

# A Framework for Robust Semantic Interpretation

Carolyn P. Rosé

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

## Abstract

This paper describes AUTOSEM, a robust semantic interpretation framework that can operate both at parse time and repair time. The evaluation demonstrates that AUTOSEM achieves a high level of robustness efficiently and without requiring any hand coded knowledge dedicated to repair.

## 1 Introduction

In order for an approach to robust interpretation to be practical it must be efficient, address the major types of disfluencies that plague spontaneously produced language input, and be domain independent so that achieving robustness in a particular domain does not require an additional knowledge engineering effort. This paper describes AUTOSEM, a semantic interpretation framework that possesses these three qualities. While previous approaches to robust interpretation have offered robust parsers paired with separate repair modules, with separate knowledge sources for each, AUTOSEM is a single unified framework that can operate both at parse time and repair time. AUTOSEM is integrated with the LCFLEX robust parser (Rosé and Lavie, to appear; Lavie and Rosé, 2000). Together AUTOSEM and LCFLEX constitute the robust understanding engine within the CARMEL natural language understanding component developed in the context of the Atlas intelligent tutoring project (Freedman et al., to appear). The evaluation reported here demonstrates that AUTOSEM's repair approach operates 200 times faster than the most similar competing approach while producing hypotheses of better quality.

AUTOSEM provides an interface to allow semantic interpretation to operate in parallel with syntactic interpretation at parse time in a lexicon driven fashion. Domain specific semantic knowledge is encoded declaratively within a meaning representation specification. Semantic constructor functions are compiled automatically from this specification and then linked into lexical entries as in the Glue Language Semantics approach to interpretation (Dalrymple, 1999). Based on syntactic head/argument relationships assigned at parse time, the construc-

tor 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.

AUTOSEM first attempts to construct analyses that satisfy both syntactic and semantic well-formedness conditions. The LCFLEX parser has the ability to efficiently relax syntactic constraints as needed and as allowed by its parameterized flexibility settings. For sentences remaining beyond the parser's coverage, AUTOSEM's repair algorithm relies entirely on semantic knowledge to compose the partial analyses produced by the parser. 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 Meaning Representation Specification

At the heart of AUTOSEM is its interpretation framework composed of semantic constructor functions compiled from a meaning representation specification. These semantic constructor functions can be used at parse time to build up semantic representations. These same constructor functions can then be 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 ex-

```

(:type <*state>
 :isa (<>)
 :instances nil
 :vars (entity time duration polarity)
 :spec ((who <*entity> entity)
        (when <*when> time)
        (how-long <*time-length> duration)
        (negation [+/-] polarity)))

(:type <*personal-state>
 :isa (<*state>)
 :instances nil
 :vars ()
 :spec ((who <*who> entity)))

(:type <busy>
 :isa (<*personal-state>)
 :instances nil
 :vars (activity)
 :spec ((frame *busy)
        (event <*event> activity)))

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

```

Figure 1: Sample meaning representation specification entries

tragrammatical input sentence.

The meaning representation specification provides a venue for expressing domain specific semantic information declaratively. AUTOSEM produces frame-based meaning representation structures. Thus, each domain specific meaning representation specification must define a 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.

AUTOSEM provides a simple formalism for defining meaning representations. Each entry corresponds to a semantic type and contains five fields: `:type`, `:isa`, `:instances`, `:vars`, and `:spec`. Some sample entries for the appointment scheduling domain are displayed in Figure 1. Some details are omitted for simplicity. 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.

### 3 Semantic Interpretation at Parse Time

```

(:type <cancel>
 :isa (<*event>)
 :instances nil
 :vars (agent activity time polarity)
 :spec ((frame *cancel)
        (engagement <*event> activity)))

```

Figure 2: Meaning representation definition of <cancel>

```

(:morph cancel
 :syntax ((cat vlex) (root cancel)
         (vform bare) (irreg-past +)
         (irreg-pastpart +)
         (irreg-prespart +)
         (subcat (*or* intrans np))
         (semtag cancel1))
 :semantics (cancel1 <cancel>
            ((subject agent)
             (object activity)
             (tempadjunct time)
             (negation polarity))))

```

Figure 3: Lexical entry for the verb “cancel”

As an extension to LCFLEX’s LFG-like 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 2 and 3 discussed further below. 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 as in CARMEL’s syntactic parsing grammar evaluated in (Freedman et al., to appear). In this way, for example, the roles assigned within an active sentence and its corresponding passive sentence remain the same.

