Towards a Self-Extending Lexicon*

Uri Zernik
Michael G. Dyer

Artificial Intelligence Laboratory
Computer Science Department
3531 Boelter Hall
University of California
Los Angeles, California 90024

Abstract

The problem of manually modifying the lexicon appears with any natural language processing program. Ideally, a program should be able to acquire new lexical entries from context, the way people learn. We address the problem of acquiring entire phrases, specifically *figurative phrases*, through augmenting a *phrasal lexicon*. Facilitating such a self-extending lexicon involves (a) *disambiguation*—selection of the intended phrase from a set of matching phrases, (b) *robust parsing*—comprehension of partially-matching phrases, and (c) *error analysis*—use of errors in forming hypotheses about new phrases. We have designed and implemented a program called RINA which uses *demons* to implement *functional-grammar* principles. RINA receives new figurative phrases in context and through the application of a sequence of *failure-driven* rules, creates and refines both the patterns and the concepts which hold syntactic and semantic information about phrases.

1. Introduction

A language understanding program should be able to acquire new lexical items from context, forming for novel phrases their linguistic patterns and figuring out their conceptual meanings. The lexicon of a learning program should satisfy three requirements: Each lexical entry should (1) be learnable, (2) facilitate conceptual analysis, and (3) facilitate generation. In this paper we focus on the first two aspects.

1.1 The Task Domain

Two examples, which will be used throughout this paper, are given below. In the first dialogue the learner is introduced to an unknown phrase: take on. The words take and on are familiar to the learner, who also remembers the biblical story of David and Goliath. The program, modeling a language learner, interacts with a native speaker, as follows:

* This work was made possible in part by a grant from the Keck Foundation.

David vs. Goliath

Native:	Remember the story of David and Goliath?
	David took on Goliath.
Learner:	David took Goliath somewhere?
Native:	No. David took on Goliath.
Learner:	He took on him. He won the fight?
Native:	No. He took him on.
	David attacked him.
Learner:	He took him on.
	He accepted the challenge?
Native:	Right.
Native:	Here is another story.
	John took on the third exam question.
Learner:	He took on a hard problem.

Another dialogue involves put one's foot down. Again, the phrase is unknown while its constituents are known:

Going Punk

Native:	Jenny wanted to go punk,
	but her father put his foot down.
Learner:	He moved his foot down?
	It does not make sense.
Native:	No. He put his foot down.
Learner:	He put his foot down.
	He refused to let her go punk.

A *figurative phrase* such as put one's foot down is a linguistic pattern whose associated meaning cannot be produced from the composition of its constituents. Indeed, an interpretation of the phrase based on the meanings of its constituents often exists, but it carries a different meaning. The fact that this literal interpretation of the figurative phrase exists is a misleading clue in learning. Furthermore, the learner may not even notice that a novel phrase has been introduced since she is familiar with down as well as with foot. Becker [Becker75] has described a space of phrases ranging in generality from fixed proverbs such as charity begins at home through idioms such as lay down the law and phrasal verbs such as put up with one's spouse and look up the name, to literal verb phrases such as sit on the chair. He suggested employing a phrasal lexicon to capture this entire range of language structures.

1.2 Issues in Phrase Acquisition

Three issues must be addressed when learning phrases in context.

(1) **Detecting failures**: What are the indications that the initial interpretation of the phrase `take him on` as "to take a person to a location" is incorrect? Since all the words in the sentence are known, the problem is detected both as a *conceptual discrepancy* (why would he take his enemy anywhere?) and as a *syntactic failure* (the expected location of the assumed physical transfer is missing).

(2) **Determining scope and generality of patterns**: The linguistic pattern of a phrase may be perceived by the learner at various levels of *generality*. For example, in the second dialogue, incorrect generalizations could yield patterns accepting sentences such as:

```
Her boss put his left foot down.
He moved his foot down.
He put down his foot.
He put down his leg.
```

A decision is also required about the *scope* of the pattern (i.e., the tokens included in the pattern). For instance, the scope of the pattern in `John put up with Mary` could be (1) `?x:person put:verb up` where `with` is associated with `Mary` or (2) `?x:person put:verb up with ?y:person`, where `with` is associated with `put up`.

