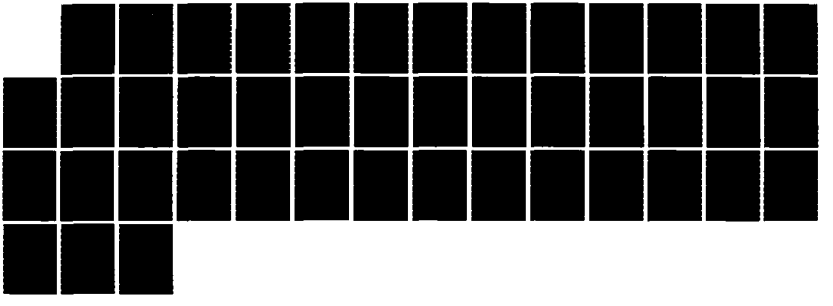
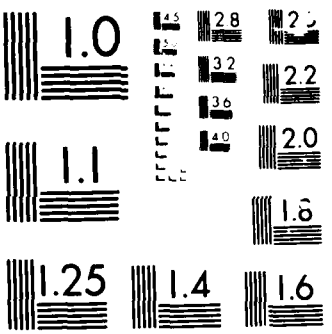
AD-A169 501 LANGUAGE ACQUISITION AND MACHINE LEARNING(U) CALIFORNIA 1/1
UNIV IRVINE DEPT OF INFORMATION AND COMPUTER SCIENCE
P LANGLEY ET AL. 01 FEB 86 UCI-ICS-TR-86-12
UNCLASSIFIED N00014-84-K-0345 F/G 9/2 NL




1.0
1.1
1.25
1.4
1.6
2.0
2.2
2.8
3.2
3.6
4.0

Information and Computer Science

LANGUAGE ACQUISITION AND MACHINE LEARNING

Pat Langley
Irvine Computational Intelligence Project
Department of Information & Computer Science
University of California, Irvine, CA 92717

Jaime G. Carbonell
Department of Computer Science
Carnegie–Mellon University
Pittsburgh, PA 15213

TECHNICAL REPORT

UNIVERSITY OF CALIFORNIA
IRVINE

12

LANGUAGE ACQUISITION AND MACHINE LEARNING

Pat Langley
Irvine Computational Intelligence Project
Department of Information & Computer Science
University of California, Irvine, CA 92717

Jaime G. Carbonell
Department of Computer Science
Carnegie–Mellon University
Pittsburgh, PA 15213

Technical Report 86–12

February 1, 1986

To appear in B. MacWhinney (Ed.), *Mechanisms of Language Acquisition.* Hillsdale, N.J.: Lawrence Erlbaum Associates, 1986.

We would like to thank Brian MacWhinney and Jeff Sokolov for their comments on an early draft of the chapter. We also thank Doug Fisher and Dan Easterlin for discussions that led to our framework for research in machine learning.

This research was supported by Contract N00014–84–K–0345 from the Information Sciences Division, Office of Naval Research. Approved for public release; distribution unlimited. Reproduction in whole or part is permitted for any purpose of the United States Government.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER Technical Report No. 4	2. GOVT ACCESSION NO. AD-A169581	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* Language Acquisition and Machine Learning		5. TYPE OF REPORT & PERIOD COVERED Interim Report 6/85–12/85
		6. PERFORMING ORG. REPORT NUMBER UCI–ICS Technical Report 86–12
7. AUTHOR*(s)* Pat Langley and Jaime G. Carbonell		8. CONTRACT OR GRANT NUMBER*(s)* N00014–84–K–0345
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Information & Computer Science University of California, Irvine, CA 92717		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Information Sciences Division Office of Naval Research Arlington, Virginia 22217		12. REPORT DATE February 1, 1986 13. NUMBER OF PAGES 38
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*		15. SECURITY CLASS. *(of this report)* Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

To appear in B. MacWhinney (Ed.), *Mechanisms of Language Acquisition*. Hillsdale, N.J.: Lawrence Erlbaum Associates, 1986.

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

language acquisition
learning from examples
heuristics learning
conceptual clustering
learning macro-operators
aggregation
characterization
negative instances

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

OVER

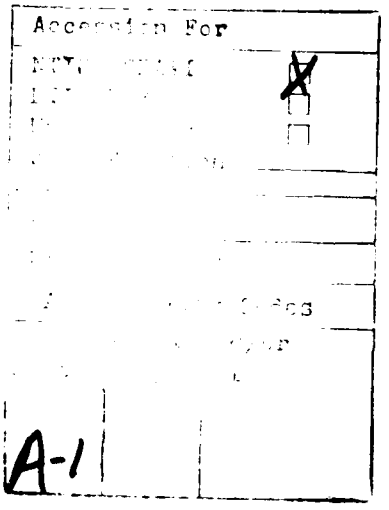
DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

20. ABSTRACT

In this paper, we review recent progress in the field of machine learning and examine its implications for computational models of language acquisition. As a framework for understanding this research, we propose four component tasks involved in learning from experience – aggregation, clustering, characterization, and storage. We then consider four common problems studied by machine learning researchers – learning from examples, heuristics learning, conceptual clustering, and learning macro-operators – describing each in terms of our framework. After this, we turn to the problem of grammar acquisition, relating this problem to other learning tasks and reviewing four AI systems that have addressed the problem. Finally, we note some limitations of the earlier work and propose an alternative approach to modeling the mechanisms underlying language acquisition.



Introduction

Despite its apparent complexity, nearly every human learns to use language in the first ten years of life, and as scientists we would like some theory to explain this phenomenon. Ideally, such a theory should explain the *processes* that lead from naive to sophisticated language use, just as chemical theories explain the processes involved in reactions. The natural places to turn for such process descriptions are the fields of artificial intelligence (AI) and cognitive science. In particular, one should look to machine learning, the subfield of artificial intelligence concerned with computational approaches to learning – i.e., with processes that lead to improved performance over time.

Over the past two decades, machine learning researchers have made considerable progress in understanding the nature of learning mechanisms, and many of their results are relevant to models of human behavior. In this paper, we examine these results and their implications for theories of language acquisition. We begin by reviewing four basic learning tasks that have been the focus of the machine learning work, describing each in terms of a common framework. We then turn to the task of grammar learning, comparing and contrasting it with other learning tasks. After this, we review some earlier computational models of grammar learning and consider some drawbacks of these models. Finally, we outline a new approach to modeling language acquisition that we hope will overcome these limitations.

An Overview of Machine Learning Research

In order to give the reader a better feel for the nature of machine learning research and its implications for models of language acquisition, let us begin by considering some common tasks that researchers in this field have addressed. We will consider the tasks in roughly historical order, based on the periods at which they first drew the major attention of machine learning researchers. Using this ordering, we have the tasks of learning from examples, learning search heuristics, conceptual clustering, and learning macro-operators. This list is not exhaustive, but the majority of AI learning research has focused on these tasks.

Considerable work has also been carried out on the task of learning grammars. Interest in this area emerged early in the history of machine learning and has remained active until the present. Despite its early role in the field, we will delay our discussion of grammar learning (and thus violate our historical ordering) so we can better see its relation to other work. However, before moving on to specific tasks, let us first present a general framework within which each task can be viewed.

The Components of Learning from Experience

Any attempt to define *learning* is as doomed to failure as attempts to define *life* and *love*. One can certainly generate a formal definition, but others can always find intuitive examples that fall outside the specified conditions or find counterexamples that fall within them. Rather than trying to define learning in general, we will focus on the more con-

strained issue of learning from experience. Most of the research in machine learning has focused on this class of problems, as opposed to learning by being told or learning by deduction from known facts. Will we not attempt to define the task of learning from experience; instead we will specify some components or subproblems which must be addressed by any system that learns from experience.

As we will see, these components are designed to characterize the five basic learning tasks that we consider in the following pages. It is possible that additional dimensions are required, but we will not know this until someone formulates a new learning task that forces expansion of the framework. Of course, other frameworks are possible that divide the learning problem along orthogonal dimensions. However, we will see that the current framework leads to useful insights about the nature of the language acquisition task, making it sufficient for our present goals. The four basic components of learning from experience are:

- *Aggregation* – the learner must identify the objects from which he will form rules or hypotheses; i.e., he must determine the appropriate *part-of* relations.[1] For instance, in understanding a visual scene, the viewer must identify the basic objects and their components. Similarly, in language acquisition, one must first group utterances into component sound-sequences (words). Thus, one may aggregate over either spatial or temporal descriptions.
- *Clustering* – the learner must identify which objects or events should be grouped together into a class; i.e., he must determine the appropriate *instance-of* relations, or generate an *extensional* definition of the rule or hypothesis. For example, a concept learner must divide objects into instances and non-instances of the concept being learned.
- *Characterization* – the learner must formulate some general description or hypothesis that characterizes instances of the rule; i.e., he must generate an *intensional* definition of the rule or hypothesis.[2] For instance, the task of language acquisition requires one to move beyond specific sentences, and to formulate general rules or grammars.
- *Storage/Indexing* – the characterization of the rule must be stored in some manner that lets one retrieve it when appropriate. For example, one may store an acquired problem solving heuristic in some form of discrimination network.

---

[1] Some readers will prefer the term *segmentation* to the term *aggregation*, but this is simply a matter of perspective. One can view part-of relations as being established in either a top-down or a bottom-up manner.

[2] This is often called the *generalization* problem. We will avoid this term because it has has two distinct meanings within machine learning. The first sense includes the process of moving from a specific hypothesis to a more general one, and is the opposite of *discrimination* learning. The second sense includes *any* process for constructing a general rule from data, and encompasses both discrimination and the first sense of *generalization*. We intend the term *characterization* to replace this second (more general) sense.

As we proceed, we will see examples of these components in concept learning, in procedural learning, and in grammar learning. All learning tasks include the components in one form or another, and all learning systems must address them in some sense.

However, in many tasks one or more components have been idealized out; that is, the solution to a subproblem is either provided by some outside source or can be effectively ignored. For instance, the programmer may divide the learner's input into distinct objects, thus solving the aggregation problem. Similarly, the learner may acquire only a few characterizations, so that storage issues are not significant. We will see examples of these and other simplifications shortly, but Table 1 summarizes the basic results of this analysis, listing the relevant components of the five learning tasks we will consider.

TABLE 1

Relevant Components of Machine Learning Tasks

Learning task	Relevant components
Learning from examples	characterization
Learning search heuristics	clustering, characterization
Conceptual clustering	clustering, characterization, storage
Learning macro-operators	aggregation
Grammar learning	aggregation, clustering, characterization, storage

Learning from Examples

The task of learning concepts from examples is the most widely studied problem in machine learning. Research on this task addresses the question of how one forms concepts from examples presented by a tutor. The general version of this task may be stated:

- *Given:* One or more classes or concepts, along with a set of instances or examples for each class. E.g., one might be given instances and non-instances of the concept *uncle*.
- *Find:* Some description or rule that correctly predicts the class to which each instance should be assigned. In the *uncle* example, one would hypothesize some general description of this concept.

In other words, given an extensional definition of one or more concepts, one must generate intensional definitions for each of those concepts. These descriptions must satisfy two constraints – each instance of a class must be covered by the description of that class, and no instance of a class may be covered by the description of any other class. Michalski (1983) has called the first of these the *completeness* condition and the second of these the *consistency* condition. In general, the description is expected to correctly classify instances that were not in the training set, so the learner must move beyond simple summaries of the original data.

The simplest version of learning from examples involves formulating a description for a single concept. In this case, we use the term *positive instances* for all examples of the concept and the term *negative instances* for all non-examples. Let us consider a simple task in which objects can be described in terms of only two attributes – `size` and `shape`. The learner might be told that a `large circle` and a `large square` are positive instances of the concept to be learned, while a `small circle` is a negative instance. One hypothesis that accounts for these data (though not the only one) is that the concept is simply `large`; this description covers the positive instances but not the negative instance. Concept learning tasks of this type were widely studied in psychology (e.g., Bruner, Goodnow, & Austin, 1956) before they were adopted by the machine learning community.