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.

Take as an example the sentence “The meeting I had scheduled was canceled by you.” as it is processed by AUTOSEM using the CARMEL grammar and lexicon, which is built on top of the COMLEX lexicon (Grishman et al., 1994). The grammar assigns deep syntactic functional roles to constituents. Thus, “you” is the deep subject of “cancel”, and “the meeting” is the direct object both of “cancel” and of “schedule”. The detailed subcategorization classes associated with verbs, nouns, and adjectives in COMLEX make it possible to determine what these relationships should be. The meaning representation entry for <cancel> as well as the lexical entry for the verb “cancel” are found in Figures 2 and 3 respectively. Some details are left out for simplicity. When “the meeting I had scheduled” is analyzed as the surface subject of “was canceled”, it is assigned the deep syntactic role of object since “was canceled” is passive. The verb “cancel” has **cancel1** as its semtag value in the lexicon. **cancel1** is defined there as being associated with the type <cancel>, and the **object** syntactic role is associated with the **activity** argument. Thus, the <cancel> function is called with its **activity** argument instantiated with the meaning of “the meeting I had scheduled”.

Next, when “by you” is attached, “you” is assigned the deep syntactic role of subject of “cancel”. The **subject** role is associated with the **agent** argument in the definition of **cancel1**. Thus, <cancel> is called a again, this time with “you” instantiating the **agent** argument and the result from the last call to <cancel> passed in through the “result so far” argument.

## 4 Semantic Interpretation for Repair

While the LCFLEX parser has been demonstrated to robustly parse a variety of disfluencies found in spontaneously generated language (Rosé and Lavie, to appear), sentences still remain that are beyond its coverage. Previous research involving the earlier GLR\* parser (Lavie, 1995) and an earlier repair module (Rosé, 1997) has demonstrated that dividing the task of robust interpretation into two stages, namely parsing and repair, provides a better trade off between run time and coverage than attempting to place the entire burden of robustness on the parser alone (Rosé, 1997). Thus, when the flexibility allowed at parse time is not sufficient to construct an analysis of an entire sentence for any reason, a fragmentary analysis is passed into the repair module. For each pair of vertices in the chart the best single clause level and noun phrase level analysis according to LCFLEX’s statistical disambiguation scores is included in the set of fragmentary analyses passed on to the repair stage. An example<sup>1</sup> is displayed in Figure 4. Here the sentence “Why don’t we make it from like eleven to one?” failed to parse. In this case, the problem is that the insertion of “like” causes the sentence to be ungrammatical. When the parser’s flexibility settings are such that it is constrained to build analyses only for contiguous portions of text, such an insertion would prevent the parser from constructing an analysis covering the entire sentence. Nevertheless, it is able to construct analyses for a number of grammatical subsets of it.

Genetic programming search (Koza, 1992; Koza, 1994) is used to search for different ways to combine the fragments. Genetic programming is an opportunistic search algorithm used for constructing computer programs to solve particular problems. Among its desirable properties is its ability to search a large space efficiently by first sampling widely and shallowly, and then narrowing in on the regions surrounding the most promising looking points.

---

<sup>1</sup>This example was generated with the grammar used in the evaluation. See Section 6. The AUTOSEM repair algorithm can be used with grammars that do not make use of AUTOSEM’s parse-time interface by using a simple conversion program that automatically builds a function for each partial analysis corresponding to its semantic type.

Sentence: Why don't we make it from like eleven to one?