(3) **Finding appropriate meanings**: The conceptual meaning of the phrase must be extracted from the context which contains many concepts, both appropriate and inappropriate for hypothesis formation. Thus there must be strategies for focusing on appropriate elements in the context.

1.3 The Program

RINA [Dyer85] is a computer program designed to learn English phrases. It takes as input English sentences which may include unknown phrases and conveys as output its hypotheses about novel phrases. The program consists of four components:

(1) **Phrasal lexicon**: This is a list of phrases where each phrase is a declarative *pattern-concept* pair [Wilensky81].

(2) **Case-frame parser**: In the parsing process, case-frame expectations are handled by spawning *demons* [Dyer83]. The parser detects comprehension failures which are used in learning.

(3) **Pattern Constructor**: Learning of phrase patterns is accomplished by analyzing parsing failures. Each failure situation is associated with a pattern-modification action.

(4) **Concept Constructor**: Learning of phrase concepts is accomplished by a set of strategies which are selected according to the context.

Schematically, the program receives a sequence of *sentence/context* pairs from which it refines its current *pattern/concept* pair. The pattern is derived from the sentence and the concept is derived from the context. However, the two processes are not independent since the context influences construction of patterns while linguistic clues in the sentence influence formation of concepts.

2. Phrasal Representation of the Lexicon

Parsing in RINA is central since learning is evaluated in terms of parsing ability before and after phrases are acquired. Moreover, learning is accomplished through parsing.

2.1 The Background

RINA combines elements of the following two approaches to language processing:

Phrase-based pattern matching: In the implementation of UC [Wilensky84], an intelligent help system for UNIX users, both PHRAN [Arens82], the conceptual analyzer, and PHRED [Jacobs85] the generator, share a *phrasal lexicon*. As outlined by Wilensky [Wilensky81] this lexicon provides a *declarative* database, being modularly separated from the control part of the system which carries out parsing and generation. This development in representation of linguistic knowledge is paralleled by theories of *functional grammars* [Kay79], and *lexical-functional grammars* [Bresnan78].

Case-based demon parsing: Boris [Dyer83] modeled reading and understanding stories in depth. Its conceptual analyzer employed demon-based templates for parsing and for generation. Demons are used in parsing for two purposes: (1) to implement syntactic and semantic expectations [Riesbeck74] and (2) to implement memory operations such as search, match and update. This approach implements Schank's [Schank77] theory of representation of concepts, and follows *case-grammar* [Fillmore68] principles.

RINA uses a declarative phrasal lexicon as suggested by Wilensky [Wilensky82], where a lexical phrase is a *pattern-concept* pair. The pattern notation is described below and the concept notation is Dyer's [Dyer83] *i-link* notation.

2.2 The Pattern Notation

To span English sentences, RINA uses two kinds of patterns: *lexical patterns* and *ordering patterns* [Arens82]. In Figure 1 we show sample lexical patterns (patterns of lexical phrases). Such patterns are viewed as the generic linguistic forms of their corresponding phrases.

```
1. ?x:(animate,agent) nibble:verb <on ?y:food>
2. ?x:(person,agent) take:verb on ?y:patient
3. ?x:(person,agent) <put:verb foot:body-part down>
```

Figure 1: **The Pattern Notation**

The notation is explained below:

(1) A *token* is a literal unless otherwise specified. For example, **on** is a literal in the patterns above.

(2) **?x:***sort* denotes a variable called **?x** of a *semantic* type *sort*. **?y:food** above is a variable which stands for references to objects of the *semantic* class **food**.

(3) *Act*:**verb** denotes any form of the **verb** *syntactic* class with the root *act*. **nibble:***verb* above stands for expressions such as: `nibbled`, `has never nibbled`, etc.

(4) By default, a pattern sequence does not specify the order of its tokens.

(5) Tokens delimited by < and > are restricted to their specified order. In Pattern 1 above, **on** must directly precede ?y:food.

Ordering patterns pertain to language word-order conventions in general. Some sample ordering patterns are:

```
active:   <?x:agent ?y:(verb,active)>
passive:  <?x:patient ?y:(verb,passive)>
                      *<by ?z:agent>
infinitive:<to ?x:verb,active> ˆ?y:agent
```

Figure 2: **Ordering Patterns**

The additional notation introduced here is:

(6) An * preceding a term, such as `*<by ?z:agent>` in the first pattern above indicates that the term is optional.

