely ambiguous. It would be helpful to recognise patterns which a user cannot be reasonably expected to pass judgment on, and not generate queries about these, perhaps allowing them to succeed by default.

5.2. Measuring the System's Learning

As more sentences are parsed and more patterns are classified, we can expect the system to grow "smarter" in the sense that it will ask the user increasingly fewer questions. Eventually, the system should reach a state of reasonably complete domain knowledge, at which time few unknown patterns would be encountered, and the user would almost never be queried. We do not know how many sentences SPQR would have to examine before attaining such a plateau, but an estimate would be in the range of 500 to 1000 [Grishman1986]. We plan to measure the decrease in the frequency of queries to the user as a function of the number of sentences parsed and the number of patterns collected in order to evaluate the system's learning. This will enable us to determine the feasibility of using this technique to bootstrap into a new domain.

5.3. Preference-Based Parsing

A long-term goal is to implement a parsing algorithm based on preference rather than on the current success/failure paradigm. This would allow the system to use statistical information on the frequency of observed patterns as one factor in weighting. Frequently occurring patterns would be assigned greater weight than unknown patterns, and bad patterns would detract from the overall weighting. This would allow the system to make intelligent "guesses" about parsing without constantly querying the user.

ACKNOWLEDGMENTS

We would like to thank Marcia Linebarger for her helpful comments and insightful suggestions.

REFERENCES

[Ayuso1987]
Damaris M. Ayuso, Varda Shaked, and Ralph M. Weischedel, An Environment for Acquiring Semantic Information. In ***Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics***, Stanford, California, July 1987, pp. 32-40.

[Ballard1986]
Bruce W. Ballard and Douglas E. Stumberger, Semantic Acquisition in TELI: A Transportable, User-Customized Natural Language Processor. In ***Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics***, New York, NY, July 1986, pp. 20-29.

[Dahl1986]
Deborah A. Dahl, Focusing and Reference Resolution in PUNDIT. In ***Proceedings of the Fifth National Conference on Artificial Intelligence***, Philadelphia, PA, 1986, pp. 1083-1088.

[Damerau1985]
Fred J. Damerau, Problems and Some Solutions in Customization of Natural Language Database Front Ends. ***ACM Transactions on Office Information Systems*** **3**(2), April 1985, pp. 165-184.

[Grishman1986]
Ralph Grishman, Lynette Hirschman, and Ngo Thanh Nhan, Discovery Procedures for Sublanguage Selectional Patterns: Initial Experiments. ***Computational Linguistics*** **12**(3), 1986, pp. 205-215.

[Gross1983]
Barbara J. Gross, TEAM: A Transportable Natural-Language Interface System. In ***Proceedings of the Conference on Applied Natural Language Processing***, Santa Monica, CA, February 1983, pp. 39-45.

[Hendrix1980]
Gary G. Hendrix, Mediating the Views of Databases and Database Users. In ***Proceedings of the Workshop on Data Abstraction, Databases, and Conceptual Modelling***, Pingree Park, CO, June 1980, pp. 131-132.

[Hirschman1982]
Lynette Hirschman and Karl Puder, Restriction Grammar in Prolog. In ***Proceedings of the First International Logic Programming Conference***, M. Van Caneghem (ed.), Association pour la Diffusion et le Developpement de Prolog, Marseille, 1982, pp. 85-90.

[Hirschman1985]
Lynette Hirschman and Karl Puder, Restriction Grammar: A Prolog Implementation. In ***Logic Programming and its Applications***, D.H.D. Warren and M. VanCaneghem (ed.), 1985.

[Hirschman1986a]
Lynette Hirschman, Discovering Sublanguage Structures. In ***Sublanguage: Description and Processing***, R. Kittredge and R. Grishman (ed.), Lawrence Erlbaum Assoc., Hillsdale, NJ, 1986.

[Hirschman1986b]
Lynette Hirschman, Conjunction in Meta-Restriction Grammar. ***Journal of Logic Programming* 4**, 1986, pp. 299-328.

[Lang1987]
François-Michel Lang, A User's Guide to the Selection Module, Logic-Based Systems Technical Memo No. 68, Paoli Research Center, Unisys, Paoli, PA, October, 1987.

[Moser1984]
M. G. Moser, Domain Dependent Semantic Acquisition. In ***Proceedings of the First Conference on Artificial Intelligence Applications***, IEEE Computer Society, Denver, CO, December, 1984, pp. 13-18.

[Palmer1986]
Martha S. Palmer, Deborah A. Dahl, Rebecca J. [Passonneau] Schiffman, Lynette Hirschman, Marcia Linebarger, and John Dowding, Recovering Implicit Information. In ***Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics***, Columbia University, New York, August 1986.

[Passonneau1986]
Rebecca J. Passonneau, A Computational Model of the Semantics of Tense and Aspect, Logic-Based Systems Technical Memo No. 43, Paoli Research Center, Unisys, Paoli, PA, November, 1986.

[Sager1981]
Naomi Sager, ***Natural Language Information Processing: A Computer Grammar of English and Its Applications***. Addison-Wesley, Reading, Mass., 1981.

[Thompson1983]
Bozena H. Thompson and Frederick B. Thompson, Introducing ASK, A Simple Knowledgeable System. In ***Proceedings of the Conference on Applied Natural Language Processing***, Santa Monica, CA, February 1983, pp. 17-24.

[Thompson1985]
Bozena H. Thompson and Frederick B. Thompson, ASK is Transportable in Half a Dozen Ways. ***ACM Transactions on Office Information Systems* 3**(2), April 1985, pp. 185-203.

[Tomita1984]
Masaru Tomita, Disambiguating Grammatically Ambiguous Sentences by Asking. In ***Proceedings of the 22nd Annual Meeting of the Association for Computational Linguistics***, Stanford, CA, July 1984, pp. 476-480.