Functions:

```
(<SCHEDULE> NIL T NIL NIL NIL NIL NIL NIL T   (<SCHEDULE> NIL T NIL NIL NIL T T NIL NIL
  NIL NIL                                     NIL NIL
:STR ((FRAME *SCHEDULE) (ATTITUDE *LET-S)     :STR ((FRAME *SCHEDULE)
  (WHAT ((FRAME *IT))))                       (WHAT ((FRAME *IT))) (NEGATIVE +)
:COV (6 5 4 3 2 1)                             (WHO ((FRAME *WE))))
:SCORE 1.4012985e-45                             :COV (6 5 4 3 2) :SCORE 3.05483e-43)

(<SCHEDULE> NIL NIL NIL NIL NIL NIL NIL       (<SCHEDULE> NIL NIL NIL NIL NIL T T NIL NIL
  NIL T NIL NIL                               NIL NIL
:STR ((FRAME *SCHEDULE) (ATTITUDE *LET-S))    :STR ((FRAME *SCHEDULE) (NEGATIVE +)
:COV (5 4 3 2 1)                             (WHO ((FRAME *WE))))
:SCORE 1.4012985e-45                             :COV (5 4 3 2)
:SCORE 2.2584231e-37)

(<INTERVAL> NIL T T T                          (<SCHEDULE> NIL T NIL NIL NIL T NIL NIL NIL
:STR ((END ((FRAME *SIMPLE-TIME)             NIL NIL
  (HOUR 1)))
  (START ((FRAME *SIMPLE-TIME)
  (HOUR 11)))
  (INCL-EXCL INCLUSIVE)
  (FRAME *INTERVAL))
:COV (11 10 9)
:SCORE 1.2102125e-38)

:STR ((FRAME *SCHEDULE) (WHAT ((FRAME *IT)))
  (WHO ((FRAME *WE))))
:COV (6 5 4)
:SCORE 1.861576e-27)

(<PRO> NIL NIL NIL                             (<SIMPLE-TIME> NIL NIL NIL NIL NIL NIL T NIL
:STR ((FRAME *PRO))                           :STR ((FRAME *SIMPLE-TIME) (HOUR 1))
:COV (11)                                     :COV (11)
:SCORE 2.2223547e-18)                         :SCORE 2.2223547e-18)

(<SIMPLE-TIME> NIL NIL NIL NIL NIL NIL NIL    (<IT>
  T NIL                                       :STR ((FRAME *IT))
:STR ((FRAME *SIMPLE-TIME) (HOUR 11))        :COV (6)
:COV (9)                                     :SCORE 2.593466e-13)
:SCORE 1.0090891e-22)
```

Ideal Program:

```
(<SCHEDULE> NIL NIL NIL NIL NIL NIL NIL
  (<INTERVAL> NIL NIL NIL NIL
    :STR ((END ((FRAME *SIMPLE-TIME) (HOUR 1)))
      (START
        ((FRAME *SIMPLE-TIME) (HOUR 11)))
        (INCL-EXCL INCLUSIVE)
        (FRAME *INTERVAL))
      :COV (11 10 9)
      :SCORE 1.2102125e-38)
  NIL NIL NIL
:STR ((FRAME *SCHEDULE) (ATTITUDE *LET-S)
  (WHAT ((FRAME *IT))))
:COV (6 5 4 3 2 1)
:SCORE 1.4012985e-45)
```

Interpretation: Let's schedule it for from eleven o'clock till one o'clock

Figure 4: Repair example

It first takes a list of functions and terminal symbols and randomly generates a population of programs. It then evaluates each program for its “fitness” according to some predetermined set of criteria. The most fit programs are then paired up and used to produce the next generation by means of a crossover operation whereby a pair of subprograms, one from each parent program, are swapped. The new generation is evaluated for its fitness, and the process continues for a preset number of generations.

As mentioned earlier, because each semantic representation built by AUTOSEM contains a pointer to the constructor function that built it, each partial analysis can itself be treated as a constructor function. Thus, the function set made available to the genetic programming search for each sentence needing repair is derived from the set of partial analyses extracted from the parser’s chart. A number of the functions produced for the example are displayed in Figure 4. Some functions have been omitted for brevity. The functions are displayed as function calls, with the name of the function followed by its arguments. The name of each function corresponds to the semantic type from the meaning representation that corresponds to the associated partial analysis. Following this is a list of place holders corresponding to each argument position associated with the semantic type, as described in Section 2. Each place holder is either `nil` if it is an open place holder, or `t` if the position has already been filled in the corresponding partial analysis. The `STR` field contains the corresponding partial analysis. This is the “result so far” parameter discussed in Section 3. The `COV` field lists the positions in the sentence covered by the partial analysis. Note that in the example sentence, the word “don’t” covers both positions 2 and 3 since the parser expands the contraction before parsing. The `SCORE` field contains the statistical score assigned by the parser’s statistical disambiguation procedure described in (Rosé and Lavie, to appear).