(7) ˆ denotes an omitted term. The concept for ?y in the third example above is extracted from the agent of the pattern including the current pattern.

(8) By convention, the *agent* is the case-frame which precedes the verb in the lexical pattern. Notice that the notion of *agent* is necessary since (a) the agent is not necessarily the *subject* (i.e., `she was taken`) and (b) the agent is not necessarily the *actor* (i.e., `she received the book, he took a blow`), and (c) in the infinitive form, the agent must be referred to since the agent is omitted from the pattern in the lexicon.

(9) *Unification* [Kay79] accounts for the interaction of lexical patterns with ordering patterns in matching input sentences.

So far, we have given a declarative definition of our grammar, a definition which is neutral with respect to either parsing or generation. The parsing procedure which is derived from the definitions above still has to be given.

2.3 Parsing Objectives

Three main tasks in phrasal parsing may be identified, ordered by degree of difficulty.

(1) **Phrase disambiguation**: When more than one lexical phrase matches the input sentence, the parser must select the phrase intended by the speaker. For example, the input `the workers took to the streets` could mean either "they demonstrated" or "they were fond of the streets". In this case, the first phrase is selected according to the principle of *pattern specificity* [Arens82]. `The pattern ?x:person take:verb <to the streets>` is more specific then `?x:person take:verb <to ?y:thing>` However, in terms of our pattern notation, how do we define pattern specificity?

(2) **Ill-formed input comprehension**: Even when an input sentence is not well phrased according to text-book grammar, it may be comprehensible by people and so must be comprehensible to the parser. For example, `John took Mary school` is telegraphic, but comprehensible, while `John took Mary to` conveys only a partial concept. Partially matching sentences (or "near misses") are not handled well by syntax-driven pattern matchers. A deviation in a function word (such as the word `to` above) might inhibit the detection of the phrase which could be detected by a semantics-driven parser.

(3) **Error-detection**: when the hypothesized phrase does not match the input sentence/context pair, the parser is required to detect the failure and return with an indication of its nature. Error analysis requires that pattern tokens be assigned a case-significance, as shown in Section 4.

Compounding requirements—disambiguation plus error-analysis capability— complicate the design of the parser. On one hand, analysis of "near misses" (`they bury a hatchet` instead of `they buried the hatchet`) can be performed through a rigorous analysis—assuming the presence of a single phrase only. On the other hand, in the presence of multiple candidate phrases, disambiguation could be made efficient by organizing sequences of pattern tokens into a discrimination net. However, attempting to perform both disambiguation and "near miss" recognition and analysis simultaneously presents a difficult problem. The discrimination net organization would not enable comparing the input sentence, the "near miss", with existing phrases.

The solution is to organize the discrimination sequence by order of generality from the general to the specific. According to this principle, verb phrases are matched by conceptual features first and by syntactic features only later on. For example, consider three initial erroneous hypotheses: (a) bury a hatchet (b) bury the gun, and (c) bury the hash. On hearing the words "bury the hatchet", the first hypothesis would be the easiest to analyze (it differs only by a function word while the second differs by a content-holding word) and the third one would be the hardest (as opposed to the second, hash does not have a common concept with hatchet).

2.4 Case-Frames

Since these requirements are not facilitated by the representation of patterns as given above, we slightly modify our view of patterns. An entire pattern is constructed from a set of *case-frames* where each case-frame is constructed of single tokens: words and concepts. Each frame has several slots containing information about the case and pertaining to: (a) its syntactic appearance (b) its semantic concept and (c) its phrase role: agent, patient. Variable identifiers (e.g., ?x. ?y) are used for unification of phrase patterns with their corresponding phrase concepts. Two example patterns are given below:

The first example pattern denotes a simple literal verb phrase:

```
{id:?x class:person role:agent}
{take:verb}
{id:?y class:person role:patient}
{id:?z class:location marker:to}
```

Figure 3: **Case Frames for "He took her to school"**

Both the agent and the patient are of the class person; the indirect object is a location marked by the preposition to. The second phrase is figurative:

```
{id:?x class:person role:agent}
{take:verb}
{marker:to determiner:the word:streets}
```

Figure 4: **Case Frames for "He took to the streets"**

The third case frame in Figure 4 above, the indirect object, does not have any corresponding concept. Rather it is represented as a sequence of words. However the words in the sequence are designated as the *marker*, the *determiner* and the *word* itself.

