Appears in Proceedings of the Eleventh National Conference of the American Association for Artificial Intelligence (AAAI-93), pp.817-822, Washington, D.C. July, 1993

Learning Semantic Grammars with Constructive Inductive Logic Programming *

John M. Zelle and Raymond J. Mooney
Department of Computer Sciences
University of Texas
Austin, TX 78712
zelle@cs.utexas.edu, mooney@cs.utexas.edu

Abstract

Automating the construction of semantic grammars is a difficult and interesting problem for machine learning. This paper shows how the semantic-grammar acquisition problem can be viewed as the learning of search-control heuristics in a logic program. Appropriate control rules are learned using a new first-order induction algorithm that automatically invents useful syntactic and semantic categories. Empirical results show that the learned parsers generalize well to novel sentences and out-perform previous approaches based on connectionist techniques.

Introduction

Designing computer systems to "understand" natural language input is a difficult task. The laboriously hand-crafted computational grammars supporting natural language applications are often inefficient, incomplete and ambiguous. The difficulty in constructing adequate grammars is an example of the "knowledge acquisition bottleneck" which has motivated much research in machine learning. While numerous researchers have studied computer acquisition of natural languages, most of this research has concentrated on learning the syntax of a language from example sentences [Wirth, 1989; Berwick, 1985; Wolff, 1982] In practice, natural, language systems are typically more concerned with extracting the meaning of sentences, usually expressed in some sort of case-role structure.

Semantic grammars, which uniformly incorporate both syntactic and semantic constraints to parse sentences and produce semantic analyses, have proven extremely useful in constructing natural language interfaces for limited domains [Allen, 1987]. Unfortunately, new grammars must be written for each semantic domain, and the size of the grammar required for more general applications can make manual construction infeasible. An interesting question for machine learning is whether such grammars can be automatically constructed from an analysis of examples in a given domain.

The semantic grammar acquisition problem presents a number of difficult issues. First, there is little agreement on what constitutes an "adequate" set of cases for sentence analysis; different tasks may require differing semantic representations. Therefore, the learning architecture must be general enough to allow mapping to (more or less) arbitrary meaning representations. Second, domain specific semantic constraints must be automatically recognized and incorporated into the grammar. This necessitates some form of constructive induction to identify useful semantic word and phrase classes. Finally, as in any learning system, it is crucial that the resulting grammar generalize well to unseen inputs. Given the generativity of natural languages, it is unreasonable to assume that the system will be trained on more than a small fraction of possible inputs.

In this paper we show how the problem of semantic grammar acquisition can be considered as learning control rules for a logic program. In this framework, the acquisition problem can be attacked using the techniques of inductive logic programming. We introduce a new induction algorithm that incorporates constructive induction to learn word classes and semantic relations necessary to support the parsing process. Empirical results show this to be a promising approach to the language acquisition problem.

Learning Case-Role Mapping

The Mapping Problem

Traditional case theory decomposes a sentence into a proposition represented by the main verb and various arguments such as agent, patient, and instrument, represented by noun phrases. The basic mapping problem is to decide which sentence constituents fill which roles. Though case analysis is only a part of the overall task of sentence interpretation, the problem is nontrivial even in simple sentences.

*This research was supported by the National Science Foundation under grant IRI-9102926 and the Texas Advanced Research Program under grant 003658114.

Consider these sentences from [McClelland and Kawamoto, 1986]:

1. The boy hit the window.
2. The hammer hit the window.
3. The hammer moved.
4. The boy ate the pasta with the cheese.
5. The boy ate the pasta with the fork.

In the first sentence, the subject, `boy`, is an agent. In the second, the subject, `hammer`, is an instrument. The role played by the subject must be determined on the grounds that boys are animate and hammers are not. In the third sentence, the subject, `hammer`, is interpreted as a patient, illustrating the importance of the relationship between the surface subject and the verb. In the last two sentences, the prepositional phrase could be attached to the verb (making `fork` an instrument of `ate`) or the object (`cheese` is an accompaniment of `pasta`). Domain specific semantic knowledge is required to make the correct assignment.

Previous Approaches

Recent research in learning the case-role assignment task has taken place under the connectionist paradigm [Miikkulainen and Dyer, 1991; McClelland and Kawamoto, 1986]. They argue that proper case-role assignment is a difficult task requiring many independent sources of knowledge, both syntactic and semantic, and therefore well-suited to connectionist techniques.