The repair process begins as the genetic programming algorithm composes the function definitions into programs that assemble the fragments produced by the parser. The genetic programming algorithm has access to a list of type restrictions that are placed on each argument position by the meaning representation specification. Thus, the algorithm ensures that the programs that are generated do not violate any of the meaning representation’s type restrictions.

Once a population of programs is generated randomly, each program is evaluated for its fitness. A simple function implements a preference for programs that cover more of the sentence with fewer steps while using the analyses the parser assigned the best statistical scores to. A score between 0 and

1 is first assigned to each program corresponding to the percentage of the input sentence it covers. A second score between 0 and 1 estimates how complicated the program is by dividing the number of function calls by the length of the sentence and subtracting this number from 1. A third score is assigned the average of the statistical scores assigned by the parser to the fragments used in the program. Using coefficients based on an intuitive assignment of relative importance to the three scores, the final fitness value of each program is  $1 - [(.55 * coverageS) + (.25 * complexityS) + (.2 * statisticalS)]$ .

A typed version of the original crossover algorithm described in (Koza, 1992; Koza, 1994) was used to ensure that new programs would not violate any type restrictions or include more than one partial analysis covering the same span of text. This was accomplished by first making for each subprogram a list of the subprograms from the alternate program it could be inserted into without violating any semantic constraints. From these two lists it is possible to generate a list of all quadruples that specify a subprogram from each parent program to be removed and which subprogram from the alternate parent program they could be inserted into. From this list, all quadruples were removed that would either cause a span of text to be covered more than once in a resulting program or would require a subprogram to be inserted into a subprogram that would have been removed. From the remaining list, a quadruple was selected randomly. The corresponding crossover operation was then executed and the resulting two new programs were returned. While this typed version of crossover is more complex than the original crossover operation, it can be executed very rapidly in practice because the programs are relatively small and the semantic type restrictions ensure that the initial lists generated are correspondingly small.

## 5 Related Work

Recent approaches to robust parsing focus on shallow or partial parsing techniques (Van Noord, 1997; Worm, 1998; Ait-Mokhtar and Chanod, 1997; Abney, 1996). Rather than attempting to construct a parse covering an entire ungrammatical sentence as in (Lehman, 1989; Hipp, 1992), these approaches attempt to construct analyses for maximal contiguous portions of the input. The weakness of these partial parsing approaches is that part of the original meaning of the utterance may be discarded with the portion(s) of the utterance that are skipped in order to find a parsable subset. Information communicated by the relationships between these fragments within the original text is lost if these fragments are not combined. Thus, these less powerful algorithms essentially trade effectiveness for efficiency. Their goal is to introduce enough flexibility to gain an accept-

able level of coverage at an acceptable computational expense.

Some partial parsing approaches have been coupled with a post-parsing repair stage (Danieli and Gerbino, 1995; Rosé and Waibel, 1994; Rosé, 1997; Van Noord, 1997; Kasper et al., 1999) The goal behind these two stage approaches is to increase the coverage over partial parsing alone at a reasonable computational cost. Until the introduction of AUTOSEM, the ROSE approach, introduced in (Rosé, 1997), was unique in that it achieved this goal without either requiring hand coded knowledge specifically dedicated to repair or excessive amounts of interaction with the user. However, although (Rosé, 1997) demonstrates that the two stage ROSE approach is significantly faster than attempting to achieve the same quality of results in a single stage parsing approach, our evaluation demonstrates that it remains computationally intractable, requiring on average 67 seconds to repair a single parse on a 330 MHz Gateway 2000. In contrast, we demonstrate that AUTOSEM is on average 200 times faster, taking only .33 seconds on average to repair a single parse while achieving results of superior quality.

## 6 Evaluation

An experiment was conducted to evaluate AUTOSEM's robustness by comparing the effectiveness and efficiency of AUTOSEM's repair approach with that of the alternative ROSE approach. The test set used for this evaluation contains 750 sentences extracted from a corpus of spontaneous scheduling dialogues collected in English. For both repair approaches we used the meaning representation developed for the appointment scheduling domain 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. Because a pre-existing semantic grammar was available that parsed directly onto this meaning representation, that grammar was used in the parsing stage to construct analyses. 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 result was graded as either Bad, Okay, or Perfect, where Perfect indicates that the result was fluent and communicated the idea from the original sentence. A grade of Okay indicates that the result communicated the correct information, but not fluently. Those graded Bad either communicated incorrect information or were missing part or all of the information communicated in the original

sentence.

