NOTICE WARNING CONCERNING COPYRIGHT RESTRICTIONS:
The copyright law of the United States (title 17, U.S. Code) governs the making of photocopies or other reproductions of copyrighted material. Any copying of this document without permission of its author may be prohibited by law.

COLLECTED PAPERS ON THE LEARNING AND RECOGNITION OF STRUCTURED PATTERNS

Department of Computer Science
Carnegie-Mellon University
Pittsburgh, PA. 15213

January 1975

"Representation of Structured Events and Efficient Procedures for Their Recognition," by Frederick Hayes-Roth.

"An Automatically Compilable Recognition Network for Structured Patterns," by Frederick Hayes-Roth, and David J. Mostow.

"Schematic Classification Problems and Their Solution," by Frederick Hayes-Roth.

"Uniform Representations of Structured Patterns and an Algorithm for Grammatical Inference[1,2]," by Frederick Hayes-Roth.

The research reported here was supported in part by the Defense Advanced Research Projects Agency under Contract F44620-73-C-0074.

COLLECTED PAPERS ON THE LEARNING AND RECOGNITION OF STRUCTURED PATTERNS

PREFACE

This collection of four papers presents many of the major findings in my research related to the representation, recognition, and learning of structured patterns. The papers have been assembled in one volume to facilitate the reader's task of pursuing these topics over several variations of focus.

The first paper presents an introduction to the problems of representation and recognition of structured patterns and suggests several directions for further research. One of these suggestions concerns a distributed processing network for pattern recognition. In the second paper (co-authored by D. J. Mostow), such a network is developed for use as a syntax system for speech understanding. The last two papers are concerned specifically with the learning problems encountered in the framework of structured patterns. The first of these explains the use of the interference matching technique to abstract schemata from examples, hypothetical class characteristics (conjunctive sets of predicates) which should perform well in related problems requiring classification of novel test items. The last paper explains how such a learning technique may be extended for rule learning or grammatical inference. A computationally universal representation for production systems is introduced, and it is shown how the techniques developed in the previous paper can be generalized to solve learning problems involving the inference of such productions.

In short, these papers provide an overview of a theory of learning and behavior which is more fully explained elsewhere.[1] While the theory is both intuitively plausible and demonstrably formally correct (over the problem domain defined by behaviors whose rules are representable as productions of this sort), its computational feasibility and practical utility have yet to be demonstrated. Several applications, including those discussed in the last paper, are now being undertaken.

F. Hayes-Roth

January 16, 1975

---

[1] Hayes-Roth, F. Fundamental mechanisms of intelligent behavior: the representation, organization, acquisition and use of structured knowledge in perception and cognition. Unpublished doctoral dissertation. Ann Arbor: The University of Michigan, 1974.

FEB 14 '75

HUNT LIBRARY
CARNEGIE-MELLON UNIVERSITY

REPRESENTATION OF STRUCTURED EVENTS

AND EFFICIENT PROCEDURES FOR THEIR RECOGNITION

Frederick Hayes-Roth
Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

ABSTRACT

Structured events are configurations of objects in logical, spatial, temporal or activity relations. A parameterized structural representation system for this class of events is discussed. Parameters in such representations are arbitrarily chosen symbols used to insure consistent references to the same object in diverse relations. All-or-none matching of two representations is the basis for pattern recognition. In this framework, descriptions of pattern or concept prototypes act as structural templates for stimuli. As a result, recognition can be performed in a natural and structural way and is unaffected by manipulations of irrelevant variables. Typical recognition procedures are reviewed and a variety of alternative approaches are considered in light of the potential combinatorial explosions which might arise in applications of these procedures. One alternative is proposed which can exploit both redundancy among partially matching templates and computational parallelism in exhaustive search (recognition) problems. Another possibility considered is to find special recognition procedures for particular recognition problems. For example, to accomplish word recognition in speech understanding systems, highly practical techniques exist to match many templates in parallel (simultaneously) using only simple bit string operations. In addition, both the possibility of additional heuristic approaches to general recognition procedures and the total abandonment of relational representations are also considered in this paper.

INTRODUCTION

Many prevalent questions in information science concerning perceptual, learning, classification, and retrieval functions share a common conceptual base. These functions all necessitate some more or less formal approach to the representation of information and techniques for comparing two or more information structures. In this paper, the notion of a

structured event, a discrete, integrated, relational information structure, is introduced. Structured events provide a desirable basis for the representation of a wide variety of knowledge. Further, transformations between events are easily described and can be used to represent general behavior rules.

The bulk of this paper addresses two basic questions which, for the sake of concreteness, are cast in the area of visual perception: (1) How should a stimulus and a memory prototype (e.g. a Gothic capital letter "A") be represented? (2) How can a stimulus representation be matched to a memory representation to accomplish recognition? The answers to these questions will be easily seen to be generalizable to other problems involving the representation and comparison of information structures. Before proceeding to a detailed discussion of these questions, however, a brief review of previous approaches and an overview of the proposed structural representation theory are provided.

PREVIOUS THEORIES OF REPRESENTATION AND MATCHING

Established theories for the representation and recognition of patterns fall into four categories. These will be called the graphic template model, spatial model, feature model, and generative (syntactic) model. Each class is considered in turn.

The graphic template model (see, for example, Lindsay & Norman, 1972) proposes a simple and seemingly desirable solution to both the representation and pattern matching problems. In brief, a memory prototype is a graphic template or replica of the type of item to be recognized. The representation of a Gothic capital "A" is conceived of as a stencil or photograph of that letter. Stimuli are recognized as "A"'s, roughly speaking, whenever the light which they emit is present where the template is light and absent where the template is dark. The advantages of such a system are that recognition is determined by a direct contact and comparison of homologous stimulus and memory structures and that all classification alternatives may conceivably be evaluated simultaneously. Its disadvantages are well known and are summarized by the statement that a template is an oversensitive representation and one which lacks any basis for generalization. Even insignificant deviations between the stimulus and template (e.g. differences in proportion, size, orientation, font) may completely inhibit proper recognition.

Spatial models underlie scaling theory approaches, signal detection theory, dimensional representations, correlational methods, discrimination analysis, and most decision theory approaches. In brief, a spatial model proposes that a prototype of a pattern class be represented as a single point in an n-dimensional (usually metric) space where each dimension reflects some multivalued attribute which takes a particular value for each stimulus. Any

stimulus item is evaluated on each dimension and is represented by the point at the corresponding coordinates in the pattern space. A stimulus is matched to each memory prototype by some arbitrary distance function, and classification is usually performed by assigning the stimulus to the class associated with the nearest prototype.

These models have the simplicity of representation and matching operations as their outstanding qualities. Their primary disadvantages are a corollary of this simplicity. These models are generally incapable of representing criterial value dependencies among a subset of stimulus attributes. They are also unable to recognize that, although a particular set of features or feature values are relevant to one pattern class, those features may be completely irrelevant to the definition of other classes. One way in which this can be seen is by considering the pattern volume corresponding to each pattern class. In spatial models the procedure of classifying all stimuli close to one prototype to the corresponding class defines a non-empty volume of points around each prototype which corresponds to the related class. If particular features values were irrelevant to some class (as "amount of green paint" is irrelevant to the pattern class "Volvo"), a decision procedure should ignore "amount of green paint" in considering the potential classification of objects as "Volvo". The appropriate pattern volume in such a case would then be a zero-volume hyperarea of some dimensionality less than that of the entire pattern space (Hayes-Roth, 1973, 1974b; Michalski, 1973; Watanabe, 1973).

Feature models like that of Pandemonium (Selfridge, 1959) posit that each class is defined by the simultaneous presence of a particular set of feature values. A prototype is represented as a set of critical feature values and their necessary frequency of occurrence in a stimulus which is to be considered an element of the corresponding class. While these models solve the dependence and irrelevance problems attributed to spatial models, they too have difficulties. Chief among these is that stimuli possessing correct features in an inappropriate configuration are improperly recognized. This is a direct result of the matching process used with feature list representations. A measure of the degree of successful matching of stimulus and prototype is usually computed by determining the proportion of features prescribed by the prototype that are found in the stimulus. Classification is then performed by assigning the stimulus to the category associated with the best matched prototype.

Generative models include grammatical approaches to representation and matching, including linguistic pattern recognition (Uhr, 1971; Shaw, 1972; Joshi, 1973; Thomason, 1973), analysis-by-synthesis (Neisser, 1967), and top-down language understanding systems (Winograd, 1973; Woods, 1970). These approaches employ syntactic and semantic rules to describe the kinds of stimuli that may be produced and encountered in the environment. A

prototype is represented or conceived of in terms of a list of productions, transitions, or transformations which must be employed to generate it from a meaningless starting symbol. Matching of stimuli and prototypes is usually done in two steps. First, the stimulus is synthesized or "forged"; the rules of grammar are employed in some arbitrary manner until a particular sequence of transitions is found which leads to the production of an acceptably close replica of the stimulus. Second, when the transitions can be given an iterpretation which is plausible in the problem domain, that interpretation is employed in some way to effect classification or understanding. One advantage of this type of approach is particularly noteworthy. Given sufficiently powerful grammars, the generative technique is capable of computing all definable recognition functions (Hopcroft & Ullman, 1969). In specific, any structural dependencies or feature irrelevancies can be appropriately incorporated into the rules of the grammar. On the other hand, the power of these approaches is intimately connected to their principal disadvantages. Among these is their usually enumerative, recursive, or non-deterministic searches of the space of possibilities (all sentences or stimuli which can be produced by the grammar). As a result of such search procedures, these programs do not provide plausible models of the apparently simple and immediate recognition that is apparently involved in the perception of a well known pattern like the letter "A". Furthermore, the performance of these systems is seriously degraded by any expansions of the possible set of rules applicable at each step.

In some sense, generative models seem related to natural mechanisms of recognition in the way Turing machines are related to human cognition. In both cases the strength of the relationship rests primarily on the ability of one machine to effect input-output transformations of a complexity equivalent to that of the natural system. Beyond that, there is little to support the notion that the machine provides a believable or otherwise promising model of the natural computations of interest.

STRUCTURAL REPRESENTATION THEORY

Because of the inadequacies of the preceding alternatives, several researchers have suggested the use of predicate-calculus or equivalent graph representations for pattern prototypes and stimuli (Evans, 1969; Narasimhan, 1969; Winston, 1970; Barrow, et al., 1972; Hayes-Roth, 1973, 1974a, 1974b, 1974c). In general, while the details of their approaches vary widely, all of these researchers have proposed some representation for structured events and procedures for all-or-none matching of two event representations to effect recognition. One such representation and related matching algorithm are reviewed here which, because of their generality, subsume the previously proposed alternatives.

Structured events (Hayes-Roth, 1973) are configurations of objects in logical, spatial,

temporal, or activity relations. To represent these configurations, several kinds of elements are used. A relation is a set composed of a predicate item and a number of named predicate object items. A predicate is any property that is asserted to obtain among one, two, or more objects. The functional role of each item in a relation is identified by its prefix symbol, "p" for predicate, and completely arbitrary type names for the predicate objects. Parameters are arbitrarily chosen symbols which name objects and are necessarily introduced in these representations to permit consistent multiple references to an object in diverse relations. Finally, a parameterized structural representation (PSR) is a two-tuple containing a set of parameters and a set of parameterized relations which are simultaneously true and which constitute the description of a single event of interest.

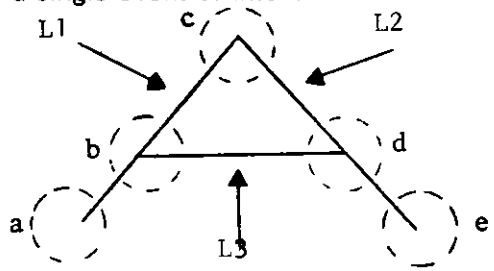


Figure 1. Structural Template as Prototype.

Although the current paper is primarily directed to the structure and formal properties of such representations, it will help to consider the concrete example in Figure 1, in which a Gothic capital letter "A" is illustrated and the eight parameters used in its description are indicated. The corresponding PSR is labelled A and is given below:

```
A: (  {a, b, c, d, e, L1, L2, L3},
      {{p:on, node:a, line:L1},
       {p:on, node:b, line:L1},
       {p:on, node:c, line:L1},
       {p:on, node:c, line:L2},
       {p:on, node:d, line:L2},
       {p:on, node:e, line:L2},
       {p:on, node:b, line:L3},
       {p:on, node:d, line:L3}})                    (1)
```

Each parameter in the parameter set {a, b, c, d, e, L1, L2, L3} identifies either a node (a terminus of a line) or a line of the line drawing of the "A". The body of the PSR contains relations asserting that each of the first five parameters is a "node" in the graphic structure and that several nodes are incident "on" several lines.

It is not claimed that the PSR (1) is an accurate model of what humans acquire when they learn the corresponding pattern. However, such a representation has several salutary

features. The representation is insensitive to attributes of an "A" which are not essential to the basic concept. A similar description would be true of any "A" which reflected these structural dependencies regardless of its orientation, font, proportion, etc. As a result, the PSR labelled A is considered a structural template. It describes those properties-- visual, in the current case--of any stimulus which are necessary for it to be classified as a proper instance of the class of block capital "A"'s. Although it is possible that additional features (unary relations) or other n-ary relations (n=2, 3, ...) might be present in any exemplar of this class, all of these may be considered superfluous here. Because the number of elementary relations in a predicate calculus representation or in a PSR may make their complete listing tedious, a compact form is also used in this paper. In that form relations with redundant predicates and permutable objects are merged and represented by their set union. If this is done for PSR A for Fig. 1, the following compact PSR is produced:

```
A: (  {a, b, c, d, e, L1, L2, L3},
      {{p:on, node:a, node:b, node:c, line:L1},
       {p:on, node:c, node:d, node:e, line:L2},
       {p:on, node:d, node:b, line:L3}})                    (2)
```

Before considering the use of structural templates in recognition, the general properties of the proposed structural representation system should be elaborated. Predicates may be chosen with complete freedom where the only consideration need be the task to be performed. Object type names, specified by the particular prefix symbols preceding objects in a relation, are also arbitrary and should be chosen to suit the specific task. Some predicates may be able to accept more objects than are of interest at any moment, and these others need not be specified at all. For example, one might conceive of a distinct predicate for each verb-sense in a language and associated predicate objects representing each of the cases associated with that verb-sense, e.g. agent, instrument, location, etc. In some instances, the order of a relation (the number of possible predicate objects) will not be the number of distinct object prefixes. For example, a symmetric binary (second order) relation would have two object items with identical prefixes (e.g., {p:equal, value:x, value:y}).

Any set of relations may be named by assigning it an arbitrary parameter symbol, and the corresponding labelled event may be cited as the object of some predicate by referencing that symbol as the appropriate predicate object (Hayes-Roth, 1973c). This is particularly useful where a total configuration comprises several perceptually or logically distinct subevents which are related as unitary wholes in some way. Examples of such representations can be found in the translation of a linguistic phrase structure tree into a corresponding PSR, with each node in the tree associated with a parameter representing the set of descendants emanating from that node.

Although the PSR's thus far described have employed only parameters as predicate objects, in some sense this is not essential. When all objects are parameters and all content words are restricted to use as predicates, the PSR's are considered to be in explicit form. Otherwise, they are in implicit form. The relationship between the two forms is evident in the following semantically equivalent PSR's for the sentence, "John who is a very tall adolescent likes Mary to run with him."

```
(   {t, u, v, w, x, y, z},
      {{p:name, person:w, word:x},{p:name, person:y, word:z},
      z:{{p:Mary}}, {p:tall, how much:v, who:w},
      x:{{p:John}}, v:{{p:very}}, t:{{p:run}},
      {p:adolescent, who:w}, u:{{p:do together, what:t, agent:w, agent:y}},
      {p:like, agent:w, object:u}})                                    (3)

(   {u, w, y},
      {{p:name, person:w, word:John}, {p:name, person:y, word:Mary},
      {p:tall, how much:very, who:w}, {p:adolescent, who:w},
      u:{{p:do together, what:run, agent:w, agent:y}},
      {p:like, agent:w, object:u}})                                    (4)
```

The advantage of an implicit form like (4) is that it is easy to read because the content words which actually represent predicates may be used as if they were constants within relations. The value of an explicit form like (3) is that it clearly identifies every proposition predicated about the event. As a result, the comparison of any two structured events is most easily performed on explicit PSRs because irreconcilable differences between two patterns will always result from some lack of correspondence between predicates and will therefore be easily interpreted. Thus, any stimulus whose representation contains corresponding relations to those of a template can be seen to match the template. On the other hand, when explicit representations are being partial-matched to generate abstractions which are potential pattern templates (Hayes-Roth, 1973, 1974a, 1974c), irreconcilable differences (residuals in the matching process) constitute sets which may be considered categories of constants. In any given recognition problem, the pattern template may specify that only members of particular categories may occur in particular relations. For example, a pattern template for recognizing the occurrence of a sentence in active voice for the active-to-passive rule of transformational grammar may require that the subject of the sentence be in the category noun phrase, one of whose members is required, in turn, to be in the category noun.

The representation of programmatic rules of behavior in this framework is achieved through the use of transformations between contingency and response event PSRs The contingency PSR describes an internal condition or state of the computing system which must

be satisfied before the transformation is invoked and the response occurs. This view of computing is like that in production systems (Minsky, 1967; Newell, 1972; Becker, 1973), web grammars (Pflatz & Rosenfeld, 1969), and pattern-directed procedure invocation (Hewitt, 1972; Rulifson, et al., 1972). The response PSR describes conditions which are to occur as a result of the successful invocation of the transformation. A computing system which is constructed in this way is called interrupt-driven or stimulus-driven, because computing is directly controlled by the detection of conditions of importance. Transformations whose contingencies are descriptions of stimulus encodings and whose responses describe appropriate classification or recognition behaviors are called categorical. Substitution transformations, which employ one or more parameters in both the contingency and response PSR's which are common between them, represent behaviors in which particular values are extracted from a stimulus and are incorporated directly into the related response configuration.

These two types of transformations alone are sufficient to represent all conceivable procedures. Although categorical transformations are really special cases of substitution transformations, they may both be induced by effective algorithms which are somewhat different in the two cases (Hayes-Roth,1974a, 1974c). Transformations have been introduced at this point to complete the structural representation story and to facilitate the reader's appreciation of the generality of the later results.

Enough detail has now been provided on the representation question to permit the consideration of all-or-none matching, the logical basis of structured pattern recognition.