Connectionist models, however, face a number of difficulties in handling natural language. Since the output structures are "flat" (nonrecursive) it is unclear how the embedded propositions in more sophisticated analyses can be handled. The models are also limited to producing a single output structure for a given input. If an input sentence is truly ambiguous, the system produces a single output that appears as a weighted average of the possible analyses, rather than enumerating the consistent interpretations. We believe that symbolic techniques are more appropriate, and our approach does not suffer from these deficiencies. In addition, empirical results demonstrate that our system trains faster and generalizes to novel inputs better than its neural counterparts.

Shift-Reduce Case-Role Parsing

Variations of shift-reduce parsing have proven practical for many symbolic natural language applications [Tomita, 1986]. Our system adopts a simple shift-reduce framework for case-role mapping [Simmons and Yu, 1992]. The process is best illustrated by way of example.

Consider the sentence: "The man ate the pasta." Parsing begins with an empty stack and an input buffer containing the entire sentence. At each step of the parse, either a word is shifted from the front of the input buffer onto the stack, or the top two elements

Action	Stack Contents
	[]
(shift)	[the]
(shift)	[man, the]
(1 det)	[[man, det:the]]
(shift)	[ate, [man, det:the]]
(1 agt)	[[ate, agt:[man, det:the]]]
(shift)	[the, [ate, agt:[man, det:the]]]
(shift)	[pasta, the, [ate, agt:[man, det:the]]]
(1 det)	[[pasta, det:the], [ate, agt:[man, det:the]]]
(2 pat)	[[ate, pat:[pasta, det:the], agt:[man, det:the]]]

Figure 1: Parsing "The man ate the pasta."

on the stack are popped and combined to form a new element which is pushed back onto the stack. The sequence of actions and stack states for our simple example is shown Figure 1. The action notation *(x label)*, indicates that the stack items are combined via the role, *label*, with the item from stack position, x, being the head.

An advantage of assuming such a constrained parsing mechanism is that the form of structure building actions is limited. The operations required to construct a given case representation are directly inferable from the representation. In general, a structure building action is required for each unique case-role that appears in the analysis. The set of actions required to produce a set of analyses is the union of the actions required for each individual analysis.

Overview of Chill

Our system, Chill, (Constructive Heuristics Induction for Language Learning) is a general approach to semantic grammar acquisition. The input to the system is a set of training instances consisting of sentences paired with the desired case representations. The output is a shift-reduce parser (in Prolog) which maps sentences into case representations. The parser may produce multiple analyses (on backtracking) for a single input sentence, allowing for true ambiguity in the training set.

The Chill algorithm consists of two distinct tasks. First, the training instances are used to formulate an overly-general parser which is capable of producing case structures from sentences. The initial parser is overly-general in that it produces many spurious analyses for any given input sentence. The parser is then specialized by introducing search control heuristics. These control heuristics limit the contexts in which certain program clauses are used, eliminating the spurious analyses. The following section details these two processes.

The Chill Algorithm

Constructing an Overly-General Parser

A shift-reduce parser is easily represented as a logic program. The state of the parse is reflected by the contents of the stack and input buffer. Each distinct parsing action becomes an operator clause that takes the current stack and input and produces new ones. The overly-general parser is built by translating each action inferable from the training problems into a clause which implements the action. For example, the clause implementing the *(1 agt)* action is:

```
op([S1,S2|SRest], Inp, [SNew|SRest], Inp) :-
     combine(S1, agt, S2, SNew).
```

Building a program to parse a set of training examples is accomplished by adding clauses to the `op` predicate. Each clause is a direct translation of a required parsing action. As mentioned above, the identification of the necessary actions is straight-forward. A particularly simple approach is to include two actions (e.g., *(1 agt)* and *(2 agt)*) for each role used in the training examples; any unnecessary operator clauses will be removed from the program during the subsequent specialization process.

Parser Specialization

General Framework The overly-general parser produces a great many spurious analyses for the training sentences because there are no conditions specifying when it is appropriate to use the various operators. The program must be specialized by including control heuristics which guide the application of operator clauses. This section outlines the basic approach used in CHILL. More detail on incorporating clause selection information in Prolog programs can be found in [Zelle and Mooney, 1993].

Program specialization occurs in three phases. First, the training examples are analyzed to construct positive and negative control examples for each operator clause. Examples of correct operator applications are generated by finding the first correct parsing of each training pair with the overly-general parser; any subgoal to which an operator is applied in a successful parse becomes a positive control example for that operator. A positive control example for any operator is considered a negative example for all operators that do not have it as a positive example.

