Learning a Second Language*

Steven L. Lytinen and Carol E. Moon
Department of Electrical Engineering and Computer Science
The University of Michigan
Ann Arbor, MI 48109

Abstract

We present a system, called IMMIGRANT, which learns rules about the grammar of a second language from instructions. We explore the implications of this task on the representation of linguistic knowledge in a natural language understanding system. We conclude that the internal representation of linguistic knowledge used in IMMIGRANT, which is unification-based, is more amenable to language learning from instructions than other representation schemes.

1 Introduction

This paper describes IMMIGRANT, a program which learns a second language from instructions. Initially, the program's knowledge base contains rules for understanding English sentences. Input to the program consists of sentences which describe linguistic rules of a second language. All input instructions describe a feature of the second language which is not found in English. Thus, the rules learned represent differences between the two languages. The program must understand these input sentences, build a representation of the instruction in terms of how it differs from a corresponding English rule, and then modify its rules about the grammar of the second language accordingly. After the program has read the set of instructions, it can process sentences which use grammatical constructions of the second language.

The purpose of this project is twofold. First, we wish to explore learning from instructions. Relatively little research has been directed at this type of learning (although there are some exceptions, e.g., Mostow, 1983, Kieras and Bovair, 1986). Clearly, however, it is an important method for acquiring knowledge. As people grow up, they are constantly being told new facts and rules about the world. People spend many years in classrooms, listening to instructors teach them. Obviously, people acquire a great deal of knowledge through communication with other people.

The second issue that this project addresses, and the issue that we will focus on in this paper, is to explore the constraints that language learning places on the representation of linguistic knowledge. There are numerous theories of grammar which have been proposed in linguistics and artificial intelligence. Which of these theories lend themselves most easily to the task of learning new grammar rules? Can we say anything about how grammatical knowledge is stored by looking at the way it is taught?

*This research was supported in part by a grant from the Horace H. Rackham School of Graduate Studies, The University of Michigan

2 Language Learning and Representation

What can the task of second language learning tell us about how linguistic knowledge should be represented? Let us answer this question by considering an example of the sort of instruction that IMMIGRANT ought to be able to process:

> In German, verbs come at the end of relative clauses.

Our program must build a representation of this statement, then use this representation to modify its parsing rules. It is fairly straightforward to represent this statement in the following way: we define the word "end" to mean the last (in the case of written text, the rightmost) location in a range, which will be specified by the object of the preposition "of" immediately following "end." This results in the following representation:

```
(LOC-RANGE VERB0 LOC0 LOC1)
(INSTANCE VERB0 VERB)
(LOC-RANGE REL0 LOC2 LOC1)
(INSTANCE REL0 RELATIVE-CLAUSE)
(AFTER LOC0 LOC2)
```

We are assuming that constituents in a sentence take up a certain range of locations within the sentence, specified by the second and third arguments of the predicate LOC-RANGE. We have represented the concept that verbs come at the end of relative clauses by specifying that the right boundary of the verb's range is the same as the right boundary of the relative clause's range. The phrase "the end of" also implies that the object after the preposition "of" contains the other object. This is represented by the assertion that the left boundary of the verb is to the right of (AFTER) the left boundary of the clause.

Since this representation would be fairly straightforward to generate from the text above, it would be nice if our system could also use this representation, or something close to it, to parse with. That way, the task of internalizing the rule would be relatively simple. The further away in form the internal parsing rules of the system are from this representation, the more difficult it will be for the system to learn from the instruction.

Let us consider the way in which grammatical information is typically represented in natural language systems, and see how close this is to the representation above.

Often, it is represented by means of context-free grammar rules, such as the following rules for English relative clauses:

C → REL V NP
C → REL NP V
...
REL → who, that, ...

As we can see, this representation is quite different from the earlier representation. Word order information is represented implicitly, by the order of the nonterminals in the right hand sides of the rules. In order to modify these rules correctly, we would have to devise a set of mapping rules, which told us how to change the right hand sides of the context-free rules depending on the predicates used in the representation of the instructions. In this case, the rule would be something like: "If the right boundary of constituent A is specified to be the same as the right boundary of constituent B, and A's left boundary is after B's left boundary, then look for productions whose left hand side is B and whose right hand side contains A. If A is not the rightmost thing on the right hand side, rewrite the rule so that it is."