ALL-OR-NONE MATCHING: THE LOGICAL BASIS OF RECOGNITION

Presuming that known pattern prototypes are represented by PSR's, the task of recognizing the occurrence of a pattern exemplar is that of detecting when a stimulus contains all of the structural relations specified by the prototype. Consider the example in Figure 2 of

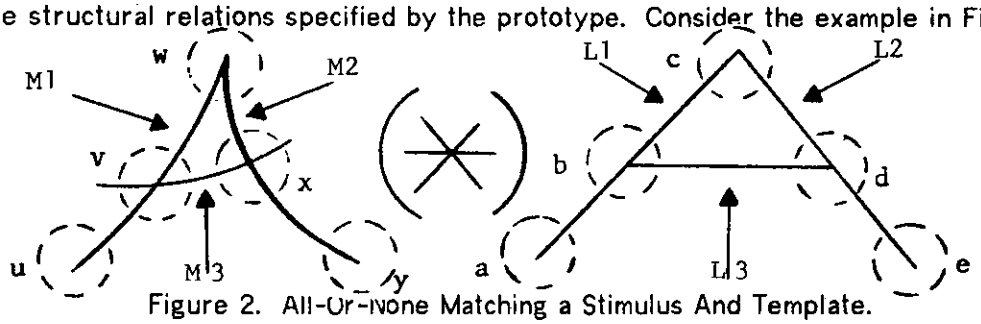


Figure 2. All-Or-None Matching a Stimulus And Template.

determining whether the stimulus S matches the structural template A previously described (2). Two problems arise as a direct result of the introduction of arbitrarily chosen parameter symbols into the representations of both the stimulus and prototype. The binding problem

concerns the determination of a distinct correspondent among the parameters in S for each parameter in the template A. Corresponding parameters represent objects which play equivalent roles in their respective patterns, those roles being defined by the relations in the template. In the current example it is intuitively apparent that each node or line in A corresponds to the node or line in S which occupies a similar locus in the drawings. The correspondences are denoted u≡a, v≡b, w≡c, x≡d, y≡e, M1≡L1, M2≡L2, and M3≡L3.

The solution of the binding problem is, however, closely allied to a solution of another problem, the matching problem. The matching problem concerns the identification of a corresponding relation in S for each relation in A such that the two relations are completely comparable when the alphabetic differences between corresponding predicate object parameters are ignored. In the current example, it is clear that every relation in A is contained in S under the stated parameter equivalences. Thus S is said to match A, denoted S(*)A, and the interpretation is given that the pattern S contains the subpattern A.

In short, having solved the binding problem, the matching problem is to determine whether every relation present in A is also present in S. On the other hand, a desirable solution to the binding problem must necessarily provide a complete solution to the matching problem, too. In the next section, two previously published all-or-none matching procedures are briefly reviewed so that the proposed alternatives may be more fully appreciated.

PREVIOUSLY PUBLISHED PROCEDURES FOR ALL-OR-NONE MATCHING

The first algorithm, the interference matching procedure (Hayes-Roth, 1974a), is a three step serial procedure for simultaneously solving the binding and matching problems. The procedure is easily described in terms of the concept of models of two or more eveents. A model of a set of events with PSRs E_1, E_2, ..., E_n comprises the following components: a set of parameter correspondences defining equivalent parameters in each of the E_i; an abstraction E which is a set of relations common to each of the E_i when alphabetic differences between corresponding parameters are ignored; and a set of residuals, R_1, R_2,..., R_n which are sets of relations present in each original E_i, respectively, which are not represented in the common abstraction E. Distinct models of a stimulus and a template will describe alternative ways of matching the stimulus to the template (e.g. identifying more than one occurrence of a particular subpattern in a stimulus pattern). Distinct models of any set of events identify alternative ways in which each element in the set is similar to the others. The former interpretation will now be elaborated as the use of models in the all-or-none interference matching procedure for pattern recognition is explained.

The three steps of the procedure as applied to the question of determining whether S matches A are as follows:

Step 1. The stimulus and template are represented as explicit PSRs S and A, respectively. In parallel, all relations in S are compared with all relations in A. Each pair of relations from S and A which contain identical or semantically equivalent predicates are matched to produce a corresponding model. The abstraction in this model contains a single relation of the same type as in both compared relations. Each parameter in this relation is assigned a new symbolic name identifying the pair of parameters from S and A which must be assumed to correspond in the original patterns to allow the interpretation that the relations are exactly equivalent. The residuals of S and A in this model are all relations originally present in S and A, respectively, except for the matched relations represented in the new abstraction. In each model, the set of parameter correspondences contains each pair of parameters assumed to correspond by the forced equivalence of matched relations from S and A. At all times during the interference matching procedure, only models which entail consistent parameter bindings are allowed; that is, each parameter in A can be assigned at most one correspondent in S.

Step 2. In parallel, all pairs of consistent and mergeable models are combined to form new, more informative models. Two models are consistent if their parameter correspondences are jointly consistent in the previously described sense. They are mergeable if all residuals in each model contain the abstraction of the other model. That is each model to be merged must reduce the unexplained residuals of the other by identifying some relations which are common to both and which are complementary to those already identified. When these conditions are satisfied, the models are combined by following these simple rules: the set of parameter correspondences of both models are merged by forming their set union; and each residual in the new model is the set intersection of the corresponding residuals in the original models.

Step 3. Each model which contains a null residual for the template pattern A identifies one occurrence of the pattern A in the stimulus S. Each distinct model represents a unique occurrence of A in S. The correspondent in S for each parameter in A is specified by the appropriate element in the merged set of parameter correspondences.

Although it is apparent that this procedure is effective, two additional points should be made. First, every possible occurrence of the template A in the stimulus S is detected and represented by a corresponding model. Second, this algorithm is in some sense optimal for the task of identifying every occurrence of A in S. This conclusion is a direct result of the fact that only those models which may lead to a solution are ever manipulated, and manipulations cease as soon as sufficient evidence is obtained that any further operations will be futile. Further, the procedure rapidly reduces the set of mergeable models by reducing residuals whenever a merge occurs and, therefore, is guaranteed to halt.

The second algorithm is called the successive refinement procedure (Barrow, et al., 1972). It solves the binding and matching problems by successively identifying fewer and fewer possible S parameter correspondents for each parameter in A and then enumeratively checking each possible binding function (assigning one S parameter to every A parameter) to see if it entails a complete match between the relations of A and S. The procedure by which possible S correspondents for each A parameter are hypothesized is to compute features of each A and S parameter and to consider any parameter q in S which exhibits all the features of a parameter r in A as a possible correspondent of r. The possibilities are successively refined (reduced) by computing more and more complex features (such as whether r occurs in a relation {p:bigger, 1:r, 2:w}, where w is a parameter whose features include f_1 and f_2).

If successive refinements identify an A parameter with no possible correspondents, recognition terminates in failure. In any other case, it may continue until such time as a complete binding function betwen the parameters of A and S is generated consistent with the current refinement and a test for matching relations is performed.

STRUCTURED PATTERN RECOGNITION: PROBLEMS AND POSSIBILITIES

Because the all-or-none matching problem is equivalent to the graph monomorphism problem (Barrow, et al., 1972) and both are members of the class of complete problems (problems solvable in polynomial time only, apparently, by non-deterministic procedures), it is possible that no deterministic algorithm for structured event recognition exists which does not require an amount of time which grows exponentially with the number of parameters and relations in the template and stimulus (Karp, 1972). For example, as the number of stimulus relations with predicate on increases in the preceding example, the number of Step 1 models increases in proportion to the product of the number of these with the number of corresponding template relations. Worse yet, in practical problems the number of pattern templates is quite large (e.g., a 1500 word vocabulary in the Carnegie-Mellon University speech understanding system, most of which require numerous alternative pronunciation templates) and each template must be separately matched to the stimulus.

Thus, there are four distinct avenues of approach for future work: (1) algorithms like the interference matching and successive refinement procedures may be heuristically improved; (2) the computation which they perform may be organized and performed in parallel or in some other efficient way; (3) special algorithms may be developed for restricted classes of pattern structures; or (4) less powerful (general) pattern representations may be used. Each of these possibilities is considered in turn.

(1) Several heurisitcs for fast pattern matching (or graph monomorphism) have already

been tried (Barrow, et al., 1972; Hayes-Roth, 1973). These usually involve reducing the search space of possible binding functions or stimulus-template relation correspondences by computing additional features of the possible correspondents. Both of the procedures presented above employ such heuristics. Additional research may yield more practical procedures than those currently available. For example, the interference matching procedure can be exogenously constrained to consider at most M models and to introduce additional relations into existing models in such a way as to minimize the potential combinatorial explosion of alternative models (e.g., the next predicate type from the template for which stimulus correspondents are enumerated by generating Step 1 models is chosen to be the one with fewest possible stimulus correspondents). Some of the desirable properties of such space limited interference matching (SLIM) procedures are discussed elsewhere (Hayes-Roth, 1974b, 1974c).

(2) Alternative organizations of the computing required for pattern matching may be sought which can reduce the computing time or steps required. For example, for real-time control, very fast recognition is desired, and it is interesting to inquire whether any algorithm could solve the recognition problem with a template of N relations in k, k*N, ..., or $k*n^{f(N)}$ steps (for some k or f(N)) and how this procedure would be affected by increasing numbers of templates or stimulus components.

For example, one possible recognition system for a set of M templates, T_1, T_2, ..., T_M which can be economically organized to allow partial results of matching the stimulus to one template to be retained and used when other similar (partially matching) templates are subsequently evaluated is as follows. Let each template T_i be represented as a set of explicit relations $R_{i,1}$, ..., R_{i,n_i} (i.e. all objects of $R_{i,j}$ are parameters). Further, let each n-th order stimulus relation wih the same predicate Q be considered as a separate instance of a first level template $Q(x_1, ..., x_n)$. Thus, the k instances of the template $Q(x_1, ..., x_n)$ may be identifed as $Q[1] = (a_{1,1}, a_{1,2}, ..., a_{1,n})$, ..., $Q[k] = (a_{k,1}, ..., a_{k,n})$, meaning that $a_{i,j}$ is the parameter occurring as the j-th predicate object in the i-th distinct stimulus relation of predicate type Q.

The recognition of each template T_i is effected by forming a network whose nodes correspond to conjunctions of relations (a level m node Q in the network for T_i corresponds to the conjunction $Q(x_1, ..., x_n) = R_{i,1}$ and $R_{i,2}$ and ... and $R_{i,m}$). Each node Q is associated with a memory M(Q) whose k elements are M(Q)[1], ..., M(Q)[k] such that $M(Q)[j] = (a_{j,1}, ..., a_{j,n})$ is the list of n argument values corresonding to the j-th instance in the stimulus of the pattern represented by $Q(x_1, ..., x_n)$. The arcs in this network connect a level m node Q to a level m-1 node A(Q) and a level one node B(Q) whose conjunctive product is equivalent to Q. Thus a template T_i with n_i relations is represented by a binary tree of n_i levels. Finally, recognition of template T_i is effected by iteratively computing all instances of each level m node Q from

the instances of A(Q) and B(Q), for m = 2, 3, ..., n_i. Notice that several templates T_{i_1}, T_{i_2}, ..., T_{i_g} might share a common subpattern (a subset of equivalent relations), and this can be reflected by their sharing a common subtree with root Q in the recognition net. As a result, a minimal amount of computing can be done and all stimulus instances of the common subpattern Q stored in M(Q).

These notions can be formalized as follows. The i-th element of any list L is denoted L[i], so the j-th argument value of the i-th instance of the template represented by Q is M(Q)[i][j]. the length of L is denoted length(L). Each node Q is actually a list (A(Q), B(Q), M(Q), N(Q), S(Q), D(Q), E(Q)), as follows:

A(Q) and B(Q) are the two nodes conjoined at Q;

M(Q) is the memory stack of all argument lists (instances) satisfying Q;

N(Q) is the number of argument values extracted from A(Q) and B(Q) instances and placed into the M(Q) stack as an instance of Q;

S(Q) is a list of two-tuples (i,j) which mean that any instance of Q must derive from a pair of argument lists, one from each of M(A(Q)) and M(B(Q)) with the requirement that the i-th argument from A(Q) must be the same ("S") as the j-th argument from B(Q). As an example, if Q2 represents ({p:on, node:a, line:L1} and {p:on, node:b, line:L1}), then if each argument list from Q1 (the node for all instances of {p:on, node:x, line:y}) is a list of the form (node, line), we will require that L1 = L1 by putting (2, 2) in S(Q2);

D(Q) similarly is a list of index pairs (i',j') requiring that the i'-th argument from A(Q) be different ("D") than the j'-th from B(Q). Continuing with the preceding example, since a is not equal to b, (1, 1) is in D(Q2);

E(Q) is a list of length N(Q) of pairs (A!OR!B, index) which say where the argument values for M(Q)are to be derived from. If E(Q)[k] = (A(Q), m), the k-th argument value for M(Q) is extracted ("E") from whatever argument was in the m-th location of the argument list used from A(Q). Similarly, if E(Q)[k] = (B(Q), m), it is taken from B(Q).

Given this network representation, a highly parallel algorithm which can be used for recognition is as follows. It is written in a slightly extended form (allowing the simple type of list indexing used above) of LEAP (Feldman & Rovnar, 1969; Feldman, 1972) which is itself an extended ALGOL compiler. It should be noticed that all blocks of code at the same level between the identifiers **parbegin** and **parend** ("parallel" **begin** and **end**) and all instantiations of

the blocks within the iterative **parfor** ("parallel" for) and **parforeach** ("parallel" foreach) can be executed in parallel. Thus, the entire procedure for computing all instances of Q from instances of A(Q) and B(Q) could conceivably be computed in three time steps and a template with N relations recognized in 3*N steps.

```
procedure eval (list Q);
begin "evalQ"
comment Let Q=(A(Q),B(Q),M(Q),N(Q),S(Q),D(Q),E(Q))
    and let each of these component names be a macro which compiles into
    Q[k] for the appropriate k (e.g., A(Q)=Q[1]). Similarly, let
    s!item=(i,j) and d!item=(i',j') and let i,j,i',j' be macros
    similarly expanded;
integer array test!count[1:length(M(A(Q))), 1:length(M(B(Q)))];
integer m, n, N(Q), i, j, i', j';
list s!item, d!item, A(Q), B(Q), M(Q), E(Q);
comment test!count[m,n]=k implies that the m-th instance of
    A(Q) and the n-th instance of B(Q) satisfy k of the total number
    of same/different tests on argument values specified by S(Q) and
    D(Q). If k=length(S(Q))+length(D(Q)), these two instances
    can be combined and the appropriate argument values (specified by
    E(Q)) extracted and placed in a list in M(Q) as an instance
    of Q;

begin "step 1"
    comment initialize test!count to zero;
    parfor m←1 step 1 to length(M(A(Q))) do
    parfor n←1 step 1 to length(M(B(Q))) do
        test!count[m,n]←0;
end "step 1";

begin "step 2"

    parbegin "step 2a"
        comment Evaluate each requirement in S(Q) in parallel
            over all pairs of instances of A(Q) and B(Q) in parallel;
        parforeach s!item such that s!item in S(Q) do
        parfor m←1 step 1 to length(M(A(Q))) do
        parfor n←1 step 1 to length(M(B(Q))) do
            test!count[m,n]←test!count[m,n] +
                (if M(A(Q))[m,i] = M(B(Q))[n,j] then 1 else 0);
```

```
        comment s!item is iteratively bound to each element
            in s(Q), a list of two elements, which simultaneously binds
            i and j;
    parend "step 2a";
        .
        .
    parbegin "step 2b"
        comment As in "step 2a" for requirements in D(Q);
        parforeach d!item such that d!item in D(Q) do
        parfor m←1 step 1 to length(M(A(Q))) do
        parfor n←1 step 1 to length(M(B(Q))) do
            test!count[m,n]←test!count[m,n] +
                (if M(A(Q))[m,i'] = M(B(Q))[n,j'] then 0 else 1);
        comment d!item is iteratively bound to each element
            in D(Q), a list of two elements, which simultaneously binds
            i' and j';
    parend "step 2b";
end "step 2";

begin "step 3"
    comment Find all pairs of instances of A(Q) and B(Q)
        which have satisfied Q;
    parfor m←1 step 1 to length(M(A(Q))) do
    parfor n←1 step 1 to length(M(B(Q))) do
        if test!count[m,n]=length(S(Q))+length(D(Q)) then
            put extract(M(A(Q))[m], M(B(Q))[n]) in M(Q)
                before 1;
    comment extract(alist,blist) constructs an instance of Q
        (a list of N(Q) elements) from alist (an instance of A(Q))
        and blist (an instance of B(Q)) according to E(Q) and
        put x in L before 1 places the element x
        in the first position (at the head) of list L;
end "step 3";
comment Now M(Q) contains all instances of the pattern which
    Q(x_1, ..., x_n) represents;
end "evalQ";
```

In sum, the proposed organization and algorithm have two major advantages: (1) they exploit redundancy during the recognition of multiple templates sharing common subtemplates

by only computing instances of these common subpatterns a single time and storing them in corresponding M(Q) lists throughout the network; (2) they allow for extensive amounts of parallelism during recognition processing. Whether or not these prove valuable depends critically on the availability of appropriate parallel systems and the number of instances of each subpattern, since the possible number of instances of Q is length(M(A(Q)))*length(M(B(Q))).

(3) A very promising direction for research in structured pattern recognition is the discovery of special representational cases where simpler (less general) recognition algorithms will suffice. An example from our current speech understanding research will make this point clear. One of the major problems in speech understanding is word recognition. This is difficult for many reasons, two of which are that each word has numerous possible pronunciations (phonetic spellings) and in any given instance a phoneme may be missing (undetected) or an extra phoneme may be inserted (incorrectly differentiated from its temporally adjacent neighbors). Furthermore, at any point in time, t, the computational subsystem that is used to identify phonemes present in the speech input may hypothesize numerous alternatives $h(t) = (p_{t,1}, p_{t,2}, \ldots, p_{t,n_t})$. One particularly efficient strategy for performing recognition in this environment is now proposed.

Frequently, the numerous phonetic spellings for each word can be efficiently represented by a finite state transition network without cycles, because the multiple spellings result from the independent combinations of phonetic alternatives at various temporal positions in the word. For example, the following eight pronunciations (spellings) of "America" can be efficiently coded by the list equivalent T_1 in (4) corresponding to the network of Figure 3: (ax,m,eh,r,ax,k,ax), (ax,m,eh,r,ax,k,ah), (ax,m,eh,r,ih,k,ax), (ax,m,eh,r,ih,k,ah), (ah,m,eh,r,ax,k,ax), (ah,m,eh,r,ax,k,ah), (ah,m,eh,r,ih,k,ax), (ah,m,eh,r,ah,k,ih),

$$T_1 = ((ax,ah),m,eh,r,(ax,ih),k,(ax,ah)) \tag{4}$$

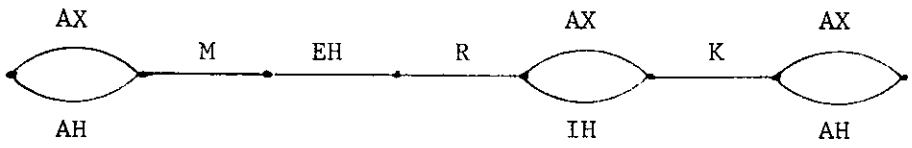


Figure 3. A transition network for the 8 pronunciations of "America."