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 in conjunction with the **LC w/restarts** parsing mode. The results are displayed in Figures 5 and 7. 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 5 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.

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.

The primary difference between ROSE and AUTOSEM is that ROSE uses a single repair function, **MY-COMB**, to combine any two fragments by referring to the meaning representation specification. While it is possible to obtain the same set of repair hypotheses with ROSE as with AUTOSEM, the ROSE approach insulates the genetic search from the semantic restrictions imposed by the meaning representation. These restrictions are visible only locally within individual applications of the **MY-COMB** function. Thus, **MY-COMB** must be able to cope with the case where the arguments passed in cannot be combined. Large portions of the programs generated by ROSE as repair hypotheses do not end up contributing to the resulting structure. The programs gener-

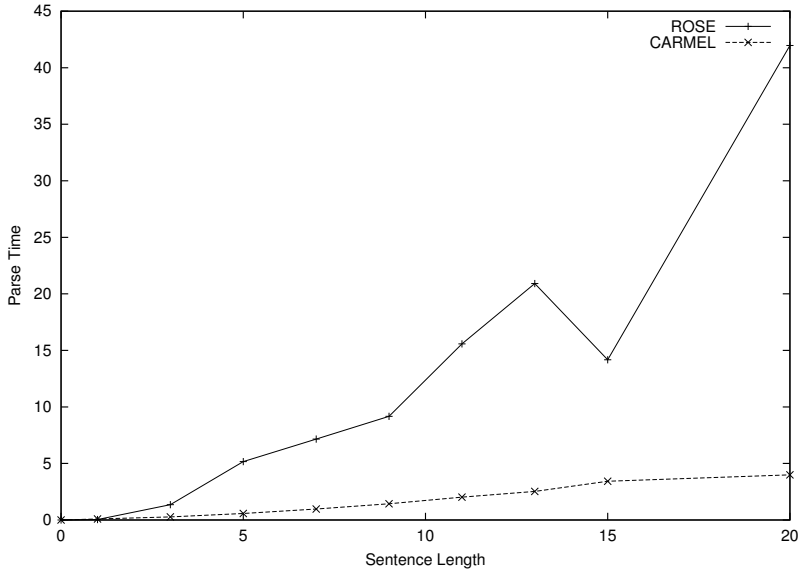


Figure 5: Processing Times for Alternative Strategies

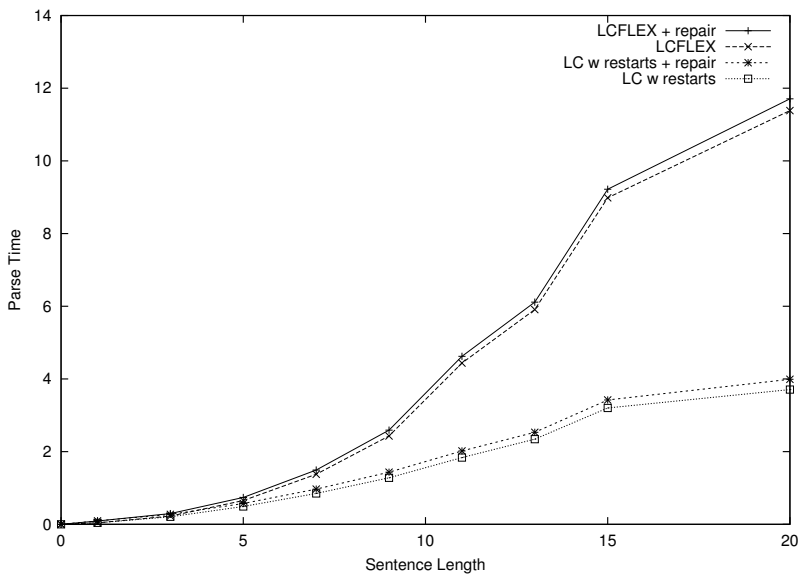


Figure 6: Processing Times for Alternative Strategies

ated by ROSE must therefore be much larger than in AUTOSEM in order to obtain the same results. Furthermore, the fitness of each repair hypothesis can only be computed by executing the program to obtain a result. The combination of all of these things makes the process of fitness evaluation in ROSE far more costly than in AUTOSEM. In contrast, AUTOSEM’s constructor function definitions make it possible for the genetic search to make use of semantic restrictions to speed up the process of converging on a high quality repair hypothesis. The tremendous speed-up offered by the AUTOSEM approach makes

it practical to apply repair more often and to use a larger generation size (50 individuals as opposed to 32) and a larger number of generations (5 as opposed to 4) for the genetic search.