The task of learning multiple concepts from examples can be reduced to the simpler problem of learning a single concept, repeated a number of times. Focusing on one of the classes, we label all objects associated with this class as positive instances, and label all objects associated with other classes as negative instances. The task of learning this single concept involves formulating some description which covers all positive instances but none of the negative instances. Suppose we repeat this process for each of the other classes, in each case producing a set of positive and negative instances, and generating a concept description for that class. The result of this scheme is a set of concept descriptions, each complete and consistent with respect to the others.[3] Since the multiple concept learning task can be subdivided in this manner, most researchers have focused on the task of learning single concepts, and we will do so here as well.

Learning from examples can be viewed as an idealized version of the general task of learning from experience. Since the tutor provides the basic objects upon which rules or descriptions are based, the aggregation subproblem is bypassed. Since the tutor assigns instances to classes for the learner, the clustering subproblem is trivialized. Finally, since only a few concepts are learned, the resulting descriptions can be stored in a simple list, thus avoiding the subproblem of storage and indexing.

In other words, the task of learning from examples can be viewed as "distilled" characterization, and it is undoubtedly for this reason that it proved so popular in the early days of machine learning research. Focusing on this simplified task let researchers deal with characterization issues to the exclusion of complicating factors, much as the early physicists ignored nuisances like air resistance. This strategy led to a variety of interesting methods for formulating general descriptions from positive and negative instances (Winston, 1975; Hayes-Roth & McDermott, 1978; Mitchell, 1982; Anderson, 1983), which later proved invaluable in studying more complex learning tasks.

Since these characterization methods have been widely used in the work on more complex learning tasks, we should briefly review the set of methods that have been developed. The vast majority of methods rely on the insight that the set of descriptions or hypotheses considered during the characterization process can be partially ordered according to their *generality*. However, since this is only a *partial* ordering, multiple paths exist through the

[3] This approach assumes that classes must be disjoint; this assumption is not necessary, but is very common among machine learning researchers.

space of hypotheses, and this leads to *search*. Even very simple spaces are only partially ordered along the dimension of generality; e.g., Figure 1 shows the space for the `size/shape` task described above. As a result, most characterization methods carry out a systematic search through the hypothesis space, but they differ widely in the details of this search.

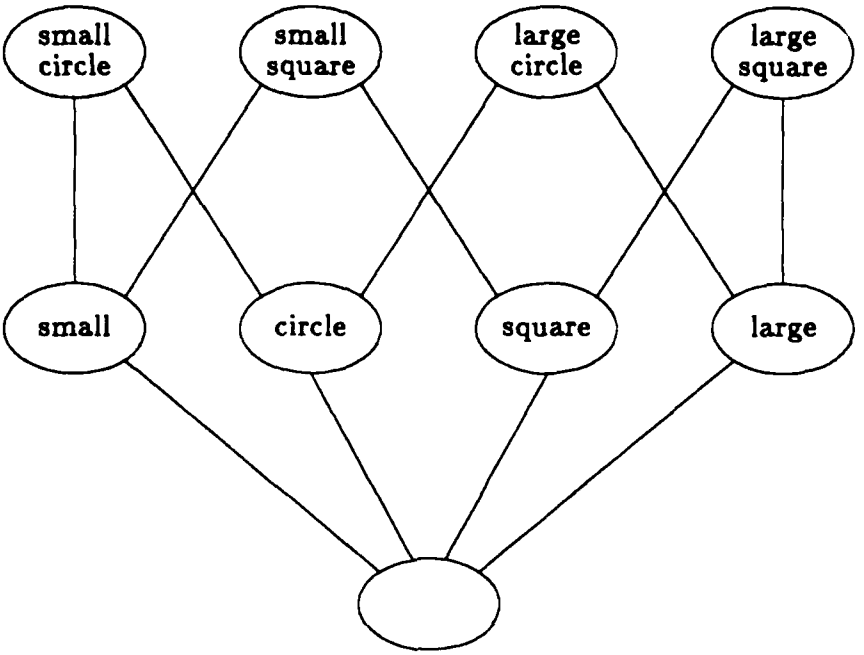


Figure 1. A partially ordered space of hypotheses.

For instance, one can start with very specific hypotheses and gradually make them more general until an acceptable description is found (specific-to-general search), or one can begin with very general hypotheses and gradually make them more specific (general-to-specific search). One can even search in both directions simultaneously (the version space method). These approaches have different implications for the nature of the resulting description; for example, specific-to-general strategies will arrive at more specific characterizations than general-to-specific methods. However, all will generate descriptions that are both consistent and complete over the given instances, provided certain assumptions are met, such as the absence of noise (misclassified instances). We will see examples of both approaches in our review of grammar learning methods.

Characterization methods also vary in their processing of the instances they are given. Some approaches incorporate data one instance at a time; these are usually called *incremental* learning methods. Other approaches examine all of the instances simultaneously, often using statistical techniques to direct their search for useful hypotheses. These are typically called *nonincremental* learning methods. Naturally, the former are more plausible models of the human learning process, while the latter are more analogous to scientific

data analysis. Most of the grammar learning systems we examine later are incremental, but we will see one example of the nonincremental approach.

Finally, characterization methods differ in the *operators* that they employ for moving through the space of hypotheses. Some systems use operators that require only the current hypothesis as input. These techniques use some knowledge of the domain to transform the current description into one or more new hypotheses, and use the data only to test these hypotheses. We will call these *model-driven* methods. Other systems employ operators that require both the current hypothesis and a new instance as input; such techniques use the new data to transform the current description into one or more new hypotheses. We will call these *data-driven* methods. Historically, there has been a strong association between data-driven and incremental approaches, and a similar correlation between model-driven and nonincremental approaches, though exceptions to this trend exist. All of the grammar learning systems we will consider are data-driven in nature.

Within the data-driven approach, we find two quite different classes of operators. The first of these finds *common structure* between instances and/or hypotheses, and is always combined with specific-to-general methods. For instance, suppose one's current hypothesis states that all members of some concept are `large`, `red`, and `square`, and that one observes the new positive instance `large`, `blue`, and `square`. Since the hypothesis fails to match the new example, we know it is overly specific and should be made more general. In this case, the technique would generate the revised hypothesis that all examples of the concept are `large` and `square`, since these features are held in common between the old hypothesis and the new instance. Such methods are usually conservative, in that they generalize only enough to cover the new instance and no more.

One interesting aspect of the "finding common structures" operators is that, given a simple attribute-value representation involving a single object (like the one above), no search is required through the hypothesis space. Given relational or structural representations (in which predicates take two or more arguments), one description may be mapped onto another in multiple ways, so that competing hypotheses must be considered and search carried out. However, to the extent that one can represent one's hypotheses entirely in terms of attributes and their values, the characterization task will be greatly simplified.[4]

In relational cases involving search, there must be some way to distinguish good hypotheses from bad ones, and this is where negative instances come into play. If hypothesis H covers any of the known negative instances, we know it is overly general; thus, we can remove H from consideration and concentrate on the alternatives. Another important aspect of these operators is that they generate hypotheses containing those features held in common by all positive instances. This is desirable if the concept can be described as a conjunction, but if a disjunction is required, the approach will lead to overly general hypotheses that cover some negative instances. Note that if the concept can be expressed

[4] A number of researchers, including MacWhinney (1978), Berwick (1979), and Wexler (this volume), have applied this approach to grammar learning. Given the inherent relational nature of language, this seems counterintuitive, but their success speaks for itself. Hedrick (1976) employed a relational version of the common structures method in his grammar learning system.

in terms of an attribute-value scheme and if we know the concept is conjunctive in form, then *we can determine the correct concept description using only positive instances.*

The second type of data-driven operator finds *differences* between instances and/or hypotheses, and is always combined with general-to-specific characterization methods. For instance, suppose one's current hypothesis is that all members of some concept must be `red`, and that the last positive instance of this concept was `small, red, and circle`. Further suppose that the next negative instance is `small, red, and square`. Since the current hypothesis matches this counterexample, we know the hypothesis is overly general and should be made more specific. In this case, a better hypothesis would be `red and circle`.

We can generate more specific hypotheses by finding the differences between the positive and negative instances, and using these differences to further constrain the hypothesis. Thus, each difference leads to an alternative description, each more specific than the one we started with. In this case, the only difference is that the positive instance is a circle while the negative instance is a square. Winston (1975) has used the term *near miss* to refer to such one-difference negative instances. They are important because they eliminate search for difference-based methods in the same way that attribute-value based representations eliminate search for commonality-based methods. One can still make progress with *far misses*, but alternative hypotheses must be considered.[5] As with specific-to-general techniques, these methods are conservative in that they discriminate only enough to ensure that the negative instance will not be covered.

The majority of research on learning from examples has employed *empirical* methods like those we have just considered, which rely on many instances to induce an adequate characterization. We will see that these methods have been widely used in other machine learning tasks, including grammar learning. However, recently some researchers have explored another approach that generates descriptions on the basis of a single positive instance.

These have been called *analytical* methods, since they reason about why the instance belongs to the specified class. They have also been called *explanation-based* methods, since they construct justifications for the instance's classification. For example, one might be given a functional specification of the concept *cup* – that it must contain liquids, that it can be lifted, and so forth. If one is then given a specific coffee cup labeled as a positive instance, one can *prove* that the physical features of this object satisfy the functional definition of *cup*. Thus, the cup's handle lets it be lifted, its concavity lets it contain liquid, etc. This proof identifies the relevant features of the instance, and these features can be used to formulate a general structural description of the concept.

We do not have the space to describe such analytical approaches in detail, but later we will see that some of the work on grammar acquisition fits into this framework, rather

[5] Langley (1980, 1982) and Anderson (1981) have employed difference-based methods in their models of first language acquisition. Unfortunately, since parents do not carefully plan their presentation of sample sentences, we cannot assume that the first language learner relies on near misses to eliminate search.

than the empirical one. We direct interested readers to Mitchell, Keller, & Kedar-Cabelli (1986) for an excellent review of explanation-based learning methods.

Learning Search Heuristics

One of the central insights of AI is that intelligence requires the ability to search, and another is that expertise involves the ability to direct search down useful paths. This search occurs within the context of some *problem space*, which can be defined as a set of problem states together with operators for moving between those states. Each operator has an associated set of legal conditions that specify when it can be applied, but these are not enough to let one discover the goal state without search. For this one needs additional *heuristic* conditions on each operator that suggest the optimal move at each point in the problem solving process.

Naturally, researchers in machine learning are concerned with how such heuristics might be learned from experience. The task of learning search heuristics may be stated as follows:

- *Given:* A problem space defined in terms of an initial state and set of legal operators;
- *Given:* A test to determine when the goal has been reached, and a search strategy for selecting operators and states.
- *Find:* Heuristic conditions for each operator that will reduce or eliminate search.

For example, Mitchell, Utgoff, and Banerji (1983) have examined heuristics learning in the domain of symbolic integration. In this case, problem states take the form of symbolic expressions such as $\int 6x^2 dx$ and $2x^3$, and the goal is to find some state in which no integral sign occurs. Similarly, operators take the form of rules for transforming one state into another, such as $\int a \cdot bx^n dx \rightarrow a \int bx^n dx$. Fifty such operators commonly occur in solving simple integration problems, and the search space generated by these operators is quite large.

As with learning from examples, researchers have explored a variety of different methods for automatically generating heuristics. However, each such method must respond to an issue that makes the heuristics learning task more difficult than the task of learning from examples. This is known as the *credit assignment* problem.

Minsky (1963) was the first to identify the difficulty of assigning credit and blame in procedural learning. This issue arises in situations where the learner receives feedback only after it has taken a sequence of actions. In order to improve its performance, the learner must assign *credit* to desirable actions and *blame* to undesirable ones. For instance, if one loses a chess game, the final move is seldom responsible for checkmate. Usually, some other (much earlier) move led to this undesirable state, but identifying this move may be very difficult.

Significant progress in heuristics learning occurred only when researchers identified effective methods for assigning credit and blame. The most obvious of these methods involves waiting until one finds the goal state through search, and then using the complete

solution path to distinguish desirable moves from undesirable ones. For the integration problem $\int 6x^2dx$, the optimal solution path would be $\int 6x^2dx \Rightarrow 2\int 3x^2dx \Rightarrow 2x^3$. Any move lying directly along this path to the goal state is marked as desirable, while any move leading one step off the solution path is marked as undesirable. Sleeman, Langley, & Mitchell (1982) have called this method *learning from solution paths*. Later, we will see that Berwick (1979) has employed a very similar approach to learn grammar rules.