Given numerous hypothesized phonemes h(t) at each time t=1,2,...,N and numerous word

templates T_1, T_2, ..., T_M as in (4), an efficient recognition procedure exists which can examine all templates and all alternative pronunciations in parallel using only bit string logical products (and) and sums (or). For each phoneme p, several inverted bit strings $s(p,j) = b_1b_2...b_M$ are computed and permanently stored in a dictionary, as follows: b_i=1 if p occurs as an alternative in the j-th arc (or in the j-th sublist) of T_i, where j=1,2,...,L and L is the length of the longest word template (list); b_i=1 if the length of T_i is less then j; in any other case, b_i=0. If any selection of phonemes p_k, p_{k+1}, ..., p_{k+L-1} from L temporally adjacent hypothesized sets h(k), h(k+1), ..., h(k+L-1) matches a template T_i, then the i-th bit, b_i, of the bitwise logical product B = $s(p_k,1)$ and $s(p_{k+1},2)$ and ... and $s(p_{k+L-1},L)$ will be 1. Moreover, if the logical sum (bitwise or) of the inverted strings $s(p_{k+j},n)$ for j=1,2,...,length(h(k)) are called C(k,n) (for each n=1, 2, ..., L), the i-th bit d_i of D(k) = C(k,1) and C(k,2) and ... and C(k+L-1,L) is 1 if any such selection of phonemes entails recognition of T_i starting at t=k.

The preceding extremely efficient parallel search of all word templates is adequate only when there is a perfect match between the sequences of hypothesized phonemes and a word template. However, there is a simple elaboration which makes it suitable for the cases of at most r insertions or deletions. Instead of the preceding sorts of computations, we use inverted bit strings y(p,j) of length M*L, where L bits are assigned successively to each word template T_1, T_2, ..., T_M, and the j-th bit of the L bits for word template T_i is 1 if phoneme p occurs in the j-th sublist of T_i or if j exceeds the length of T_i. Let the L bits corresponding to T_i (those starting at position (i-1)*L+1) in such a string B be denoted Z(B,i). The preceding algorithm could be recast in terms of such strings y as follows. Let C(k,n) be the bitwise logical sum of all strings $y(p_{k,j},n)$ as before, and let D(k) = C(k,1) or C(k+1,2) or ... or C(k+L-1,L). Then $Z(D(k),i) = 111....1 = (1)^L$ only if some selection of temporally adjacent phonemes matches T_i starting at t=k. In this case, the number of 1 bits in Z(D(k),i), count(Z(D(k),i)), is equal to L.

Now suppose a single deletion occurred in word T_i at time t=k+m (m in {1,2,...,L-1}). Then the first m bits of Z(D(k),i) will be 1 but the rest will probably be 0. On the other hand, the last L-m-1 bits of Z(D(k-1),i) will all be 1. Similarly, if a single insertion occurred at time k+m, the last L-m bits of Z(D(k),i) will probably be 0 but the last L-m-1 bits of Z(D(k+1),i) will be 1. Thus, an efficient way to recognize T_i at time k allowing at most 1 insertion or deletion or both is to determine whether count(Z(D(k-1) or D(k) or D(k+1)),i) > L-2. In general, to recognize T_i with a maximum of r insertions or deletions the recognition algorithm is "accept T_i if count(Z(convolve(k,r),i)) > L-r-1," where convolve(k,r) = D(k-r) or D(k-r+1) or ... or D(k+r).

In sum, in the special case of word recognition, a very powerful yet simple technique exists for evaluating many templates (with errors) in parallel and obviates a more general, combinatorial procedure. Such techniques may possibly occur in other limited recognition problem domains and are clearly worth seeking.

(4) Finally, less general pattern representations can be used. For example, the preceding framework for word recognition might be considered a feature representation since recognition could be performed on bit strings wherein each bit corresponds to a particular attribute of the stimulus. The need to consider abandoning general structural representation and matching will necessarily arise whenever, in specific contexts, a combinatorial explosion in all-or-none matching actually occurs. In general however it is not possible to determine this on a priori considerations and, as a result, nothing further will be said here.

CONCLUSIONS

Real-world pattern recognition problems appear to require relational (graph) representations. General all-or-none matching procedures for such problems were discussed. The possibility exists that these procedures will require computing time which increases exponentially with pattern complexity. Thus, special approaches aimed at preventing this predicament are necessarily sought. For example, the availability of parallel processors and extensive memory could make the proposed recognition network an efficient, general procedure. On the other hand, an analysis of the problem domain (e.g., word recognition) may yield specially tailored, extremely efficient procedures for recognition. In general, it is apparent that continued research into structured pattern recognition is warranted and promises to yield practical techniques which may, incidentally, provide valuable insights into the interesting relationship between representational and computational complexity.

REFERENCES

Barrow, H.G., Ambler, A.P., & Burstall, R.M. Some techniques for recognising structures in pictures. In S. Watanabe (Ed.), Frontiers of pattern recognition. New York: Academic Press, 1972.

Becker, J.D. A model for the encoding of experiential information. In R. C. Shank & K. M. Colby, Computer models of thought and language. San Francisco: Freeman, 1973.

Evans, T.G. Descriptive pattern-analysis techniques. In A. Grasselli (Ed.), Automatic interpretation and classification of images. New York: Academic Press, 1969.

Feldman, J.A., & Rovnar, F. An Algol-based associative language. Communications of the ACM, 1969, 12, 439-449.

Feldman, J.A., Low, J.R., Swinehart, D.C., & Taylor, R.H. Recent developmentments in Sail. Proceedings AFIPS Fall Joint Conference, 1972, 1193-1201.

Hayes-Roth, F. A structural approach to pattern learning and the acquisition of classificatory power. Proceedings of the First International Joint Conference on Pattern Recognition, 1973.

Hayes-Roth, F. An optimal network representation and other mechanisms for the recognition of structured events. Proceedings of the Second International Joint Conference on Pattern Recognition, 1974a.

Hayes-Roth, F. Schematic classification problems and their solution. Pattern Recognition, 1974b, 6, 105-113.

Hayes-Roth, F. Fundamental mechanisms of intelligent behavior: the representation, organization, acquisition, and use of structured knowledge in perception and cognition. Unpublished doctoral dissertation. Ann Arbor: The University of Michigan, 1974c.

Hayes-Roth, F. Uniform relational representations and an algorithm for grammatical inference. Pittsburgh: Computer Science Department, Carnegie-Mellon University, 1974d.

Hewitt, C. Description and theoretical analysis (using schemata) of PLANNER: a language for proving theorems and manipulating models in a robot. Cambridge: MIT Project MAC, 1972.

Hopcroft, J. E., & Ullman, D. D. Formal languages and their relation to automata. Reading, Massachusetts: Addison-Wesley 1969.

Joshi, A. K. Remarks on some aspects of language structure and their relevance to pattern analysis. Pattern Recognition, 1973, 5, 365-382.

Karp, R.M. Reducibility among combinatorial problems. In A.E. Miller & J.W. Thatcher (Eds.), Complexity of computer computations. New York: Plenum Press, 1972.

Lindsay, P. H., & Norman, D. A. Human information processing. An introduction to psychology. New York: Academic Press, 1972.

Michalski, R. S. AQVAL/1--computer implementation of a variable-valued logic system VL_1 and examples of its application to pattern recognition. Proceedings of the First International Joint Conference on Pattern Recognition, 1973.

Minsky, M.L. Computation: finite and infinite state machines. Englewood Cliffs, N.J.: Prentice-Hall, 1967.

Narasimhan, R. On the description, generation, and recognition of classes of pictures. In A. Grasselli (Ed.), Automatic interpretation and classification of images. New York: Academic Press, 1969.

Neisser, U. Cognitive psychology. New York: Appleton, 1967.

Newell, A. A theoretical exploration of mechanisms for encoding the Stimulus. In A. W. Melton & E. Martin (Eds.), Coding processes in human memory. Washington: Winston, 1972.

Pflatz, J.L., & Rosenfeld, A. Web grammars. Proceedings of the First International Joint Conference on Artificial Intelligence, 1969.

Rulifson, J. F., Derksen, J. A., & Waldinger, R. J. QA4: a procedural calculus for intuitive reasoning. Menlo Park: Stanford Research Institute, 1972.

Selfridge, O. G. Pandemonium: a paradigm for learning. In D. V. Blake & A. M. Uttley (Eds.), Proceedings of the Symposium on the Mechanisation of Thought Processes. London: H. M. Stationery Office, 1959.

Shaw, A.C. Picture graphs, grammars, and parsing. In S. Watanabe (Ed.), Frontiers of pattern recognition. New York: Academic Press, 1972.

Thomason, M. G. Finite fuzzy automata, regular fuzzy automata, and pattern recognition. Pattern Recognition, 1973, 5, 383-390.

Uhr, L. Flexible linguistic pattern recognition. Pattern Recognition, 1971, 3, 363-370.

Watanabe, S. Subspace method in pattern recognition. Proceedings of the First International Joint Conference on Pattern Recognition, 1973.

Winograd, T. A program for understanding natural language. Cognitive Psychology, 1972, 3, 1-191.

Winston, P.H. Learning structural descriptions from examples. AI-TR-76. Cambridge: MIT Artificial Intelligence Laboratory, 1970.

Woods, W. A. Transition network grammars for natural language analysis. Communications of the ACM, 1970, 13, 591-606.

AN AUTOMATICALLY COMPILABLE RECOGNITION NETWORK FOR STRUCTURED PATTERNS[1]

Frederick Hayes-Roth and David J. Mostow
Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

ABSTRACT

A new method for efficient recognition of general relational structures is described and compared with existing methods. Patterns to be recognized are defined by templates consisting of a set of predicate calculus relations. Productions are representable by associating actions with templates. A network for recognizing occurrences of any of the template patterns in data may be automatically compiled. The compiled network is economical in the sense that conjunctive products (subsets) of relations common to several templates are represented in and computed by the network only once. The recognition network operates in a bottom-up fashion, in which all possibilities for pattern matches are evaluated simultaneously. The distribution of the recognition process throughout the network means that it can readily be decomposed into parallel processes for use on a multi-processor machine. The method is expected to be especially useful in errorful domains (e.g., vision, speech) where parallel treatment of alternative hypotheses is desired.

INTRODUCTION

The work described in this paper was motivated by certain problems involved in the task of recognizing structured patterns, especially the problem of parsing continous spoken speech. From the point of view of the language parser, an essential quality of speech is its errorful nature. Ambiguities in acoustic segmentation, phonetic labelling, word hypothesization, and semantic interpretation necessitate understanding systems which can deal efficiently with multiple alternative hypotheses about each portion of the input (Newell, et al.,

[1]This work was supported in part by Advanced Research Projects Agency Contract F44620-73-C-0074 to Carnegie-Mellon University.

1973). The usual methods of dealing with such multiple hypotheses typically entail an expensive search through a combinatorial space, since they consider only one hypothesis for each portion of input at a time, and then exploit contextual relationships to eliminate certain combinations of adjacent hypotheses as impossible. The data structure and associated recognition procedure described in this paper can be thought of as effectively reversing this process by first exploiting context -- thereby eliminating all but a few combinations from consideration -- and then testing contextually related hypotheses for adjacency. Since the contextual information is statically embedded in the data structure itself, comparatively little work needs to be done at recognition time. This work requires only the computation of a few, simple operations rather than a complex search. Moreover, the method provides an efficient way to handle the spurious insertions, deletions, and repetitions characteristic of speech.

TEMPLATE GRAMMARS

In this section, we define template grammars for recognizing relational structures. We then define a template normal form (TNF) for template grammars and describe an algorithm for translating a given template grammar into an equivalent TNF grammar which is economical in that it maximally exploits repeated subtemplates in the original grammar. The construction of an automatically compilable recognition network (ACORN) from a TNF grammar is straightforward and is described in the next section.[1] The definitions we use are tailored to natural language understanding, but are immediately generalizable to other applications (e.g., vision).

A relation $r(t_1, t_2; x_1, ..., x_n)$ is an n-ary predicate corresponding to some element or pattern in the language. For example, the relation $\mathbf{tell}(t_1, t_2)$ holds if the word "tell" occurs in the input utterance beginning at time t_1 and ending at time t_2. In general, t_1 and t_2 are temporal arguments specifying the time interval containing a recognized occurrence of the relation, and $x_1, ..., x_n$ are selected arguments and features of the occurrence. A relation is called primitive if it corresponds to a primitive element (terminal symbol) of the language, non-primitive if it corresponds to a pattern of elements (non-terminal symbol), and top-level if it corresponds to a complete pattern (sentential form).

A template T is a Boolean combination of relations r_i, i=1, ..., |T|, restricted as follows. It

---

[1] In actuality, our compiler will compile the input grammar directly into a recognition network, without the intermediate step of translation into TNF. For the purposes of this discussion, however, we have divided ACORN compilation into the two phases of translation into TNF and construction of the recognition network from the TNF grammar.

must be either a disjunction

$$r_1(t_1, t_2; x_1, ..., x_n) \text{ or } r_2(t_1, t_2; x_1, ..., x_n) \text{ or } ... \text{ or } r_d(t_1, t_2; x_1, ..., x_n), |T| = d \geq 1,$$

or a conjunction

$$r_1(t_{1_1}, t_{2_1}; x_{1_1}, ..., x_{n_1}) \text{ and } ... \text{ and } r_p(t_{1_p}, t_{2_p}; x_{1_p}, ..., x_{n_p}) \text{ and}$$
$$\text{not } r_{p+1}(t_{1_{p+1}}, t_{2_{p+1}}; x_{1_{p+1}}, ..., x_{n_{p+1}}) \text{ and } ... \text{ and not } r_q(t_{1_q}, t_{2_q}; x_{1_q}, ..., x_{n_q}) \; (q \geq p \geq 1).$$

In the first case (disjunction), the symbolic arguments $(t_1, t_2; x_1, ..., x_n)$ are the same for each r_i, $i = 1, ..., d$. In the second case, a weaker condition must be satisfied: the relations must have enough symbolic arguments in common for the template to be connected, that is, for any partition of the $p+q$ relations $r_1, ..., r_{p+q}$ into two non-empty sets A and B, there must exist relations $r_a(t_{1_a}, t_{2_a}; x_{1_a}, ..., x_{n_a}) \in A$ and $r_b(t_{1_b}, t_{2_b}; x_{1_b}, ..., x_{n_b}) \in B$ such that $\{t_{1_a}, t_{2_a}\} \cap \{t_{1_b}, t_{2_b}\} \neq \emptyset$.

A template grammar is a set of rules of the form [<template> => <relation>; <action>]. The action optionally associated with each rule specifies what should be done in the event that an instance of the template is recognized in the input and the rule is invoked. Thus a template grammar is actually a production system of the sort described by Newell (1973).

Table 1. Sample Grammar (G_{AP})

1. [Ford(t_1, t_2) => TOPIC(t_1, t_2; expr); expr←"FORD"]
2. [Rockefeller(t_1, t_2) or Rocky(t_1, t_2) => TOPIC(t_1, t_2; expr); expr←"ROCKEFELLER"]
3. [Kissinger(t_1, t_2) => TOPIC(t_1, t_2; expr); expr←"KISSINGER"]
4. [or(t_1, t_2) and TOPIC(t_2, t_3; expr) => TOPIC*(t_1, t_3; expr);]
5. [TOPIC(t_1, t_2; $expr_1$) and TOPIC*(t_2, t_3; $expr_2$) =>
   TOPIC*(t_1, t_3; expr); expr ← $expr_1 \cup expr_2$]
6. [UTTERANCE(t_1, t_4) and about(t_2, t_3) and TOPIC(t_3, t_4; expr) =>
   TOPIC*(t_3, t_4; expr);]
7. [UTTERANCE(t_1, t_6) and tell(t_1, t_2) and me(t_2, t_3) and
   nothing(t_3, t_4) and about(t_4, t_5) and TOPIC*(t_5, t_6; expr) =>
   REJECT(t_1, t_6; expr); SUPPRESS(expr)]
8. [UTTERANCE(t_1, t_6) and tell(t_1, t_2) and me(t_2, t_3) and
   not nothing(t_3, t_4) and about(t_4, t_5) and TOPIC*(t_5, t_6; expr) =>
   REQUEST(t_1, t_6; expr); RETRIEVE(expr)]

As an example, consider the sample template grammar G_{AP} (Table 1) which is part of a much larger grammar for analyzing spoken queries to a wire-service news retrieval system (Frost, 1974). G_{AP}'s top-level relations are **REQUEST** and **REJECT**. A sample instance of **REQUEST** is the utterance "Tell me all about Rocky." An instance of **REJECT** is the utterance "Tell me nothing about Ford, Rockefeller, or Kissinger." The primitive relation **UTTERANCE**(t_1, t_2), used in rules 6, 7, and 8, simply signifies that the entire utterance spans the time interval $[t_1, t_2]$; this makes the beginning and ending times of the utterance accessible as arguments to other relations, without violating the framework of the template grammar. Rule 2 illustrates the use of features and actions. The feature, expr, of **TOPIC** is the semantic expression eventually passed to the actual news retrieval routine. The action of Rule 2 gives expr the value "ROCKEFELLER." Rule 5 is an example of recursion. It handles phrases of the form "topic$_1$, topic$_2$, ..., topic$_{n-1}$, or topic$_n$." The action of Rule 5 forms a compound semantic expression from the expressions associated with its individual constituents. Thus the instance "Ford, Rockefeller, or Kissinger" of the relation **TOPIC*** has expr = {"FORD", "ROCKEFELLER", "KISSINGER"}. Rule 6 shows how context sensitivity can be embedded in a template grammar. It states that any instance of **TOPIC** which occurs at the end of an utterance, and whose left context is **ABOUT**, constitutes an instance of **TOPIC***. Rule 8 illustrates the use of negation. It states that any utterance of the form "Tell me ... about X" is a request for information about X unless the gap "..." contains the word "nothing." Thus "Tell me about Ford," "Tell me all about Ford," and "Tell me everything you know about Ford," are all instances of **REQUEST**. This illustrates the capacity of a template grammar to ignore redundant portions of the input.

A template grammar is in <u>template normal form</u> (TNF) if the following conditions are satisfied:

(1) The template of each rule has one of the following types:

<relation$_1$> <u>or</u> <relation$_2$> <u>or</u> ... <u>or</u> <relation$_d$>, d≥1	(disjunctive type)
<relation$_1$> <u>and</u> <relation$_2$>	(conjunctive type)
<relation$_1$> <u>and not</u> <relation$_2$>	(negative type)

The relations in a disjunctive template have the same symbolic arguments; the relations in a conjunctive or negative template are connected.

(2) Every non-primitive relation appears on the right side of exactly one rule. Hence we can define the <u>type</u> of a relation to be the type of its unique defining template; a primitive relation is simply said to be of primitive type.

It is clear that any template grammar G can be translated into an equivalent grammar G* in TNF by means of adding new relations and rules. The task of the automatic translator is to do this in such a way as to minimize the number of new relations added.

