A Syntactic Framework for Semantic Interpretation

Carolyn Penstein Rosé
Learning Research and Development Center
University of Pittsburgh
Pittsburgh, PA 15260
rosecp@pitt.edu

1 Introduction

This paper describes the AUTOSEM syntactic framework for semantic interpretation and the CARMEL robust understanding engine in which it operates. This work has been developed as part of an effort to build tools for authoring broad coverage natural language understanding interfaces for intelligent tutoring systems. Its purpose is to provide a solid foundation for interpretation that will facilitate rapid development of domain specific interfaces [Freedman et al., 2000].

Rather than having its roots in a single syntactic theory, the underlying design for the AUTOSEM interpretation framework grows out of a desire to benefit from the the advantages of a number of alternative approaches. From a computational perspective, domain specific semantic grammars offer a number of advantages for our task. For example, they tend to produce less ambiguity since they only construct analyses relevant to a limited number of contexts. Furthermore, they constrain the parser to construct analyses covering only those portions of text that the system is prepared to interpret. However, from a development standpoint, semantic grammars are expensive to produce. In our experience roughly two person years are required to develop a solid semantic grammar for a non-trivial domain on the order of complexity of the English Spontaneous Scheduling Task [Lavie et al., 1996, Suhm et al., 1994, Woszcyna et al., 1993, Woszcyna et al., 1994]. Regardless of the syntactic theory used, separating syntactic analysis from semantic interpretation offers advantages from a development standpoint. Rather than writing a new grammar for every domain, it is only necessary to write mapping rules that transform syntactic analyses into semantic representations. Nevertheless, completely separating syntactic analysis from semantic analysis has disadvantages precisely where semantic grammars offer their advantages. With a broad coverage syntactic grammar and large-scale lexicon, parsers produce a great deal of ambiguity. Furthermore, they waste time parsing portions of text that the system does not possess the semantic knowledge to be able to interpret. Even for in-domain portions of text, such syntactic approaches construct analyses corresponding to syntactic constructions and lexical entries involving in-domain words but with word senses that are never used within that domain or do not make sense when combined with the larger context of the sentence as a whole.

In order to benefit from the advantages of both of these disparate general approaches, we have developed a syntactic framework that separates syntactic and semantic knowledge but interleaves the process of syntactic and semantic interpretation. Thus, we reap the benefits from a processing standpoint of the semantic grammar approach while also reaping the advantages from a development standpoint of the syntactic grammar approach. AUTOSEM is a lexicon driven approach to semantic interpretation similar to the Glue Language Semantics approach [Dalrymple, 1999, Dalrymple et al., 1993] in that the lexicon provides a clean interface between syntactic and semantic knowledge. Semantic constructor functions are linked into lexical entries where the entry may act as a lexical head corresponding to the associated semantic concept. Syntactic functional roles are matched to argument positions in the semantic constructor functions. Thus, as syntactic functional roles are assigned at parse time, semantic interpretation can proceed in parallel.

The AUTOSEM lexicon was built using the large scale COMLEX lexicon [Grishman et al., 1994] as a foundation. The COMLEX lexicon, made available through the Linguistic Data Consortium, offers a wealth of syntactic features for nouns, verbs, adjectives, determiners, and auxiliaries obtained through years of corpus analysis. The most important subset of these features for our purposes has been the over ninety different verb subcategorization tags, each associated with one or more specific verb case frames. Using these subcategorization tags as an interface between syntactic analysis and semantic interpretation is not straightforward, however, since each one corresponds to a wide range of surface syntactic realizations resulting from passivization and extraction transformations. Thus, each one corresponds to a number of different ways of assigning surface syntactic roles to semantic roles. Encoding a separate mapping from syntactic roles to semantic roles for each of these is not desirable from a development standpoint. We avoid this problem by only encoding a mapping from deep syntactic functional roles to semantic roles. Thus, it is the job of the syntactic grammar to construct a deep syntactic functional analysis of its input text.

Writing a grammar to compute deep syntactic functional roles is more complex than writing one that computes surface syntactic functional roles since there is both a one-to-many mapping between surface syntactic functional roles and deep syntactic functional roles as well as a one-to-many mapping between deep syntactic functional roles and surface syntactic functional roles. The complexity arises from the interaction between subcategorization case frames, passivization, and extraction. Thus, the job of maintaining the module of the grammar responsible for assigning deep syntactic functional roles to verb arguments is incredibly difficult. Managing this complexity is one contribution of the research reported in this paper. Our approach has been to encode each of the COMLEX verb subcategorization tags into a feature representation that allows the wide range of possible surface realizations to be computed automatically. Based on this computation, the grammar rules responsible for this analysis are also generated automatically. The automatic generation of grammar rules assures that if the feature representations of the subcat patters are correct, grammatical coverage will also be perfect within the limitations of the coverage of the lexicon. Maintaining the feature encodings of the subcategorization tags is arguably far simpler than maintaining the corresponding grammar rules. These and surrounding issues are discussed below.

The second contribution of this work is the handling of domain specific semantic knowledge, allowing it to be used not only at parse time, but also within a post-parsing repair stage. Thus, added robustness not possible at parse time can be achieved for free. No additional knowledge dedicated to robustness is required. In AUTOSEM, domain specific semantic knowledge is encoded declaratively within a meaning representation specification. Semantic constructor functions implemented in Lisp are compiled automatically from this specification and then linked by hand into lexical entries, as described above. The grammar, as already mentioned, allows the parser to assign deep syntactic roles to constituents. Based on these head/argument relationships, the constructor functions enforce semantic selectional restrictions and assemble meaning representation structures by composing the meaning representation associated with the constructor function with the meaning representation of each of its arguments. Thus, syntactic and semantic analyses are constructed in parallel.

Broad coverage grammatical coverage is essential for making practical systems robust. However, writing more grammar rules is not the complete answer to making an understanding system robust. In addition to the problem that some input will be blatantly ungrammatical, it is also the case that each speaker has his own idiosyncratic language patterns, some of which will inevitably not be within the grammatical coverage of any grammar [Lehman, 1989]. If it were possible to write a grammar covering every hypothetical speaker's idiosyncratic linguistic patterns, the resulting system would create so much ambiguity that it would be intractable. Thus, we rely on flexibility in our processing model to achieve an acceptable level of robustness. We view syntactic analysis as a vehicle for guiding semantic interpretation, not as an end in itself. Thus, we are willing to sacrifice syntactic correctness when necessary in order to arrive at a plausible semantic interpretation.