Using this view of patterns enables the recognition of "near misses" and facilitate error-analysis in parsing.

3. Demons Make Patterns Operational

So far, we have described only the linguistic notation and indicated that unification [Kay79] accounts for production of sentences from patterns. However, it is not obvious how to make pattern unification operational in parsing. One approach [Arens82] is to generate word sequences and to compare generated sequences with the input sentence. Another approach [Pereira80] is to implement unification using PROLOG. Since our task is to provide *lenient parsing*, namely also ill-formed sentences must be handled by the parser, these two approaches are not suitable. In our approach, parsing is carried out by converting patterns into demons.

Conceptual analysis is the process which involves reading input words left to right, matching them with existing linguistic patterns and instantiating or modifying in memory the associated conceptual meanings. For example, assume that these are the phrases for take: in the lexicon:

```
?x:person take:verb ?y:person ?z:locale
        John took her to Boston.
?x:person take:verb ?y:phys-obj
        He took the book.
?x:person take:verb off ?y:attire
        He took off his coat.
?x:person take:verb on ?y:person
        David took on Goliath.
?x:person take:verb a bow
        The actor took a bow.
?x:thing take:verb a blow
        The wall took a blow.
?x:person take:verb <to the streets>
        The workers took to the streets.
        The juvenile took to the streets.
```

Figure 5: **A Variety of Phrases for TAKE**

where variables ?x, ?y and ?z also appear in corresponding concepts (not shown here). How are these patterns actually applied in conceptual analysis?

3.1 Interaction of Lexical and Ordering Patterns

Token order in the lexical patterns themselves (Figure 5) supports the derivation of simple active-voice sentences only. Sentences such as:

```
Mary was taken on by John.
A weak contender David might have left alone,
      but Goliath he took on.
David decided to take on Goliath.
```

Figure 6: **A Variety of Word Orders**

cannot be derived directly by the given lexical patterns. These sentences deviate from the order given by the corresponding lexical patterns and require interaction with language conventions such as passive voice and infinitive. *Ordering patterns* are used to span a wider range of sentences in the language. Ordering patterns such as the one's given in Figure 2 depict the word order involving verb phrases. In each pattern the case-frame preceding the verb is specified. (In active voice, the agent appears imediately before the verb, while in the passive it is the patient that precedes the verb.)

3.2 How Does It All Work?

Ordering patterns are compiled into demons. For example, D_AGENT, the demon anticipating the agent of the phrase is generated by the patterns in Figure 2. It has three clauses:

```
If the verb is in active form
  then the agent is immediately before the verb
If the verb is in passive form
  then the agent may appear, preceded by by.
If the verb is in infinitive
  then the agent is omitted.
  Its concept is obtained from the function verb.
```

Figure 7: **The Construction of D_AGENT**

In parsing, this demon is spawned when a verb is encountered. For example, consider the process in parsing the sentence

`David decided to take on Goliath.`

Through identifying the verbs and their forms, the process is:

- decided (active, simple)
  Search for the agent before the verb, anticipate an infinitive form.
- take (active, infinitive)
  Do **not** anticipate the agent. The **actor** of the "take on" concept which is the agent, is extracted from the agent of "decide".

4. Failure-Driven Pattern Construction

Learning of phrases in RINA is an iterative process. The input is a sequence of sentence-context pairs, through which the program refines its current hypothesis about the new phrase. The hypothesis pertains to both the pattern and the concept of the phrase.

4.2 The Learning Cycle

The basic cycle in the process is:

(a) A sentence is parsed on the background of a conceptual context.

(b) Using the current hypothesis, either the sentence is comprehended smoothly, or a failure is detected.

(c) If a failure is detected then the current hypothesis is updated.