Requiring mapping rules like this makes the internalization of instructions a difficult task. We would need a large number of mapping rules, telling us different changes to make to the context-free rules depending on the representation built by the parser. While it might be possible to come up with a complete set of these mapping rules, it would certainly be easier if they were not necessary. Instead, we would like to be able to use our initial representation more directly.

3 The Representation of Knowledge in IMMIGRANT

Grammatical knowledge in IMMIGRANT is represented in a format which corresponds much more closely to instructions such as our example above. The representation of linguistic knowledge that we are using is similar to what is used in unification grammars (Shieber, 1986). In this approach, we explicitly represent word order information, as well as information about the functional relations between words.

IMMIGRANT's representation of linguistic knowledge has another advantage. In typical natural language systems, different types of linguistic knowledge are represented differently. For example, syntactic information is often expressed in terms of a context-free grammar, while semantic information might be represented in terms of case frames or selectional restrictions, and pragmatic information might be given in a frame notation or first-order predicate calculus. However, in IMMIGRANT, the same unification-style notation is used to capture all these types of knowledge. Thus, semantic representations produced by IMMIGRANT's language understander look exactly the same as the system's syntactic rules.

Let us look at IMMIGRANT's initial knowledge about English relative clauses. For clauses in which the clause "gap" (i.e., the missing constituent in the clause) is the subject, the following rule holds:

```
[C]
  (1)             = REL               <1>
  (2)             = V                 <2>
  (3)             = NP                <3>
  (1 rborder)     = (2 lborder)       <4>
  (2 rborder)     = (3 lborder        <5>
  (2 head subj)   = (1 head mod)      <6>
  (2 head obj)    = (3 head)          <7>
  (rborder)       = (3 rborder)       <8>
  (lborder)       = (1 lborder)       <9>
```

This rule is for sentences such as "The man who saw Mary was John." The equivalent graphic representation of this rule is the directed acyclic graph (DAG) shown in Figure 1. Each of the above equations represents a constraint or fact about this particular type of an English clause. Items enclosed in parentheses indicate path names (i.e., sequences of slots) and how they should be filled. So, for instance, equation one states that a constituent in a clause (to be placed in the slot '1') must be a relative pronoun (REL). The fact that this is the leftmost constituent is explicitly represented by equation nine, which states that the left-hand border of (1), (1 LBORDER), is the same as the left-hand border (LBORDER) of the entire clause, (C). Equations four and five specify the adjacency relationships of REL, V, and NP. Thus, we have taken the same sort of approach to representing phrase structure information as Functional Unification Grammar (Kay, 1985), in that this information is part of the unification structures. We have achieved this, however, with adjacency links, which do not have the privileged status that Kay's PATTERN features had.

The functional relationships between the verb and noun phrases is also explicitly represented in our rules. Equation six indicates that the NP before the clause is the subject of the verb in the clause[1], while the seventh equation states that the head of the noun phrase within the clause is the object of the verb. In parsing, these two constraints would correspond to some primitive parsing actions, which would manipulate the semantic representations of the pronoun and verb and the noun phrase and verb accordingly.

Similarly, the system's initial rule for clauses in which the clause gap is the object is shown graphically in Figure 2. These two rules constitute the system's original rules for English clauses. The system assumes that these rules apply to German (or any other second language) unless it is instructed otherwise, so that English rules serve as default rules for German in this way.

The program has several knowledge bases for each of the languages it works with, all of which contain rules in the unification format. The rules are stored as DAGs in the parser. One rule base, the parser rule base, contains rules which the parser uses in producing its output, which is a DAG representing both the syntactic structure and the semantic representation of the input sentence. These are rules such as the ones we have seen for clauses in figures 1 and 2.

Knowledge about specific words in the input language

[1]This is accomplished by linking the representation of the NP with the Relative Pronoun via a MOD link (see Figure 1), which is added by the rule which combines NP's with relative clauses.