AUTOSEM first attempts to construct analyses that satisfy both syntactic and semantic well-formedness conditions. The LCFLEX parser (Rosé and Lavie, to appear), which is the heart of the CARMEL robust understanding engine, has the ability to efficiently relax syntactic constraints as needed and as allowed by its parameterized flexibility settings. For sentences remaining beyond the coverage of AUTOSEM's syntactic knowledge, AUTOSEM's repair stage relies entirely on semantic knowledge to compose the partial analyses produced by the parser using the same semantic constructor functions used at parse time. Each semantic representation built by AUTOSEM's interpretation framework contains a pointer to the constructor function that built it. Thus, each partial analysis can be treated as a constructor function with built in knowledge about how the associated partial analysis can be combined with other partial analyses in a semantically meaningful way. Genetic programming search [Koza, 1992, Koza, 1994] is used to efficiently compose the fragments produced by the parser. The function definitions compiled from the meaning representation specification allow the genetic search to use semantic constraints to make effective use of its search space. Thus, AUTOSEM operates efficiently, free of any hand coded repair rules or any knowledge specifically dedicated to repair unlike other approaches to recovery from parser failure [Danieli and Gerbino, 1995, Van Noord, 1997, Kasper et al., 1999].

2 The Architecture of the CARMEL Robust Understanding Engine

Because our goal is to produce dialogue based tutoring systems that can effectively interact with students through natural language, robustness and efficiency are essential. While a solid syntactic foundation for interpretation is fundamental to achieving our ultimate goal of providing reusable components for understanding, the effective interpretation of language input that falls outside of the syntactic coverage of the system has been of utmost importance. Because the output from the language understanding component is used by the system to evaluate the student's knowledge and for determining how best to respond to the student's contribution, the semantic interpretation of the student's utterance is of primary importance, while the correctness of the syntactic analysis of the student's utterance is secondary. Furthermore, in the case of ungrammatical student input, a theoretically correct syntactic analysis would be impossible. Thus, the CARMEL approach embodies the preference for producing analyses that satisfy both syntactic and semantic well-formedness conditions. When this is impossible, syntactic constraints are relaxed at parse time as needed and as allowed by the parser's parameterized flexibility parameters [Rosé and Lavie, pear]. When the parser fails to construct an analysis covering the student's whole contribution, parsed fragments are combined using only semantic well-formedness conditions to guide the search [Rosé, 2000].

2.1 Historical Context

The underlying philosophy behind the design of the CARMEL understanding engine has its roots in the ROSE system for robust interactive interpretation [Rosé, 1997, Rosé, 1999]. The ROSE approach was based on an analogy between human/computer interaction through a natural language interface and language interaction between speakers of different languages with a small shared language base. Humans who share a very small language base are able to communicate when the need arises by simplifying their speech patterns and negotiating until they manage to transmit their ideas to one another

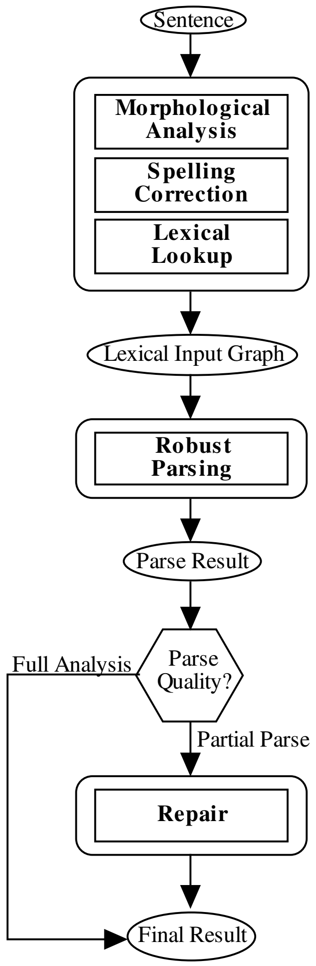

Figure 1: **CARMEL architectural overview**

[Hatch, 1983]. As the speaker is speaking, the listener "throws his net" in order to catch those fragments of speech which are comprehensible to him which he then attempts to fit together semantically. His subsequent negotiation with the speaker builds upon this partial understanding.

Based on this collaborative model of communication, the ROSE approach interpreted extragrammatical input in two stages. The first stage, Repair Hypothesis Formation, was responsible for assembling a set of hypotheses about the meaning of the ungrammatical utterance. This stage was itself divided into two steps, Partial Parsing and Combination. The Partial Parsing step was similar to the concept of the listener "casting his net" for comprehensible fragments of speech. A robust parser was used to obtain an analysis of islands of the speaker's sentence in cases where it was not possible to obtain an analysis for the entire sentence. In the Combination step, the fragments from the partial parse were assembled into a set of alternative meaning representation hypotheses. A genetic programming approach was used to search for different ways to combine the fragments in order to avoid requiring any handcrafted repair rules. In ROSE's second stage, Interaction with the user, the system generated a set of queries, negotiating with the speaker in order to narrow down to a single best meaning representation hypothesis. The CARMEL approach is based on the same collaborative model. However, only the Hypothesis Formation stage is addressed here.

2.2 Robust Understanding in Three Stages

Input understanding in the CARMEL understanding engine proceeds in three distinct stages, each of which has been built to be robust to noisy, extragrammatical input. The lexical preprocessing stage, which is the initial stage, is responsible for performing morphological analysis on each word in the input sentence and retrieving all lexical entries that match the root form of each word from the lexicon. Subsequently, the parsing stage is responsible for performing a syntactic and semantic analysis with these lexical entries in so far as the parser's flexibility settings will allow. When necessary, a repair stage is responsible for assembling the pieces of a fragmentary parse when no complete parse is possible.

2.2.1 The Lexical Preprocessing Stage

The key to robustness in the lexical preprocessing stage is the coordination of morphological analysis, lexical lookup, and spelling correction. For each word, the morphological analyzer first constructs a list of possible segmentations of it into morphemes. For each possible segmentation, the associated lexical entries are retrieved from the lexicon. If none of the root forms from any of the possible segmentations correspond to an entry in the lexicon, a list of possible replacement words are obtained from the spelling corrector [Elmi and Evens, 1998] based on the form of the word, the previous word and the next word. Based on its internal heuristics, the spelling corrector chooses either to split the word, combine the word with a neighboring word, or replace with whole word with one or more alternative words. Thus, when the spelling corrector is allowed to return more than one hypothesis, the output may be a graph instead of an ordered list. The morphological analyzer, the lexical lookup mechanism, and the spelling corrector all use the same lexicon. Thus, after spelling correction, the lexical lookup mechanism is guaranteed to be able to retrieve lexical entries for each word in the input. This spelling corrector has previously proved to be instrumental in processing typed student input in the CIRCSIM-TUTOR project [Glass, 1999].