Let $G = \{[T_i \Rightarrow r_i; A_i] : i = 1, ..., |G|\}$ be a template grammar. Define U(G) to be the set of rules in G which fail to satisfy condition (1) of TNF, i.e., $U(G) = \{T \in G : T$ is a conjunction of relations and $|T| > 2\}$. Define $N(G) = \sum_{T \in U(G)} (|T| - 2)$. Then N(G) = 0 iff U(G) = ∅ iff G satisfies condition (1). Moreover, N(G) is a measure of how far off G is from satisfying (1). We are now ready to describe the algorithm for translating the input grammar G into an equivalent TNF grammar G*.

Step 0. Let $G_0 = G$ and set $j \leftarrow 0$.

Step 1. If $N(G_j) = 0$, go to Step 2. Otherwise, find a pair of relations $\langle r_1, r_2 \rangle$ such that $r_1(t_1, t_2; x_1, ..., x_n)$ and (not) $r_2(t_3, t_4; y_1, ..., y_m)$ is a connected subtemplate of at least one template $T \in U(G_j)$, i.e., one for which $\{t_1, t_2\} \cap \{t_3, t_4\} \neq \emptyset$. Define the new relation r_{new} to be this subtemplate. Replace the subtemplate by r_{new} wherever possible in G_j, and add a new rule

$$[r_1(t_1, t_2; x_1, ..., x_n) \text{ and (not) } r_2(t_3, t_4; y_1, ..., y_m) \Rightarrow r_{new}(t_5, t_6; z_1, ..., z_k);].$$

(A later example will illustrate the criteria used in selecting the arguments $t_5, t_6, z_1, ..., z_k$ of the new relation from the arguments $t_1, t_2, t_3, t_4, x_1, ..., x_n, y_1, ..., y_m$ of its constituent relations r_1 and r_2.) Call the resulting equivalent grammar G_{j+1}. Let $j \leftarrow j+1$ and go to Step 1.

Note that $N(G_{j+1}) < N(G_j)$, since each iteration of Step 1 shortens at least one template in $U(G_j)$. Hence Step 1 must terminate after $J \leq N(G)$ iterations, giving a G_J such that $N(G_J) = 0$. The heuristic we use in an effort to obtain a minimal G_J (i.e., to minimize J) is simple: each iteration through Step 1, choose the subtemplate which occurs the greatest number of times in $U(G_j)$, i.e., the pair of relations which will give the maximum reduction in $N(G_j)$.

It remains to satisfy condition (2). This is done by

Step 2. For every relation r which appears in the grammar G_J on the right side of $d \geq 2$ rules $[T_i \Rightarrow r; A_i]$, $i = 1, ..., d$, rewrite each rule $[T_i \Rightarrow r; A_i]$ as $[T_i \Rightarrow r_i; A_i]$, where r_i is a unique new relation, and add the new rule $[r_1 \text{ or } r_2 \text{ or } ... \text{ or } r_d \Rightarrow r;]$. Call the resulting grammar G*. By its construction, G* is equivalent to G_J, and satisfies conditions (1) and (2). Therefore G* is the desired TNF grammar equivalent to G.

As an example, consider the application of this translation algorithm to the sample grammar G_{AP}. The templates T_1, T_2, and T_3 of rules 1, 2, and 3 are disjunctions, and $|T_4| = |T_5| = 2$, so $U(G_0) = U(G_{AP}) = \{T_6, T_7, T_8\}$, and $N(G_0) = |T_6|-2 + |T_7|-2 + |T_8|-2 = 1 + 4 + 4 = 9$. The relation pair which occurs the greatest number of times in $U(G_0)$ is **<UTTERANCE, about>**, which occurs in T_6, T_7, and T_8. However, all three occurrences of this pair fail to satisfy the connectivity condition, since $\{t_1, t_4\} \cap \{t_2, t_3\} = \emptyset$ in T_6 and $\{t_1, t_6\} \cap \{t_4, t_5\} = \emptyset$ in T_7 and T_8. Because several relation pairs occur twice, arbitrarily choose the pair **<tell, me>**, which occurs in T_7 and T_8, for replacement. Accordingly, the following rule is added to the grammar:

9. [**tell**(t_1, t_2) and **me**(t_2, t_3) => **TELL•ME**(t_1, t_3);].

Rule 9 states that if the word "tell" occurs in the time interval $[t_1, t_2]$ in the spoken input, and the word "me" occurs in the interval $[t_2, t_3]$, then the relation **TELL•ME** occurs in the concatenated interval $[t_1, t_3]$. We rewrite rules 7 and 8 as

7*. [**UTTERANCE**(t_1, t_5) and **TELL•ME**(t_1, t_2) and
nothing(t_2, t_3) and **about**(t_3, t_4) and **TOPIC***(t_4, t_5; expr) =>
REJECT(t_1, t_5); **SUPPRESS**(expr)]

8*. [**UTTERANCE**(t_1, t_5) and **TELL•ME**(t_1, t_2) and
not **nothing**(t_2, t_3) and **about**(t_3, t_4) and **TOPIC***(t_4, t_5; expr) =>
REQUEST(t_1, t_5); **RETRIEVE**(expr)].

Note that $N(G_1) = N(G_0) - 2 = 7$.

Now repeat Step 1. A choice must be made among several relation pairs which occur twice; arbitrarily choose the pair **<about, TOPIC*>**, add the new rule

10. [**about**(t_1, t_2) and **TOPIC***(t_2, t_3) => **ABOUT•TOPIC***(t_1, t_3);],

and rewrite rules 7* and 8* as

7**. [**UTTERANCE**(t_1, t_4) and **TELL•ME**(t_1, t_2) and
nothing(t_2, t_3) and **ABOUT•TOPIC***(t_3, t_4; expr) =>
REJECT(t_1, t_4); **SUPPRESS**(expr)]

8**. [**UTTERANCE**(t_1, t_4) and **TELL•ME**(t_1, t_2) and
not **nothing**(t_2, t_3) and **ABOUT•TOPIC***(t_3, t_4; expr) =>
REQUEST(t_1, t_4); **RETRIEVE**(expr)].

As a result, $N(G_2) = N(G_1) - 2 = 5$. Note that the minimization heuristic rules out the choice of the relation pair <**nothing, about**> for replacement; to do so would reduce $N(G_1)$ by only 1, since the relation **nothing** is negated in T_8 but not in T_7.

Again, Step 1 allows a choice among relation pairs occurring twice; arbitrarily choose the pair <**UTTERANCE,TELL•ME**>. At this point, care must be exercised in defining a new relation. In previous applications of Step 1, relations were simply concatenated. Here, however, relations overlap. When **UTTERANCE**(t_1, t_4) and **TELL•ME**(t_1, t_2) are conjoined into a new relation, t_1 as well as t_2 and t_4 must be retained, since t_1 appears elsewhere in rules 7^{**} and 8^{**} (as an argument of **REJECT** and **REQUEST**). Previously, such a problem did not arise, because the argument shared by the pair of relations appeared nowhere else in the rule, and consequently could safely be omitted from the new relation. In the case at hand, the translator adds a new rule

11. **TELL•ME**(t_1, t_2) and **UTTERANCE**(t_1, t_3) => **TELL•ME-UTTERANCE**(t_2, t_3; t_1);]

and rewrites rules 7^{**} and 8^{**} as

7^{***}. [**TELL•ME-UTTERANCE**(t_2, t_4; t_1) and **nothing**(t_2, t_3) and
ABOUT•TOPIC*(t_3, t_4; expr) =>
REJECT(t_1, t_4); **SUPPRESS**(expr)]

8^{***}. [**TELL•ME-UTTERANCE**(t_2, t_4; t_1) and not **nothing**(t_2, t_3) and
ABOUT•TOPIC*(t_3, t_4; expr) =>
REQUEST(t_1, t_4); **RETRIEVE**(expr)],

with the result that $N(G_3) = N(G_2) - 2 = 3$. In our mnemonic convention for naming new relations, the "+" in "**TELL•ME-UTTERANCE**" denotes concatenation, and the "-" denotes overlapping.

There is now only one relation pair which occurs twice, namely <**TELL•ME-UTTERANCE, ABOUT•TOPIC***>. The translator adds a new rule

12. [**TELL•ME-UTTERANCE**(t_2, t_4; t_1) and **ABOUT•TOPIC***(t_3, t_4; expr) =>
TELL•ME-UTTERANCE-ABOUT•TOPIC*(t_2, t_3; t_1, t_4, expr)]

and rewrites rules 7^{***} and 8^{***} as

7^{****}. [**TELL•ME-UTTERANCE-ABOUT•TOPIC***(t_2, t_3; t_1, t_4, expr) and **nothing**(t_2, t_3) =>
REJECT(t_1, t_4); **SUPPRESS**(expr)]

8****. [TELL•ME-UTTERANCE-ABOUT•TOPIC*(t_2, t_3; t_1, t_4, expr) and not **nothing**(t_2, t_3) =>
REQUEST(t_1, t_4); RETRIEVE(expr)].

Since $|T_7****| = |T_8****| = 2$, we now have $U(G_4) = \{T_6\}$ and $N(G_4) = N(G_3) - 2 = 1$.

At this point, there are two connected pairs of relations, each occurring once in T_6. Arbitrarily, choose the pair <about, TOPIC>. In conjoining **about**(t_2, t_3) and TOPIC(t_3, t_4; expr), the translator discards t_2, since it appears nowhere else in rule 6, but must retain t_3, which appears as an argument of the relation TOPIC*. Accordingly, it adds the new rule

13. [**about**(t_1, t_2) and TOPIC(t_2, t_3; expr) => **(ABOUT)TOPIC**(t_2, t_3; expr);]

and rewrites rule 6 as

6*. [UTTERANCE(t_1, t_3) and **(ABOUT)TOPIC**(t_2, t_3; expr) => **TOPIC***(t_2, t_3; expr);].

In our naming convention, the parentheses in "(ABOUT)TOPIC" are a mnemonic for context sensitivity, indicating that an occurrence of TOPIC with **about** as its left-hand context constitutes an occurrence of (ABOUT)TOPIC. At this point, $|T_6*| = 2$, $U(G_5) = \emptyset$, and $N(G_5) = N(G_4)-1 = 0$, so G_5 satisfies condition (1) of TNF.

Now apply Step 2 to G_5. The only relations which occur on the right side of more than one rule are TOPIC, which occurs on the right side of rules 1, 2, and 3, and TOPIC*, which occurs on the right side of rules 4, 5, and 6*. The translator rewrites rules 1, 2, and 3 as

1*. [**Ford**(t_1, t_2) => **TOPIC/1**(t_1, t_2; expr); expr←"FORD"]
2*. [**Rockefeller**(t_1, t_2) or **Rocky**(t_1, t_2) => **TOPIC/2**(t_1, t_2; expr); expr←"ROCKEFELLER"]
3*. [**Kissinger**(t_1, t_2) => **TOPIC/3**(t_1, t_2; expr); expr←"KISSINGER"]

and adds the new rule

14. [**TOPIC/1**(t_1, t_2; expr) or **TOPIC/2**(t_1, t_2; expr) or **TOPIC/3**(t_1, t_2; expr) =>
TOPIC(t_1, t_2; expr);].

Similarly, rules 4, 5, and 6* are rewritten as

4*. [**or**(t_1, t_2) and TOPIC(t_2, t_3; expr) => **TOPIC*/4**(t_1, t_3; expr);]
5*. [TOPIC(t_1, t_2; $expr_1$) and TOPIC*(t_2, t_3; $expr_2$) =>
TOPIC*/5(t_1, t_3; expr); expr ← $expr_1 \cup expr_2$]
6**. [UTTERANCE(t_1, t_3) and **(ABOUT)TOPIC**(t_2, t_3; expr) => **TOPIC*/6**(t_2, t_3; expr);]

and the following new rule is added:

15. [TOPIC*/4(t_1, t_2; expr) or TOPIC*/5(t_1, t_2; expr) or TOPIC*/6(t_1, t_2; expr) =>
TOPIC(t_1, t_2; expr);].

The resulting grammar G_{AP}^*, shown in Table 2, is in TNF.

Table 2. Sample Grammar in TNF (G_{AP}^*)

1*. [Ford(t_1, t_2) => TOPIC/1(t_1, t_2; expr); expr←"FORD"]
2*. [Rockefeller(t_1, t_2) or Rocky(t_1, t_2) => TOPIC/2(t_1, t_2; expr); expr←"ROCKEFELLER"]
3*. [Kissinger(t_1, t_2) => TOPIC/3(t_1, t_2; expr); expr←"KISSINGER"]
4*. [or(t_1, t_2) and TOPIC(t_2, t_3; expr) => TOPIC*/4(t_1, t_3; expr);]
5*. [TOPIC(t_1, t_2; $expr_1$) and TOPIC*(t_2, t_3; $expr_2$) =>
TOPIC*/5(t_1, t_3; expr); expr ← $expr_1$ ∪ $expr_2$]
6**. [UTTERANCE(t_1, t_3) and (ABOUT)TOPIC(t_2, t_3; expr) => TOPIC*/6(t_2, t_3; expr);]
7****. [TELL+ME-UTTERANCE-ABOUT+TOPIC*(t_2, t_3; t_1, t_4, expr) and nothing(t_2, t_3) =>
REJECT(t_1, t_4); SUPPRESS(expr)]
8****. [TELL+ME-UTTERANCE-ABOUT+TOPIC*(t_2, t_3; t_1, t_4, expr) and not nothing(t_2, t_3) =>
REQUEST(t_1, t_4); RETRIEVE(expr)]
9. [tell(t_1, t_2) and me(t_2, t_3) => TELL+ME(t_1, t_3);]
10. [about(t_1, t_2) and TOPIC*(t_2, t_3) => ABOUT+TOPIC*(t_1, t_3);]
11. [TELL+ME(t_1, t_2) and UTTERANCE(t_1, t_3) => TELL+ME-UTTERANCE(t_2, t_3; t_1);]
12. [TELL+ME-UTTERANCE(t_2, t_4; t_1) and ABOUT+TOPIC*(t_3, t_4; expr) =>
TELL+ME-UTTERANCE-ABOUT+TOPIC*(t_2, t_3; t_1, t_4, expr);]
13. [about(t_1, t_2) and TOPIC(t_2, t_3; expr) => (ABOUT)TOPIC(t_2, t_3; expr);]
14. [TOPIC/1(t_1, t_2; expr) or TOPIC/2(t_1, t_2; expr) or TOPIC/3(t_1, t_2; expr) =>
TOPIC(t_1, t_2; expr);]
15. [TOPIC*/4(t_1, t_2; expr) or TOPIC*/5(t_1, t_2; expr) or TOPIC*/6(t_1, t_2; expr) =>
TOPIC(t_1, t_2; expr);]

THE RECOGNITION NETWORK[1]

Given a template grammar in TNF, a corresponding recognition network (ACORN) is constructed as follows. For each relation r appearing in the TNF grammar, there is a unique

[1]The recognition network used here is based on the structure described in Hayes-Roth (1974b).

node, node(r), in the network. (Hence minimizing the number of relations in the TNF grammar is equivalent to minimizing the number of nodes in the network.) For every rule [T => r; A], an arc is drawn from node(s_i) to node(r) for each relation s_i in the template T. Each node(s_i) is said to be a constituent of node(r), and node(r) a derivative of node(s_i). A node may have zero, one, or more derivatives. The recognition network for the sample grammar G_{AP}, constructed from the TNF grammar G_{AP}^*, is shown in Figure 1.

Node(r) contains various information: its type (i.e., the type of relation r); the action A in the rule [T => r; A], if any; and the correspondence between the arguments of relation r and the arguments of its constituent relations s_i. This correspondence consists of two parts, a set of tests and a generator. The tests represent any requirements for agreement between the arguments supplied by the constituents node(s_i). The generator is a list of the arguments which are to be supplied in turn to the derivatives of node(r). The arguments are encoded according to a canonical numbering scheme best described by an example. Consider node(TELL+ME). Its constituents are node(TELL), which supplies arguments t_1, t_2, and node(ME), which supplies arguments t_3, t_4. Let L be the concatenated argument list (t_1, t_2, t_3, t_4). Then node(TELL+ME) can specify its arguments by their indices in L. Thus node(TELL+ME)'s only test is L(2) = L(3), denoted by "2:3" below node(TELL+ME) in the network. (See Figure 1.) Similarly, node(TELL+ME)'s generator is the list (L(1), L(4)), denoted by "(1, 4)" above node(TELL+ME) in the network. Arguments which are not supplied by a node's constituents but instead originate at the node itself are specified by negative indices. For example, node(TOPIC/2)'s generator is denoted by "(1, 2; -1);" the -1 specifies the argument expr, which originates at node(TOPIC/2). The action stored in node(TOPIC/2) assigns this argument the value "ROCKEFELLER".

All of the recognition network components described so far are static. There is also associated with each node(r) a dynamic instance list IL. Each instance in the instance list of node(r) represents a single recognized occurrence (instantiation) of the relation r in the input utterance. An instance has four components: a unique identification number I; the time interval [t_1, t_2] containing the occurrence; the values x_1, ..., x_n of selected arguments and features of the occurrence; and a support set SS containing one or two instance identification numbers. An instance is denoted I:(t_1, t_2; x_1, ..., x_n; SS). During the recognition process, instances are created and deleted dynamically.