An alternative method involves assigning credit and blame during the search process itself. Rather than waiting until a complete solution has been found, one may note regularities in the search tree as it is generated. For instance, one might notice that an existing state has been revisited. This may result from a loop or from a longer path to that state, but in either case the move that led to the state is undesirable. (In integration, loops can easily occur when integration by parts is attempted.) Similarly, moves that lead to dead ends should be avoided if possible. Anzai & Simon (1979) and Langley (1985) have called this approach *learning while doing*, since it lets one assign credit and blame while search is being carried out.

Once a credit assignment technique has been used to label moves as desirable or undesirable, one can easily identify positive and negative instances of each operator. These can be passed to a characterization method, which in turn generates general descriptions of the conditions under which each operator should be applied. When the heuristic conditions associated with an operator are used to determine when that operator should be applied, search is reduced or even eliminated. Langley & Ohlsson (1984) have called this basic method the *problem reduction approach to heuristics learning*, since it involves separately identifying the heuristic conditions for each operator and then recombining them into a system that requires little or no search. The vast majority of research on heuristics learning has taken this approach, though there has been considerable variation in both the credit assignment methods and characterization methods employed.

Like the task of learning from examples, the heuristics learning problem is an idealized case of the general task of learning from experience. Since the problem space is provided by the programmer, the basic objects upon which rules are based (the problem states) are given at the outset. As a result, there is no significant aggregation problem. Since the problem spaces that have been examined seldom have more than a few operators (the fifty that Mitchell et al. examined was very unusual), and since no more than a few heuristics are learned for each operator, there is no significant storage or indexing problem.

However, the heuristics learning task differs from learning from examples in that the programmer does not provide the clusters from which descriptions (in this case heuristics) are generated. The learning system must of its own accord cluster instantiations of each operator into groups of positive and negative instances, using complete solution paths or some other credit assignment method. In fact, within the context of learning search heuristics, the subproblem of clustering is identical to the problem of assigning credit and blame. Once the system has determined the positive and negative instances for each operator, it must still employ some characterization method to determine the heuristic conditions for that operator. However, the same methods that have proven so useful in learning from examples can also be applied in this situation. To summarize, the task of

heuristics learning involves less idealization than the task of learning from examples, since the former requires one to address issues of both clustering *and* characterization.

Conceptual Clustering

We associate the notion of taxonomies with biology, but many (if not all) of the sciences progressed through a stage of taxonomy formation before moving on to discovering laws and theories. The task confronting scientists in this stage can be stated:

- *Given:* A set of objects and their associated descriptions.
- *Find:* A hierarchical classification tree that covers all objects, and which places similar objects in the same classes.

For instance, one might observe many different species or plants and animals, and then attempt to formulate a taxonomy which places similar species in the same categories. The most frequent examples come from biology, but taxonomies have also played an important role in astronomy (classifying stars and galaxies) and chemistry (classifying substances).

At first glance, this task appears quite similar to the problem of learning from examples. However, it differs from the simpler task on three dimensions. First, objects are not assigned to classes by a tutor, so that a distinction between positive and negative instances is not inherent in the data. Second, the goal is to generate extensional definitions of each class rather than general descriptions (intensional definitions). Finally, since a taxonomy is hierarchical, one must discover concepts at multiple levels of abstraction, as contrasted with the single level concepts that occur in learning from examples.

Despite the apparent complexity of the taxonomy formation task, statisticians and biologists have developed computational methods for automating this process. These techniques share the general names of *cluster analysis* and *numerical taxonomy*, and a variety of them have been proposed (Everitt, 1980). Most of the methods employ some measure of the *distance* between objects or clusters in a N-dimensional feature space, attempting to group together objects that are close to each other. Unlike most statistical methods, cluster analysis and numerical taxonomy have little theoretical justification and are largely heuristic in nature. Moreover, different methods tend to produce radically different taxonomies unless the data are very regular, and the resulting hierarchies are often difficult to interpret.

In response to these limitations, Michalski & Stepp (1983) have formulated a related task they call *conceptual clustering*. This task differs from the traditional taxonomy formation task in two respects:

- In addition to generating a hierarchy containing clusters of objects, one must also *characterize* those clusters.
- In evaluating potential clusters, one should consider the characterizations of these clusters as well as the objects they contain.

The authors argued that by including characterizations in the evaluation process, the resulting clusters will be easier to understand than those generated by traditional methods.

Given this revised framework, it is not surprising that Michalski and Stepp used established characterization methods as subroutines in their approach to taxonomy formation. Other methods for conceptual clustering (Langley & Sage, 1984; Fisher, 1984) differ in various respects, but all take advantage of standard characterization techniques in some manner.

Methods for conceptual clustering can be viewed in terms of three different levels of search, each involving a different problem space. The first of these involves search for clusterings or groupings of objects at a given level of the hierarchy. The second involves search for descriptions or characterizations of object clusters; this is identical to the search carried out by systems that learn from examples. The final search is through the space of possible hierarchies within which the clusters and their descriptions are contained.

Methods for dealing with each of these subproblems can vary along a number of dimensions. We have already seen some varieties of characterization methods, and similar variations exist in the search for clusterings and hierarchies. For instance, one may search for clusterings exhaustively or using heuristic search techniques; in particular, Michalski & Stepp employed a hill-climbing method to find useful clusterings. Similarly, one may construct hierarchies from the top down or from the bottom up; Michalski & Stepp used a top-down approach, while most numerical taxonomy methods operate in a bottom-up fashion.[6] The interested reader is directed to Fisher & Langley (1985) for a more detailed discussion of conceptual clustering methods in these terms.

The conceptual clustering task can be viewed as another variant on the general problem of learning from experience. Like the task of learning from examples, it ignores the problem of *aggregation*, since the basic objects and their descriptions are given to the learner. However, it differs from the simpler task in that it explicitly addresses the problem of *clustering* objects into groups without aid from a tutor. Unlike traditional clustering techniques, it also addresses the *characterization* problem, since one must form general descriptions for each cluster. Finally, it begins to deal with the *storage/indexing* problem, since objects and classes are stored in a hierarchy that can be used in classifying novel objects.

In other words, conceptual clustering forces one to address three of the four components of learning from experience, more than either learning from examples or heuristics learning. Later in the paper, we will see that the clustering problem also arises in the grammar learning task, and we will examine some responses to this problem by several grammar learning systems.

Learning Macro-Operators

The notion of *chunks* was originally proposed by Miller (1956) to explain short-term memory phenomena. The term *chunk* denotes some familiar pattern that one can remember or manipulate as a single entity. Chunks can be perceptual or action-oriented, and can involve either spatial or sequential relations. In practice, machine learning re-

[6] Researchers in numerical taxonomy (Everitt, 1980) use the term *divisive* for top-down methods and *agglomerative* for bottom-up approaches.

searchers' interest in issues of procedural learning has led them to emphasize sequential action structures.

Like the work on heuristics learning, the chunking research has focused on learning in the context of search through some problem space. In this case, the goal is to discover sequences of operators, or *macro-operators*, that achieve useful results in the problem space (e.g., bringing one closer to the goal). Since relatively little work has been done on the acquisition of spatial or perceptual chunks, we will focus on macro-operators here. [7]

A number of mechanisms for generating macro-operators have been described in the literature, though they have not always been cast in these terms. For example, Lewis (1978) and Neves & Anderson (1981) discuss a process called *composition* that combines two production rules into a single, more powerful rule whenever the original rules apply in sequence. They have used this to explain the Einstellung effect, in which problem solvers prefer a well-practiced solution to some problem even when more efficient solutions are possible. More recently, Anderson (1983) has described a more selective version of composition that combines only those rules used to achieve a common goal.

Korf (1982) has described a quite different method for generating macro-operators that involves the notion of decomposable subgoals, while Iba (1985) has employed a third method that combines rules when it notes peaks in a numeric evaluation function. Finally, Laird, Rosenbloom, & Newell (1986) have described a method called *chunking* that is evoked only when a goal is achieved; however, this method differs from Anderson and Neves' composition in that it constructs the resulting macro-operator from memory elements involved in the goal, rather than from the rules used to reach the goal.

Despite the differences in these approaches to forming sequential chunks, some common themes have emerged. First, most of the work has occurred within a heuristic search framework, in which macro-operators are composed from primitive legal operators. Second, goals play a central role in determining when most of the chunking methods are evoked. For this reason, most of the methods are embedded within a means-ends analysis framework like that used by Newell, Shaw, & Simon's GPS (1960), which allows intelligent generation of subgoals.

The task of forming macro-operators can be viewed as another variant on the general task of learning from experience. In this case, the structure to be learned is some configuration of actions or operators – a sequential chunk. Methods for learning macro-operators directly address the *aggregation* issue, since they decide which components to include as parts of the higher level structure. In most chunking methods, the *characterization* problem is made trivial, since new rules are based directly on existing rules, for which the level of generality is already known. Even in methods that address issues of characterization (such as Laird, Rosenbloom, & Newell's approach), chunks are based on single instances, so that the *clustering* problem is bypassed. Finally, none of the research in this area explicitly addresses storage issues, though much of the work is embedded within production system

[7] However, later we will see that sequential chunks also arise in the grammar learning task, where they correspond to structures such as noun phrases and verb phrases.

frameworks like Anderson's ACT (1983), which have implications for how knowledge is indexed in long term memory.

Grammar Learning

Now that we have considered a number of machine learning tasks, let us turn to the problem of language acquisition. The overall task of language acquisition is very complex and involves many levels, including: learning to recognize and generate words; learning the meanings of words; learning grammatical knowledge; and learning pragmatic knowledge. Each of these subproblems is interesting in its own right, but since the majority of AI work on language acquisition has dealt with grammar learning, we will focus on this issue in the current section.

Some of the earliest work in machine learning addressed the problem of grammar acquisition, and this is still an active area of research in the field. The basic task may be stated in the following manner:

- *Given:* A set of grammatical sentences from some language;
- *Find:* A procedure for recognizing and/or generating all grammatical sentences in that language.

The learned procedure may take many different forms, such as a set of rewrite rules, an augmented transition network, or a production system. Note that one is given only legal sentences from the language to be learned, and that no "negative instances" are presented. Solomonoff (1959) carried out some of the earliest AI work on this problem, followed by Knowlton (1962), Garvin (1967), and Horning (1969). Wolff (1978, 1982) and Berwick (1979) have described more recent grammar learning systems in this tradition.

However, we know from the child language data that the human learner does not hear sentences in isolation; rather, the sentences usually describe some event or object in the immediate environment. This observation leads to a different formulation of the grammar learning task, which may be stated:

- *Given:* A set of grammatical sentences from some language, each with an associated meaning;
- *Find:* A procedure for mapping sentences onto their meanings or vice versa.

This view of grammar acquisition differs significantly from the first one we examined. Grammatical knowledge may again be represented in a variety of ways, but it must contain more than information about sentence structure – it must also relate this structure to meaning. We will see that the second view of grammar learning leads to quite different models of the learning process. Kelley (1967), Siklóssy (1968), and Klein & Kuppin (1970) carried out the earliest work in this "semantic" tradition. More recent systems have been described by Hedrick (1976), Reeker (1976), Anderson (1977), Selfridge (1981), Sembugamoorthy (1981), Langley (1982), Smith (1982), and Hill (1983).

Since we review some of the earlier work on grammar learning in detail in the following section, we will not delve deeply into particular methods here. However, we should note

that both versions of the grammar learning task can be viewed as further variants on the general problem of learning from experience. However, they differ from the other four tasks in an important respect. Like the chunking task, they address the problem of *aggregation*, since the grammar learner must form sequential chunks such as noun phrase and verb phrase. Like the conceptual clustering task, they address the *clustering* problem, since one must group words into disjunctive classes like noun and verb without the aid of a tutor.