In the second phase, a general first-order induction algorithm is employed to learn a control rule for each operator. This control rule comprises a Horn-clause definition that covers the positive control examples for the operator but not the negative. The induction algorithm used by CHILL is discussed in the following subsection.

The final step is to "fold" the control information back into the overly-general parser. A control rule is easily incorporated into the overly-general program by unifying the head of an operator clause with the head of the control rule for the clause and adding the induced conditions to the clause body. The definitions of any invented predicates are simply appended to the program. As an example, the *(1 agt)* clause of `op` is typically modified to:

```
op([A,[B,det:the]],C,[D],C) :-
    animate(B), combine(A,agt,B,D).
animate(boy). animate(girl). ...
```

Here, a new predicate has been invented representing the concept "animate."[1] This rule may be roughly interpreted as stating: "If the stack contains two items, the second of which is a completed noun phrase whose head is animate, then attach this phrase as the agent of the top of stack."

Inducing Control Rules The induction task is to generate a Horn-clause definition which covers the positive control examples for an operator, but does not cover the negative. There is a growing body of research in inductive logic programming which addresses this problem. Our algorithm implements a novel combination of bottom-up techniques found in systems such as CIGOL [Muggleton and Buntine, 1988] and GOLEM [Muggleton and Feng, 1992] and top-down methods from systems like FOIL [Quinlan, 1990] and CHAMP [Kijsirikul *et al.*, 1992].

```
Let Pos := Positive Examples
Let Neg := Negative Examples
Let Def := Positive examples as unit clauses.
Repeat
  Let OldDef := Def
  Let S be a sampling of pairs of clauses in OldDef
  Let OldSize := TheorySize(OldDef)
  Let CurrSize := OldSize
  For each pair of clauses <C1, C2> in S
    Find_Generalization(C1,C2,Pos,Neg,NewClause,NewPreds)
    Reduce_Definition(Pos,OldDef,NewClause,NewPreds,NewDef)
    If TheorySize(NewDef) < CurrSize then
      CurrSize := TheorySize(NewDef)
      Def := NewDef
Until CurrSize = OldSize % No compaction achieved
Return Def
```

Figure 2: CHILL Induction Algorithm

Space does not permit a complete explanation of the induction mechanism, but the general idea is simple. The intuition is that we want to find a small (hence general) definition which discriminates between the positive and negative examples. We start with a most specific definition (the set of positive examples) and introduce generalizations which make the definition more compact (as measured by a CIGOL-like size metric). The search for more general definitions is carried out in a hill-climbing fashion. At each step, a number of possible generalizations are considered; the one producing the greatest compaction of the theory is implemented, and the process repeats. The basic algorithm is outlined in Figure 2.

The heart of the algorithm is the *Find_Generalization* procedure. It takes two clauses in the current definition and constructs a new clause that (empirically) subsumes them and does not cover any

[1]Invented predicates actually have system generated names. They are renamed here for clarity.

negative examples. *Reduce_Definition* proves the positive examples using the current definition augmented with the new generalized clause. Preferential treatment is given to the new clause (it is placed at the top of the Prolog definition) and any clauses which are no longer used in proving the positive examples are deleted to produce the reduced definition.

Find_Generalization employs three levels of effort to produce a generalization. The first is construction of the least general generalization (LGG) [Plotkin, 1970] of the input clauses. If the LGG covers no negative examples, further refinement is unnecessary. Otherwise, the clause is too general, and an attempt is made to refine it using a FOIL-like mechanism which adds literals derivable either from background or previously invented predicates. If the resulting clause is still too general, it is passed to *Invent_Predicate* which invents a new predicate to discriminate the positive examples from the negatives which are still covered.

Predicate invention is carried out in a manner analogous to CHAMP. The first step is to find a minimal-arity projection of the clause variables such that the set of ground tuples generated by the projection when using the clause to prove the positive examples is disjoint with the ground tuples generated in proving the negative examples. These ground tuples form positive and negative example sets for the new predicate, and the top-level induction algorithm is recursively invoked to create a definition of the predicate.

Experimental Results

The crucial test of any learning system is how well the learned concept generalizes to new input. CHILL has been tested on a number of case-role assignment tasks.