The recognition process is bottom-up, as follows. Initially all instance lists are empty. A lexical analyzer is invoked and begins to scan for occurrences of primitive relations in the input utterance. Since the lexical analyzer receives imperfect, incomplete information from the phonetic labelling routine, the best it can do is to identify possible occurrences. When it finds a possible occurrence of a relation r, it adds a new element to the instance list of node(r)

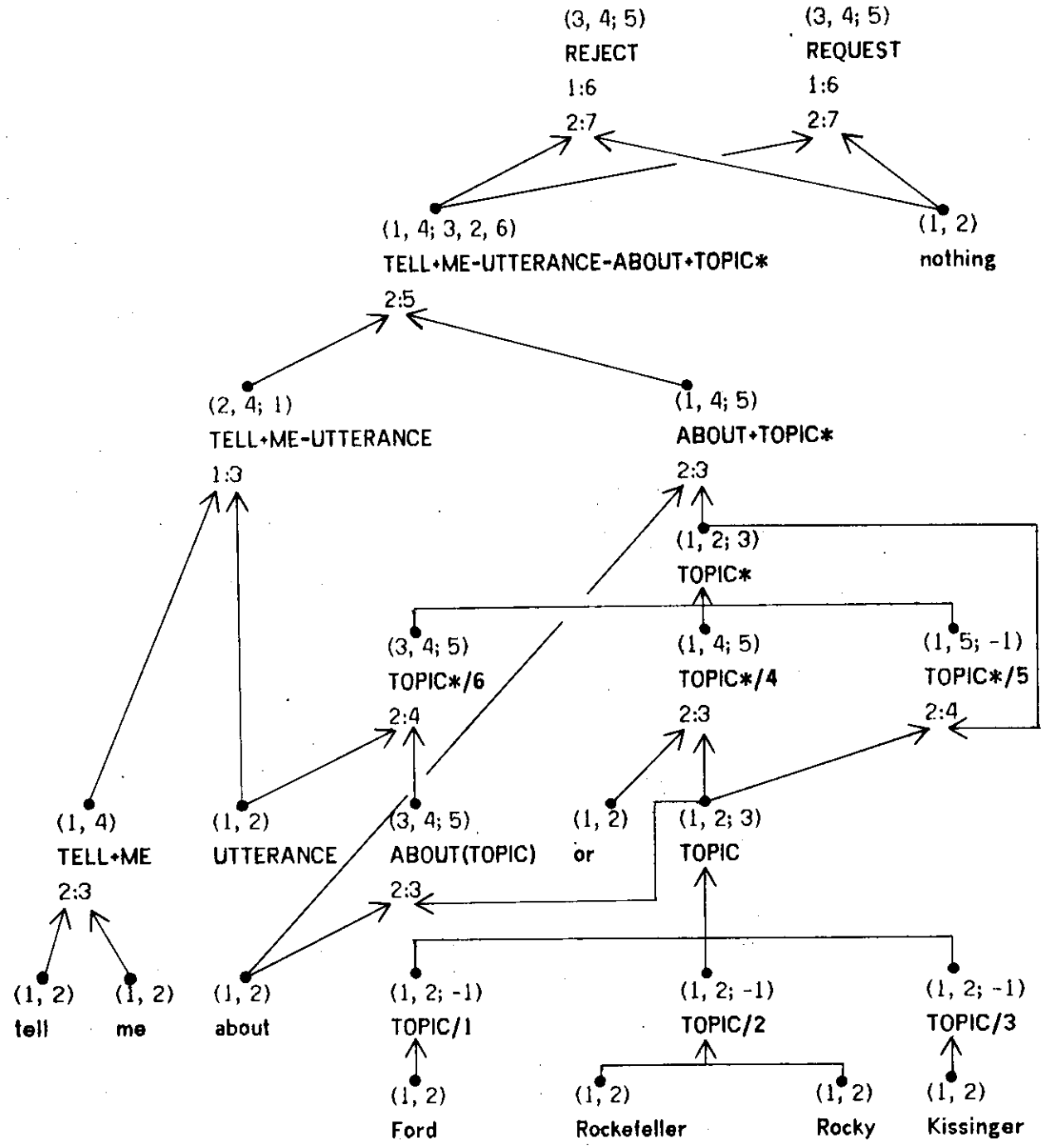


Figure 1. Sample Recognition Network (an ACORN). See text for an explanation of the tests i:j below the nodes and the generators (i_1, i_2; i_3, ..., i_k) above them.

containing the appropriate information. To understand the recognition process, imagine each node(r) as having a demon. The node(r) demon continuously monitors the instance list of each constituent node(s_i) of node(r). Whenever a new instance is added to the instance list of node(s_i), the node(r) demon adds a reference to this new instance to its node(s_i) add set. Similarly, whenever an existing instance of s_i is deleted, the node(r) demon saves a copy of it in its node(s_i) delete set. Add sets and delete sets are referred to collectively as change sets.[1] The demon then activates (wakes) node(r) itself by invoking code stored in node(r). [2]

When node(r) is activated, it updates its instance list according to the information in its constituents' instance lists and change sets. If node(r) can derive (construct) any new instances from instances of its constituents, it does so, adding the new instances to its instance list. The support set of each instance contains the identification numbers of the instances from which it has been derived. Node(r) deletes from its instance list any instances supported by (derived from) the defunct instances listed in its constituents' delete sets. The exact way in which all this is done depends, of course, on the type of node(r).

If node(r) is disjunctive, then it has d constituents node(s_1), ..., node(s_d). For each instance I:(t_1, t_2; x_1, ..., x_n; SS) in a node(s_i) add set, node(r) adds a new element I_{new}:(t_1, t_2; z_1, ..., z_k; {I}) to its own instance list, computing z_1,..., z_k from the values of x_1, ..., x_n according to the generator stored in node(r). I_{new}'s support set is {I} because the instance I_{new} of r is derived from (supported by, dependent on) the instance I of r's constituent relation s_i. For each defunct instance I in a node(s_i) delete set, node(r) deletes all instances I_{old}:(t_1, t_2; z_1, ..., z_k; SS) supported by I, i.e., such that $I \in SS$. (Actually, for disjunctive r, all instances of r will have support sets of size one, so $I \in SS$ iff $SS = \{I\}$. However, for conjunctive r, $|SS| = 2$; hence the set notation).

If node(r) is conjunctive, then it has exactly two constituents, node(s_1) and node(s_2), with respective instance lists IL_1 and IL_2, add sets AS_1 and AS_2, and delete sets DS_1 and DS_2. First node(r) deletes any of its instances I_{old}:(t_1, t_2; z_1, ..., z_k; SS) which were derived from instances in DS_1 or DS_2, i.e., those for which $SS \cap (DS_1 \cup DS_2) \neq \emptyset$. Then node(r) looks for new instance pairs I_1:(t_1, t_2; x_1, ..., x_n; SS_1) in IL_1 and I_2:(t_3, t_4; y_1, ..., y_m; SS_2) in IL_2 such that (t_1, t_2; x_1, ..., x_n) matches (t_3, t_4; y_1, ..., y_m) according to the tests stored in node(r). For

---

[1] Demons and change sets are already used in the Hearsay II system (Lesser, et al., 1974).

[2] One of the major virtues of the recognition network is that all the nodes contain similar code; in an actual implementation the differences between nodes would be parameterized and only the parameters stored in the nodes. For clarity of presentation, however, we treat nodes as autonomous entities.

each such matching pair, node(r) adds a new element I_{new}:(t_1, t_2; z_1, ..., z_k; {I_1, I_2}) to its instance list, using its generator to select z_1, ..., z_k from x_1, ..., x_n, y_1, ..., y_m. It is sufficient to check only those pairs of instances I_1, I_2 of which one or both are new, or more formally, such that either $I_1 \in AS_1$ and $I_2 \in IL_2$ or $I_1 \in IL_1$ and $I_2 \in AS_2$. For example, suppose the input utterance is "Tell me nothing about Rockefeller," and the lexical analyzer finds an instance I_1:(0, 18; ...) of **tell** and an instance I_2:(18, 23; ...) of **me**. Then the test stored in node(**TELL+ME**) becomes 18 = 18, which is true, so node(**TELL+ME**) adds a new instance I_{new}:(0, 23; ;{I_1, I_2}) to its instance list to represent the occurrence of "tell me" in the concatenated time interval [0, 23]. (Time is measured in centiseconds since the beginning of the utterance.) Now suppose the lexical analyzer mistakenly identifies the syllable "fell" in "Rockefeller" as the word "tell," and adds an instance I_3:(257, 269; ...) to node(**tell**)'s instance list. This may happen, for example, if the phonetic labeller correctly identifies the "F" in "Rockefeller" as an unvoiced consonant but can't tell if it's an "F," a "T," or a "P." No harm is done, however, since when node(**TELL+ME**) matches I_3 against I_2, the test 269 = 18 fails, and no new instance of **TELL+ME** is derived from I_3. This example shows how the ACORN automatically weeds out spurious instances hypothesized by the lexical analyzer on the basis of incomplete phonetic information.

Finally, if node(r) is negative, then it has two constituents, node(s_1) and node(s_2), where $r = (s_1$ <u>and not</u> $s_2)$. Let IL_1, IL_2, AS_1, AS_2, DS_1, DS_2 be the instance lists, add sets, and delete sets of node(s_1) and node(s_2). First node(r) deletes any of its instances I_{old}:(t_1, t_2; z_1, ..., z_k; SS) derived from defunct instances in DS_1, <u>i.e.</u>, those for which $SS \cap DS_1 \neq \emptyset$. Then node(r) looks for any instance pairs I_1:(t_1, t_2; x_1, ..., x_n; SS_1) in IL_1 and I_2:(t_3, t_4; y_1, ..., y_m; SS_2) in AS_2 such that (t_1, t_2; x_1, ..., x_n) matches (t_3, t_4; y_1, ..., y_m) according to the tests stored in node(r). For each such pair, node(r) deletes all of its instances I_{old}:(t_1, t_2; z_1, ..., z_k; SS) which depended on I_1, <u>i.e.</u>, such that $I_1 \in SS$. This is done since each such I_{old}, previously an instance of (s_1 <u>and not</u> s_2), is now invalidated by a new instance of s_2. Adding instances of node(r) is also a bit tricky, and proceeds as follows. First node(r) constructs the set IS of all instances $I_1 \in IL_1$ which match some I_2 in DS_2. Then node(r) looks for all instances I_1:(t_1, t_2; x_1, ..., x_n; SS_1) in $AS_1 \cup IS$ which match none of the instances in IL_2. For each such I_1, node(r) adds a new instance I_{new}:(t_1, t_2; z_1, ..., z_k; {I_1}) to its instance list.

To illustrate this, let us continue with our sample utterance. Suppose that at some point the lexical analyzer has recognized all the words in the utterance except the word "nothing," and node(**TELL+ME-UTTERANCE-ABOUT+TOPIC***) has I_4:(23, 41; 0, 274, "ROCKEFELLER"; ...) on its instance list. Since the instance list of node(**nothing**) is empty, node(**REQUEST**) will have an instance I_5:(0, 274, "ROCKEFELLER", {I_4}) on its instance list. Now suppose that the lexical analyzer finally recognizes the word "nothing," and puts the instance I_6:(23, 41; ...) on

node(**nothing**)'s instance list. This activates both of node(**nothing**)'s derivatives. Node(**REJECT**) matches I_6 against I_4, tests 23 = 23 and 41 = 41, and accordingly adds a new instance I_7:(0, 274; "ROCKEFELLER"; $\{I_4, I_6\}$) to its instance list. Node(**REQUEST**) matches I_6 against I_4, tests 23 = 23 and 41 = 41, and accordingly deletes I_5:(0, 274; "ROCKEFELLER"; $\{I_4\}$) from its instance list. This example shows how information is accumulated and corrected dynamically during the ACORN recognition process. It also illustrates the ACORN's state-saving nature and its sharing of information between top-level nodes.

Once node(r) has examined its constituents' change sets and, if appropriate, revised its own instance list, it goes back to sleep. Meanwhile, the demons sitting on the derivatives of node(r) have been watching its instance list and, when changes occur, activate their nodes. This chain reaction continues, fuelled by new instances generated by the lexical analyzer, until the lexical analyzer has stopped, all nodes are asleep, and all change sets are empty.

At this point each instance I:(t_1, t_2; x_1, ..., x_n; SS) of a non-primitive node, node(r), may be interpreted as a partial parse of the interval [t_1, t_2], with relevant syntactic and semantic features given by x_1, ..., x_n. For example, when the recognition of our sample utterance terminates, the instance I:(41, 274; "ROCKEFELLER"; $\{I_{\text{ABOUT}}, I_{\text{TOPIC*}}\}$) of **ABOUT+TOPIC*** may be considered to be a partial parse of the input interval [41, 274] containing "about Rockefeller." Parse trees can easily be reconstructed from the information contained in the support sets, especially if an appropriate scheme is used for assigning identification numbers to instances. Specifically, an ACORN assigns the unique number <node(r), i> to the i^{th} successive instance of node(r). This is easily implemented by storing a counter for i in each node. Moreover, this scheme insures that each node can assign identification numbers to its instances independently, without worrying about numbering conflicts, a property which is desirable for the implementation of the recognition network on a multi-processor machine. Parses of the entire utterance are given by instances of top-level nodes. Thus the instance I_7:(0, 274; "ROCKEFELLER"; $\{I_4, I_6\}$) of **REJECT** constitutes a total parse of the sample utterance, and supplies the semantic feature, expr, required by the action SUPPRESS(expr).

RELATIONSHIP TO EXISTING PARSERS AND PATTERN-MATCHERS

The original motivation which led to the ACORN concept was the development of a general automatic recognition system for spoken utterances, visual scenes, and other structured patterns in which context is a fruitful source of information. Since the speech understanding ACORN treats an utterance as a relational structure, it is related both to natural language parsers and to general pattern-matching mechanisms.

The ACORN's closest relative among natural language parsers is PARRY (Colby, 1974), a

program which simulates a paranoid individual being interviewed by a psychiatrist. PARRY uses the following methods to interpret its unrestricted idiomatic English input. Given an input sentence, e.g., "Tell me what is your current primary occupation," PARRY replaces each word by a canonical synonymic form, dropping any words it doesn't recognize, such as "current" in this example. This results in (TELL ME WHAT BE YOU MAIN JOB). PARRY then breaks up the sentence into short phrases, using function words such as "what" as reliable phrase boundary indicators. This method of segmentation yields ((TELL ME) (WHAT BE YOU MAIN JOB)). PARRY tests each phrase against a large library of stored templates. If no match is found for a phrase, PARRY omits one word at a time from the phrase, and tests each shortened version. Thus if (WHAT BE YOU MAIN JOB) is not in PARRY's library of patterns, it tries (BE YOU MAIN JOB), (WHAT YOU MAIN JOB), (WHAT BE MAIN JOB), and ultimately (WHAT BE YOU JOB), which matches a stored pattern. If there is still no match, PARRY assumes the phrase is unimportant and ignores it. Having reduced the input sentence to a few templates, PARRY attempts to match the pattern of templates against a library of such patterns. If necessary, it ignores some of the templates in order to get a match. Finally, an action associated with the matched pattern tells the response routines how to react. PARRY manages to handle teletyped input in unrestricted English fast enough (1 second response) and accurately enough to perform impressively on Turing's imitation test.

While the approach underlying PARRY is very successful with typed input, it appears to be too risky for spoken input. Unlike the "perfect" typed input which PARRY receives, the input to the syntax routine of a speech understanding system such as Hearsay II (Lesser, et al., 1974) is highly imperfect. For example, often the best that Hearsay II's phonetic labelling routine can do is, in effect, to say "this phone is an unvoiced consonant, probably F, T, or P." Each level of the system must be able to handle multiple alternative hypotheses about each portion of the input, and hope that other levels will be able to rule out most of the hypotheses on various grounds. PARRY can say, with confidence, "this portion of the input is such-and-such (e.g., the word "oh"), so I'll ignore it;" Hearsay II can only say "if this portion of the input is "oh," I can ignore it; but if it's really the word "no," then I'll need it." Thus in order to be used in a speech understanding system, PARRY's techniques must be implemented in a non-deterministic fashion. An ACORN can be thought of as a non-deterministic version of a PARRY-like system in which all possibilities are followed simultaneously in parallel.

Woods' augmented transition network (ATN) (Woods, 1970) is a mechanism for parsing natural language. It works top-down, uses backtracking, and produces a formal parse of the input sentence. In contrast, an ACORN works bottom-up, does no backtracking, and dispenses with a formal parse, extracting only those features of the utterance which are relevant to the particular application. Both structures are augmented with actions and consequently have the power of a Turing machine. An ACORN can be thought of as a bottom-up version of an ATN.

Miller (1974) has proposed a parser for spoken English which builds multiple partial parse trees representing alternative hypotheses about portions of the input utterance. In order to assemble the partial parse trees into a complete parse tree, Miller's parser performs a complicated and heuristic search for legal combinations of hypotheses. An ACORN differs from Miller's parser in handling all combinations simultaneously rather than sequentially, and in the simplicity of the matching operations it uses.

Current artificial intelligence programming systems such as PLANNER (Hewitt, 1972), QA4 (Rulifson, 1972), and SAIL (Feldman & Rovnar, 1969) can match a given relational template against a data base. However, the method they use is an exhaustive iterative search: one relation of the template is instantiated, and the data base is searched for instances of that relation which are consistent with the rest of the template. For example, SAIL, in matching the template (COLOR ⊗ ?X ≡ RED) and (SHAPE ⊗ ?X ≡ SPHERICAL), might look at all objects in its data base which have RED as the value of the attribute COLOR, and test the SHAPE attribute of each one in turn for a value of SPHERICAL, finally finding the object BALL which satisfies both conditions. If several templates are to be matched against the data base, they must be matched one at a time. In contrast, the associative matching operation performed by ACORNs effectively tests all the relations of all the templates simultaneously.

The ACORN's nearest ancestor among general pattern-matching methods is hierarchical synthesis (Barrow, et al., 1972). Consider the task of matching an input against a template. For example, the template might be a schematic representation of a building, and the input might be a set of line segments. Certain substructures, such as rectangles, may occur at several places in the template. A recognition algorithm employing hierarchical synthesis would replace the single, many-component template for "building" with a hierarchy of templates. The top-level template might define a building in terms of doors, windows, and walls. These components would in turn be defined by lower level templates, and so on. The lowest-level templates (e.g., "rectangle") would be defined in terms of line segments. The recognition algorithm would proceed by locating all instances of low-level templates such as "rectangle," grouping them into (or "synthesizing") higher-level templates such as "door" and "window," and so on up to "building." Barrow, et al., have noted that using hierarchical synthesis speeds up recognition considerably. Hierarchical synthesis is efficient for two reasons. First, it can exploit the repetition of subtemplates by recognizing all instances of a single subpattern just once. Second, before considering whether or not the entire pattern specified by a template is present, it can insure that all necessary subpatterns are present.

However, hierarchical synthesis as described by Barrow, et al., depends on a hierarchy defined a priori by the user. This limitation is transcended by the interference matching method (Hayes-Roth, 1974a), which does hierarchical synthesis in parallel in all possible

directions, thereby obviating the need for a predefined hierarchy. In interference matching, a template is represented as a set of relations. Each relation is a predicate with one or more symbolic variables. E.g., "line(a, b)" asserts that there exists a line segment joining the points a and b. The input is also a set of relations, whose arguments are constants. A partial match consists of an assignment of input constants to the symbolic variables of a subset of the relations in the template such that all of the relations in the subset hold true. Interference matching works by finding partial matches and combining them into complete matches.

Like interference matching, the ACORN method is an improved version of hierarchical synthesis in that it requires no predefined hierarchy. The ACORN compiler itself determines an economical hierarchy, and embeds it in the form of a recognition network. Hierarchy selection can be factored out into a separate compilation phase because the choice of hierarchy depends only on the templates and not on the individual input utterance. In interference matching, on the other hand, hierarchy selection depends on the input pattern, and is therefore a part of the recognition process. Thus the ACORN method combines the convenience of automatic hierarchy selection with the efficiency which comes from using a predefined hierarchy in the recognition process.