The opportunity for *characterization* also exists in the second (semantic) version of the task, since semantic features can often be used to predict when a class like noun or verb is appropriate. Finally, most representations of grammatical knowledge (such as ATNs and production systems) have implications for the *storage/indexing* problem, and this carries over into the work on grammar learning. In other words, the task of grammar learning is the only task that forces one to address all four components of learning from experience, making it (in principle, at least) the most challenging of the problems we have examined.

Machine Learning Research on Grammar Acquisition

Now that we have reviewed the types of tasks that machine learning researchers have focused on, let us consider some examples of AI systems that address the grammar learning process. Considerable work has been done in this area, and we will not have time to cover it all here. Instead, we will examine four specific systems that we feel will clarify the nature of this work and its relation to other problems in machine learning. The interested reader is directed to reviews by McMaster, Sampson, & King (1976), Pinker (1979), and Langley (1982).

We will examine four AI systems that implement quite different approaches to grammar learning: Wolff's SNPR (1978, 1982), Berwick's LPARSIFAL (1979, 1980), Anderson's LAS (1977), and Langley's AMBER (1980, 1982). We will see that these systems differ on a variety of dimensions, the most important involving whether they learn from isolated sentences or from sentence-meaning pairs. In each case, we describe the inputs and outputs of the system, its representation of acquired grammatical knowledge, its learning mechanisms, and the relation of these mechanisms to the four components of learning from experience. We close with some comments on the role of negative instances in grammar learning.

Wolff's SNPR System

Wolff (1978, 1982) has developed SNPR, a program that acquires grammatical knowledge in a very data-driven manner. The system begins with a sequence of letters, and generates a phrase structure grammar (stated as rewrite rules) that summarizes the observed sequence. SNPR is not provided with any punctuation or with any pauses between words or sentences; it must determine these boundaries on its own. The program processes the strings in a semi-incremental manner, first examining a subset of the data, then processing another segment, and so forth.

One of SNPR's strategies is to look for common sequences of symbols, and to define *chunks* in terms of these sequences. For example, given the sequence thedogchasedthecatthecatchasedthedog ..., the program might define chunks like the, dog, cat, and chased. This example is somewhat misleading, since the system always builds chunks from pairs of symbols, but it conveys the basic idea. Whenever a chunk is created, the component symbols are replaced by the symbol for that chunk. In this case, the sequence the-dog-chased-the-cat-the-cat-chased-the-dog would result. This process can be applied recursively to generate hierarchical chunks.

In addition, when SNPR finds a number of different symbols (letters or chunks) that precede or follow a common symbol, it may define a disjunctive class in terms of the first set. For instance, in the above sequence we find the subsequences the-dog-chased and the-cat-chased. Based on this regularity, Wolff's program might define the disjunctive class noun = {dog, cat}. It would then substitute the symbol for this new class into the letter sequence for the member symbols. In this case, the sequence the-noun-chased-the-noun-the-noun-chased-the-noun would be generated. Additional classes such as 'verb' and 'determiner' would be defined and replaced in the same manner.

These two basic methods are applied recursively, so that chunks can be defined in terms of disjunctive classes. This leads to constructs such as noun phrases, prepositional phrases, verb phrases, and ultimately to sentences. Thus, the interleaving of chunks and disjuncts leads SNPR to construct phrase structure grammars which summarize the letter sequences it has observed.

From this description we see that Wolff's learning system employs two operators – one for forming disjunctive classes such as noun, and another for defining chunks or *conjunctive* structures, such as dog. SNPR also includes operators for generalization (by discarding some data) and recursion, but we will not focus on them here. The system employs a numeric evaluation function to determine which of its operators should be applied in a given situation. This function measures two features of the grammar that would result – the *compression capacity* or the degree to which a given grammar compresses the original data, and the *size* of the grammar. At each point in its learning process, SNPR selects that step which gives the greatest improvement in compression capacity per unit increase in size. Thus, the system can be viewed as carrying out a hill-climbing search through the space of possible phrase structure grammars.

Now that we have described Wolff's SNPR in process terms, let us reexamine the system in terms of the four components of learning from experience. The first operator is clearly responsible for generating sequential chunks, and thus addresses the *aggregation* problem. Similarly, the second operator is responsible for forming disjunctive classes or extensional definitions, and thus addresses the *clustering* component. The most interesting feature of SNPR is that these operators both compete for attention through the evaluation function, and interact in that chunks are later used in disjunctive classes, which are in turn used in higher level chunks. Thus, the solution to both aggregation and clustering is inherently intertwined, with both using co-occurrence statistics to determine which step to take. Note that this data-intensive approach to chunking differs radically from the work on macro-operators, in which chunks are determined on the basis of a single instance.

Wolff's system is also interesting in that it makes no explicit attempt to characterize its disjunctive classes (e.g., noun and verb) after they have been extensionally defined. However, the interaction between aggregation and clustering can lead to multiple chunks which reference the same disjunctive class. For instance, the symbol noun may occur in the rewrite rules for noun phrase and prepositional phrase. Taken together, one can view the set of chunks that refer to noun as an intensional definition or characterization of that class. Thus, SNPR arrives at characterizations of a sort, though it does so indirectly.

Similarly, Wolff does not explicitly address the details of the storage process, but the notion of efficient storage is a major motivation behind his work. Rewrite rules are commonly used within computer science to store grammars for compilers, and recent versions of the AI programming language Prolog incorporate efficient implementations of such rewrite rules. Moreover, although SNPR's heuristics are concerned with efficient storage rather than efficient access and retrieval, the two measures are certainly correlated. Now that we have considered SNPR's relation to the components of learning, let us turn to some incremental approaches to grammar acquisition.

Berwick's LPARSIFAL System

Berwick (1979, 1980) has described LPARSIFAL, a system that learns grammars from a sequence of legal English sentences. The program incrementally modifies its grammar after each input sentence, unless that sentence can already be parsed by the grammar. The input sentences differ from Wolff's in that each one is composed of a sequence of separate words, and the sentences themselves are separated from each other. No meanings are associated with either words or sentences. Grammatical knowledge is represented as a set of rules, but ones quite different from the rewrite rules used by Wolff's SNPR. In order to understand the nature of these rules, we must review Marcus' (1980) PARSIFAL, the natural language system upon which Berwick's work is based.

PARSIFAL differs from most AI natural language systems in that it employs a look-ahead method to avoid the need to backtrack on the vast majority of sentences. The system employs two data structures – a buffer containing the words in the sentence (the input) and a stack of nodes representing phrase structures (the output). The conditions of rules can examine only the first three items in the buffer and the top item in the node stack. There are four available actions:

(1) *Create* a node and push it onto the stack;

(2) *Remove* the top node on the stack and put it in the buffer, pushing existing items to the right;

(3) *Attach* the first buffer item to the top node on the stack, moving the remaining items in the buffer to the left;

(4) *Switch* the first and second items in the buffer.

The first of these actions can be instantiated in different ways. For instance, one rule may create a noun-phrase node, while another may create a verb-phrase node. However, the last three actions are completely determined by the situation in which they apply. PARSIFAL

operates in cycles, applying the first rule that matches, altering the stack and buffer accordingly, applying the next rule that matches, and so forth. This process continues until the sentence has been completely analyzed and a parse tree has been constructed.

Now let us return to Berwick's LPARSIFAL, which operates within this framework. The system begins with a knowledge of $\bar{X}$ theory and an interpreter for applying grammar rules to parse sentences. Although the program can learn rules "from scratch", our discussion will be simplified if we assume that LPARSIFAL has already acquired a few rules for parsing simple sentences, such as active statements like `The boy bounced the ball`.

When given a new sentence, LPARSIFAL attempts to parse it using the existing rules. If it encounters some problem, the system attempts to create a new rule that will handle the problem-causing situation. The program determines the action on this rule using a generate and test strategy, first seeing if *attach* will let it continue parsing the sentence, and if this fails, seeing if *switch* will suffice. Assuming one of these ultimately leads to a successful parse, the program constructs a new rule containing that action.

The conditions of the new rule are based on the state of the parse when the impasse was encountered. This includes the top of the stack and the contents of the input buffer, including lexical features associated with the words in the buffer. Upon adding the new rule to memory, the system checks to see if any existing rules have identical actions. If there are none, the rule is inserted at the beginning of the rule list.

However, if a rule with the same action and the same $\bar{X}$ context is found, LPARSIFAL compares the two condition sides to determine what they hold in common. The resulting mapping is used to construct a more general rule with the same action. Differing conditions are dropped from the resulting general rule or, in some cases, lead to the creation of syntactic classes like nouns and verbs. In the latter case, the words that differ in the two conditions are replaced by the name of the class, and the words are stored as members of that class. If the old rule contains a class where the new rule contains a word, the word is added to that class.

The reader will note that LPARSIFAL's method for combining rules is identical to one of the data-driven characterization operators we considered in the context of learning from examples. This is the "finding common structures" operator, which is often used in conjunction with specific-to-general strategies for learning concepts from examples. There are three interesting aspects to Berwick's use of this method.

First, the system decides for itself into which of the existing rules it should incorporate the new "instance" (a given buffer-node combination). LPARSIFAL determines this by examining the action and the $\bar{X}$ context associated with existing rules and the new situation, much as a system that learns from examples uses the name of the class associated with an instance. Second, the system represents instances and condition sides purely in terms of attribute-value pairs. As a result, there is never more than one way to incorporate a new instance into an existing rule, so that absolutely no search through the rule space is required. This leads directly to the third point. Since no search problem exists, the

program does not require negative instances to prune the search tree, and LPARSIFAL can learn grammars without computing such negative instances.[8]

However, recall that the "finding common structures" approach relied on an important assumption – that there exists a conjunctive characterization of the data. If an adequate description requires a disjunct of some form, then this approach will lead to an overly general characterization, and only negative instances will reveal the difficulty. Thus, Berwick's approach relies on the assumption that each action/$\bar{X}$ context combination has at most one associated set of conditions. If this assumption were violated, his system would acquire overly general grammatical rules, though it would never realize this fact.

Now let us reconsider LPARSIFAL's approach to grammar acquisition as it relates to the four components of learning from experience. We have seen that the system addresses the issue of *clustering*, since it decides which instances (combinations of buffer items and nodes) to compare to one another. We have also seen that it attempts to *characterize* the resulting clusters by finding common features. The system's response to the *storage* problem is to create rules that are indexed for easy retrieval, a common approach that we have seen in other contexts. However, the program does not form any new sequential chunks beyond those it starts with, so that it bypasses the *aggregation* problem.

Upon reflection, LPARSIFAL feels quite different not only from Wolff's SNPR, but from every other grammar learning system that has been proposed. The reason for its distinctiveness becomes apparent when we recall another class of learning problems that addresses clustering and characterization but not aggregation – the task of *heuristics learning*. We would argue that Berwick has successfully transformed the grammar learning task into the task of learning search heuristics, a counterintuitive (but apparently useful) approach.

The relation will become apparent if we consider a conservative approach to the heuristics learning task. Suppose one begins with a set of heuristic rules that are overly specific, and which thus lead to a state in which no move is proposed. At this point, one falls back on those operators whose legal (but not heuristic) conditions are met. If applying one of these operators eventually leads to the goal state, then a new heuristic rule is created based on the successful move. This specific rule may then be combined with other rules that involve the same operator. The operators correspond to Berwick's four actions, and the learned heuristics correspond to his acquired grammar rules. This analogy is not as forced as it appears at first glance. Ohlsson (1983) has described UPL, a heuristics learning system that uses a nearly identical strategy to learn rules for puzzles like the Tower of Hanoi.[9]

Although Berwick's approach is an elegant one, it clearly addresses different issues in grammar learning than other systems. For instance, Wolff's SNPR generates sequential

[8] Berwick's system could identify negative instances of each action using the *learning from solution paths* method described earlier. Later, we will see another approach to "constructing" negative instances from grammatical sentences.

[9] We should note that Berwick reported the first version of LPARSIFAL in 1979, when very few results had been achieved in heuristics learning.

chunks like noun phrase and verb phrase, while LPARSIFAL does not. Similarly, Berwick's program learns the conditions under which to apply specific "parsing" operators, while other systems do not. The main overlap lies in the formation of syntactic classes like nouns and verbs, which both SNPR and LPARSIFAL (and many other systems) define. What is interesting about the latter system is that it *clusters* objects at two entirely different levels – the level of instances of each operator, and the level of words that should be grouped into one syntactic class.