The crucial point in this scheme is to obtain from the parser an intelligible analysis of failures. As an example, consider this part of the first dialog:

1 Program: `He took on him. He won the fight?`
2 User: `No. He took him on. David attacked him.`
3 Program: `He took him on.`
`He accepted the challenge?`

The first hypothesis is shown in Figure 8.

pattern:	?x:person take:verb <on ?y:person>
concept:	?x win the conflict with ?y

Figure 8: **First Hypothesis**

Notice that the preposition on is attached to the object ?y, thus assuming that the phrase is similar to `He looked at Mary` which cannot produce the following sentence: `He looked her at.` This hypothesis underlies Sentence 1 which is erroneous in both its form and its meaning. Two observations should be made by comparing this pattern to Sentence 2:

- The object is not preceded by the preposition on.
- The preposition on does not precede any object.

These comments direct the construction of the new hypothesis:

pattern:	?x:person take:verb on ?y:person
concept:	?x win the conflict with ?y

Figure 9: **Second Hypothesis**

where the preposition on is taken as a modifier of the verb itself, thus correctly generating Sentence 3. In Figure 9 the conceptual hypothesis is still incorrect and must itself be modified.

4.3 Learning Strategies

A subset of RINA's learning strategies, the ones used for the `David and Goliath` Dialog (Section 1.1) are described in this section. In our exposition of failures and actions we will illustrate the situations involved in the dialogues above, where each situation is specified by the following five ingredients:

(1) the input sentence (**Sentence**),

(2) the context (not shown explicitly here),

(3) the active pattern: either the pattern under construction, or the best matching pattern if this is the first sentence in the dialogue (**Pattern1**).

(4) the failures detected in the current situation (**Failures**),

(5) the pattern resulting from the application of the action to the current pattern (**Pattern2**).

Creating a New Phrase

A *case-role mismatch* occurs when the input sentence can only be partially matched by the active pattern. A *goal mismatch* occurs when the concept instantiated by the selected pattern does not match the goal situation in the context.

Sentence:	David took on Goliath.
Pattern1:	?x:person take:verb ?y:person ?z:location
Failures:	Pattern and goal mismatch
Pattern2:	?x:person take:verb

David's physically transferring Goliath to a location fails since (1) a location is not found and (2) the action does not match David's goals. If these two failures are encountered, then a new phrase is created. In absence of a better alternative, RINA initially generates `David took him somewhere.`

Discriminating a Pattern by Freezing a Prepositional Phrase

A *prepositional mismatch* occurs when a preposition P matches in neither the active pattern nor in one of the lexical prepositional phrases, such as:

<on ?x:platform>	(indicating a spatial relation)
<on ?x:time-unit>	(indicating a time of action)
<on ?x:location>	(indicating a place)

Sentence:	David took on Goliath.
Pattern1:	?x:person take:verb
Failures:	Prepositional mismatch
Pattern2:	?x:person take:verb <on ?y:person>

The preposition on is not part of the active pattern. Neither does it match any of the prepositional phrases which currently exist for on. Therefore, since it cannot be interpreted in any other way, the ordering of the sub-expression `<on ?y:person>` is frozen in the larger pattern, using < and >.

Two-word verbs present a difficulty to language learners [Ulm75] who tend to ignore the separated verb-particle form, generating: `take on him` instead of `take him on`. In the situation above, the learner produced this typical error.

Relaxing an Undergeneralized Pattern

Two failures involving on: (1) *case-role mismatch* (`on ?y:person` is not found) and (2) *prepositional mismatch* (on appears unmatched at the end of the sentence) are encountered in the situation below:

Sentence:	David took him on.
Pattern1:	?x:person take:verb <on ?y:person>
Failures:	Prepositional and case-role mismatch.
Pattern2:	?x:person take:verb on ?y:person

The combination of these two failures indicate that the pattern is too restrictive. Therefore, the < and > freezing delimiters are removed, and the pattern may now account for two-word verbs. In this case on can be separated from `take`.

Generalizing a Semantic Restriction

A *semantic mismatch* is marked when the semantic class of a variable in the pattern does not subsume the class of the corresponding concept in the sentence.

Sentence:	John took on the third question.
Pattern1:	?x:person take:verb on ?y:person
Failures:	Semantic mismatch
Pattern2:	?x:person take:verb on ?y:task

As a result, the type of ?y in the pattern is generalized to include both cases.

Freezing a Reference Which Relates to a Metaphor

An *unrelated reference* is marked when a reference in the sentence does not relate to the context, but rather it relates to a *metaphor* (see elaboration in [Zernik85]). The reference `his foot` cannot be resolved in the context, rather it is resolved by a metaphoric *gesture*.