In real-world applications, input is matched against several top-level templates. Current methods of hierarchical synthesis (Barrow, et al., 1972) and interference matching (Hayes-Roth, 1974a) involve matching the input against one template at a time. Such an approach is clearly undesirable for tasks such as speech recognition, which may involve large numbers of templates. The ACORN compiler takes a whole set of templates and produces a single, unified recognition network for it; common subtemplates are shared not just within top-level templates but also between them. An instance of a subtemplate is recognized just once -- not separately for each top-level template in which it occurs. Hence recognition time depends not on the total number of templates, but just on the number of templates which are matched by some portion of the input. Therefore we expect recognition time to be very much sublinear in the size of the template set. This property is encouraging, since the number of templates required to recognize a significant subset of English would probably be several thousand.

In sum, an ACORN can be looked at as a bottom-up version of an ATN, a parallel implementation of a non-deterministic version of a PARRY-like system, a powerful pattern-matcher for efficient associative retrieval, or an improved mechanism for hierarchical synthesis, with automatic hierarchy selection and subtemplate sharing between templates.

APPLICATIONS, IMPLICATIONS, AND EXTENSIONS

In order for an ACORN to be efficient, the templates and input data characteristic of the chosen problem domain should tend to be asymmetric, so that a template will usually match a given portion of the input in at most one way. Let us illustrate with a negative example. Suppose the template we wish to match is $K_5(a, b, c, d, e)$, the complete graph on five vertices, represented by the conjunction of relations **line**(a, b) and **line**(a, c) and ... and **line**(d, e). Then any occurrence of K_5 (as a subgraph, say) in the input corresponds to 5! = 120 instances of T, since there are 5! different ways to bind the variables a, b, c, d, e to the five vertices of the K_5 in the input. For symmetries on a larger scale, the problem grows combinatorially worse. Clearly, an ACORN would be inefficient in such a domain, since it would insist on finding all instances of every template.

Fortunately, many interesting applications do not have this bothersome property. Speech, in particular, is highly asymmetric, partly because it is embedded in a one-dimensional ordered temporal domain. If **tell**(t_1, t_2) is true, then $t_1 < t_2$, so **tell**(t_2, t_1) cannot be true. Symmetries at a higher level can occur only if there is more than one syntactically and semantically valid way to group the input words into phrases, i.e., if the input is inherently ambiguous. In the presence of sufficient semantic and contextual information, most natural language utterances are fairly unambiguous, or at worst have a small number of possible meanings (i.e., two or three rather than 120). Hence speech should be amenable to recognition by ACORN.

What are the advantages of ACORNs for speech understanding? The bottom-up template-oriented approach is especially conducive to handling natural, idiomatic, conversational natural language robustly. Consider the problem in spoken speech of spurious insertions such as "oh," "um," "er." We wish to treat them the same as silences. We do this by adding rules like [**oh**(t_1, t_2) => **SILENCE**(t_1, t_2);] to our template grammar, and inserting the relation **SILENCE** in the templates between every two adjacent relations whose occurrences in input are likely to be separated by spurious insertions. This solution is not as expensive as it may seem. Since spurious insertions are recognized automatically, it is not necessary to test for each possible spurious insertion at each point in the input, which would indeed be expensive, and would be the only apparent solution in a top-down system such as an ATN.

This example also illustrates the reason for non-deterministic application of Colby's methods in a speech understanding system. Even if a spurious insertion is recognized, the corresponding portion of the input must not be discarded, since it may have been recognized incorrectly. If an ACORN recognizes an instance of "oh" in the interval [t_1, t_2], it puts the instance (t_1, t_2; ...) on the instance list of **SILENCE**, without discarding any information. That

way, if the interval actually contains the word "no," it is still there for the lexical analyzer to find. In contrast, when PARRY ignores information, it throws it away altogether.

Another phenomenon common to conversational speech is the idiomatic expression, e.g., "How are you?" Using ACORNs, we can simply include explicit template rules for such expressions, e.g.,

$$[\text{how*are*you}(t_1, t_2) \Rightarrow \text{GREETING}(t_1, t_2); \text{REPLY}(\text{"Fine, how are you?"})],$$

thereby short-circuiting the detailed syntactic parse which would be attempted by a more formal system such as Woods'.

The two techniques just described can be combined. Certain idioms such as "by the way" carry essentially no useful information and can be treated as spurious insertions by rules like

$$[\text{by}(t_1, t_2) \text{ and } \text{the}(t_2, t_3) \text{ and } \text{way}(t_3, t_4) \Rightarrow \text{SILENCE}(t_1, t_4);].$$

Some expressions occur either as meaningless idioms or as meaningful phrases, depending on context. Consider, for example, the utterance "I see, could I see the midnight digest?," which occurred in an actual experimental protocol. The first occurrence of "I see" is idiomatic and can be ignored; the second is crucial to the meaning of the utterance. An ACORN, in processing this utterance, would recognize both occurrences as instances of SILENCE, without discarding any information. The first occurrence would be ignored, as desired, but the second one would still be available to match other templates. PARRY can also ignore a spurious occurrence of an ordinarily meaningful expression, but only by omitting elements in the input one at a time and attempting a match each time. The ACORN is in effect performing this operation in parallel rather than iteratively.

Spurious deletions can also be handled by ACORNs. To handle spurious deletions, we want to permit partial matching of templates. We can do this within the ACORN framework simply by adding extra templates corresponding to commonly occurring partial matches of the original templates. The obvious weakness of this method is that it requires a priori knowledge of which deletions are likely to occur. The success of the method would require many iterations over a large corpus of test utterances, with new templates added as needed. Hopefully this process would converge, after a reasonable number of such iterations, to acceptable performance with respect to handling spurious deletions. (This method of "massive iteration" seems to have worked successfully for Colby.)

Partial templates could be used for another purpose as well. Although the bottom-up approach has several advantages, as described above, it is useful to have certain properties associated with top-down processing. One such property is the ability to focus the attention of lower-level modules on critical portions of input. Another is the ability to hypothesize words from above, for lower-level modules to confirm or reject. Although we earlier referred to a lexical analyzer which finds all instances of primitive relations (words) in the input utterance, this would in practice be too expensive. The actual Hearsay II system seeks to constrain hypothesization as much as possible; to do this it applies high-level information to cut down the number of possible words considered for each portion of the input. Thus it is desirable to have a speech understanding ACORN generate intermediate partial information telling the lower level modules which portions of the input they should concentrate on processing, and which words are likely to occur at a given place in the input, on the basis of the already recognized portions of the surrounding context.

This top-down extension to the basic bottom-up mechanism requires knowledge about the predictive value of partial templates. For example, we know that "What time" often occurs in the phrase "What time is it?" We can incorporate this information in an ACORN by including a rule

$$[\ \textbf{what}(t_1, t_2)\ \underline{\text{and}}\ \textbf{time}(t_2, t_3) \Rightarrow \textbf{WHAT•TIME}(t_1, t_3);\ \text{TEST}(t_3, t_{any};\ \text{"is it"})\ ],$$

where TEST is the action invoked upon recognition of the template. The effect of the TEST is to look for the missing instance of "is it" starting at the time t_3 in the input utterance. If it is found, it is added to the instance list for "is it," leading to the desired completion of the full template "What time is it."

In the above example, a partial template was used to predict downwards in the network. Partial templates can also be good upward predictors. For example, given an instance of the partial template T_1 = "time is it," the probability $P(T_2|T_1)$ that it occurs as part of the template T_2 = "What time is it" may approach certainty. If $P(T_2|T_1)$ is high enough, say .99, we may wish to save processing time by simply assuming that T_2 does in fact occur. We could obtain values for the various Bayesian probabilities $P(T_i|T_j)$ by using the network to recognize a large set of utterances and collecting appropriate data automatically. The ultimate extension of this is a learning system, which would modify its Bayesian probabilities continously and add nodes for new partial templates when appropriate.

Note that the utility of upward predictors derives from the redundant nature of speech. Upward prediction offers a robust method of dealing with unintelligible portions of an utterance by making reasonable guesses about what they contain. A system which insisted on

making a complete parse of the input utterance would necessarily give up in failure when confronted with an unintelligible fragment.

EVALUATION AND CONCLUSIONS

A full evaluation of the ACORN method must of course await implementation, which is currently in progress. In the meantime, there are several properties we expect the method to have.

Efficiency

The expected efficiency of ACORNs derives from several sources. Since the recognition process is organized so that contextual constraints are evaluated before adjacency constraints in the search through the space of combinations of hypotheses (see Introduction), a maximum of information is precomputed and stored statically in the network structure itself. Moreover, the state-saving nature of the recognition process eliminates most of the potentially costly recomputation which would be done by an equivalent top-down system such as an ATN. Finally, information sharing between templates reduces both time and memory costs. We expect recognition time to be considerably sublinear in the number of templates, which is crucial for any system which must handle hundreds or thousands of templates. This sharing of information between templates will provide a significant improvement over alternative methods which test templates separately.

Robustness

Using an ACORN makes it possible to dispense with a formal parse. This informality should contribute both to efficiency -- e.g., by permitting the immediate recognition of idioms like "how are you" -- and to robustness with respect to ungrammaticality, spurious insertions, and unintelligibility. Even when an ACORN cannot fully parse an utterance, it still provides a partial parse. E.g., if ACORN expects a complete sentence but is given an utterance consisting only of a noun phrase, it will recognize the noun phrase (assuming that the noun phrase is an instance of some node in the network). A top-down system like an ATN could also do this -- but only by adding noun phrase as a special case of a sentential form. When one considers the multitude of sentence fragments which commonly occur as utterances in conversational speech (e.g., "Where did you go?" "Out." "What did you do?" "Nothing."), it appears that some sort of bottom-up approach is necessary to handle them all efficiently.

Simplicity and Decomposability

ACORNs are organized so as to factor recognition processing into simple, universal operations performed at the nodes. This should make them fairly easy to implement; in particular, a large ACORN shouldn't be much more complicated to build and maintain than a small one. Moreover, the fact that each operation is performed locally, involving only a node and its constituents, means that ACORNs should be readily decomposable for implementation on parallel processing machines. Since little inter-process synchronization will be required, the speed-up factor should be good.

Flexibility

To expand an ACORN to handle new cases, one need only add appropriate new rules to the template grammar and recompile the recognition network. Of course, since compilation will be an expensive process, we plan to implement facilities for editing an already-compiled network. This should make ACORNs quite flexible to work with.

Generality

Finally, we expect ACORNs to have a broad range of applications, since they seem well-suited to recognizing any sort of relational pattern which embodies little symmetry and much contextual structure. Both spoken utterances and real-world scenes appear to be in this class. At present, we are implementing an ACORN to handle syntax and semantics for Hearsay II. We hope that our ACORN will do this efficiently. If it does, the ACORN method may be indicated as an effective overall organization for general recognition systems.

REFERENCES

Barrow, H.G., Ambler, A.P., & Burstall, R.M. Some techniques for recognising structures in pictures. In S. Watanabe (Ed.), Frontiers of pattern recognition. New York: Academic Press, 1972.

Colby, K.M., Faught, B., & Parkison, R.C. Pattern-matching rules for the recognition of natural language dialogue expressions. Memo AIM-234. Stanford: Stanford Artificial Intelligence Laboratory, Stanford University, 1974.

Feldman, J.A., & Rovnar, F. An Algol-based associative language. Communications of the ACM, 1969, 12, 439-449.

Frost, M. The news service system. Operating Note SAILON 72-2. Stanford: Stanford Artificial Intelligence Laboratory, Stanford University, 1974.

Hayes-Roth, F. An optimal network representation and other mechanisms for the recognition of structured events. Proceedings of the Second International Joint Conference on Pattern Recognition, 1974a.

Hayes-Roth, F. The representation of structured events and efficient procedures for their recognition. Pittsburgh: Department of Computer Science, Carnegie-Mellon University, 1974b.

Hewitt, C. Description and theoretical analysis (using schemata) of PLANNER: a language for proving theorems and manipulating models in a robot. Cambridge: MIT Project MAC, 1972.

Lesser, V. R., Fennel, R. D., Erman, L. D., & Reddy, D. R. Organization of the HEARSAY II speech understanding system. Proceedings IEEE Symposium on Speech Understanding, 1974.

Miller, P.L. A locally-organized parser for spoken input. Communications of the ACM, 1974, 11, 621-630.

Newell, A., Barnett, J., Forgie, J., Green, C., Klatt, D., Licklider, J.C.R., Munson, J., Reddy, R., & Woods, W. Speech understanding systems: final report of a study group. New York: American Elsevier, 1973.

Newell, A. Production systems: models of control structures. In W.C. Chase (Ed.), Visual information processing. New York: Academic Press, 1973.

Rulifson, J.F., Derksen, J.A., & Waldinger, R.J. QA4: a procedural calculus for intuitive reasoning. Menlo Park: Stanford Research Institute, 1972.

Woods, W.A. Transition network diagrams for natural language analysis. Communications of the ACM, 1970, 13, 591-606.

Pattern Recognition Pergamon Press 1974. Vol. 6, pp. 105–113. Printed in Great Britain

SCHEMATIC CLASSIFICATION PROBLEMS AND THEIR SOLUTION*

FREDERICK HAYES-ROTH†
Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Pennsylvania, U.S.A.

(*Received* 18 *December* 1973; *revised* 25 *April* 1974)

Abstract—The necessity arises in a variety of tasks to classify items on the basis of the presence of one of a number of criterial sets of co-related feature values. Such sets are called class characteristics. Because such classification problems require the identification of characteristics on the basis of limited training information, they entail a difficult search problem. Consideration of the differences between the theoretical models underlying characteristic and volume pattern generators suggests a schematic approach. Schemata, sets of commonly co-occuring features values, are probabilistic indicators of class membership whenever the characteristics are unknown but the characteristic model prevails. Formal and algorithmic solutions to the classification problem when exemplars are simple (consist only of M feature or attribute values) are described. The relevance of these procedures to problems involving general (relational) data structures is also indicated.

Classification Nominal Non-spatial Characteristic Interference Matching Schema Attribute Relation Performance.

INTRODUCTION

Several researchers[1–4] have recently argued for the necessity in many problem contexts to identify different sets of features, as well as distinct sets of feature values, associated with each of several alternative pattern classes. For example, in classifying a test item X as an element of one of the classes $C_1, C_2, \ldots, C_N$, the simultaneous presence in X of all values prescribed by a *schema* S_i might be criterial for the choice C_i, where $S_i = \{s_{i1}, s_{i2}, \ldots, s_{in_i}\}$ is a set of required feature values. Such a *schematic* approach to classification problems is to be distinguished from more typical *volume* approaches in which each test item is located at a point in M-dimensional space determined by its values on each of M feature dimensions and classified as an element of that class to whose exemplars it is most "proximate" in one of a variety of arbitrary ways.

Clearly both sorts of techniques may have advantages. Volume approaches are characterized by a simultaneous dependence on each of a common set of feature dimensions and a corresponding inability to recognize the irrelevance of some features for some classes. In the language of volume approaches, a schema of m features would correspond to an m-dimensional hyperplane in M-space.[4] The greatest advantage of the volume approach is the simplicity of the related solution: each test point can occur in at most one pattern volume and the associated classification response is trivial. In the schematic approach, however, a test item may *match* (contain) the schemata of a number of alternative classes, and the classification decision is not obvious.

The current paper provides a detailed description of the schematic approach to classification problems. In the next section two knowledge models, *spatial* and *characteristic*, are introduced which, it is suggested, constitute the underlying theoretical bases for the volume and schematic approaches, respectively. Consideration of the differences between the models illuminates the relative advantages and suitability of each approach for a specific task. Following that, *simple* (non-relational) and *general* (relational) schematic classification problems are defined. Both a complete formal solution and an efficient heuristic solution to the simple classification problem are then presented. A formal solution to the more general relational classification problem is available elsewhere.[5] Finally, the optimal properties and desirability of the schematic solutions are considered.

KNOWLEDGE MODELS UNDERLYING CLASSIFICATION PROCEDURES

A knowledge model is a theory describing the way in which feature values of pattern exemplars are gener-

* This study was supported in part by National Institutes of Health Grant GM-01231 to The University of Michigan.

ated. For example, a model of computer program malfunctions would relate erratic program behavior to the kinds of bugs which could cause it. A knowledge model underlying classification procedures must relate the kinds of commonalities one finds in the feature value sets of each exemplar to the process which generates them.

One prominent class of volume classification techniques associates all items whose vector representations are closely together in M-dimensional space as likely elements of the same category or class. The obvious knowledge model, the *spatial model*, which underlies this is that exemplars of a given class tend to share all feature values in common and deviations from the class central or mean values on each dimension are stochastically generated but severly constrained. Also, the assumption of independence of the deviations on each dimension is usually made, although in some cases, pairwise linear correlations of these deviations are considered. In short, this spatial model postulates that exemplars are ballistically produced, and all exemplar feature deviations from the target point (the ideal) are, in a sense, probabilistic errors.

The model underlying schematic classification is the *characteristic model*. It postulates that a given class, say *dog*, is defined by a characteristic, a set of mandatory features and that any other features present or absent (e.g. color of eyes or shape of tail) are irrelevant to the class concept. Employing such a model, one would predict that all exemplars of each class would manifest the corresponding class characteristic and that all noncriterial features would occur probabilistically.

A generalization of this single characteristic model would permit a number of distinct characteristics to define the class concept disjunctively. For example, an exemplar X of the class of *periodicals* would necessarily reflect at least one of the following characteristics: "X is a newspaper," "X is a magazine," or "X is a scholarly journal." This generalized characteristic model is assumed to underlie the schematic classification procedures developed later in this paper.

TWO TYPES OF SCHEMATIC CLASSIFICATION PROBLEMS

The *simple* schematic classification problem entails the identification of those schemata (feature sets) which are common to exemplars of each pattern class and the application of those schemata to effect classification of test items. Each class $C_1, C_2, \ldots, C_N$ is associated with a reference set $R_i = \{E_{ij} : j = 1, 2, \ldots, |R_i|\}$ of known exemplars of C_i. Each exemplar E_{ij} is an ordered set of feature values, $E_{ij} = \{f_{ijk} : k = 1, 2, \ldots, M\}$, where f_{ijk} is an element of F_k, the finite set of all possible values of the kth feature or attribute dimension. In addition, some assumptions regarding the sampling procedure used to generate these reference sets must be made. In this paper these assumptions are: (1) each reference set R_i represents a random sample of size $|R_i|$ taken with replacement from a much larger population P_i of exemplars of C_i; and (2) the relative cardinalities (population sizes) of $P_1, P_2, \ldots, P_N$ are

$$p_1, p_2, \ldots, p_N \left(\sum_{i=1}^{N} p_i = 1 \right)$$

and that in random samples taken from

$$P = \bigcup_{i=1}^{N} P_i$$

for test classification, each class type will occur with frequency determined by the multinomial distribution with corresponding parameters $p_1, p_2, \ldots, p_N$. The p_i may be assumed known or may be estimated from the cardinalities of the reference sets R_i.