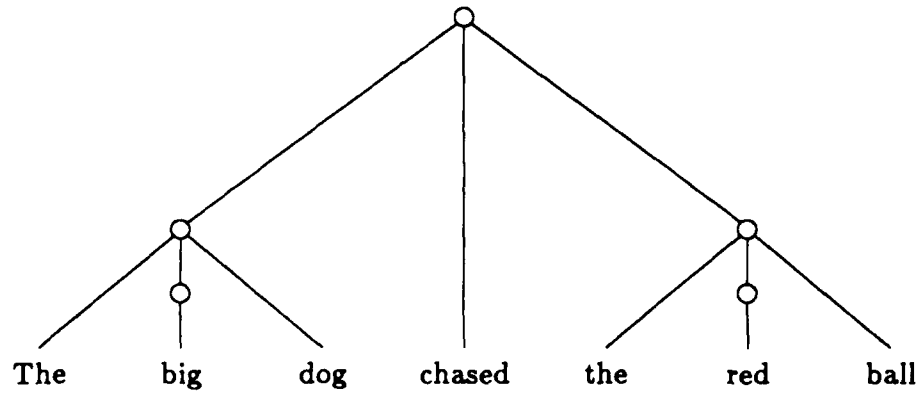


Figure 2. Inferred parse tree for a simple sentence.

Anderson's LAS System

Anderson (1977) has developed LAS, a program that learns to understand and generate sentences in both English and French. The system accepts legal sentences and their associated meanings as input, with meaning represented in terms of a semantic network. The goal is to acquire a *mapping* from sentences onto their meanings and vice versa, rather than simply learning to recognize grammatical utterances. LAS represents grammatical knowledge as an augmented transition network (ATN), with both semantic and syntactic information stored on each link.

In addition to this basic information, LAS is provided with additional knowledge that constrained the learning process. This information included:

- Connections between concepts and their associated words;
- The main topic of each sentence;
- Knowledge that some concepts (like shapes) were more significant than others; the words for these concepts eventually developed into the class of nouns;
- The *graph deformation condition*, which roughly states that if two words occur near each other in a sentence, the concepts associated with those words must occur near each other in the meaning of that sentence.

These sources of information are sufficient to enable LAS to determine a unique parse tree for any given sentence-meaning pair. For instance, suppose the system is given the sentence `The big dog chased the red ball` and its associated meaning. We can represent this meaning using node-link-node triples, with each triple specifying a connection in a semantic network: `(event-1 action chase)`, `(event-1 agent agent-1)`, `(agent-1 type dog)`, `(agent-1 size big)`, `(event-1 object object-1)`, `(object-1 type ball)`, `(object-1 color red)`.[10] Given this information, LAS would generate the parse `((The (big) dog) chased (the (red) ball))`, where parentheses indicated the level of the tree. Figure 2 presents a graphic version of this parse tree.

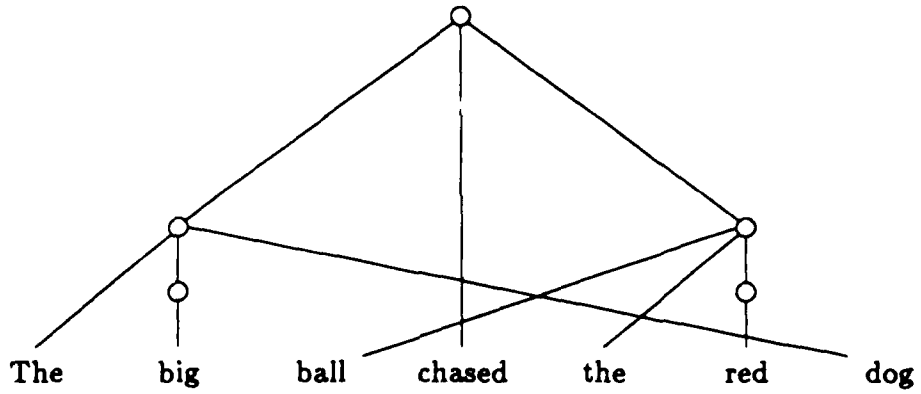


Figure 3. A sentence that violates the graph deformation condition.

Let us consider the graph deformation condition in somewhat more detail. Stated more formally, this says that the parse tree for a sentence must be a graph deformation of the network representing that sentence's meaning, and that the branches in this parse tree must not cross each other. This assumption constrains the space of grammars that LAS considers, but it does not eliminate search by itself. Given the same meaning representation as above, a variety of associated sentences would satisfy the constraint. These include `The dog big chased the ball red`, `The red ball chased the dog big`, and `Chased the big dog the red ball`. If LAS observed any of these sentences paired with the same meaning, it would find them acceptable and generate their parse trees. Since the parses would be different, the system would acquire a different ATN in each case.

However, the sentence `The big ball chased the red dog` violates the graph deformation condition. When one orders the words in this fashion, there is no way to redraw the tree from Figure 2 so that the lines do not cross. As Figure 3 shows, the agent `dog` is too far from the agent node and the object `ball` is too far from the object node for this to be possible. As a result, LAS would reject this sentence as unacceptable, and would never consider learning a grammar which generated such sentences.

[10] LAS actually used a different set of links in its network representation, but we have used mnemonic ones for the sake of clarity.

Given the parse tree for a sentence, it is a simple matter to generate an augmented transition network that will parse that sentence. For instance, suppose LAS is given the parse tree ((The (big) dog) chased (the (red) ball)), shown graphically in Figure 2. Using the knowledge it has been given (including the graph deformation condition), the program can transform this structure directly into the (initial) ATN shown in Figure 4.

Since the parse tree has three branches at the top level, LAS would generate a top-level ATN with three links – one for the first structure (The (big) dog), one for the second structure chased, and one for the third (the (red) ball). Since the first and third components themselves contain internal structure, LAS would build a sub-ATN for both of these, each with three links. For example, the first sub-ATN (call it NP1) would have links for The, (big), and dog. Similarly, since the second element for each of the sub-ATNs has internal structure, LAS would create even lower level ATNs for these, each having one link (in one case for big and in the other for red).

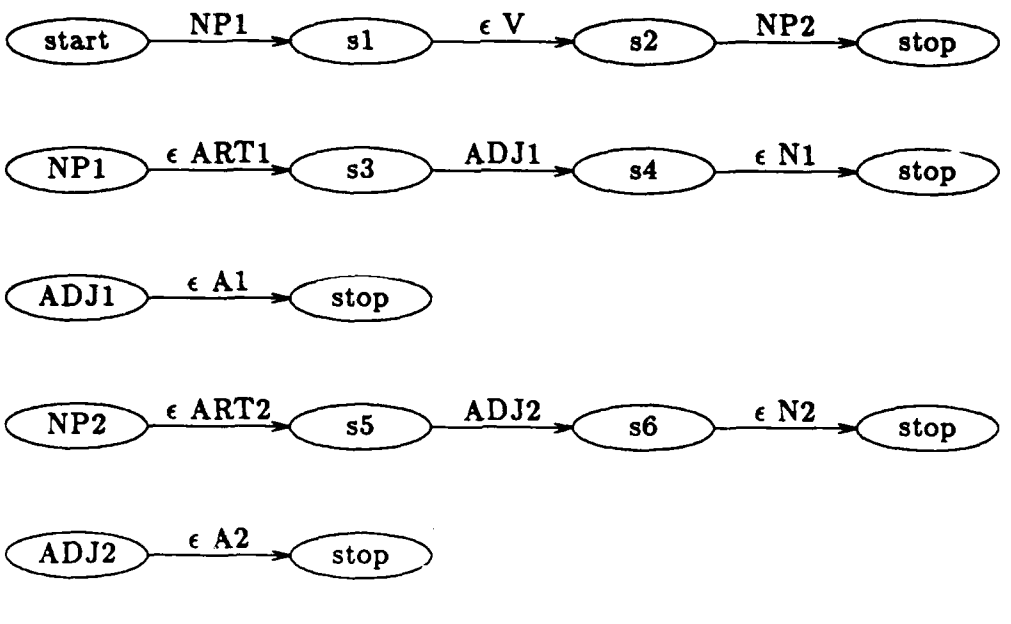


Figure 4. Initial ATN based on a single sentence.

In other words, there is a direct mapping from a parse tree to an ATN for generating that parse tree. Note from Figure 4 that specific words are never used as tests on the ATN's links. Instead, LAS defines a word class that initially has a single member, and uses this class in the test. Also note that the initial grammar employs different subnetworks for

constructs that we view as equivalent, such as the two networks for noun phrases. As the system progresses, such distinctions gradually disappear.

After it has constructed an initial ATN, LAS attempts to incorporate new parse trees with as little modification as possible. For instance, given the new sentence `The small cat chased the long string`, the system would note that its ATN would parse this quite well, if only the certain classes were expanded. In this case, the class `ADJ1` = {`big`} must be extended to `ADJ1` = {`big, small`}, the class `NOUN1` = {`dog`} must be extended to `NOUN1` = {`dog, cat`}, and so forth. However, LAS is cautious about taking such steps, carrying them to completion only when the concepts associated with the words play the same semantic role as in earlier sentence meanings.

In addition to expanding word classes, LAS employs two other mechanisms for producing more general grammars. First, when the system finds two word classes that share a significant number of elements, it combines them into a single class. Second, if LAS finds two sub-ATNs to be sufficiently similar, it combines them into a single subnetwork. A special case of this process actually leads to recursive networks for parsing noun phrases, so that arbitrarily deep embeddings can be handled. These steps occasionally lead the system to learn overly general ATNs, which generate constructions such as `foots` instead of `feet`, and it has no mechanisms for recovering from such errors.

Figure 5 presents a revised ATN that LAS might construct after hearing the second sentence `A tall man followed the big dog`. In this case, the word `followed` has been added to the syntactic class `V`. Moreover, the two classes `N1` and `N2` have been combined into the single class `N` with members `dog`, `ball`, and `man`.[11] Similarly, the classes `ART1` and `ART2` have been collapsed, as have the classes `A1` and `A2`. More important, the two noun phrase ATNs have been combined into the single ATN `NP`, based on their similar structure and components. An analogous combination has occurred for the `ADJ1` and `ADJ2` networks, generating a much simpler grammar than we had after the first example.

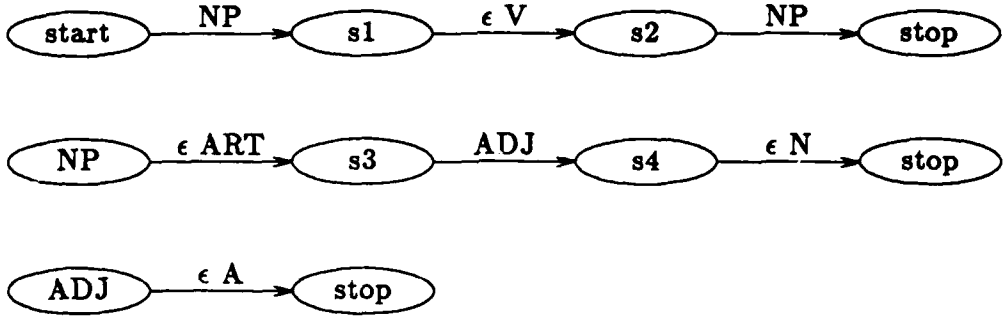
Now let us reconsider Anderson's LAS in terms of the four components of learning from experience. First, we see that the system employs the meanings of sentences, their main topic, and the graph deformation condition to determine a unique parse tree. This in turn determines an augmented transition network, which can be viewed as a hierarchically organized set of *sequential chunks*. In other words, LAS used the above information to solve the *aggregation* problem for each sentence it is given. Note that this approach is quite different from chunking methods that have been used for building macro-operators, employing knowledge about language to determine the chunks.