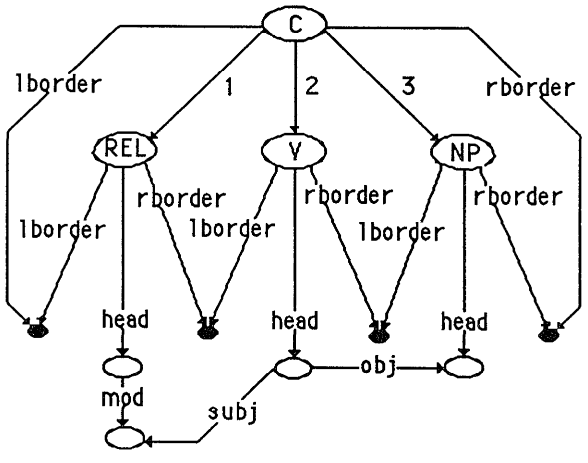

Figure 1: First English Relative Clause Rule

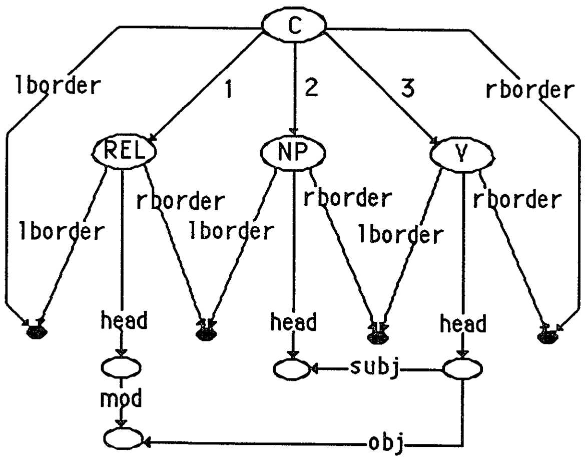

Figure 2: Second English Relative Clause Rule

are stored in a lexicon. This knowledge is encoded in a similar format. For example, the definition of the word "verb" would look like this:

```
VERB: N
      (head number) = SING
      (head person) = THIRD
      (head rep)    = V
```

For these rules the syntactic category is given in addition to a list of constraints. The first two constraints give grammatical information about the noun "verb." The last constraint gives a semantic representation of the word, in terms of the parser. That is, to the parser, a "verb" means the symbol V.

By storing all knowledge in the same format, the program can easily integrate new information. Since the parser produces a representation of the input rules which has the same format as all other knowledge in the system, the comparison of old and new information is easily facilitated.

4 Learning a Rule

IMMIGRANT's first step in learning a new rule is to parse an instruction, thereby producing a representation of the new rule in DAG form. We will not discuss the parsing of instructions in this paper, except to say that IMMIGRANT uses a combination of syntactic rules like the clause rules in Figures 1 and 2, as well as lexical entries such the one for "verb" presented earlier, to arrive at the representation.

During parsing, IMMIGRANT also categorizes the rule. This categorization affects the way that the rule will be subsequently processed. We have identified several rule types thus far, including the two categories that we discuss in this paper: *constraint-addition* rules and *reordering* rules.

Independent of rule type, the next step in processing the instruction is to identify which English rule(s) are relevant to the new rule. Once this is done, IMMIGRANT attempts to unify the new rule with the existing English rule(s). What happens at this point depends on the category of the new rule.

First we will consider the case of constraint-addition rules. These rules provide additional constraints for a preexisting English rule. An example is the German rule that cases must agree between various constituents in a sentence, such as:

The case of a prepositional object must match the case required by the preposition.

For these types of rules, since nothing in the new rule contradicts the existing English rule, unification of the new rule with the English rule succeeds. The result of unification provides the new rule for the second language, ready to be used in parsing. The parser rule base for the second language is updated with the result of the unification, replacing the existing English rule. For the case rule above, the existing English rule is the one which combines a preposition with its object:

```
PP:
   (1)  = PREP
   (2)  = NP
   (1 rborder) = (2 lborder)
   (lborder) = (1 lborder)
   (rborder) = (2 rborder)
   (head) = (1 head)
   (head prep-obj) = (2 head)
```

IMMIGRANT's parse of this input produces the representation in Figure 3. It indicates that a preposition's CASE property must be equal to (i.e., unify with) the CASE property of its PREP-OBJ. Since prepositional phrases are not mentioned in the instructions, there is no reference to a prepositional phrase in the representation. However, since IMMIGRANT knows that English prepositions are only found in prepositional phrases, the PREP node in the representation of the instruction is unified with

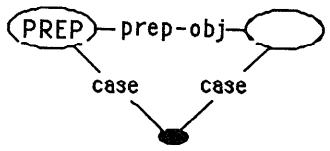

Figure 3: Representation Produced by IMMIGRANT Parser for Case Agreement Example

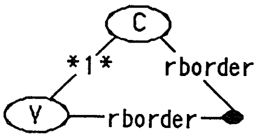

Figure 4: Representation Produced by IMMIGRANT Parser for Relative Clause Example

the PREP node in the PP rule above. Unification succeeds, resulting in the addition of the following constraint to the existing English rule:

```
(1 case) = (head prep-obj case)
```

This becomes the program's rule for German prepositional phrases.