Each pattern class C_i is assumed to be defined by an unknown number of characteristics C_{im} $m = 1, 2, \ldots, |C_i|$. Each characteristic is a simple schema of the form $C_{im} = \{c_{imk} : k = 1, 2, \ldots, M\}$ where each c_{imk} is either a feature value in F_k or is ϕ (*nil*), indicating the irrelevance of the kth feature value to the characteristic C_{im}. Each characteristic C_{im} is assumed to predict membership in class C_i in the sense that every item in C_i must manifest one of the associated class characteristics C_{im}, and items in any other class C_j should manifest such characteristics only by the chance combination of the component feature values. Classification of a test item X is thus performed by finding which characteristics C_{im} are matched by X and choosing the class C_i associated with the strongest or best predictor. The details of such a procedure are considered after the more general schematic problem is introduced.

The *general* (relational) schematic classification problem is similar to the simple problem just discussed in all ways but one. In this problem, the exemplars of each class may be sets of objects or events described

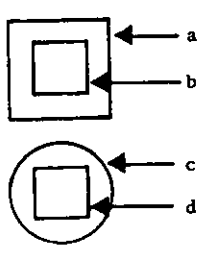


Fig. 1. A typical exemplar in a general schematic classification task.

in terms of both simple feature values and relations among objects, actions, subpatterns, etc. An example of the sort of exemplar which might be encountered in a general classification problem is the following *parameterized structural representation* (PSR) for the configuration in Fig. 1:

$$(\{p = \text{square}, o_1 = a\}, \{p = \text{square}, o_1 = b\},$$
$$\{p = \text{circle}, o_1 = c\}, \{p = \text{square}, o_1 = d\},$$
$$\{p = \text{contains}, o_1 = a, o_2 = b\}, \{p = \text{contains}, o_1 = c, o_2 = d\},$$
$$\{p = \text{above}, o_1 = a, o_1 = b, o_2 = c, o_2 = d\},$$
$$\{p = \text{same}, o_1 = a, o_1 = b, o_1 = d\}). \quad (1)$$

In this representation, the symbols, a, b, c and d are called *parameters*, because their only function is to facilitate consistent reference to objects occurring in several relations. Each relation is represented as a set comprising a predicate (p) and the ordered predicate objects (o_1 and o_2) which it relates. The predicates used in this example carry their ordinary English interpretations.

Suppose that this item were an exemplar of the class C_1 defined by the following characteristic: "C_1 contains any items which contain at least four geometric figures with two vertically organized pairs of nested figures such that the interior figures in both pairs are the same." This characteristic is easily represented in terms of the following PSR:

$$(\{p = \text{figure}, o_1 = a'\}, \{p = \text{figure}, o_1 = b'\},$$
$$\{p = \text{figure}, o_1 = c'\}, \{p = \text{figure}, o_1 = d'\},$$
$$\{p = \text{contains}, o_1 = a', o_2 = b'\}, \{p = \text{contains}, o_1 = c', o_2 = d'\},$$
$$\{p = \text{above}, o_1 = a', o_1 = b', o_2 = c', o_2 = d'\},$$
$$\{p = \text{same}, o_1 = b', o_2 = d'\}). \quad (2)$$

It is clear that the item in Fig. 1 described by (1) matches the characteristic (2). This follows from the fact that each parameter (e.g. a) in (1) corresponds to the similar primed parameter (e.g. a') in (2) and that each relation in (2) is matched by a corresponding relation in (1). Here, however, matching is a more general phenomenon. The relation $\{p = \text{square}, o_1 = a\}$ matches $\{p = \text{figure}, o_1 = a'\}$ because the former implies the latter if the equivalence of a and a' is assumed. The same is true of the matching relations $\{p = \text{same}, o_1 = a, o_1 = b, o_1 = d\}$ and $\{p = \text{same}, o_1 = b', o_1 = d'\}$ under the assumed pairwise equivalences $\{b, b'\}$ and $\{d, d'\}$.

Procedures for the generation of appropriate schematic representations of characteristics and for matching a test item to a set of possible predictive characteristics of this general sort are complex and beyond the scope of this paper. Both are discussed in Hayes-Roth[4] and fully detailed in Hayes-Roth.[5] For the current purposes, however, it will suffice to note that both the simple and general classification problems are identical in their formal structures. That is, in both cases, schemata which are essentially frequent subrepresentations of exemplars of each pattern class are identified and then used to classify matching test items. In this paper, detailed solutions are provided only for the simple problem. However, because of the significant similarity between the simple and relational problems, the generality of the provided solutions is immediately apparent.

A FORMAL SOLUTION TO THE SIMPLE SCHEMATIC PROBLEM

A few definitions must be given before proceeding. A *schema* S is any set $\{f_k : k = 1, 2, \ldots, M\}$ such that $f_k \in F_k$ or $f_k = \phi$. The *order* of S, $o(S)$, is the number of non-nil f_k in S. The *null schema* $\phi^M = \{f_k : f_k = \phi, k = 1, 2, \ldots, M\}$ is of order zero and is matched by any schema or exemplar. The interpretation given a schema S is that it represents a subset of features (those not equal to ϕ) which are co-related, occur simultaneously, in one or more exemplars of a pattern class. The schema T is a *subschema* of the schema S if S matches T, and this is denoted $S(*)T$. $S(*)T$ whenever each non-nil feature value in T also occurs in S. The notation $R_i S$ denotes the set of all elements E_{ij} in the reference set R_i of C_i which match S. R denotes the union

$$\bigcup_{i=1}^{N} R_i,$$

and $\bar{R}_i$ denotes the set of exemplars of all classes other than C_i: $\bar{R}_i = R - R_i$. As an example of this notation, $|\bar{R}_i S|$ denotes the number of exemplars of classes other than C_i which match the schema S. To denote that X is an exemplar of the pattern class C_i (or an element of the corresponding population P_i) $X \in C_i$ is written.

Given these definitions, the schematic classification problem is formalized as follows. Consider all possible schemata $S_1, S_2, \ldots, S_k$ matched by a test item X. Which of these is the best indicator or strongest predictor of membership in a single class C_i and which class is so indicated?

The solution proposed here rests on a particular interpretation of "best indicator." A schema S of X is said to indicate or predict a class C_i, denoted $S \Rightarrow C_i$, to the extent that the *a posteriori* conditional odds that X is an element of the corresponding population P_i, $Pr[X \in C_i | X(*)S]\ Pr[X \notin C_i | X(*)S]$, exceed 1. The *performance* value of S as an indicator of C_i, $U(S \Rightarrow C_i)$, is

a weighted sum of the expected number of correct less incorrect classifications (expected gain):

$$U(S \Rightarrow C_i) = E\{a_c(\text{number of correct classifications of } C_i \text{ indicated by } S) - a_w(\text{number of incorrect classifications of } C_i \text{ indicated by } S)\}$$
$$= a_c E\{N_c(S \Rightarrow C_i)\} - a_w E\{N_w(S \Rightarrow C_i)\}, \quad (3)$$

where a_c and a_w denote the respective significances of correct and wrong classifications. E denotes the expectation operation. $N_c(S \Rightarrow C_i)$ and $N_w(S \Rightarrow C_i)$ denote the number of correct and wrong classifications of test items as members of C_i indicated by S. In brief, the classification rule used is to classify X as an exemplar of the class C_i only if there exist some i and j such that $X(*)S_j$ and $U(S_j \Rightarrow C_i) \geq U(S_k \Rightarrow C_h)$ for all k such that $X(*)S_k$ and for all h and $i \in \{1, 2, \ldots, N\}$.

$U(S \Rightarrow C_i)$ can be computed in a straightforward way. The *a priori* probability densities. $P'[X \in C_i] \sim R[0, 1]$ (assuming p_i unknown). $P'[X(*)S|X \in C_i] \sim R[0, 1]$ and $P'[X(*)S|X \notin C_i] \sim R[0, 1]$ represent our initial uncertainty about the true probabilities, respectively, that any item drawn at random will be an exemplar of C_i, that any exemplar of C_i will match S, and that any non-exemplar of class C_i will match S. ($R[0, 1]$ is the rectangular (uniform) density function over all values in the interval from 0 to 1.) The reference sets R_i and $\bar{R}_i$ can be considered samples which provide additional information about these probabilities and, when combined with these prior distributions, yield posterior density distributions for the variables of principal interest.

Briefly, each variable can be considered a binomial parameter about which the uncertainty at the outset, reflected by the distribution $R[0, 1]$, is maximal. The estimation of the posterior probability $P''[X \in C_i]$, when p_i is unknown, may be considered as an example. The number of sampled exemplars which are elements of C_i is simply $|R_i|$. Those which are not contained in C_i are elements of $\bar{R}_i$, and their number is $|\bar{R}_i|$. The posterior density distribution for $P''[X \in C_i]$ is then

$$P''[X \in C_i] \sim Be(1 + |R_i|, 1 + |\bar{R}_i|), \quad (4)$$

the Beta distribution with parameters $1 + |R_i|$ and $1 + |\bar{R}_i|$ (Ferguson[10]). Similarly, the other posterior densities of interest are:

$$P''[X(*)S|X \in C_i] \sim Be(1 + |R_i/S|, 1 + |R_i| - |R_i/S|); \quad (5)$$

$$P''[X(*)S|X \in C_i] \sim Be(1 + |\bar{R}_i/S|, 1 + |\bar{R}_i| - |\bar{R}_i/S|). \quad (6)$$

From these facts, the following results are immediately derivable:

$$E\{N_c(S \Rightarrow C_i)\} = E\{P''[X(*)S|X \in C_i]\, P''[X \in C_i]\}$$
$$= \frac{1 + |R_i|}{2 + |R_i| + |\bar{R}_i|} \frac{1 + |R_i/S|}{2 + |R_i|}$$
$$= \frac{1 + |R_i|}{2 + |R_i|} \frac{1 + |R_i/S|}{2 + |R|}. \quad (7)$$

$$E\{N_w(S \Rightarrow C_i)\} = E\{P''[X(*)S|X \notin C_i]\, P''[X \notin C_i]\}$$
$$= \frac{1 + |\bar{R}_i/S|}{2 + |\bar{R}_i|} \frac{1 + |\bar{R}_i|}{2 + |R_i| + |\bar{R}_i|}$$
$$= \frac{1 + |R_i|}{2 + |\bar{R}_i|} \frac{1 + |\bar{R}_i/S|}{2 + |R|}. \quad (8)$$

$$U(S \Rightarrow C_i) = \frac{a_c(1 + |R_i|)\dfrac{1 + |R_i/S|}{2 + |R_i|} - a_w(1 + |\bar{R}_i|)\dfrac{1 + |\bar{R}_i/S|}{2 + |\bar{R}_i|}}{2 + |R|}. \quad (9)$$

This general solution is provided for cases in which the true population proportions p_i are unknown. When the p_i are known however, the terms $(1 + |R_i|)/(2 + |R|)$ and $(1 + |\bar{R}_i|)/(2 + |R|)$ are replaced by p_i and $(1 - p_i)$, so that

$$U(S \Rightarrow C_i) = a_c p_i \frac{1 + |R_i/S|}{2 + |R_i|} - a_w(1 - p_i)\frac{1 + |\bar{R}_i/S|}{2 + |\bar{R}_i|}. \quad (10)$$

The essential terms in these equations, $|R_i/S|$ and $|\bar{R}_i/S|$, are called the *positive* and *negative* sampling frequencies of S in R_i, respectively.

A COMPLETE ALGORITHMIC SOLUTION TO THE SIMPLE SCHEMATIC PROBLEM

There is one outstanding problem in applying the rule suggested in the preceding section for classification decisions. The number of non-trivial schemata matched by $X = \{x_1, \ldots, x_M\}$ is $2^M - 1$ which, for most realistic problems, is vastly too many to consider. One major algorithmic approach that suggests itself (Michalski[21]) is to look for a small number of low-order schemata which are moderately diagnostic for the reference set of each class. In this approach, each additional schema is especially chosen for its diagnosticity with respect to those exemplars not yet explained by (matching) the previously chosen schemata. The fundamental problem with such an approach is that it

is not very robust. The choice of most desirable schemata is likely to be overly dependent upon the exemplars encountered, especially when the size of the reference set is small in comparison to the number of feature values and or characteristics associated with that set. Consider what must be done if a high-order schema occurs frequently throughout one reference set. Any particular lower-order subschema chosen to represent the higher-order one is likely to perform badly whenever the entire set or some other subset of co-related features in the original schema is actually criterial.

Rather than following the typical sequential procedures which essentially assemble high performing schemata by addition of one feature at-a-time, the approach considered here operates simultaneously on the entire sets of features composing each exemplar and avoids choices among alternative schemata which are potentially correct representations of criterial characteristics. The *simple interference matching* (SIM) algorithm solves the classification problem by first generating a *maximal decomposition* of each reference set R_i as follows. A maximal abstraction of simple exemplars $E_{i1}, E_{i2}, \ldots, E_{iH}$, denoted

$$\prod_{j=1}^{H} {}^{*}E_{ij},$$

is the schema representing the set of feature values common to all of them. Thus,

$$\prod_{j=1}^{H} {}^{*}E_{ij} = E_{i1}{}^{*}E_{i2}{}^{*} \ldots {}^{*}E_{iH} = \{f_k : f_k = f_{ijk}$$

if $f_{i1k} = f_{i2k} = \ldots = f_{iHk}$, otherwise $f_k = \phi$, $k = 1, 2, \ldots, M\}$. The operation denoted by * and Π^{*} is called the *interference product* because of its similarity to physical interference processes. A maximal abstraction *cluster* in the reference set R_i is a set of exemplars in R_i with a unique non-null maximal abstraction not matched by any other exemplar in R_i. That is $h = \{E_{ij_1}, E_{ij_2}, \ldots, E_{ij_H}\}$ is a cluster in R_i if $E_{ij_n} \in R_i$ $(n = 1, \ldots, H)$,

$$\prod_{n=1}^{H} {}^{*}E_{ij_n} \neq \phi^{M},$$

and for no $j \in \{1, 2, \ldots, |R_i|\} - \{j_1, j_2, \ldots, j_H\}$ is it true that

$$E_{ij}(^{*}) \prod_{n=1}^{H} {}^{*}E_{ij_n}.$$

Finally, the maximal decomposition of R_i, denoted D_i, is the set

$$D_i = \{A_{ij} : A_{ij} \text{ is the maximal abstraction of a cluster } R_i/A_{ij} \text{ in } R_i, j = 1, 2, \ldots, |D_i|\}. \quad (11)$$

The decomposition D_i is very significant. It represents the minimal complete set of non-redundant (highest-order) schemata which occur in R_i with distinct frequencies. Consider any nth order schema A_{ij} in D_i. The number of non-trivial lower-order schemata which it matches is $2^n - 2$. Such schemata may be entirely redundant with A_{ij} with respect to the number of exemplars in R_i which match them. Let T be a subschema of A_{ij} so that T represents a *submaximal* abstraction of the exemplars in R_i A_{ij}. Then, if any exemplar E_{ik} matches A_{ij}, it also matches T. Furthermore, if there is any E_{ik} which does not match A_{ij} but does match T, there must be some other lower-order maximal abstraction $A_{im} \in D_i$ and associated cluster R_i A_{im} such that $A_{im}(^{*})T$ and $|R_i\ A_{im}| > |R_i\ A_{ij}|$. Thus, every schema T is either redundant with some A_{ij} in D_i in the sense that it is a lower-order schema with the same positive sampling frequency or is itself an element of D_i with a larger associated cluster R_i T.

Because the formal solution given in the previous section requires the identification of schemata with greatest performance measures, the maximal decomposition D_i is an extremely useful structure. The performance of a schema $A_{ij} \in D_i$ as an indicator of C_i, $U(A_{ij} \Rightarrow C_i)$, is an increasing function of $|R_i\ A_{ij}|$ and a decreasing function of $|\bar{R}_i\ A_{ij}|$. Any schema T which is redundant (in the above sense) with A_{ij} is such that $|R_i\ T| = |R_i\ A_{ij}|$, in all probability $|\bar{R}_i\ A_{ij}| < |\bar{R}_i\ T|$, but in no case can it be that $|\bar{R}_i\ T| < |\bar{R}_i\ A_{ij}|$ since $X(^{*})A_{ij}$ implies $X(^{*})T$. Thus, an algorithmic solution need only bother to identify (in the first phase) schemata in D_i. Not only are these the highest performing schemata, but all schemata of non-zero frequency in R_i not present in D_i are subschemata of at least one other schema in D_i.

A complete algorithmic solution to the first part of the simple schematic classification problem, identifying the schemata with highest performance values, is as follows. For each $i = 1$ to N by 1, initialize the list D_i to be empty, and for each $j = 1$ *to* $|R_i$ by 1, repeat the following two-step interference procedure.

Step 1. For each $k = 1$ to $|D_i|$ by 1 (if D_i list is not empty), compute a new schema S, a maximal abstraction, by the interference product

$$S = \prod_{E_{ik} \in R_i\ (A_{ij}{}^{*}E_{ik})}^{*} E_{ik} \quad (12)$$

and if $S \notin D_i$, concatenate S to the D_i list.

Step 2. Concatenate E_{ij} to the D_i list, unaltered.

At the conclusion of this procedure, each schema A_{ik} in D_i is maximally performing and non-redundant. However, there may be some redundant lower-order schemata which will in fact be criterial for some

classification tasks on novel test items. Suppose S_1 is a redundant subschema of a higher-order schema S and an item X is to be classified. It is possible that all of the following would be true: not $[X(*)S]$, $X(*)S_1$, and

$$U(S_1 \Rightarrow C_i) = \max_{A_j \ldots X(*)A_j} \max_{j=1,\ldots,N} U(A_j \Rightarrow C_j).$$

In such a case, S_1 is a critical characteristic schema. Although such situations are very improbable when the reference sets are very large and conform to the assumptions made earlier, for a complete solution each of the schemata in D_i should be iteratively reduced (informationally) by setting some non-nil feature values to o to produce redundant lower-order schemata which are concatenated to the D_i list if not already contained in it. The entire expanded list, denoted D_i^*, should then be sorted according to the performance value of each contained schema. Although this procedure ultimately entails a consideration of many redundant schemata, this is unavoidable when a complete solution is required.

The ordered list D_i^* for each reference set answers the first part of the classification problem. The second part, which class is most strongly indicated, is easily answered in terms of these sorted lists. The classification response to a test item X is C_i whenever, for some $A_{ij} \in D_i^*$ such that $X(*)A_{ij}$, $U(A_{ij} \Rightarrow C_i) \geq U(A_{kn} = C_k)$ for all i, j, k and n such that $A_{kn} \in D_k^*$ is matched by X. Of course, all lists can be scanned simultaneously if they are merged and sorted *in toto* by performance values; then the first A_{ij} in the combined list matched by X would entail the classification decision C_i.

AN EFFICIENT SPACE LIMITED ALGORITHM FOR THE SIMPLE PROBLEM

There is one principal weakness of the complete solution just discussed. The number of schemata in each D_i^* list is likely to be astronomical due to the random pairings of non-criterial features in each exemplar. Nevertheless, with minor modifications, the general SIM algorithm provides an efficient and desirable procedure.