Anderson's approach to the clustering problem also differs from methods used for conceptual clustering. LAS uses a bottom up (agglomerative) approach to form disjunctive word classes, but this process operates in two stages. In the first stage, the system extends its word classes incrementally, expanding them whenever required to parse new sentences. In the second stage, it combines classes (nonincrementally) if they have enough common

[11] This combination could occur after `dog` had been added to `N2`, causing the two sets to have a 50% overlap. Actually, we doubt that LAS would collapse word classes on the basis of such slim evidence, but we have assumed that it would for the sake of simplicity.

members. Moreover, both mechanisms are limited by semantic constraints. Thus, LAS's *clustering* method is evoked by syntactic regularities, but is filtered by semantic information. Syntax is used to generate possible clusters, while semantics is used to test whether they are appropriate.

LAS determines the semantic constraints on its ATN's links from a single sentence-meaning pair, assuming that all portions of the semantic network relating concepts at the same level in the ATN are relevant. For instance, given the parse tree ((The (big) dog) chased (the (red) ball)), LAS would assume that only the agent, action, and object relations would be relevant to the top-level ATN. Thus, Anderson's response to the *characterization* problem also differs from the traditional approaches, producing a general rule from a single instance. In this sense, it is similar to the *explanation-based learning* methods that have recently been formulated by Mitchell, Keller, & Kedar-Cabelli (1986) and others.



V = {chased, followed} N = {dog, ball, man}
ART = {the, a} A = {big, red, tall}

Figure 5. Initial ATN based on a single sentence.

Finally, Anderson's decision to use augmented transition networks constitutes a response to the *storage* issue. The use of ATNs have implications for retrieval, since the connecting arcs act as direct pointers to successor states, giving them a top-down, expectation-driven flavor. Assuming the system has parsed the first part of a sentence, the most likely steps to follow can be easily retrieved. In addition, LAS combined sub-ATNs whenever possible. Although this was primarily intended as an induction technique, it also led to an efficient storage of grammatical knowledge. This is another dimension on which Anderson's work differs from other machine learning efforts, but this is not surprising, considering that ATNs were designed to handle linguistic phenomena.

Langley's AMBER System

Langley (1980, 1982) has described AMBER, a cognitive simulation of the early stages of child grammar acquisition. Like LAS, the system accepts sentence/meaning pairs as input, using a semantic network to represent meaning. Again, the goal is to learn a mapping, in this case from meanings to sentences. AMBER represents this grammatical knowledge as production rules for generating sentences, including both semantic constraints and information about what has already been said in the conditions of rules.

AMBER is also similar to LAS in that it requires knowledge of the meanings of content words (like `small`, `ball`, and `bounce`), as well as information about the main topic of each sentence. In addition, the system assumes that utterances having no associated meaning (like `the` and `ing`) are function words, and that these play a quite different role than content words. Although AMBER does not assume Anderson's graph deformation condition, we will see that something analogous arises from the system's strategy for generating sentences.

The reader will recall that LAS used each sample sentence-meaning pair to generate a parse tree, which formed the basis for its ATN. Instead, AMBER employs information about the main topic of the sentence to transform the semantic network representation of meaning into a tree, in which the top node corresponds to the main topic. Consider the following sentences:

```
The big dog chased the red ball.
The red ball was chased by the big dog.
The dog that chased the red ball was big.
```

Although these sentences describe the same event, they differ in their main topic. In the first construction, the chasing action is emphasized; in the second, the ball is highlighted; and in the final sentence, the dog is emphasized. In each case, AMBER transforms the network representation of the meaning into a different tree structure.

Based on the resulting tree, AMBER proceeds to generate an utterance to describe the structure. In doing so, it employs the notion of goals and subgoals. The system's top level goal is to describe the entire tree. In order to achieve this high level goal, it creates subgoals to describe nodes lower in the tree. At the outset, AMBER can handle only one subgoal at a time, leading the system to generate one-word "sentences". Much of the system's learning consists of acquiring rules that let it deal with multiple subgoals, and then identifying the relative order in which those subgoals should be achieved.

However, even in its early stages AMBER places two important constraints on this process. First, it never creates a goal at level L while another goal at the same level is still active. Second, once it has deactivated a given goal, it cannot reactivate it for the current utterance. Thus, in describing an event in which a big dog chased a red ball, AMBER might not mention all aspects of the event; for instance, it might fail to mention that the dog was big or that the action involved chasing. However, it would never say dog, followed by `chase`, and then return to `big`, since the concepts *big* and *dog* occur in the same subtree. As a result of this goal-processing strategy, AMBER is guaranteed to

generate only utterances which obey Anderson's graph deformation condition, even though this constraint is not explicitly included in the system.

AMBER begins with the ability to say one content word at a time. Based on differences between these utterances and the sample sentences it is given, the system generates new rules that let it generate combinations of words and phrases in the correct order. For instance, upon describing the shape of an object (say ball) without mentioning its color (say red), AMBER would acquire a rule stating that it should only describe shape after it had described color. Such rules must be constructed a number of times before gaining enough "strength" to take control from the default rules. Similarly, if the agent was omitted entirely from the system-generated sentence while the object was described, the system would construct a rule stating that the object should only be described once the agent had been mentioned. This last situation leads AMBER to constructions like Daddy ball in which the action is omitted. Of course, such omissions eventually disappear as the system progresses.

Whenever AMBER successfully predicts all of the content words in an adult sentence, it turns its attention to function words like is, the, and ing. In the early stages, the system simply omits these terms and creates rules to produce them in the future. However, these initial rules include only limited conditions based on the semantic role played by the associated content word. For instance, ing would only be produced following an action word, but no additional constraints are included. Once such rules gain sufficient strength, they begin to generate errors of commission by applying in inappropriate situations. In these cases, AMBER invokes a discrimination learning mechanism to identify differences between positive and negative instances of the overly general rule. This creates more conservative rules with additional conditions, which (after gaining sufficient strength) eliminate the errors of commission.

Now let us reconsider Langley's system in light of our four components of learning. First, we see that AMBER uses the meaning of a sentence, together with the main topic, to determine a tree structure that is very similar to a parse tree. In fact, this tree contains all of the information in a parse tree except the word order, which is available from the sentence itself. As in LAS, this tree structure tells AMBER which basic chunks it should form, solving the *aggregation* problem.

Although their basic response to this problem is the same, the two system differ in their implementation details. In particular, Langley's system does not require an explicit statement of the graph deformation condition, since this falls out of the model's mechanisms for processing goals.[12] In this sense, AMBER's approach to chunking is similar to Anderson's (1983) composition learning method, which creates chunks based on goal trees.

Unlike the other language acquisition systems we have discussed, AMBER does not formulate explicit syntactic word classes. Rather, the system states its "grammar" entirely in terms of semantic roles like agent, action, color, and shape. This corresponds to

[12] Anderson's (1981) ALAS employs a very similar response to the chunking problem, and to modeling first language acquisition in general. The two systems employ quite similar representations and learning mechanisms, though AMBER accounts for somewhat earlier stages than ALAS.

children's early utterances, though eventually they move beyond the semantic stage to more abstract syntactic classes like nouns and verbs. In any case, this means that AMBER has no explicit response to the *clustering* problem for content words. The system does have to group sentences into positive and negative instances for each function word, but this is easily done by seeing whether each word occurs in the expected position.

Langley's model has two distinct responses to the *characterization* problem. Once it has identified positive and negative instances for the various function words, AMBER invokes a discrimination mechanism to determine the semantic conditions for each word. We discussed this method earlier in the context of learning from examples. Basically, it is a data-intensive technique that begins with general hypotheses and generates more specific ones as errors of commission occur, using a differencing technique to determine new conditions. Anderson (1983) and Langley (1985) have used similar methods for other domains, including learning from examples and heuristics learning. AMBER combines the discrimination process with a strengthening mechanism that serves to direct search through the space of hypotheses, as well as modeling the gradual nature of children's mastery of function words.

In contrast, AMBER employs a quite different strategy to identify conditions on rules for content word order. In this case the system learns from a single instance, rather than relying upon a method that requires multiple observations. Learning occurs when the system correctly generates one content word but omit another content word it should have produced. To determine the relevant conditions, the system finds the *path* through the semantic network that connects the two content words, and includes all links along this path as conditions in the new production rule. Although the details differ, this strategy is similar to that used by LAS, in that the system reasons about the meaning of a sample sentence to decide which conditions are relevant. Although AMBER must relearn the resulting rules many times before they affect behavior, the same conditions would be determined in each case. Thus, Langley's system uses a simple form of explanation-based learning to acquire rules for content words, rather than the empirical method it uses for function words.

Although it makes no explicit response to *storage* issues, AMBER is implemented as a production system model. Newell and Simon (1972) have argued that production systems are a viable model of human long term memory, accounting for a variety of robust phenomena exhibited in human cognition. Moreover, Forgy (1979) has proposed a method for efficiently storing large numbers of production rules that takes advantage of shared features, and for efficiently matching against these rules by retaining partial matches. AMBER is implemented in PRISM (Langley & Neches, 1981), a production system architecture that employs Forgy's storage and matching methods to provide reasonable performance even when large numbers of rules are involved. Thus, the system provides a plausible response to issues of the storage and retrieval of grammatical knowledge.

Negative Instances in Grammar Learning

Before concluding our review, we should add a few words about the role of negative instances in grammar learning. In an earlier section, we saw that many learning methods rely on negative instances to direct their search through the space of hypotheses. Characterization methods that find common structure employ such instances to determine when a description is overly general, and thus should be eliminated. Characterization methods that find differences use negative instances to determine how overly general descriptions should be made more specific. We found that negative instances are heavily used in learning from examples, where they are provided by the tutor. However, they are also used in heuristics learning and conceptual clustering, where they must be generated by the learning system itself.

Only one of the grammar learning systems we discussed (Langley's AMBER) actually employs negative instances, but a number of other systems have also used this type of information, including Reeker's PST (1976) and Anderson's ALAS (1981). At first glance, the use of negative instances may seem odd, since these models are given only examples of *legal* sentences. However, AMBER and its relatives are not dealing with positive and negative instances at the level of the entire sentence. Rather, they are learning the conditions on rules or networks that deal with only *parts* of sentences.[13] Moreover, they are not acquiring the ability to judge grammaticality, but to *map* sentences onto their meanings and vice versa.

The presence of sentence meanings makes a major difference. Since a particular word or phrase may fail to occur in the presence of a particular meaning, negative instances become possible. As a result, one can use difference-based characterization methods such as discrimination (Langley, 1985) that require comparisons between positive and negative instances. Let us consider a brief example of how this can occur. Suppose the learner knows that ed may occur after a verb or action word, but not exactly when. Each case in which the ending does occur is marked as a positive instance of ed, while each case in which it fails to occur is marked as a negative instance. Based on this clustering, one can systematically search the space of characterizations to determine which semantic conditions best predict the occurrence of the ending. Similar methods could be used for content words or larger structures, such as phrases.

Let us repeat that we do *not* mean that children receive negative evidence in the form of ungrammatical sentences. However, we do mean that one can *generate* negative instances from sentences paired with their meanings, and use this information in the grammar learning process. The ability to do this relies on an important assumption that has not been clearly stated in earlier papers taking this approach: there must be a one-to-one mapping between sentences and meanings.[14] If this *uniqueness* assumption does not hold, then

[13] Berwick (1979) could also have employed negative instances at the rule level by noting which actions failed to allow a successful parse. However, LPARSIFAL did not employ this information, since its search for rules was already sufficiently constrained.

[14] We direct the reader to other chapters in this volume for a fuller treatment of the uniqueness assumption. In particular, the chapters by Clark, by Pinker, and by MacWhinney and Sokolov all

one cannot infer that a missing word or phrase implies a negative instance; the construct might be perfectly acceptable, but the speaker has simply decided to describe the meaning another way.

Thus, the uniqueness assumption guarantees that a missing construct constitutes a negative instance of that construct. This considerably simplifies the learning task, since one can then use the inferred negative instances to eliminate overly general rules or to formulate more specific ones. Of course, learning mechanisms that can handle noise (such as strength-based methods) might still learn if this assumption is not met, and in this case, more frequent constructs would come to be preferred. Still, the greater the degree to which the assumption is violated, the more difficult the grammar learning task will become.