2.2.2 The Robust Parsing Stage

	Bad	OK	Perfect	Total Acceptable
No Robustness	47.16	11.67	41.17	52.84 (60.39)
Restarts	32.83	21.00	46.17	67.17 (76.77)
LCFLEX Skip 3	26.67	20.00	53.33	73.33 (83.80)

Table 1: **Translation Quality Evaluation**

The key to robustness in the parsing stage is the LCFLEX robust parser (Rosé and Lavie, to appear). LCFLEX belongs to a family of flexible parsing approaches designed to avoid failure when faced with extra-grammatical input. Many robust approaches to symbolic parsing relax syntax and grammar constraints. In some cases, semantic information is used to compensate for the lack of grammaticality in order to partially or completely drive the analysis process. The earliest approaches to robust parsing involved hand-coded grammar specific heuristics for selecting a subset of analyses to extend when each input word was processed [Hobbs et al., 1991, McDonald, 1993b, McDonald, 1993a, McDonald, 1992]. With these approaches, the work of making the parser robust and efficient must be redone by hand for every new grammar developed. The use of hand-coded heuristics also constrains these approaches

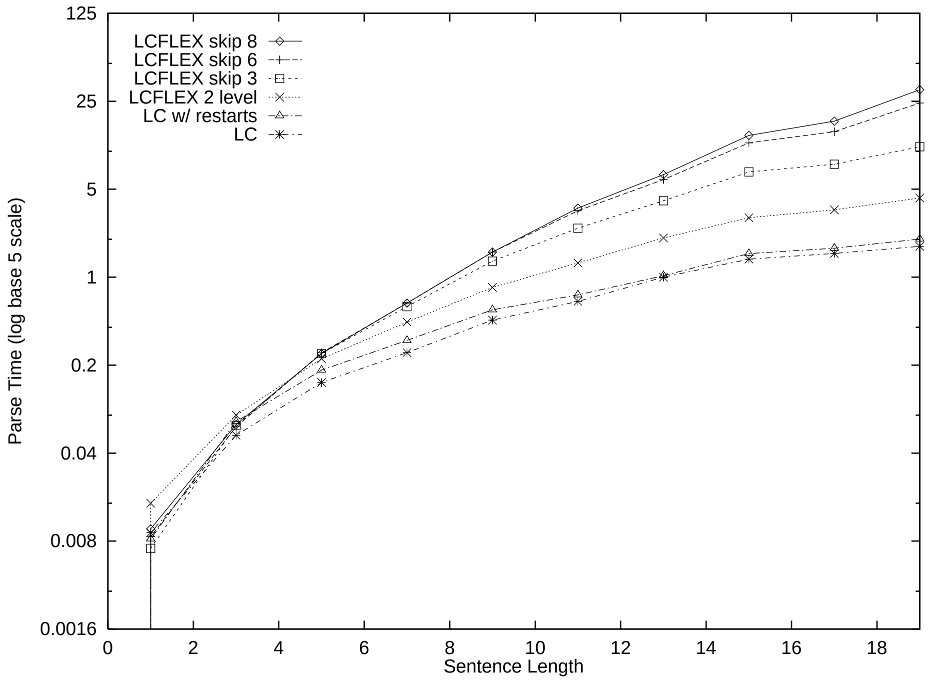

Figure 2: **Avg Parse Time vs. Sentence Length for** LCFLEX**, Various Skipping Settings**

to handle only those cases that are consistent with the simplifying assumptions represented by those heuristics.

Some recent approaches to robust parsing focus instead on shallow or partial parsing techniques [Van Noord, 1997, Abney, 1996, Ait-Mokhtar and Chanod, 1997, Worm, 1998]. Rather than attempting to construct a parse covering an entire ungrammatical sentence, these approaches attempt to construct analyses for maximal contiguous portions of the input. Thus, they address the computational expense problem by limiting the flexibility of their parsing algorithms. The result is a practical form of robust parsing not involving any grammar specific heuristics, but more shallow in its resulting analysis than could possibly be achieved using other types of flexibility. For certain applications, this level of analysis may certainly be sufficient. However, if a more complete analysis is required, additional post-parsing processing is necessary, where robustness must also be applied.

The LCFLEX parser used within the CARMEL understanding engine is a flexible left-corner parser designed for efficient, robust interpretation. Its approach to robustness was originally inspired by the earlier GLR* parser [Lavie, 1995]. LCFLEX extended the robustness capabilities of GLR* and supports various types of flexibility including word skipping, non-terminal inserting, and flexible unification. While the additional "flexibility features" offered by LCFLEX could have just as well been implemented within GLR*, our previous experience with GLR* demonstrates that these additional degrees of flexibility render GLR* intractable [Rosé and Lavie, 1997]. One attractive feature of LCFLEX is that its "flexibility features" can easily be controlled by setting global parameters. Thus, it is possible to tailor the parser's behavior to the specific needs of individual applications where it is used. The LCFLEX parser robustly applies AUTOSEM's broad coverage syntactic parsing grammar by introducing various types of flexibility only as needed. Because broad coverage syntactic grammars naturally produce a great deal of ambiguity, LCFLEX's ambiguity packing and pruning strategy as well as its statistical disambiguation make it an ideal choice for our application.

The LCFLEX approach to robust parsing has already proven to be both effective and efficient in the

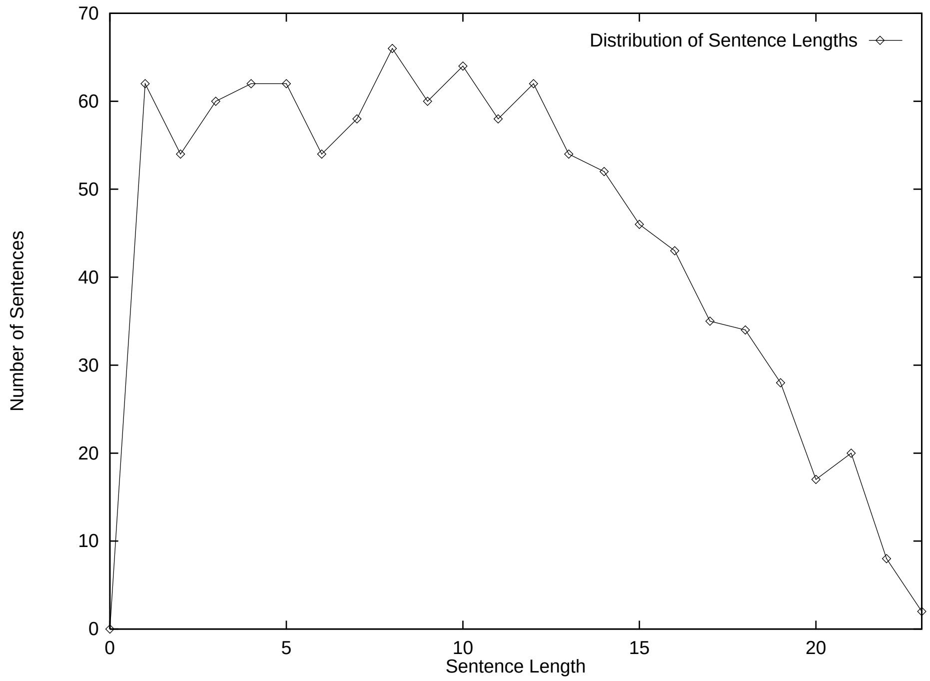

Figure 3: **Distribution of Sentence Lengths**

context of a large-scale speech-to-speech translation system. A previously reported evaluation (Rosé and Lavie, to appear) demonstrates the practical value of this approach. Here LCFLEX is evaluated using a semantic grammar developed for the spoken language ESST (English Spontaneous Scheduling Task) domain. The test set used in the evaluation contains 607 sentences of transcribed spontaneous speech from the ESST task. The distribution of sentence lengths is found in Figure 3. Of these 607 sentences, a subset of 200 sentences was selected for grading. These 200 sentences were contiguous sentences from five dialogues in the ESST corpus. We selected this set of contiguous sentences in order to ensure a realistic distribution of sentence lengths and in-domain vs. out-of-domain sentences. Each translation was graded by three independent judges, receiving a grade of Perfect, OK, or Bad. For each strategy, the number of sentences assigned each grade was totaled for each grader. These totals were then averaged to arrive at the percentages presented in Table 1. Of the 200 sentences in the test set, 25 were determined to be out of domain. Thus, the percentages in parentheses in the final column are percentages of acceptable translations out of the in-domain sentences.