Comparison with Connectionism

In the first experiment, CHILL was tried on the baseline task reported in [Miikkulainen and Dyer, 1991] using 1475 sentence/case-structure examples from [McClelland and Kawamoto, 1986] (hereafter referred to as the M & K corpus). The corpus was produced from a set of 19 sentence templates generating sentences/case-structure pairs for sentences like those illustrated above. The sample actually comprises 1390 unique sentences, some of which allow multiple analyses. Since our parser is capable (through backtracking) of generating all legal parses for an input, training was done considering each unique sentence as a single example. If a particular sentence was chosen for inclusion in a training or testing set, the pairs representing all correct analyses of the sentence were included in that set.

Training and testing followed the standard paradigm of first choosing a random set of test examples (in this case 740) and then creating parsers using increasingly larger subsets of the remaining examples. All reported results reflect averages over five trials. During testing, the parser was used to enumerate all analyses for a given test sentence. Parsing of a sentence can fail in two ways: an incorrect analysis may be generated, or a correct analysis may not be generated. In order to account for both types of inaccuracy, a metric was introduced to calculate the "average correctness" for a given test sentence as follows: $Accuracy = (\frac{C}{P} + \frac{C}{A})/2$ where P is the number of distinct analyses produced, C is the number of the produced analyses which were correct, and A is the number of correct analyses possible for the sentence.

CHILL performs very well on this learning task as demonstrated by the learning curve shown in Figure 3. The system achieves 92% accuracy on novel sentences after seeing only 150 training sentences. Training on 650 sentences produces 98% accuracy.

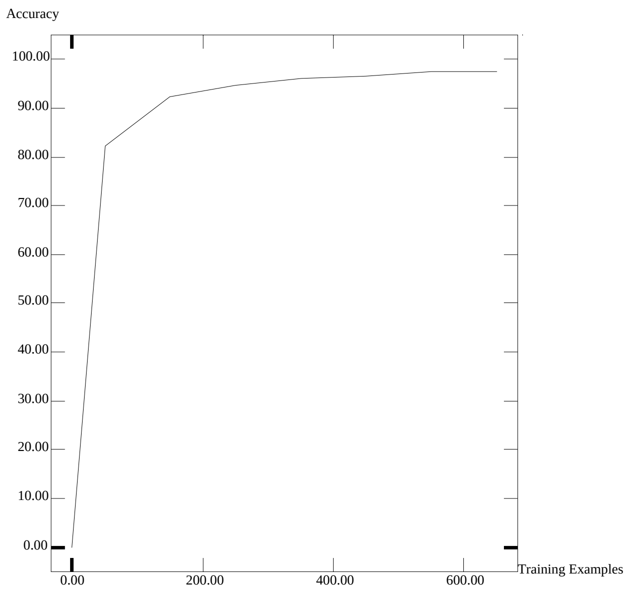

Figure 3: M & K corpus Accuracy

Direct comparison with previous results is difficult, as connectionist learning curves tend to be expressed in terms of low level measures such as "number of correct output bits." The closest comparison can be made with the results in [Miikkulainen and Dyer, 1991] where an accuracy of 95% was achieved at the "word level" training with 1439 of the 1475 pairs from the M & K corpus. Since the output contains five slots, assuming independence of errors gives an estimate of 0.95^5 or 78% completely correct parses. Interestingly, the types of inaccuracies differ substantially between systems. Neural networks always produce an output many of which contain minor errors, whereas CHILL tends to produce a correct output or none at all. From an engineering standpoint, it seems advantageous to have a system which "knows" when it fails; connectionists might be more interested in failing "reasonably."

With respect to training time, the induction algorithm employed by CHILL is a prototype implemented in Prolog. Running on a SparcStation 2, the creation of the parsers for the examples in this paper required from a few minutes to half an hour of CPU time. This compares favorably with backpropagation training times usually measured in hours or days.

It is also noteworthy that CHILL consistently invented interpretable word classes. One example, the invention of `animate`, has already been presented. This concept is implicit in the analyses presented to the system, since only animate objects are assigned to the agent role. Other invented classes clearly picked up on the distribution of words in the input sentences. The system regularly invented semantic classes such as `human`, `food`, and `possession` which were used for noun generation in the M & K corpus.