Our representation provides a very natural mechanism for rules such as these, which add restrictions to existing English rules. Constraints, such as those for case, can be explicitly represented in the rules and simply need to be appended to the existing rule in order to form the new rule.

For other types of rules, unification fails because these rules contain information which contradicts an existing English rule. Reordering rules specify that constituents in the second language are not ordered in the same way as in English. Our German relative clause rule is of this type:

In German, verbs come at the end of relative clauses.

IMMIGRANT's parse of this input produces the representation in Figure 4. It indicates that the right hand border of the clause is equal to the right hand border of the verb; i.e., the verb is the right-most constituent of the clause. Since the input sentence does not specify a specific relationship between the clause and the verb (except to imply that the clause contains the verb), the variable *1* is used in the representation. The variable will be instantiated when the parser reexamines its old rules.

Once the instruction is parsed, the representation is unified with each of the system's original English clause rules (which were shown in Figures 1 and 2). During unification, the variable *1* is instantiated with the arc label from the

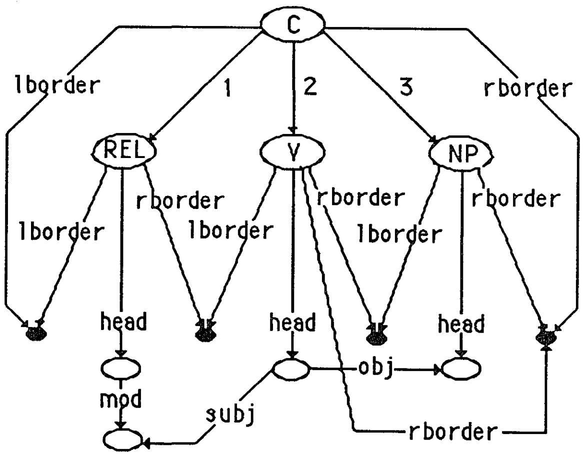

Figure 5: Unification Failure

original rule. This time, unification fails because of the incompatability of adjacency links due to reordering, so IMMIGRANT knows that a modification of the original rule must be made to form the new rule. For the first rule, which describes English relative clauses with no subject, the unification failure is due to the following set of constraints:

```
(2 rborder) = (rborder)
(2 rborder) = (3 lborder)
(3 rborder) = (rborder)
```

The problem is that the right border for the second constituent is equal to two different nodes and that two constituents claim to occur last in the relative clause. This is shown graphically in Figure 5.

To resolve the inconsistency, the system relies on the type of rule to tell it what to do. In this case, since the rule is a reordering rule, the required change to the existing rules must involve changing the paths indicating adjacency relationships. For this type of rule, having constituent ordering information represented explicitly aids in determining what part of the old rule needs to change and how it needs to be changed.

Armed with this knowledge, IMMIGRANT knows that it must alter an adjacency rule (i.e., a rule involving RBORDER and/or LBORDER links). Thus, it must throw out one of the three constraints from above. Since adjacency links from the new rule must be preserved, (presumably the instructor is not giving faulty instructions), precedence is given to the newer constraint, (2 rborder) = (rborder). This means the other two equations must be modified. For the most part, the system is able to maintain the order of the original DAG. It adds the new constraint and then readjusts the adjacency links for all nodes so that the links are consistent. This reordering results in the rule in Figure 6. This will be our new rule for German relative clauses with missing subjects. It reflects the new information about the verb being at the end of the clause.