## 7 Current Directions

In this paper we described AUTOSEM, a new robust semantic interpretation framework. Our evaluation demonstrates that AUTOSEM achieves a greater level of robustness 200 times more efficiently than the most similar competing approach.

In AUTOSEM, the mapping between syntactic

	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 7: Interpretation quality with and without repair

functional roles and semantic arguments is completely determined in the current version. In some cases, such as with copular constructions and with adjunct prepositional phrases, it would be useful to introduce some non-determinism so that, for example, semantic selectional restrictions between the object of the preposition and the semantic structure that the prepositional phrase is attaching to can more easily play a role in selecting the appropriate semantic relationship. Exploring approaches for achieving this non-determinism efficiently is one of our current objectives.

## 8 Acknowledgements

Special thanks are due to the JANUS multilingual speech-to-speech translation project for making their interlingua specification and semantic parsing and generation grammars available for the evaluation reported here. This research was supported by NSF Grant IRI-94-57637 and Grant 9720359 to CIRCLE, a center for research on intelligent tutoring.

## References

- S. Abney. 1996. Partial parsing via finite-state cascades. In *Proceedings of the Eighth European Summer School In Logic, Language and Information, Prague, Czech Republic*.
- S. Ait-Mokhtar and J. Chanod. 1997. Incremental finite-state parsing. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*.
- M. Dalrymple. 1999. *Semantics and Syntax in Lexical Functional Grammar*. The MIT Press.
- M. Danieli and E. Gerbino. 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*.
- R. Freedman, C. P. Rosé, M. A. Ringenber, and K. VanLehn. to appear. Its tools for natural language dialogue: A domain-independent parser and planner. In *Proceedings of the Intelligent Tutoring Systems Conference*.
- R. Grishman, C. Macleod, and A. Meyers. 1994. COMLEX syntax: Building a computational lexicon. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING-94)*.
- D. R. Hipp. 1992. *Design and Development of Spoken Natural-Language Dialog Parsing Systems*. Ph.D. thesis, Dept. of Computer Science, Duke University.
- W. Kasper, B. Kiefer, H. Krieger, C. Rupp, and K. Worm. 1999. Charting the depths of robust speech parsing. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*.
- J. Koza. 1992. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press.
- J. Koza. 1994. *Genetic Programming II*. MIT Press.
- A. Lavie and C. P. Rosé. 2000. Optimal ambiguity packing in unification-augmented context-free grammars. In *Proceedings of the International Workshop on Parsing Technologies*.
- A. Lavie. 1995. *A Grammar Based Robust Parser For Spontaneous Speech*. Ph.D. thesis, School of Computer Science, Carnegie Mellon University.
- J. F. Lehman. 1989. *Adaptive Parsing: Self-Extending Natural Language Interfaces*. Ph.D. thesis, School of Computer Science, Carnegie Mellon University.
- C. P. Rosé and A. Lavie. to appear. Balancing robustness and efficiency in unification augmented context-free parsers for large practical applications. In J. C. Junqua and G. Van Noord, editors, *Robustness in Language and Speech Technologies*. Kluwer Academic Press.
- C. P. Rosé and A. Waibel. 1994. Recovering from parser failures: A hybrid statistical/symbolic approach. In *Proceedings of The Balancing Act: Combining Symbolic and Statistical Approaches to Language workshop at the 32nd Annual Meeting of the ACL*.
- C. P. Rosé. 1997. *Robust Interactive Dialogue Interpretation*. Ph.D. thesis, School of Computer Science, Carnegie Mellon University.
- G. Van Noord. 1997. An efficient implementation of the head-corner parser. *Computational Linguistics*, 23(3).
- K. Worm. 1998. A model of robust processing of spontaneous speech by integrating viable fragments. In *Proceedings of COLING-ACL 98*.