We conducted a set of experiments, with varying amounts of skipping flexibility. In LCFLEX, the amount of skipping flexibility is controlled by a global skip parameter, which determines the total number of words allowed to be skipped in the process of obtaining an analysis. We tested the parser with five different settings: (1) "LC": no skipping allowed - equivalent to a non-robust left-corner parser; (2) "LC with restarts": allowing only inter-analysis skipping (of prefixes and suffixes of the input string); (3) "LCFLEX skip 3": a maximum of 3 words can be skipped within an analysis; (4) "LCFLEX skip 6": similar, with a maximum of 6 skipped words; and (5) "LCFLEX skip 8": similar, with a maximum of 8 skipped words. Average parse times as a function of sentence length for all tests appear in Figure 2. As expected, greater levels of skipping flexibility result in increasing parse times. While a skip parameter value of 3 does incur a significant time cost, parse times remain feasible for even relatively long sentences. Increasing the skip parameter above 3 did not improve the overall translation quality produced.

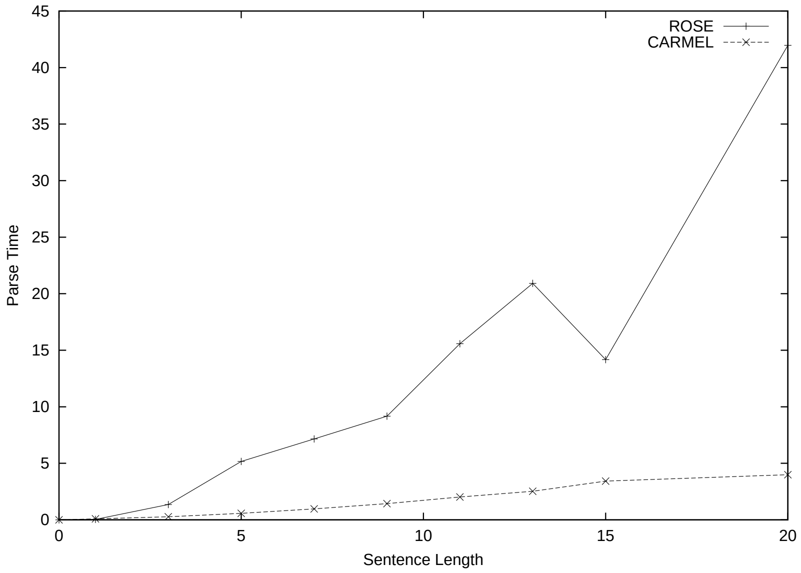

Figure 4: **Processing Times for Alternative Strategies**

Additional experiments (Rosé and Lavie, to appear) demonstrate that LCFLEX's other flexibility features also offer additional robustness, skipping offers the most economical parse-time approach to robustness.

2.2.3 The Repair Stage

When the flexibility allowed at parse time is not sufficient to construct an analysis of a sentence deviating too far from the grammar's coverage, a fragmentary analysis is passed into the repair module that quickly assembles the fragments. The same semantic constructor functions used for building semantic analyses at parse time are used in the Genetic Programming (GP) based repair stage to assemble the fragments produced by the parser. The GP based repair algorithm is described in depth in [Rosé, 2000] in comparison to the earlier ROSE approach. The evaluation reported there demonstrates that the current AUTOSEM approach operates 200 times faster than the earlier ROSE approach while achieving better performance.

The test set used for this evaluation contains 750 sentences extracted again from the ESST corpus. For both repair approaches we used the meaning representation developed for the ESST task that was used in previous evaluations of the ROSE approach [Rosé, 1997]. It consists of 260 semantic types, each expressing domain specific concepts for the appointment scheduling domain such as `busy`, `cancel`, and `out-of-town`. The ROSE meaning representation specification was easily converted to the format used in AUTOSEM. The final meaning representation structures for the first 300 sentences were then passed to a generation component, and the resulting texts were graded by a human judge not involved in developing the research reported here.

Each sentence was parsed in two different modes. In `LC w/restarts` mode, the parser was allowed to construct analyses for contiguous portions of input starting at any point in the sentence. In LCFLEX mode, the parser was allowed to start an analysis at any point and skip up to three words within the analysis. Because the AUTOSEM repair algorithm runs significantly faster than the ROSE repair algorithm, repair was attempted after every parse rather than only when a parse quality heuristic indicated a need as in the ROSE approach [Rosé, 1997]. We compared the results of both AUTOSEM and ROSE

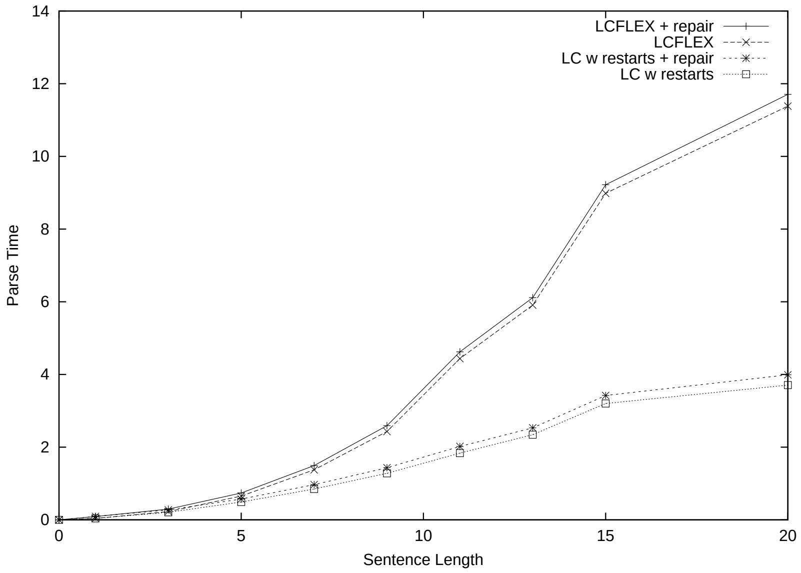

Figure 5: **Processing Times for Alternative Strategies**

	Bad	Okay	Perfect	Total Acceptable
LC w/restarts	92 (30.7%)	61 (20.3%)	147 (49.0%)	209 (69.3%)
LCFLEX	72 (24.0%)	67 (22.3%)	161 (53.7%)	228 (76.0%)
LC w/restarts + ROSE	75 (25.0%)	68 (22.7%)	157 (52.3%)	225 (75.0%)
LC w/restarts + AUTOSEM	64 (21.3%)	84 (28.0%)	152 (50.7%)	236 (78.7%)
LCFLEX + AUTOSEM	64 (21.3%)	78 (26.0%)	158 (52.7%)	236 (78.7%)