There is a second relative clause rule in the English

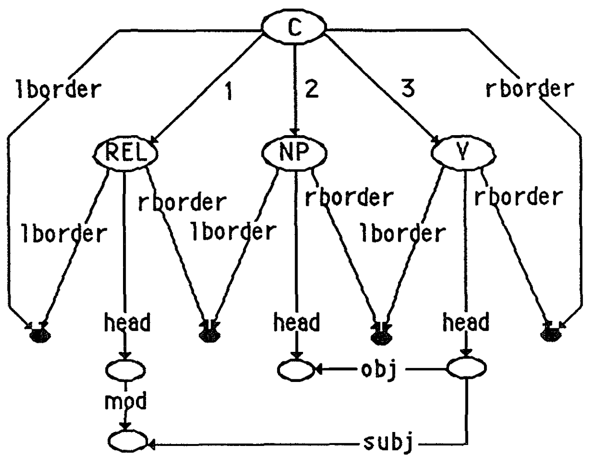

Figure 6: New Rule for German Relative Clauses

parser rule base that must also be unified with the new instruction. For the second rule, the reordering turns out to be a degenerate case, since the verb is already at the end of this type of English relative clause. Therefore, unification succeeds in this case[2], and the result is added to the German rule base. After these two rules are added, the system is ready to accept German sentences containing German relative clause constructions.

5 Conclusion and Future Plans

The integration of new information is facilitated by the use of the same representation scheme for both new and old knowledge. Not only can these two kinds for information be compared easily, but the program can also utilize the same procedure for learning as it does for understanding. It uses the unification procedure to interpret the input sentence stating a rule about a second language. This exact procedure is later used to combine the new information with the old.

Currently, we have the details worked out for a small number (about fifteen) of examples of the learning of rules such as the one above. Our immediate future plans are to implement many more examples of the learning of simple linguistic rules, to see if our representation can accommodate a broad range of grammatical modifications.

We have also begun to extend the project by studying the role that examples can play in learning from instructions. Often, when a tutor teaches a student a new rule, he accompanies the instructions with an example of when the new rule should be used. We have begun to try to understand the role of these examples better, and to expand our program so that it can learn from them.

It seems that examples can play many roles. First, they can be used by the learner to guide the process of finding which constraints from an existing rule must be modified.

[2]In fact, the English rule is not modified at all by this unification, since the new rule is redundant with the old one.

We can see this if we reconsider the processing of the relative clause instructions discussed earlier. If these instructions were accompanied by an example German sentence containing a relative clause, IMMIGRANT could simply parse the German example using its existing English relative clause rules, by relaxing the adjacency constraints on these rules[3]. In the syntactic representation of the German parse, the sub-DAG for the clause in the sentence would contain, among other things, an instantiation of the new German rule for clauses. This happens because the border constraints from the initial sentence will propagate up the DAG, eventually filling in the border links that were taken away from the English rule. The German clause rule could then be extracted from the parse of the example. This approach could limit the amount of search that the program currently has to do in order to find the relevant existing English rules which must be changed.

In addition to the grammatical rules which we have discussed in this paper, we also plan to work with instructions about word meanings in a second language, such as:

> In German, the verb "haben" with the adverb "gern" means "like."

Again, it seems that an example accompanying this instruction could help to facilitate the learning of the new rule. If both a German example using "haben gern" and its English translation were given, IMMIGRANT could parse the English translation, unifying the resulting semantic representation with its parse of the German sentence. The new rule for "haben gern" could be extracted from the result of unification, causing IMMIGRANT to modify its existing rules for "haben" (assuming it already had rules for this word). For this instruction, then, we would need both the English and the German parse of the example. The constraints on where "haben," "gern," and the object of "like" must occur in the word order from this new construction would be derived from the examples.

References

Kay, M. (1985). Parsing in functional unification grammar. In *Natural Language Parsing: Psychological, Computational, and Theoretical Perspectives.* Cambridge University Press, Cambridge, England, pp. 251-278.

Kieras, D., and Bovair, S. (1986). The acquisition of procedures from text: A production-system analysis of transfer of training. *Journal of Memory and Language 25*, pp. 507-524.

Mostow, J. (1983). Operationalizing advice: A problem-solving model. In *Proceedings of the International Machine Learning Workshop*, University of Illinois, June 1983.

Shieber, S. (1986). *An Introduction to Unification-based Styles of Grammar.* CSLI, Palo Alto, CA.

[3]Knowledge as to what type of constraints to relax would come from the categorization of instructions. In this case, since the new rule is a reordering rule, adjacency constraints would be relaxed.