A Research Proposal

Our review of computational approaches to language acquisition would not be complete without some evaluation of this work, and some suggestions for future efforts. For instance, we might evaluate various systems in terms of the psychological and linguistic validity, but this would not really be fair. Of the four grammar learning systems we have examined, only Langley's AMBER (1980, 1982) is intended as a psychological model of first language acquisition, and thus made a serious attempt to account for child language data. Other models have been proposed by Kelley (1967), Reeker (1976), MacWhinney (1978a, 1983), Selfridge (1979), and Hill (1983), but the majority of AI research on language learning has not attempted to explain the observed phenomena. We would like to encourage more work of this sort, but even ignoring this issue, the existing systems suffer on other dimensions. In this section, we consider their limitations and outline an alternative approach that we are using in our own work.

Limitations of Previous Research

One problem with the existing work is its focus on grammar learning to the exclusion of other aspects of language acquisition. A few systems, such as Siklóssy's ZBIE (1968) and Selfridge's CHILD (1979), learn to associate words with concepts, but the majority assume that these connections are present at the outset. Wolff's SNPR (1978, 1982) acquires words themselves as well as grammars, and MacWhinney (1983) has modeled the development of morphophonology, but these are distinct exceptions to the rule. In addition, all existing systems focus on generating or understanding *correct* sentences rather than *interesting* ones. We know of no system that acquires pragmatic knowledge for determining what one should talk about in a given context. Ultimately, we would like an integrated theory of language acquisition that incorporates all of the above components.

The previous work has also ignored interactions between the process of language acquisition and other aspects of cognition, such as concept formation. Thus, it fits well with the traditional machine learning focus on isolated tasks like those we reviewed earlier in

---

make use of this assumption, though not always by the same name. For example, Clark calls it the *principle of contrast*.

the paper. However, it is clear that concept learning has major implications for language acquisition, and a complete model would take their interaction into account. One can make similar arguments for other components of intelligence.

A more subtle criticism concerns only those systems that learn mappings between sentences and their meanings. However, this assumption holds for all psychological models of grammar learning, an important class of learning systems. The problem is that the representation of meaning is provided by the programmer, and this leaves considerable room for hand-crafting the input. Similar problems arise for other machine learning tasks, but the nature of the grammar learning task emphasizes the issue. There are two ways in which such "cheating" can be embedded in the meanings presented to a model of language acquisition.

First, one may employ concepts and features that are well-suited to the language being learned. For instance, some systems allow a *progressive* feature to be associated with the action of an event. This considerably simplifies the acquisition of the English ing construct, while the progressive concept is useless for other languages that make orthogonal distinctions. Second, one may include in the meaning representation only those features that are relevant to the learning task. This lets one avoid modeling the process of focusing attention on important aspects of the environment. All existing cognitive models of language acquisition suffer from such hand-crafting; the "kludges" have moved out of the models (which are often quite general) and into the inputs.

An Alternative Approach to Modeling Language Acquisition

We have been somewhat unfair in criticizing machine learning's emphasis on isolated, idealized tasks, and equally unfair in criticizing models of grammar learning for their carefully crafted inputs. Simplifying assumptions are always helpful when one is first attempting to understand a problem area, and the simplifications that occurred were natural ones. However, the history of artificial intelligence reveals a recurring trend – after the components of a problem are reasonably well understood, more "complex" problems may become easier to solve than the original "simple" ones.

An example from vision research should clarify the trend. Early work in this area focused on the idealized problem of constructing three-dimensional models from very well-lit scenes. Methods for solving this problem involved considerable computation and search. However, when the "harder" task of working with shadowed scenes was attempted, many of the difficulties disappeared. In retrospect, the reason is obvious; the presence of shadows provided constraints that were absent in the original, idealized task, and this significantly reduced the space of possible interpretations.

We believe that research on computational models of language acquisition would benefit from similar strategy. Since this work is still in its early stages, we will spend the remainder of the paper on the constraints we have set ourselves, rather than on solutions to them. Such constraints can occur at two distinct levels. First, one can make the modeling problem more difficult for oneself, using the resulting constraints to direct search through the space of possible models. Second, one can make the language acquisition task more difficult for

the learning system, providing additional constraints for the system to use itself. We plan to use both of these strategies in our work on language acquisition.

One way to constrain our search for mechanisms of language acquisition is to model human behavior in this domain. In addition to the intrinsic interest of this endeavor, human children are still our best examples of language learning systems, making them obvious objects of study. MacWhinney (1978b) has proposed nine criteria that should be satisfied by computational models of human language acquisition. Here we will list only four broad classes of constraints that our model should meet, but each of these is sufficient to rule out many of the approaches that have been previously explored.

First, it is clear that children acquire language in an incremental fashion, so our learning mechanisms must have this characteristic as well. Second, it is clear that humans learn not only to judge the grammaticality of sentences, but to map sentences onto their meanings, and our model must do the same. Third, the model should be consistent with our knowledge of the human cognitive architecture. For instance, Newell and Simon (1972) and Anderson (1983) have argued that production systems are central to human cognition; this makes production systems an obvious framework to consider, though certainly not the only one. Finally, children progress through clearly identifiable stages during their acquisition of language, and our model should account for these stages. Thus, the model should progress from the one word stage, through a telegraphic stage, and eventually produce complete adult sentences. We have not decided the level of detail we should strive to explain, but even the highest levels significantly limit the space of models.

Towards an Integrated Model of Learning

In addition to developing a psychologically plausible model of first language acquisition, we hope to develop a more complete model that moves beyond grammar learning in isolation. The planned system must learn to recognize and generate words like `bounce` and `ball`, and it must also associate these words with particular concepts. The model will have to learn the mapping between combinations of words (sentences) and their meanings, and to acquire heuristics for generating useful sentences. Moreover, these different components must be integrated into a single model of the language acquisition process.

The advantage of this approach is that the various learning tasks should feed into each other, reducing the learner's reliance on carefully crafted inputs. For instance, the model will initially learn the meanings of words based on repeated situations in which a given word and concept cooccur. This knowledge can then be used to aid the grammar learning process, much as existing grammar learning systems use word meanings. However, once an initial grammar has been acquired, this can be used in turn to learn new word meanings from context (Granger, 1977). Such positive feedback would let the system move away from its initial reliance on sentence-meaning pairs.

We also hope to integrate the language acquisition process with a model of both concept formation and problem solving. It is clear that children have many concepts in memory before they associate words with them, and the concept formation process must account for their origin. Presumably, some concepts will be acquired later than others, and this

may account for the fact that certain words are learned relatively late. Thus, the model of concept formation may contribute to explaining phenomena that appear entirely linguistic at first glance. The causal arrow may point in the other direction as well, since language may be used to communicate new concepts once it has advanced sufficiently.

We believe the problem solving process is also important to language acquisition, since it is responsible for the generation of goals, and for the creation of plans to achieve these goals. Many of children's early utterances seem to revolve around goals such as easing hunger and getting attention. If we hope to explain these utterances, we must account for the origin of these goals, and thus the need for a model of problem solving. Also, it seems quite likely that an explanation of pragmatic rules and their acquisition will revolve around goals, and a full account must explain how these goals originate.

Learning in a Reactive Environment

Machine learning researchers have traditionally focused on abstract, symbolic tasks like learning from examples and heuristics learning. Not surprisingly, they have attempted to cast the task of language acquisition in the same mold, providing well-defined inputs such as sentence-meaning pairs, and expecting clean categorical rules or grammars as outputs. However, humans learn language in the context of a complex physical world, and our model of the acquisition process should reflect this fact.

In the World Modelers Project, we have implemented a complex simulated environment in which our learner will perceive and act, much as a human child does in the real world. There are three central motivations for using this simulated environment: (1) to provide our learning model with (qualitatively) the same class of inputs as a human learner might find in the real world, rather than some mathematical abstraction of preselected information with no surrounding context; (2) to investigate reactive learning, in which the learner can experiment with different ways of generating language or action, and directly observe the behavioral consequences of its linguistic or physical acts; and (3) to provide situations in which learning can be guided by the pursuit of goals, rather than being an end in itself.

The simulated environment supports three-dimensional objects (such as furniture and toys), and these objects obey standard physical laws involving gravity, friction, and torque. The learner itself has a (simplified) physical body that lets it move around and affect its surroundings, as well as senses that let it observe these surroundings. Carbonell & Hood (1985) describe the simulated environment in more detail, but the important point is that it has much the same flavor as the environment in which a child learns language. To date, we have constructed only very simple agents, but our long-term goal is to develop an integrated model of learning (including the acquisition of language) within this environment.

The planned agent will perceive its surroundings through various senses (sight and hearing), and store the resulting descriptions in memory. This has important implications for the language acquisition task, since it means that the programmer need no longer spoon-feed the meanings of sentences to the model. In fact, such direct transfer of information is explicitly forbidden; the learner will have access *only* to what it can see and hear.

This will force us to deal explicitly with two issues we raised earlier: giving the model hand-crafted features like *progressive* to ease the learning task; and limiting the agent's attention by presenting only relevant aspects of an event. Let us consider some responses to these issues that we plan to explore in our model of learning in a reactive environment.

Rather than provide the system with arbitrary high-level features, we must show how these concepts arise naturally out of an integrated cognitive architecture. For instance, any system that interacts with a physical world must have some representation for time and be able to use this in describing events. The system must be able to distinguish between events that are currently occurring and those which are not. Thus, such a system will already have one of the features needed to state progressive rules (like the English `ing`), which are used only in describing ongoing events. However, this feature arises naturally from the architecture itself and has many uses, rather than being given by the programmer specifically for the grammar learning task. We believe that most other features can be handled in an analogous manner.

Similarly, we should not simplify the grammar learning task by providing only the relevant features of an event. Instead, we must model the process by which the learner focuses on relevant information and ignores other features, and this requires a model of *attention*. We believe that existing concepts and schemas generate goals and expectations, and that humans use these expectations to filter the overwhelming information provided by their senses. Attention is initially focused by specific object and procedural concepts such as `ball` and `bounce`, since the child's early interactions with the world lead to such concepts. This bias helps account for the dominance of content words in early speech. Only later, after he has mastered content words, does the child turn to function words. If the learner is unable to account for these with existing schemas, he must "loosen up" his filter and examine other features that he previously ignored. Different features will prove useful in different languages, and this is the point at which grammar can influence the learner's knowledge structures in significant ways.

In addition, placing the learner in a physical world should lead naturally to a variety of goals, such as easing hunger. If the agent has only limited manipulation capabilities (e.g., an object may be out of reach), then it will have significant motivation to communicate its goals, in the hope that another agent (e.g., a parent) will satisfy them. Thus, goals will play a central role not only in our model of problem solving and concept formation, but in our model of language use as well. In general, the agent will talk about what it wants, rather than describing random objects and events. Such goal-driven focus of attention should constrain the otherwise combinatorially intractable problem of correlating linguistic utterances with physical objects or actions. Moreover, these goals will arise naturally from the learner's interaction with the world, rather than being provided by the programmer.

This opens the way to modeling the development of discourse strategies. We expect that the learner will usually be accompanied by an "adult" who has both sophisticated language skills and a repertoire of actions available to it. (This agent will be directly controlled by the programmer, so we can "put words into its mouth".) Thus, the learner will be able to make demands, ask questions, and exhibit a variety of linguistic behaviors

beyond simple declarative sentences. Different types of sentences can be used to satisfy different goals, and the model must acquire the proper distinctions from experience.

Also, the presence of world knowledge provides an explanation for why understanding often *appears* to precede production (generation). Although the process of concept formation leads to knowledge of the world without need for language, this does not mean that the knowledge cannot be used in linguistic contexts. Upon hearing a sentence that it only partly understands, the agent may well take the appropriate action based on its previous experience.

For example, suppose the child hears `Go to the door`, but only knows the meaning of `door`. Since moving towards an object is a common strategy used to explore one's surroundings, the child may perform the desired action, even with no knowledge of syntax and little knowledge of semantics. Selfridge (1981) has used a similar approach in his model of first language acquisition. However, he provided the learner with the necessary knowledge structures, rather than modeling the process through which this knowledge is acquired, as we plan.