Figure 6: **Interpretation quality with and without repair**

in conjunction with the `LC w/restarts` parsing mode. The results are displayed in Figures 4 and 6. Because the ROSE approach only runs the full repair algorithm when its parse quality heuristic indicates a need and the parser returns more than one partial analysis, it only attempted repair for 14% of the sentences in the corpus. Nevertheless, although the AUTOSEM repair algorithm ran for each sentence, Figure 4 demonstrates that processing time for parsing plus repair in the AUTOSEM condition was dramatically faster on average than with ROSE. Average processing time for the ROSE algorithm was 200 times slower than that for AUTOSEM on sentences where both repair algorithms were used. In addition to the advantage in terms of speed, the AUTOSEM repair approach achieved an acceptable grade (Okay or Perfect) on approximately 4% more sentences. Both ROSE repair and AUTOSEM repair used GP to search for high quality hypotheses for how to assemble the fragments produced by the parser. However, while AUTOSEM composes semantic constructor functions whose definitions contain implicit knowledge about how they may be composed, in ROSE fragments produced by the parser are composed using instances of a single function that refers to a meaning representation specification to heuristically decide how to combine its two input fragments. While both approaches have access to the same semantic information, it is used more efficiently in AUTOSEM since using the semantic constructor functions themselves as the function set allows the genetic search to use semantic constraints to guide its hypothesis formation process on a global level rather than just locally within instances of the combination function.

Parsing in `LC w/restarts` mode plus repair was also compared with parsing in LCFLEX mode with skipping up to three words. Again, `LC w/restarts` + AUTOSEM repair achieved a slightly higher number of acceptable grades, although LCFLEX achieved a slightly higher number of Perfect grades.

On long sentences (between 15 and 20 words), LCFLEX mode required almost three times as much time as `LC w/restarts` mode plus AUTOSEM repair. This evaluation confirms our previous results that two stage approaches offer a better processing time versus robustness trade-off [Rosé, 1997].

3 Broad Coverage Linguistic Knowledge Sources

AUTOSEM makes use of two primary domain independent knowledge sources, namely a broad coverage syntactic parsing grammar and a large scale lexicon than can easily be augmented with domain specific vocabulary and idiomatic construction entries. Its single domain specific knowledge source is a meaning representation specification, allowing domain specific semantic information to be entered declaratively. Thus, applying AUTOSEM to a new domain requires only to design a meaning representation, enter domain specific vocabulary and relevant idiomatic constructions into the lexicon, and finally to link the semantic constructor functions into the lexicon.

3.1 The Meaning Representation Specification

```
(:type <*event>
 :isa (<>)
 :instances nil
 :vars (agent time polarity)
 :spec ((polarity [+/-] polarity)
        (agent <*who> agent)
        (time <*time> time)))

(:type <draw>
 :isa (<*event>)
 :instances nil
 :vars (theme)
 :spec ((frame *draw)
        (theme <*thing> theme)))

(:type [+/-]
 :isa (<>)
 :instances (+ -)
 :vars nil
 :spec nil)
```

```
(:type <magnitude>
 :isa (<*thing>)
 :instances nil
 :vars (ref-object)
 :spec ((frame *magnitude)
        (ref-object <vector> ref-object)))

(:type <velocity-vector-mag>
 :isa (<magnitude>)
 :instances nil
 :vars (timept quantity orientation object)
 :spec ((ref-object <velocity-vector>
          (<velocity-vector> timept quantity
             orientation object))))
```

Figure 7: **Sample meaning representation entries for the Newtonian Physics domain**

The meaning representation specification defines the frame based language that is used for this purpose. In particular, it defines the set of semantic types that together specify the set of frames and atomic feature values that make up the domain specific frame-based language, which slots are associated with each frame, and what range of frames and atomic feature values may fill each of those slots. When the meaning representation specification is compiled, constructor functions are created corresponding to each semantic type. As mentioned in the previous section, these semantic constructor functions are first used at parse time to build up semantic representations in parallel with syntactic ones as in the Glue Language Semantics approach [Dalrymple, 1999, Dalrymple et al., 1993]. These same constructor functions are then used in a repair stage to compose the fragments returned by the parser in the cases where the parser is not able to obtain a complete analysis for an extra-grammatical input sentence. Here we describe the knowledge sources that are part of the AUTOSEM approach. A detailed example of the AUTOSEM interpretation process is given in the Section 4..

AUTOSEM provides a simple formalism for defining meaning representations. Some sample entries are displayed in figure 7. Each entry corresponds to a semantic type and contains five fields: `:type`, `:isa`, `:instances`, `:vars`, and `:spec`. Some sample entries for the Newtonian physics domain are displayed in Figure 7. The `:type` field simply contains the name of the type. The `:vars` field contains a list of variables, each corresponding to a semantic role. The `:spec` field associates a frame and set of slots with a type. For each slot, the `:spec` field contains the name of the slot, the most general type restriction on the slot, and a specification of where the slot filler comes from. This third piece of information can be either a variable name, indicating that whatever is bound to that variable is what should fill that slot, or a function call to another semantic constructor function, allowing types to specify constraints at more than one level of embedding. Similar to the `:spec` field, the `:instances` field associates a list of atomic values with a type.

Inheritance relations are defined via the `:isa` field. Types inherit the values of each subsuming type's `:instances`, `:vars`, and `:spec` fields. If the subsumed type contains a specification for a slot in its `:spec` field that is also contained in the subsuming type, the more restrictive type restriction is maintained in the resulting type. Inheritance is computed by means of a transitive closure. An error is returned if subsuming types conflict either with each other or with the subsumed type, or if a loop is detected within the inheritance relations contained in the specification.

When each type is compiled into a constructor function, the resulting function has an argument corresponding to each variable associated with the type. The body of the function contains code for testing type restrictions on the structures instantiating the arguments and for inserting those structures into the corresponding slots in the resulting structure.

3.2 The Lexicon

```
(:morph draw
 :syntax ((cat vlex) (root draw) (vform bare)
          (irreg-past +) (irreg-pastpart +)
          (features vveryvingpast)
          (proot (*or* near on upon to at from))
          (subcat (*or* pp np-advp np-to-inf-oc
             np-for-np intrans np np-as-np advp))
          (semtag draw1))
 :semantics (draw1 <draw> (subject agent) (negation polarity) (object theme)))

(:construct (<clause>)
 :syntax ((cat clause) (semtag know1))
 :constraints (((x1 root) = (*or* have get))
               ((x1 object root) = (*or* clue idea))
               (x0 = x1)
               ((x0 root) <= know)
               ((x0 object) = *remove*)))
```

Figure 8: **Sample Lexical Entries**

As mentioned above, the AUTOSEM lexicon is built on top of the large scale COMLEX lexicon [Grishman et al., 1994] available from the Linguistic Data Consortium. The AUTOSEM lexicon contains both regular lexical entries for individual words as well as construct entries for idiomatic and otherwise non-compositional expressions. In general, we have taken a compositional approach to semantic interpretation, building up the meaning of an expression from the meanings of its constituent parts. However, idiomatic expressions are not correctly interpreted compositionally. Thus, these non-compositional expressions are represented as construct entries so that their meaning can be associated with the whole expression as a unit.