Phrase classes useful to making parsing distinctions were also invented. For example, the structure `instrumental_phrase` was invented as:

```
instr_phrase([]).
instr_phrase([with, the, X]) :- instrument(X).
instrument(fork). instrument(bat). ...
```

It was not necessary in parsing the M & K corpus to distinguish between instruments of different verbs, hence instruments of various verbs such as hitting and eating are grouped together. Where the semantic relationship between words is required to make parsing distinctions, such relationships can be learned. CHILL created one such relation: `can_possess(X,Y) :- human(X), possession(Y)`; which reflects the distributional relationship between humans and possessions present in the M & K corpus. Notice that this invented rule itself contains two invented word categories.

Although there is no *a priori* reason to suppose CHILL must invent interpretable categories, the naturalness of the invented concepts supports the empirical results indicating that CHILL is making the "right" generalizations.

A More Realistic Domain

The M & K corpus was designed specifically to illustrate the case mapping problem. As such, it does not necessarily reflect the true difficulty of semantic grammar acquisition for natural language applications. Another experiment was designed to test CHILL on a more realistic task. A portion of a semantic grammar was "lifted" from an extant prototype natural language database designed to support queries concerning tourist information [Ng, 1988]. The portion of the grammar used recognized over 150,000 distinct sentences. A simple case grammar, which produced labellings deemed useful for the database query task, was devised to generate a sample of sentence/case-structure analyses. The example pair shown in Figure 4 illustrates the type of sentences and analyses used.

An average learning curve for this domain is shown in Figure 5. The curve shows generalization results on

```
Show me the two star hotels in downtown LA with double rates
below 65 dollars.

[show, theme:[hotels, det:the,
                      type:[star, mod:two],
                      loc:[la, casemark:in, mod:downtown],
                      attr:[rates, casemark:with, mod:double,
                                   less:[nbr(65), casemark:below,
                                                  unit:dollars]]]
       dative:me]
```

Figure 4: Example from Tourist Domain

500 sentences which differed from any used in training. The results are very encouraging. With only 50 training examples, the resulting parser achieved 93% accuracy on novel sentences. With 300 training examples, accuracy is 99%.

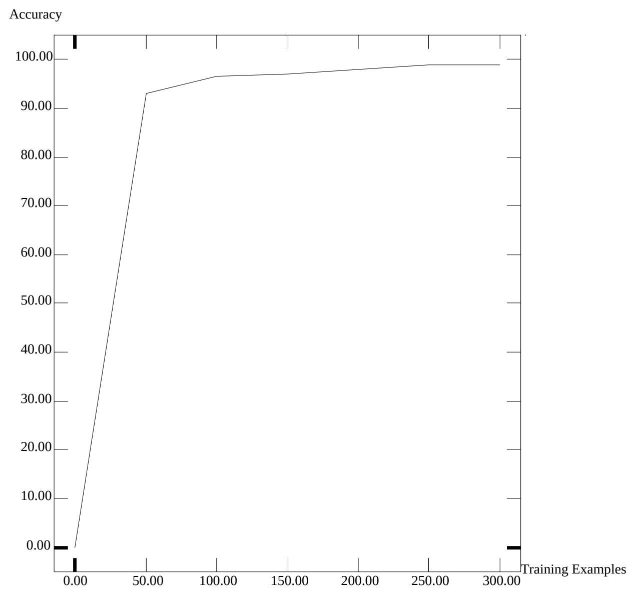

Figure 5: Tourist Domain Accuracy

Related Work

As noted in the introduction, most AI research in language acquisition has not focused on the case-role mapping problem. However, a number of language acquisition systems may be viewed as the learning of search control heuristics. Langley and Anderson [Langley, 1982; Anderson, 1983] have independently posited acquisition mechanisms based on learning search control in production systems. These systems were cognitively motivated and addressed the task of language generation rather than the case-role analysis task examined here.

Berwick's LPARSIFAL [Berwick, 1985] acquired syntactic parsing rules for a type of shift-reduce parser. His system was linguistically motivated and incorporated many constraints specific to the theory of language assumed. In contrast, CHILL uses induction techniques to avoid commitment to any specific model of grammar.

More recently an exemplar-based acquisition system for the style of case grammar used in CHILL is described in [Simmons and Yu, 1992]. Unlike CHILL, their system depends on an analyst to provide appropriate word classifications and requires detailed interaction to guide the parsing of training examples.

Recent corpus-based natural-language research has addressed some issues in automated dictionary construction [Lehnert *et al.*, 1992; Brent and Berwick, 1991]. These systems use manually constructed parsers to "bootstrap" new patterns from analyzable text. They do not employ machine learning techniques to generalize the acquired templates or construct new features which support parsing decisions. In contrast, CHILL is a first attempt at applying modern machine learning methods to the more fundamental task of constructing efficient parsers.