Of course, to the extent that such non-linguistic strategies are useful, the child will have little reason to learn word meanings and grammar. But in many cases, this approach will lead to behaviors that the adult does not desire, and the learner will observe his displeased response. For example, the above strategy would produce the same response to `Close the door` (the child would go to the door), but in this case it would be incorrect. At this point, the parent might demonstrate his intention by closing the door and repeating the word `close`. Only in a reactive environment can such interaction be modeled and exploited for learning.

This approach to language learning should work in the opposite direction as well. Suppose the child says `ball` to request that an adult bounce the ball to him, and then sees the adult place the ball in his pocket. This is a violated expectation, and as we described earlier, such failures can be used to generate the negative instances that are so useful to learning mechanisms. Experiences of this type will encourage the child to use complete sentences to achieve his goals, rather than isolated words or telegraphic sentences.

We can say little more about the model at this point, since it exists only in the most abstract terms. In fact, we have said more about the task we have set ourselves than the model. However, we feel the nature of this task is central, since it will force us to deal with issues that have been ignored (or at least postponed) in previous work on language acquisition. We feel that the goals of modeling human behavior, developing an integrated model of learning, and examining learning in a reactive environment will lead us down paths that have never been traversed, but which are essential if we hope to understand the full nature of language acquisition

Summary

In closing, let us briefly review the main points of the paper. We have seen that the field of machine learning has addressed a number of distinct tasks, including learning from examples, heuristics learning, conceptual clustering, and learning macro-operators. Significant work has also been carried out on the problem of learning grammars from sample sentences. We described each of these tasks in terms of four components or subproblems – aggregation, clustering, characterization, and storage. We found that only the grammar learning task forces one to address all four of the components, making it the most complex of the learning problems we examined.

We also saw that in the area of language acquisition, machine learning researchers have focused almost exclusively on grammatical knowledge, and we reviewed four systems that acquire such knowledge. These systems differed along a number of dimensions, including their representation of grammars, their reliance on sentence meanings, and the actual learning mechanisms they employed. Each system also had its own response to the four components of learning given above. Finally, we discussed some problems with existing computational approaches to language acquisition, and outlined an alternative approach in which we plan to integrate different aspects of the learning process, and in which we plan to model learning in a complex, reactive environment.

We have no illusions that developing an integrated model of language learning will be easy. Nor do we believe that we will succeed in any absolute sense. However, we do believe that the attempt to construct such an integrated model will lead to questions that have never before been asked, and to some tentative answers that future researchers will expand and improve upon. In the long run, we expect that this strategy will lead to our common goal – a fuller understanding of the mechanisms that underly language acquisition.

References

Anderson, J. R. (1977). Induction of augmented transition networks. *Cognitive Science, 1*, 125–157.

Anderson, J. R. (1981). A theory of language acquisition based on general learning principles. *Proceedings of the Seventh International Joint Conference on Artificial Intelligence* (pp. 165–170). Vancouver, B.C., Canada.

Anderson, J. R. (1983). *The architecture of cognition.* Cambridge, Mass.: Harvard University Press.

Anzai, Y., & Simon, H. A. (1979). The theory of learning by doing. *Psychological Review, 86*, 124–140.

Berwick, R. (1979). Learning structural descriptions of grammar rules from examples. *Proceedings of the Sixth International Conference on Artificial Intelligence* (pp. 56–58). Tokyo, Japan.

Berwick, R. (1980). Computational analogues of constraints on grammars: A model of syntactic acquisition. *Proceedings of the 18th Annual Conference of the Association for Computational Linguistics* (pp. 49–53). Toronto, Ontario, Canada.

Bruner, J. S., Goodnow, J. J., & Austin, G. A. (1956). *A study of thinking.* New York: Wiley.

Carbonell, J. G., & Hood, G. (1985). The world modelers project: Objectives and simulator architecture. *Proceedings of the Third International Machine Learning Workshop* (pp. 14–16). Skytop, PA.

Clark, E. (1986). The principle of contrast: A constraint on language acquisition. In B. MacWhinney (Ed.), *Mechanisms of language acquisition.* Hillsdale, N.J.: Lawrence Erlbaum.

Everitt, B. (1980). *Cluster Analysis.* Heinemann Educational Books, Ltd.

Fisher, D. (1984). *A hierarchical conceptual clustering algorithm.* (Technical Report) Department of Information and Computer Science, University of California, Irvine.

Fisher, D., & Langley, P. (1985). Approaches to conceptual clustering. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (pp. 691–697). Los Angeles, CA.

Forgy, C. L. (1979). *On the efficient implementation of production systems.* Dissertation, Department of Computer Science, Carnegie–Mellon University, Pittsburgh, PA.

Garvin, P. I. (1967). The automation of discovery procedure in linguistics. *Language, 43*, 172–178.

Granger, R. H. (1977). Foul–Up: A program that figures out words from context. *Proceedings of the Fifth International Joint Conference on Artificial Intelligence* (pp. 172–178). Cambridge, MA.

Hayes–Roth, F., & McDermott, J. (1978). An interference matching technique for inducing abstractions. *Communications of the ACM, 21*, 401–410.

Hedrick, C. (1976). Learning production systems from examples. *Artificial Intelligence, 7*, 21–49.

Hill, J. A. C. (1983). *A computational model of language acquisition in the two-year-old.* Dissertation, Department of Computer Science, University of Massachusetts, Amherst.

Horning, J. J. (1969). *A study of grammatical inference.* (Technical Report No. CS 139) Computer Science Department, Stanford University, Stanford, CA.

Iba, G. Learning by discovering macros in puzzle solving. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (pp. 640–642). Los Angeles, CA.

Kelley, K. L. (1967). *Early syntactic acquisition.* (Technical Report P-3719) The Rand Corporation, Santa Monica, CA.

Klein, S., & Kuppin, M. A. (1970). *An interactive, heuristic program for learning transformational grammars.* (Technical Report No. 97) Computer Sciences Department, University of Wisconsin, Madison.

Knowlton, K. (1962). *Sentence parsing with a self–organizing heuristic program.* Dissertation, Massachusetts Institute of Technology, Cambridge, MA.

Korf, R. E. (1982). A program that learns to solve Rubik's cube. *Proceedings of National Conference on Artificial Intelligence* (pp. 164–167). Pittsburgh, PA.

Laird, J. E., P. S. Rosenbloom, & A. Newell. (1986). SOAR: The anatomy of a general learning mechanism. To appear in *Machine Learning, 1.*

Langley, P. (1980). A production system model of first language acquisition. *Proceedings of the Eighth International Conference on Computational Linguistics* (pp. 183–189). Tokyo, Japan.

Langley, P. (1982). Language acquisition through error recovery. *Cognition and Brain Theory, 5*, 211–255.

Langley, P. (1985). Learning to search: From weak methods to domain-specific heuristics. *Cognitive Science, 9*, 217–260.

Langley, P., & Neches, R. T. (1981). *PRISM User's Manual.* (Technical Report) Computer Science Department, Carnegie–Mellon University, Pittsburgh, PA.

Langley, P., & Ohlsson, S. (1984). Automated cognitive modeling. *Proceedings of the National Conference on Artificial Intelligence* (pp. 193–197). Austin, TX.

Langley, P., & Sage, S. (1984). Conceptual clustering as discrimination learning. *Proceedings of the Fifth Biennial Conference of the Canadian Society for Computational Studies of Intelligence* (pp. 95–98). London, Ontario, Canada.

Lewis, C. H. (1978). *Production system models of practice effects.* Dissertation, Department of Psychology, University of Michigan, Ann Arbor.

MacWhinney, B. (1978a). The acquisition of morphophonology. *Monographs of the Society for Research in Child Development, 43.*

MacWhinney, B. (1978b). Conditions on acquisitional models. *Proceedings of the Annual Conference of the Association for Computing Machinery.* New York, NY.

MacWhinney, B. (1983). Hungarian language acquisition as an exemplification of a general model of grammatical development. In D. I. Slobin (Ed.), *The cross-linguistic study of language acquisition.* Hillsdale, N.J.: Lawrence Erlbaum.

MacWhinney, B., & Sokolov, J. (1986). Acquiring syntax lexically. In B. MacWhinney (Ed.), *Mechanisms of language acquisition.* Hillsdale, N.J.: Lawrence Erlbaum.

Marcus, M. (1980). *A theory of syntactic recognition for natural language.* Cambridge, MA: MIT Press.

McMaster, I., Sampson, J. R., & King, J. E. (1976). Computer acquisition of natural language: A review and prospectus. *International Journal of Man-Machine Studies, 8,* 367–396.

Michalski, R. S. (1983). A theory and methodology of inductive learning. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach.* Palo Alto, Ca: Tioga Press.

Michalski, R. S., & Stepp, R. (1983). Learning from observation: Conceptual clustering. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach.* Palo Alto, Ca: Tioga Press.

Miller, G. A. (1956). The magical number seven, plus or minus two. *Psychological Review, 63,* 81–97.

Minsky, M. (1963). Steps toward artificial intelligence. In E. A. Feigenbaum & J. Feldman (Eds.), *Computers and Thought.* New York: McGraw–Hill, Inc.

Mitchell, T. M. (1982). Generalization as search. *Artificial Intelligence, 18,* 203–226.

Mitchell, T. M., Keller, R. M., & Kedar-Cabelli, S. (1986). Explanation-based generalization: A unifying view. *Machine Learning, 1.*

Mitchell, T. M., Utgoff, P., & Banerji, R. B. (1983). Learning problem solving heuristics by experimentation. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach.* Palo Alto, CA: Tioga Publishing Co.

Newell, A., Shaw, J. C., & Simon, H. A. (1960). Report on a general problem–solving program for a computer. *Information Processing: Proceedings of the International Conference on Information Processing* (pp. 256–264).

Newell, A., & Simon, H. A. (1972). *Human problem solving.* Englewood Cliffs, N.J.: Prentice–Hall, Inc.

Neves, D. M., & Anderson, J. R. (1981). Knowledge compilation: Mechanisms for the automatization of cognitive skills. In J. R. Anderson (Ed.), *Cognitive skills and their acquisition* Hillsdale, N. J.: Lawrence Erlbaum Associates.

Ohlsson, S. (1983). A constrained mechanism for procedural learning. *Proceedings of the Eighth International Joint Conference on Artificial Intelligence* (pp. 426–428). Karlsruhe, West Germany.

Pinker, S. (1979). Formal models of language learning. *Cognition, 7*, 217–283.

Pinker, S. (1986). The bootstrapping problem in language acquisition. In B. MacWhinney (Ed.), *Mechanisms of language acquisition.* Hillsdale, N.J.: Lawrence Erlbaum.

Reeker, L. H. (1976). The computational study of language acquisition. In M. Yovits & M. Rubinoff (Eds.), *Advances in Computers*, Volume 15. New York: Academic Press.

Selfridge, M. (1981). A computer model of child language acquisition. *Proceedings of the Seventh International Joint Conference on Artificial Intelligence* (pp. 92–96). Vancouver, B.C., Canada.

Sembugamoorthy, V. (1981). A paradigmatic language acquisition system. *Proceedings of the Seventh International Joint Conference on Artificial Intelligence* (pp. 106–108). Vancouver, B.C., Canada.

Siklóssy, L. (1972). Natural language learning by computer. In H. A. Simon & L. Siklóssy (Eds.), *Representation and meaning: Experiments with information processing systems.* Englewood Cliffs, NJ: Prentice–Hall.

Sleeman, D., Langley, P., & Mitchell, T. (1982). Learning from solution paths: An approach to the credit assignment problem. *AI Magazine, 3*, 48–52.

Smith, D. E. (1982). *Focuser: A strategic interaction paradigm for language acquisition.* Dissertation, Department of Computer Science, Rutgers University, New Brunswick, NJ.

Solomonoff, R. (1959). A new method for discovering the grammars of phrase structure languages. *Proceedings of the International Conference on Information Processing.*

Winston, P. H. (1975). Learning structural descriptions from examples. In P. H. Winston (Ed.), *The psychology of computer vision.* New York: McGraw–Hill.

Wolff, J. G. (1978). Grammar discovery as data compression. *Proceedings of the AISB/GI Conference on Artificial Intelligence* (pp. 375–379). Hamburg, West Germany.

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

END

DTIC

8-86