Regular lexical entries contain three fields: `:morph`, `:syntax`, and `semantics`. The `:morph` and `:syntax` fields are taken from COMLEX. The `:morph` field contains the root form of the word. The `:syntax` field contains the list of features assigned to that root form in the COMLEX lexicon. Often the `:syntax` field contains a disjunction of feature structures wherever there are multiple entries associated with a single root form in COMLEX. For example, COMLEX contains seven entries for "draw", although only one is displayed in Figure 8. To the original `:syntax` field, we have added a `semtag` feature. Semtag feature values correspond to mappings between syntactic roles and the arguments of semantic constructor functions. The `:semantics` field defines the specific mappings associated with each `semtag` feature value. Each mapping needs only be defined the first time the corresponding semtag value is used in a lexical entry.

Construct entries are used for non-compositional idiomatic expressions as well as bound collocation [Cruse, 1986], which are compositional although they display idiom-like behavior. Idioms can be seen as sharing properties both with words and with phrases. To accommodate this phenomenon, construct entries can be seen as blurring the distinction between lexical entries and grammar rules. Our formalism for expressing construct entries is equivalent in power to the formalism for representing idiomatic expressions in [van den Berg et al., 1994]. They contain a `:construct` field to replace the `:morph` field. It can contain a list of one or more words, grammar symbols, or a combination of the two. Thus, construct entries can leverage off of syntactic generalizations from the grammar and easily specify how the corresponding expression should be inflected. Construct entries contain a constraints field wherein constraints can be placed on the feature structures dominated by the grammar symbols found in the construct field. For example, the construct entry in Figure 8 represents the idiom "to have a clue". It matches against clauses. If the verb and direct object match the associated constraints, the verb root is changed to "know" and the direct object is deleted. Our design principle is one idiom, one rule. Our construct entry design makes it possible with one rule to match the great deal of variation allowed by the "have a clue" idiom: "I don't have a clue in the world", "I don't have any idea", "I haven't got the foggiest clue", "Get a clue!", "Haven't you any clue?", "I have got a clue." At the same time, it does not misfire on expressions that look similar but do not mean the same thing such as "I have got to clue you in."

3.3 Broad Coverage English Parsing Grammar

The underlying principle behind our approach to syntactic analysis is to provide a solid foundation for semantic interpretation. We have incorporated aspects of both Functional Grammar [Halliday, 1985] and Lexical Functional Grammar [Bresnan, 1982] in our approach. Our usage of Functional Grammar has been primarily limited to our analysis of tense. As mentioned, the focus of our approach has been on constructing deep syntactic functional analyses. Deep functional syntactic representations have been demonstrated to translate directly into quasi logical formulas [van Genabith and Crouch, 1996] and thus provide useful input for semantic interpretation [Halvorsen, 1997] [Dalrymple, 1999] [Dalrymple et al., 1993].

The underlying grammar formalism used by LCFLEX is a unification-augmented context-free grammar formalism that was originally developed for the Generalized LR Parser/Compiler [Tomita, 1990]

Sentence: Should the vector have been drawn to point down?

```
((mood *yes-no-interrogative)
 (root draw)
 (tense past)
 (modal should)
 (negation -)
 (passive +)
 (object
     ((root vector)
      (agr 3s)
      (definite +)
      (adjunct
          ((root point)
           (tense infinitive)
           (negation -)
           (passive -)
           (subject
               ((root vector)
                (agr 3s)
                (definite +)))
           (modifier ((root down)))))))
 (semantics
     (*multiple*
          ((frame *orient)
           (theme ((frame *vector)))
           (direction ((frame *down))))
          ((frame *draw)
           (theme ((frame *vector)))))))
```

Figure 9: **Sample Parser Output**

[Tomita, 1987] at the Center for Machine Translation at Carnegie Mellon University. Several recent practical systems for structural analysis of natural language have been designed to support unification based grammar specifications within a general CFG framework, that allows the system to incorporate known CFG parsing algorithms as the core of the system's implementation. Two examples of such systems are the Core Language Engine (CLE) [Alshawi (ed.), 1992] and the Alvey Natural Language Tools (ANLT) [Carroll, 1993].

LCFLEX's grammar formalism supports grammatical specification in an LFG framework, that consists of context-free grammar rules augmented with feature bundles that are associated with the non-terminals of the rules. For each context-free rule, the grammar formalism allows the specification of the feature structure associated with a left-hand side non-terminal of the rule as a function of the feature structures that are associated with the non-terminals on the right-hand side of the rule. At run-time, the parser computes the actual feature values of a particular left-hand side non-terminal as a by-product of the operation that creates the left-hand side non-terminal from the identified constituents on the right-hand side of the rule. Feature structure computation is, for the most part, specified and implemented via unification operations. This allows the grammar to constrain the applicability of context-free rules. The LHS constituent of a context-free rule is created only if the associated feature structure unification associated with the rule constituents is successful as well.

As an extension to LCFLEX's pseudo-unification grammar formalism, AUTOSEM provides the insert-role function as an interface to allow semantic interpretation to operate in parallel with syntactic interpretation at parse time. When the insert-role function is used to insert a child constituent into the slot corresponding to its syntactic functional role in a parent constituent, the child constituent's semantic representation is passed in to the parent constituent's semantic constructor function as in the Glue Language Semantics approach to interpretation [Dalrymple, 1999]. AUTOSEM's lexicon formalism allows semantic constructor functions to be linked into lexical entries by means of the semtag feature. Each semtag feature value corresponds to a semantic constructor function and mappings between syntactic functional roles such as subject, direct object, and indirect object and semantic roles such as agent, activity, or time. See Figures 7 and 8 for example meaning representation definitions and lexical entries respectively. Note that the syntactic features that appear in this example are taken from the COMLEX lexicon [Grishman et al., 1994].