Sentence:	Her father put his foot down.
Pattern1:	?x:person put:verb down ?y:phys-obj
Failures:	Goal mismatch and unrelated reference
Pattern2:	?x:person put:verb down foot:body-part

Since, (1) putting his foot on the floor does not match any of the goals of Jenny's father and (2) the reference `his foot` is related to the domain of metaphoric gestures rather than to the context. Therefore, `foot` becomes frozen in the pattern. This method is similar to a method suggested by Fass and Wilks [Fass83]. In their method, a metaphor is analyzed when an apparently *ill-formed* input is detected, e.g.: `the car drank a lot of gas`.

4.4 Concept Constructor

Each pattern has an associated concept which is specified using Dyer's [Dyer83] *i-link* notation. The concept of a new phrase is extracted from the context, which may contain more than one element. For example, in the first dialogue above, the given context contains some salient *story points* [Wilensky82] which are indexed in episodic memory as two *violated expectations*:

- David won the fight in spite of Goliath's physical superiority.
- David accepted the challenge in spite of the risk involved.

The program extracts meanings from the given set of points. Concept hypothesis construction is further discussed in [Zernik85].

5. Previous Work in Language Learning

In RINA, the stimulus for learning is comprehension failure. In previous models language learning was also driven by detection of failures.

PST [Reeker76] learned grammar by acting upon differences detected between the input sentence and internally generated sentences. Six types of differences were classified, and the detection of a difference which belonged to a class caused the associated alteration of the grammar.

FOUL-UP [Granger77] learned meanings of single words when an unknown word was encountered. The meaning was extracted from the script [Schank77] which was given as the context. A typical learning situation was `The car was driving on Hwy 66, when it careened off the road`. The meaning of the unknown verb `careened` was guessed from the $ACCIDENT script.

POLITICS [Carbonell79], which modeled comprehension of text involving political concepts, initiated learning when semantic constraints were violated. Constraints were generalized by analyzing underlying metaphors.

AMBER [Langley82] modeled learning of basic sentence structure. The process of learning was directed by mismatches between input sentences and sentences generated by the program. Learning involved recovery from both *errors of omission* (omitting a function word such as `the` or `is` in `daddy bouncing ball`) and *errors of commission* (producing `daddy is liking dinner`).

Thus, some programs acquired linguistic patterns and some programs acquired meanings from context, but none of the above programs acquired new **phrases**. Acquisition of phrases involves two parallel processes: the formation of the pattern from the given set of example sentences, and the construction of the meaning from the context. These two processes are not independent since the construction of the conceptual meaning utilizes linguistic clues while the selection of pattern elements of new figurative phrases bears on concepts in the context.

6. Current and Future Work

Currently, RINA can learn a variety of phrasal verbs and idioms. For example, RINA implements the behavior of the learner in `David vs. Goliath` and in `Going Punk` in Section 1. Modifications of lexical entries are driven by analysis of failures. This analysis is similar to analysis of *ill-formed* input, however, detection of failures may result in the augmentation of the lexicon. Failures appear as semantic discrepancies (e.g., goal-plan mismatch), or syntactic discrepancies (e.g., case-role mismatch). Finally, references in figurative phrases are resolved by metaphor mapping.

Currently our efforts are focussed on learning the conceptual elements of phrases. We attempt to develop strategies for generalizing and refining acquired concepts. For example, it is desirable to refine the concept for "take on" by this sequence of examples:

```
David took on Goliath.
The lakers took on the Celtics.
I took on a hard task.
I took on a new job.
In selecting the name "Towards a Self-Extending
      Lexicon", we took on an old name.
```

The first three examples "deciding to fight someone", "playing against someone" and "accepting a challenge" could be generalized into the same concept, but the last two examples deviate in their meanings from that developed concept. The problem is to determine the desired level of generality. Clearly, the phrases in the following examples:

```
take on an enemy
take on an old name
take on the shape of a snake
```

deserve separate entries in the phrasal lexicon. The question is, at what stage is the advantage of further generalization diminished?

Acknowledgments

We wish to thank Erik Mueller and Mike Gasser for their incisive comments on drafts of this paper.