Future Research

The generalization results in the experiments so far undertaken are quite encouraging; however, further testing on larger, more realistic corpora is required to determine the practicality of these techniques. Another avenue of research is "deepening" the analyses produced by the system. Applying our techniques to actually construct natural language systems will require either modifying the parser to produce final representations (e.g., database queries) or adding additional learning components which map the intermediate case structures into final representations.

Conclusion

Methods for learning semantic grammars hold the potential for substantially automating the development of natural language interfaces. We have presented a system that employs inductive logic programming techniques to learn a shift-reduce parser that integrates syntactic and semantic constraints to produce case-role representations. The system first produces an overly-general parser which it then constrains by inductively learning search-control heuristics that eliminate spurious parses. When learning heuristics, constructive induction is used to automatically generate useful semantic and syntactic classes of words and phrases. Experiments on two reasonably large corpora of sentence/case-role pairs demonstrate that the system learns accurate parsers that generalize well to novel sentences. These experiments also demonstrate that the system trains faster and produces more accurate results than previous, connectionist approaches and creates interesting and recognizable syntactic and semantic concepts.

References

Allen, James F. 1987. *Natural Language Understanding.* Benjamin/Cummings, Menlo Park, CA.

Anderson, John R. 1983. *The Architecture of Cognition.* Harvard University Press, Cambridge, MA.

Berwick, B. 1985. *The Acquisition of Syntactic Knowledge.* MIT Press, Cambridge, MA.

Brent, Micheal R. and Berwick, Robert C. 1991. Automatic acquisition of subcategorization frames from tagged text. In *Speech and Natrual Language: Proceedings of the DARPA Workshop.* Morgan Kaufmann. 342–345.

Kijsirikul, B.; Numao, M.; and Shimura, M. 1992. Discrimination-based constructive induction of logic programs. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, San Jose, CA. 44–49.

Langley, P. 1982. Language acquisition through error recovery. *Cognition and Brain Theory* 5.

Lehnert, W.; Cardie, C.; Fisher, D.; McCarthy, J.; Riolff, E.; and Soderland, S. 1992. University of massachusetts: Muc-4 test results and analysis. In *Proceedings of the Fourth DARPA Message Understanding Evaluation and Conference.* Morgan Kaufmann. 151–158.

McClelland, J. L. and Kawamoto, A. H. 1986. Mechnisms of sentence processing: Assigning roles to constituents of sentences. In Rumelhart, D. E. and McClelland, J. L., editors 1986, *Parallel Distributed Processing, Vol. II.* MIT Press, Cambridge, MA. 318–362.

Miikkulainen, R. and Dyer, M. G. 1991. Natural language processing with modular PDP networks and distributed lexicon. *Cognitive Science* 15:343–399.

Muggleton, S. and Buntine, W. 1988. Machine invention of first-order predicates by inverting resolution. In *Proceedings of the Fifth International Conference on Machine Learning*, Ann Arbor, MI. 339–352.

Muggleton, S. and Feng, C. 1992. Efficient induction of logic programs. In Muggleton, S., editor 1992, *Inductive Logic Programming.* Academic Press, New York. 281–297.

Ng, H. T. 1988. A computerized prototype natural language tour guide. Technical Report AI88-75, Artificial Intelligence Laboratory, University of Texas, Austin, TX.

Plotkin, G. D. 1970. A note on inductive generalization. In Meltzer, B. and Michie, D., editors 1970, *Machine Intelligence (Vol. 5).* Elsevier North-Holland, New York.

Quinlan, J.R. 1990. Learning logical definitions from relations. *Machine Learning* 5(3):239–266.

Simmons, R. F. and Yu, Y. 1992. The acquisition and use of context dependent grammars for English. *Computational Linguistics* 18(4):391–418.

Tomita, M. 1986. *Efficient Parsing for Natural Language.* Kluwer Academic Publishers, Boston.

Wirth, Ruediger 1989. Completing logic programs by inverse resolution. In *Proceedings of the European Working Session on Learning*, Montpelier, France. Pitman. 239–250.

Wolff, J. G. 1982. Language acquisition, data compression, and generalization. *Language and Communication* 2:57–89.

Zelle, J. M. and Mooney, R. J. 1993. Combining FOIL and EBG to speed-up logic programs. In *Proceedings of the Thirteenth International Joint conference on Artificial intelligence*, Chambery, France.