In order to provide consistent input to the semantic constructor functions, AUTOSEM assumes a syntactic approach in which deep syntactic functional roles are assigned. Feature structures produced by our grammar factor out those aspects of syntax that modify the surface realization of a sentence but do not change its deep functional analysis. See Figure 9 for an example parser output. These aspects include tense, negation, mood, modality, and syntactic transformations such as passivization and extraction. In this way, for example, the roles assigned within an active sentence and its corresponding passive sentence remain the same. Thus, one important issue in the development of our grammar was to allow for such a deep syntactic functional analysis of a wide range of passivization and extraction patterns. In order to do this, we first defined a feature representation for each of COMLEX's verb subcategorization tags. The feature representations were then used to generate the verb phrase rules automatically. We cover the 91 verb subcategorization tags documented in the COMLEX manual. These tags cover a wide range of patterns of syntactic control and predication relationships. Each tag corresponds to one or more case frames. Each case frame corresponds to a number of different surface realizations due to passivization, relative clause extraction, and wh-movement. Altogether there are 519 syntactic patterns covered by the 91 subcategorization tags. However, some of these patterns have identical surface realizations, although they represent different relationships among the constituent parts. Thus, in LCFLEX's unification-augmented pseudo-unification formalism, it is possible to combine these 519 patterns into 85 disjunctive grammar rules. With our approach of generating the verb phrase rules automatically from feature representations of the COMLEX subcategorization tags, we simplify the task of maintaining that component of the grammar. Rather than maintain rules for 519 separate patterns, we need only maintain 116 feature representations, 1 for each case frame associated with the subcategorization tags. Maintaining feature representations is also simpler than maintaining the grammar itself since in the grammar, each case frame is covered by parts of a number of different complex rules, one corresponding to each possible surface realization. But in the feature representations, all of the information encoded about a single case frame is together in one feature representation.

The feature representation for each case frame is a list of *subcat elements*, each of which corresponds to a syntactic constituent in the case frame. Each subcat element contains the syntactic category of the associated constituent, markers if any, an annotation indicating whether the associated marker is optional and whether it is subject to stranding. The corresponding functional role or roles are also noted. Direct objects and indirect objects are assumed to be passivizable. In cases where they are not, or where constituents with other roles are passivizable, this is also noted. Constituents that are not subject to any sort of movement are also marked. In cases where what would ordinarily be the deep syntactic subject should not be analyzed as a subject, as in "Mary became a good worker", the alternate role is noted on the first subcat element for the case frame. Finally, if there are restrictions on the syntactic type of the subject, such as the category or that it be a pleonastic or not be a pleonastic, are also noted on first subcat element for the case frame.

4 The AUTOSEM Interpretation Process

As deep syntactic functional roles are assigned to constituents using the `insert-role` function, the syntactic head's constructor function is called with the child constituent's semantic representation bound to the parameter indicated in the head constituent's semtag features definition in the lexicon.

Take as an example the sentence "Should the vector have been drawn to point down?" displayed in Figure 9. This sentence expresses two propositions, one about drawing a vector and one about the vector pointing down. The verb `draw` as displayed in Figure 8, has as one of its subcategorization tags `np-to-inf-oc`, which indicates that the verb can take a noun phrase direct object that acts also as the subject of a to-infinitival clausal argument. In this case, the verb `draw` is passivized, so its surface syntactic subject is its deep syntactic object. Thus, "the vector" is both the deep object of "draw" as well as the deep subject of "point". When a verb, such as `draw`, appears in its past participle form following a form of the verb `to be`, an analysis is constructed for `draw` with the `passive` feature assigned the value +. When a passivized verb with subcategorization tag `np-to-inf-oc` appears with only a to-infinitival clause as its argument, as in "drawn to point down", one analysis the grammar produces is a structure with a representation of "point down" as a modifying clause, and features that indicate that the surface syntactic subject should be assigned two roles, namely, deep object and subject of the modifying clause.

Each time a constituent is assigned a functional role in relationship to `draw`, the constructor function for the semantic type `<draw>` is called. The semtag `draw1` indicates how the syntactic functional roles assigned by the grammar correspond to argument positions in the constructor function for `<draw>`. Thus, when the constructor function is called, the corresponding argument position is instantiated with the constituent's semantic interpretation. Since the same constructor function is called with different arguments a number of times in order to construct an analysis incrementally, an argument is included in every constructor function that allows a "result so far" to be passed in and augmented. Its default value, which is used the first time the constructor function is executed, is the representation associated with the corresponding type in the absence of any arguments being instantiated. Each time the constructor function is executed, each of its arguments that are instantiated are first checked to be certain that the structures they are instantiated with match all of the type restrictions on all of the slots that are bound to that argument. If they are, the instantiated arguments' structures are inserted into the corresponding slots in the "result so far". Otherwise the constructor function fails.

Interleaving syntactic and semantic analysis in this way allows semantic selectional restrictions to eliminate some of the ambiguity generated by the broad coverage syntactic grammar. It also allows the parser to avoid wasting time parsing portions of an input text that it does not have sufficient semantic knowledge to interpret because the parser fails to analyze any constituent that it cannot construct both a syntactic analysis and a semantic analysis for. At any time, a subset of semantic types can be selected from the complete meaning representation in order to restrict the parser further based on context specific semantic expectations. Additional context-specific semantic restrictions can also be introduced to specific types on the fly for the same purpose. For example, the meaning representation allows the theme associated with the `*draw` frame to be any object, but if in a particular context it was only important for the parser to interpret statements about drawing vectors, a further restriction could be imposed on the corresponding slot.

5 Current Directions

This paper describes a syntactic framework for semantic interpretation. In a recent evaluation of syntactic coverage over a corpus of 100 spontaneous typed utterance between 1 and 20 words long each containing one or more sentences, the parser was able to construct analyses covering 87% of the words in the corpus with an average fragmentation of 1.2 fragments per utterance and an average parse time of .1 seconds per utterance with a 330 mhz Pentium II. Currently a detailed evaluation of syntactic correctness is being conducted.

The AUTOSEM framework is currently being used to develop robust language understanding interfaces for three different intelligent tutoring systems. We are currently applying machine learning techniques to the problem of constructing the domain specific semantics that are built on top of its underlying syntactic framework.

6 Acknowledgements

This research was supported by NSF grant number 9720359 to CIRCLE, the Center for Interdisciplinary Research on Constructive Learning Environments at the University of Pittsburgh and Carnegie-Mellon University.

The author would like to acknowledge Alon Lavie for his involvement in the early development of the LCFLEX robust parser and in the writing of documents describing earlier incarnations of the work reported here.

This paper is dedicated to Eric and Rachel Rosé, who are most dear to the author's heart, and without whose patient understanding this work would not be possible.

References

[Abney, 1996] Abney, S. (1996). Partial parsing via finite-state cascades. In *Proceedings of the Eighth European Summer School In Logic, Language and Information, Prague, Czech Republic*.

[Ait-Mokhtar and Chanod, 1997] Ait-Mokhtar, S. and Chanod, J. (1997). Incremental finite-state parsing. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*.

[Alshawi (ed.), 1992] Alshawi (ed.), H. (1992). *The Core Language Engine*. MIT Press, Cambridge, MA.

[Bresnan, 1982] Bresnan, J. (1982). *The Mental Representation of Grammatical Relations*. The MIT Press.

[Carroll, 1993] Carroll, J. A. (1993). *Practical Unification-Based Parsing of Natural Language*. PhD thesis, University of Cambridge, Computer Laboratory.

[Cruse, 1986] Cruse, D. A. (1986). *Lexical Semantics*. Cambridge University Press.