References

[Arens82] Arens, Y., "The Context Model: Language Understanding in a Context," in *Proceedings Fourth Annual Conference of the Cognitive Science Society,* Ann Arbor, Michigan (1982).

[Becker75] Becker, Joseph D., "The Phrasal Lexicon," pp. 70-73 in *Proceedings Interdisciplinary Workshop on Theoretical Issues in Natural Language Processing,* Cambridge, Massachusets (June 1975).

[Bresnan78] Bresnan, Joan, "A Realistic Transformational Grammar," pp. 1-59 in *Linguistic Theory and Psychological Reality,* ed. M. Halle J. Bresnan G. Miller, MIT Press, Harvard, Massachusets (1978).

[Carbonell79] Carbonell, J. G., "Towards a Self-Extending Parser," pp. 3-7 in *Proceedings 17th Annual Meeting of the Association for Computational Linguistics,* La Jolla, California (1979).

[Dyer83] Dyer, Michael G., *In-Depth Understanding: A Computer Model of Integrated Processing for Narrative Comprehension,* MIT Press, Cambridge, MA (1983).

[Dyer85] Dyer, Michael G. and Uri Zernik, "Parsing Paradigms and Language Learning," in *Proceedings AI-85,* Long Beach, California (May 1985).

[Fass83] Fass, Dan and Yorick Wilks, "Preference Semantics, Ill-Formedness and Metaphor," *American Journal of Computational Linguistics* 9(3-4), pp.178-187 (1983).

[Fillmore68] Fillmore, C., "The Case for Case," pp. 1-90 in *Universals in Linguistic Theory,* ed. E. Bach R. Harms, Holt, Reinhart and Winston, Chicago (1968).

[Granger77] Granger, R. H., "FOUL-UP: A Program That Figures Out Meanings of Words from Context," pp. 172-178 in *Proceedings Fifth IJCAI* , Cambridge, Massachusets (August 1977).

[Jacobs85] Jacobs, Paul S., "PHRED: A Generator for Natural Language Interfaces," UCB/CSD 85/198,. Computer Science Division, University of California Berkeley, Berkeley, California (January 1985).

[Kay79] Kay, Martin, "Functional Grammar," pp. 142-158 in *Proceedings 5th Annual Meeting of the Berkeley Linguistic Society,* Berkeley, California (1979).

[Langley82] Langley, Pat, "Language Acquisition Through Error Recovery," *Cognition and Brain Theory* 5(3), pp.211-255 (1982).

[Pereira80] Pereira, F. C. N. and David H. D. Warren, "Definite Clause Grammars for Language Analysis- A Survey of the Formalism and a Comparison with Augmented Transition Networks," *Artificial Intelligence* 13, pp.231-278 (1980).

[Reeker76] Reeker, L. H., "The Computational Study of Language Learning," in *Advances in Computers,* ed. M. Yovits M. Rubinoff, Academic Press, New York (1976).

[Riesbeck74] Riesbeck, C. K., "Computational Understanding: Analysis of Sentences and Context," Memo 238, AI Laboratory (1974).

[Schank77] Schank, Roger and Robert Abelson, *Scripts Plans Goals and Understanding,* Lawrence Erlbaum Associates, Hillsdale, New Jersey (1977).

[Ulm75] Ulm, Susan C., "The Separation Phenomenon in English Phrasal Verbs, Double trouble," 601, University of California Los Angeles (1975). M.A. Thesis.

[Wilensky81] Wilensky, R., "A Knowledge-Based Approach to Natural Language Processing: A progress Report," in *Proceedings Seventh International Joint Conference on Artificial Intelligence*, Vancouver, Canada (1981).

[Wilensky82] Wilensky, R., "Points: A Theory of Structure of Stories in Memory," pp. 345-375 in *Strategies for Natural Language Processing*, ed. W. G. Lehnert M. H. Ringle, Laurence Erlbaum Associates, New Jersey (1982).

[Wilensky84] Wilensky, R., Y. Arens, and D. Chin, "Talking to UNIX in English: an Overview of UC," *Communications of the ACM* **27**(6), pp.574-593 (June 1984).

[Zernik85] Zernik, Uri and Michael G. Dyer, *Failure-Driven Aquisition of Figurative Phrases by Second Language Speakers*, 1985. (submitted to publication).