[Dalrymple, 1999] Dalrymple, M. (1999). *Semantics and Syntax in Lexical Functional Grammar*. The MIT Press.

[Dalrymple et al., 1993] Dalrymple, M., Lamping, J., and Saraswat, V. (1993). LFG semantics via constrants. In *Proceedings of the European Chapter of the Association for Computational Linguistics (EACL-93)*.

[Danieli and Gerbino, 1995] Danieli, M. and Gerbino, E. (1995). Metrics for evaluating dialogue strategies in a spoken language system. In *Working Notes of the AAAI Spring Symposium on Empirical Methods in Discourse Interpretation and Generation*.

[Elmi and Evens, 1998] Elmi, M. and Evens, M. (1998). Spelling correction using context. In *Proceedings of COLING/ACL 98*.

[Freedman et al., 2000] Freedman, R., Rosé, C. P., Ringenberg, M. A., and VanLehn, K. (2000). Its tools for natural language dialogue: A domain-independent parser and planner. In *Intelligent Tutoring Systems: Seventh International Conference*.

[Glass, 1999] Glass, M. S. (1999). *Broadening Input Understanding in an Intelligent Tutoring System*. PhD thesis, Illinois Institute of Technology.

[Grishman et al., 1994] Grishman, R., Macleod, C., and Meyers, A. (1994). COMLEX syntax: Building a computational lexicon. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING-94)*.

[Halliday, 1985] Halliday, M. A. K. (1985). *An Introduction to Functional Grammar*. Edward Arnold: A division of Hodder and Stoughton.

[Halvorsen, 1997] Halvorsen, P. K. (1997). Situation semantics and semantic interpretation in constraint-based grammars. Technical Report 101, CSLI: Stanford University.

[Hatch, 1983] Hatch, E. (1983). Simplified input and second language acquisition. In Andersen, R., editor, *Pidginization and Creolization as Language Acquisition*. Newbury House Publishers.

[Hobbs et al., 1991] Hobbs, J. R., Appelt, D. E., Bear, J., and Tyson, M. (1991). Robust processing of real-world natural-language texts. Technical report, SRI International.

[Kasper et al., 1999] Kasper, W., Kiefer, B., Krieger, H., Rupp, C., and Worm, K. (1999). Charting the depths of robust speech parsing. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*.

[Koza, 1992] Koza, J. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press.

[Koza, 1994] Koza, J. (1994). *Genetic Programming II*. MIT Press.

[Lavie, 1995] Lavie, A. (1995). *A Grammar Based Robust Parser For Spontaneous Speech*. PhD thesis, School of Computer Science, Carnegie Mellon University.

[Lavie et al., 1996] Lavie, A., Gates, D., Gavalda, M., Mayfield, L., and Levin, A. W. L. (1996). Multi-lingual translation of spontaneously spoken language in a limited domain. In *Proceedings of COLING 96, Kopenhagen*.

[Lehman, 1989] Lehman, J. F. (1989). *Adaptive Parsing: Self-Extending Natural Language Interfaces*. PhD thesis, School of Computer Science, Carnegie Mellon University.

[McDonald, 1992] McDonald, D. (1992). An efficient chart-based algorithm for partial-parsing of unrestricted texts. In *Proceedings of the 3rd Conference on Applied Natural Language Processing*.

[McDonald, 1993a] McDonald, D. (1993a). Efficiently parsing large corpora. In *Proceedings of the ACL Workshop on Very Large Corpora: Academic and Industrial Perspectives*.

[McDonald, 1993b] McDonald, D. (1993b). The interplay of syntactic and semantic node labels in partial parsing. In *Proceedings of the Third International Workshop on Parsing Technologies*.

[Rosé, 1997] Rosé, C. P. (1997). *Robust Interactive Dialogue Interpretation*. PhD thesis, School of Computer Science, Carnegie Mellon University.

[Rosé, 1999] Rosé, C. P. (1999). A genetic programming approach for robust language interpretation. In Spector, L., Langdon, W. B., O'Reilly, U.-M., and Angeline, P., editors, *Advances in Genetic Programming 3*. The MIT Press.

[Rosé, 2000] Rosé, C. P. (2000). A framework for robust semantic interpretation. In *Proceedings of the 1st meeting of the North American Chapter of the Association for Computational Linguistics*.

[Rosé and Lavie, 1997] Rosé, C. P. and Lavie, A. (1997). An efficient distribution of labor in a two stage robust interpretation process. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*.

[Rosé and Lavie, pear] Rosé, C. P. and Lavie, A. (to appear). Balancing robustness and efficiency in unification augmented context-free parsers for large practical applications. In Junqua, J. C. and Noord, G. V., editors, *Robustness in Language and Speech Technologies*. Kluwer Academic Press.

[Suhm et al., 1994] Suhm, B., Levin, L., Coccaro, N., Carbonell, J., Horiguchi, K., Isotani, R., Lavie, A., Mayfield, L., Rosé, C. P., Dykema, C. V.-E., and Waibel, A. (1994). Speech-language integration in a multi-lingual speech translation system. In *Proceedings of the AAAI Workshop on Integration of Natural Language and Speech Processing*.

[Tomita, 1987] Tomita, M. (1987). An Efficient Augmented Context-free Parsing Algorithm. *Computational Linguistics*, 13(1-2):31–46.

[Tomita, 1990] Tomita, M. (1990). The Generalized LR Parser/Compiler - Version 8.4. In *Proceedings of International Conference on Computational Linguistics (COLING'90)*, pages 59–63, Helsinki, Finland.

[van den Berg et al., 1994] van den Berg, M., Bod, R., and Scha, R. (1994). A corpus based approach to semantic interpretation. In *Proceedings of the Ninth Amsterdam Colloquium*.

[van Genabith and Crouch, 1996] van Genabith, J. and Crouch, D. (1996). Direct and underspecified interpretations of lfg f-structures. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*.

[Van Noord, 1997] Van Noord, G. (1997). An efficient implementation of the head-corner parser. *Computational Linguistics*, 23(3).

[Worm, 1998] Worm, K. (1998). A model of robust processing of spontaneous speech by integrating viable fragments. In *Proceedings of COLING-ACL 98*.

[Woszcyna et al., 1994] Woszcyna, M., Aoki-Waibel, N., Buo, F. D., Coccaro, N., Horiguchi, K., Kemp, T., Lavie, A., McNair, A., Polzin, T., Rogina, I., Rosé, C. P., Schultz, T., Suhm, B., Tomita, M., and Waibel, A. (1994). JANUS 93: Towards spontaneous speech translation. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*.

[Woszcyna et al., 1993] Woszcyna, M., Coccaro, N., Eisele, A., Lavie, A., McNair, A., Polzin, T., Rogina, I., Rosé, C. P., Sloboda, T., Tomita, M., Tsutsumi, J., Waibel, N., Waibel, A., and Ward, W. (1993). Recent advances in JANUS: a speech translation system. In *Proceedings of the ARPA Human Languages Technology Workshop*.