The principal ways to limit the extensive processing involved are: (1) limit the number of schemata contained in each D_i; (2) limit the total number of schemata over all classes to those n with highest performance values; or (3) reject all schemata indicating C_i whose performance values are below some value r_i. In general, these criteria result in a limitation on memory spaces (on the number of schemata in each list D_i or overall) and, as a result, on the time of processing. Hence, the resulting modified algorithms are called *space limited interference matching* (SLIM) procedures.

To produce a SLIM algorithm, the only mandatory modification to the previous procedure which is needed is to limit the number of schemata held in working memory according to the selected criterion, and this is straight-forward. At each stage in processing, memory will be occupied with those schemata so far producible on the basis of exemplars sampled with maximum expected performance values.

Other techniques for producing major improvements in processing speed and effectiveness in SLIM procedures include: (1) limited positive and or negative sampling; (2) elimination of negative sampling; (3) intersection matching; (4) inverse organization of the D_i lists; and (5) effective list pruning through competition for space based on conditional performance values. Each of these is discussed in turn.

Limited positive sampling restricts the extent to which a newly produced schema S is evaluated for frequency of occurrence, $|R_i\ S|$. A schema S is produced by the interference operation applied to a newly introduced exemplar E_{ij} and a previously computed abstraction A_{ik}. Rather than computing $|R_i\ S|$ by checking each E_{ij} for a match with S, a limited sample $R_i' \subset R_i$ can be so evaluated. This significantly reduces the number of computations without biasing the results or jeopardizing robustness of the procedure. Limited negative sampling similarly restricts the computation of $|\bar{R}_i\ S|$ also required for the computation of $U(S \Rightarrow C_i)$.

Negative sampling can be completely eliminated in a reasonable way by assuming that the feature values occurring in a schema S which indicates C_i are independently distributed in $\bar{R}_i$, the non-exemplars of C_i. The validity of this assumption is both questionnable and testable in any particular case. Nevertheless, if this assumption is made, the frequency with which S would be expected to occur in $\bar{R}_i$ is simply computable as the joint probability (product) of each of the marginal probabilities associated with each non-nil f_k in S, i.e.

$$\text{estimated } |\bar{R}_i\ S| = \prod_{\substack{f_k \in S \\ f_k \neq \phi}} (|\bar{R}_i\ \{f_k\}| / |\bar{R}_i|) \times |\bar{R}_i|. \quad (13)$$

The simplicity of this computation can afford significant reductions in processing time compared to both unlimited and limited negative sampling.

Intersection matching is an extremely efficient process for computing the sampling frequency of a schema S in a set R. Each feature value f_k is associated with a bit string $B(f_k) = (b_1, b_2, \ldots, b_{|R_i|})$ such that $b_j = 1$ if the kth feature value (f_{jk}) of exemplar E_{ij} equals f_k; otherwise $b_j = 0$. Then limited or unlimited sampling can be effected by anding the corresponding bits of all strings $B(f_k)$ of non-nil f_k in S. If unlimited sampling is desired,

the entire bit strings are anded. Limited sampling is performed by reducing the length of each $B(f_k)$ to include only those exemplars in a reduced subset R_i' of R_i. In either case, the resultant jth bit is 1 only if $E_{ij}(*)S$. The cardinality of R_i/S or R_i'/S is then the number of 1 bits in the resultant bit string. Similarly, negative sampling can be effected by anding together the appropriate inverted bit strings from the set $\bar{R}_i$.

A major processing savings may be achieved through an inverse organization of the D_i lists, as follows. Each feature value f_k is associated with a bit string $d_i(f_k)$ such that the jth bit is one only if the jth schemata in the D_i list contains the value f_k. Because the SLIM procedure requires the evaluation and storage of schemata only when they are not already in the D_i list, these $d_i(f_k)$ bit strings can be used to expedite the related searches. The and product $d_i(S) = d_i(f_1) \wedge d_i(f_2) \wedge \ldots \wedge d_i(f_t)$ for all non-nil $f_k \in S$ would then identify all schemata in D_i matching a newly produced schema S. If, in addition, a bit string $o_i(t)$, which is one in bit j only if the jth schema in the D_i list is of order t, is anded to $d_i(S)$, the resultant string is not identically zeroes only if S is in D_i already. Such a search technique is desirable whenever the capacity of the D_i lists is very large in comparison to the order of the schema S.

Finally, the overall expected utility or *total* performance of the D_i list of schemata can be maximized (in expectation) and, conversely, redundant schemata can be eliminated through competition if schemata are dynamically ordered by *conditional* performance values. Two sorts of redundancy are necessarily controlled. First, any schema S' which matches a lower-order, higher performing schema S is assigned a null *conditional* positive sampling set (frequency zero). This assignment reflects the fact that whenever S' indicates class C_i, S indicates C_i more strongly, because $S'(*)S$ and $X(*)S'$ imply $X(*)S$. Thus the simultaneous presence of both S and S' in a D_i list is unnecessary, and S' is truly redundant with S. A more complicated sort of redundancy arises when a schema S in D_i matches a lower performing schema S'. This may occur whenever S is a characteristic of C_i and the number of characteristics of C_i exceeds one. Such a lower-order schema S' may be produced whenever an exemplar E_{ij} introduced by the interference procedure contains a distinct characteristic T and $E_{ij}{}^*S = \phi^M$. To eliminate the possibility that schemata reflecting such accidental interference patterns proliferate (as they might if $o(S)$ is large), the redundant lower-order schema S' is assigned the conditional positive sampling set defined by

$$R_i/S' = (R_i - \bigcup_{k \in K} R_i/S_k)/S' \tag{14}$$

where $K = \{k: U(S_k \Rightarrow C_i) > U(S' \Rightarrow C_i)$ and $S_k(*)S'\}$. This conditional sampling set estimates the frequency with which the schema S' occurs in exemplars of C_i which do not also match S. Whenever a low-order schema S' is actually characteristic and not simply redundant with a higher performing schema, this will necessarily be reflected by a substantial number of elements in R_i/S' as defined by (14). Conversely, the emptiness of R_i/S' indicates that S' is completely redundant with higher performing schemata.

If conditional performance values of schemata are computed as in equations (9) and (10) using, when appropriate, the conditional positive sampling frequences defined above, the performance values so obtained estimate the net weighted gain of correct less incorrect classifications attributable to the corresponding schemata in the presence of the higher performing current alternatives. That is, higher performing, lower-order schemata preclude the application in classification of a matching lower performing schema, and higher performing, higher-order schemata make less probable the necessity of a matched lower-order schema as an indicator of class C_i. Competition for storage space on the D_i list based on conditional performance values then acts to preserve the overall maximally performing set of schemata at each moment in time.

It is argued here that such a measure is the best to use for space allocation decisions. One principal alternative to this technique would be to define the conditional positive sampling set as follows:

$$R_i/S' = (R_i - \bigcup_{j \in J} R_i/S_j)/S' \tag{15}$$

where $J = \{j: U(S_j \Rightarrow C_i) > U(S' \Rightarrow C_i)\}$. Comparing this to (14), it can be seen that this alternative considers the predictive value of a schema only in marginal terms, that is, in terms of what predictive power it adds to the existing set of higher performing schemata regardless of whether they match it or not. A simple example will betray the undesirability of such a technique. Suppose C_i comprises four characteristics: $C_{i1} = \{F_1 = f_1, F_2 = f_2\}$, $C_{i2} = \{F_2 = f_2, F_3 = f_3\}$, $C_{i3} = \{F_3 = f_3, F_4 = f_4\}$, $C_{i4} = \{F_1 = f_1, F_4 = f_4\}$, where an element $F_j = f_j$ signifies that the jth feature value must equal f_j. Although overlapping, these feature set characteristics are distinct. Evaluating performance in terms of the sampling set defined by (15) would be disastrous in this case. At most two of the characteristics would be considered to have non-null positive sampling sets, and either the pair $\{C_{i1}, C_{i3}\}$ or the pair $\{C_{i2}, C_{i4}\}$ would be thus evaluated. Which pair would be assigned positive sampling frequencies greater than zero and which frequencies of zero would

be determined only by chance. The usefulness of the derived schemata for classification would then be extremely tenuous. Only if all test items presented manifest one of the lucky two positively performing characteristics would classification be successful, contrary to the assumption that any of the four characteristics ought to be criterial for classification. Such errors do not arise if the conditional frequency is computed according to equation (14).

In addition to maximizing the expected total performance of the schemata in the D_i list, making performance values conditional upon higher performing schemata may be viewed as pruning unneeded schemata from further consideration when more valuable alternatives are present. This viewpoint is illustrated in the following example.

An example

Suppose C_1 is associated with two characteristics, C_{11} = {color = red, temperature = medium, pressure = lowest} and C_{12} = {color = pink, pressure = low, size = largest} in a study involving exemplars comprising 100 values on attributes including size, color, temperature, and pressure. Suppose also that each attribute takes on 5 different values, so that the number of possible distinct schemata indicating membership in class C_1 is $(5+1)^{100}-1$. Under the assumptions that C_{11} occurs with probability 2/3 in R_1, that C_{12} occurs with probability 1/3 in R_1, that each value is equiprobable on each non-criterial attribute dimension, and that C_{11} and C_{12} occur in $\bar{R}_1$ only by chance combinations of the corresponding independent random variables, it is possible to indicate the extreme efficiency of SLIM procedures.

The probability of n chance (non-criterial) feature value matches among k exemplars randomly chosen either only from $R_1 C_{11}$ or only from $R_1 C_{12}$ is approximately:

$$p(n,k) = \binom{97}{n}(1/5)^{n(k-1)}[1-(1/5)^{k-1}]^{97-n}. \quad (16)$$

For example, the probability that any (at least 1) chance matches occur among 7 exemplars is approximately:

$$1 - p(0.7) = 1 - [1-(1/5)^6]^{97} = 97(1/5)^6 = 0.006. \quad (17)$$

After sampling about 20 exemplars from R_1, therefore, it would be extremely unlikely that either C_{11} or C_{12} would not have been produced. Even if space for schemata was limited to 5, say, it would be very improbable that all spaces would have been occupied by schemata matching only C_{11}. Each time a new exemplar from $R_1 C_{11}$ is interfered with an existing schema $A_{ij} \in D_i$ which matches C_{11}, either the new schema S (equation 12) is a lower-order schema matching C_{11} or $S = A_{ij}$. In the former case, the positive sampling frequency of A_{ij} is reduced to zero to account for the redundancy between S and A_{ij}; this makes A_{ij} the weakest competitor for space on the D_i list. In the latter case, no changes are made to the D_i list. In short, until C_{11} is produced by interference matching at most one high performing schema in D_1 is likely to match C_{11}. The second highest performing schema is, similarly, likely to be one matching C_{12}. Because some extremely improbable sampling circumstances might make other schemata more prevalent than ones matching either C_{11} or C_{12}, a few additional (here, 3) memory spaces might be required to produce a proper solution to the classification problem.

In the future, simulations will be run to compute the actual performance of SLIM procedures under a variety of constraints. For the present, however, it should be clear that only a small amount of redundancy in the storage used to hold alternative schemata is necessary to offset large differences in the positive sampling frequencies of alternative characteristics.

CONCLUSIONS

A complete solution to the simple schematic classification problem was developed using the simple interference matching technique. It produces a set of schematic characteristics of the reference set R_i of a pattern class C_i by interfering the representations of all subsets of exemplars to identify clusters in R_i and their corresponding maximal abstractions. Each of these abstractions is a potential indicator of membership in C_i by the assumptions underlying the characteristic model.

A number of heuristics can be employed to design practical SLIM procedures whose results approximate the most useful results of the complete procedure. In such space limited procedures, the ranking of schemata by conditional performance values acts to maintain in memory schemata of the highest possible order relative to the large set of potentially redundant lower-order subschemata with similar positive sampling frequencies. This, in turn, acts to increase the probability that subsequent interference of a newly sampled exemplar and a previously produced schema will result in the production and identification of a subschema which is in fact better performing, until the true characteristic schemata are so produced. Conversely, when higher-order schemata match lower-order, higher performing schemata, the evidence is strong that the former are redundant with the latter in the sense that both match the same characteristic, and only the latter can possibly be applied to effect classification. By appropriate

reduction of the performance values of such redundant schemata. SLIM procedures insure that useless schemata are pruned from overcrowded storage.

Although these properties suggest that SLIM procedures may be optimal with respect to the overall performance of the schemata produced given the particular storage constraint, the proof of such a claim would both require an excessive number of very strong assumptions (e.g. assumptions about the joint distributions and numbers of compatible and incompatible distinct characteristics in each exemplar and over each class) and be very complicated statistically. In the absence of such a proof, the importance of SLIM procedures is argued on the basis of their three obvious properties: they are associated with an important and reasonable underlying model and, therefore, are widely applicable; because they utilize as much information as possible at each stage of processing and avoid unnecessary and arbitrary choices, they are apparently quite robust; and they are easily computed, primarily in terms of extremely efficient bit string logical products. It seems likely, therefore, that SLIM techniques will prove valuable wherever the characteristic model is an appropriate description of the process by which exemplars are generated.

REFERENCES

1. F. Hayes-Roth. A structural approach to pattern learning and the acquisition of classificatory power. *Proc. Int. Jt. Conf. Pattern Recognition* (1973).
2. R. S. Michalski. AQVAL 1—Computer implementation of a variable-valued logic system VL_1 and examples of its application to pattern recognition. *Proc. Int. Jt. Conf. Pattern Recognition* (1973).
3. B. Sklar and O. E. Drummond. On the use of custom features for pattern recognition. Paper presented at *Int. Jt. Conf. Pattern Recognition* (1973).
4. S. Watanabe. Subspace method in pattern recognition. *Proc. Int. Jt. Conf. Pattern Recognition* (1973).
5. F. Hayes-Roth. *Mechanisms of abstraction in a system for the structural representation of knowledge, with applications to knowledge acquisition and problem solving.* Michigan Mathematical Psychology Program. Ann Arbor (1973).
6. T. S. Ferguson. *Mathematical statistics.* Academic Press. New York (1967).

UNIFORM REPRESENTATIONS OF STRUCTURED PATTERNS AND AN ALGORITHM FOR GRAMMATICAL INFERENCE[1,2]

Frederick Hayes-Roth
Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

ABSTRACT

Many events (patterns) may be described by structural (conjunctive relational) representations, and general computational behavior may be represented in terms of a set of grammatical rules (productions, transformations) relating two such event representations as contingency and response components. Uniform representations and graphs of structural descriptions and rules are introduced. An abstraction of a set of uniform representations corresponds to a common subgraph of the corresponding uniform graphs. Every rule $F = [(\forall x_1,...,x_n)\ C(x_1,...,x_n) \Rightarrow R(x_1,...,x_n)]$ which can be induced from a training set $I = \{(C_i, R_i) : i = 1, ..., N\}$ of contingency-response (input-output) pairs is identified with a common subgraph of the uniform graphs of the causal inferences $C_i \Rightarrow R_i$. A general learning problem is formulated for which three cases are distinguishable on the basis of if and how substitutions from input to output patterns are to be made. Category (unary) and n-ary predicate learning in this framework are discussed. Examples of rule learning applications are drawn from the domains of transformational grammar and speech understanding. The properties (both desirable and undesirable) of the proposed approach and the differences between it and previous approaches are also considered.

1. INTRODUCTION

The current paper is motivated by the observation that rules of behavior (e.g., rules of transformational grammar, pattern classification, and general computation) may be directly abstracted from examples of their use if appropriate restrictions are placed on the representation of the rules and training examples. Specifically, if rules are restricted to the form $F = [(\forall x_1,...,x_n)\ C(x_1,...,x_n) \Rightarrow R(x_1,...,x_n)]$ where both $C(x_1,...,x_n)$ and $R(x_1,...,x_n)$ are conjunctive products of variable forms of the predicate calculus, a graph representation exists

for which it is true that the rule F is a common subgraph of the graphs representing the training examples. Many previously intractable learning problems can be solved in this way, and examples in this paper include the induction of the equi-noun phrase deletion rule of transformational grammar and a variety of rules for use in a real-time speech understanding system. However, numerous details of the proposed representation and abstraction procedures must be considered in solving this general learning problem. Nonetheless, the essential ideas in this paper are easily stated: (1) a uniform graph representation exists for any conjunctive clause of the predicate calculus which insures that one graph is a subgraph of another one if and only if the associated clauses describe two patterns where the first is a subpattern of the second; (2) training examples for the induction of an unknown rule can be provided such that the graph representation of each example contains the unknown rule as a subgraph; and (3) a previously published algorithm for extracting abstractions (subrepresentations) common to a set of graphs is thus effective for the induction of rules. Before proceeding to consider general rule learning, the foundation is laid by reviewing related research concerning the hypothesization of classification rules for non-metric data.

In recent pattern recognition research (Barrrow, et al., 1972; Shaw, 1972; Eden & Halle, 1962; Evans, 1968, 1969; Hayes-Roth, 1973, 1974a, 1974b; Michalski, 1973; Watanabe, 1973; Winston, 1970), an emphasis has been placed on the structural (relational) representation of pattern prototypes and procedures for the recognition of stimuli which exhibit prototypic structure. The assumption that each pattern class is identified by one or more prototypic structural descriptions has been called the characteristic model (Hayes-Roth, 1974a) to distinguish it from the more traditional spatial or multidimensional pattern recognition model. In addition, algorithms have been described which efficiently search the space of plausible pattern characteristics (prototypes), i.e., the set of all structural representations manifested by the training exemplars of a pattern class (Hayes-Roth, 1974a; Stoffel, 1974). If there are N mutually exclusive pattern classes (or responses) R_1, ..., R_N and the training exemplars of R_i are $I_i = \{E_{i,1}, ..., E_{i,n_i}\}$, these algorithms are developed from the following two observations. (1) Each proposition P which is true of some exemplars in I_i is a potential basis for classification of any novel item Y for which P(Y) is true as a member of R_i. Of course, the plausibility of (confidence in, support for) the rule $[(\forall Y)\ P(Y) \Rightarrow Y \in R_i]$ should be a strictly increasing function of the positive frequency of P in I_i, which is defined as $|I_i/P| = |\{E_{i,j} : E_{i,j} \in I_i \ \& \ P(E_{i,j})\}|$, and a strictly decreasing function of the negative frequency of P in I_i', $|I_i'/P| = |\{E_{k,j} : E_{k,j} \in I_k \ \& \ k \neq i \ \& \ P(E_{k,j})\}|$. In words, the greater the frequency with which the proposition P is true of positive instances (I_i) of R_i and the less the frequency with which P is true of non-instances (I_i') of R_i, the higher the probability that P is a characteristic of (criterial for) class R_i. (2) If all of the propositions which are true of each exemplar are given as the description of the exemplar, those schemata (conjunctive sets of propositions) which occur most frequently among the descriptions of exemplars of a single class are likely to be plausible hypothetical class characteristics.

An understanding of the details of such pattern learning programs requires only a small amount of additional terminology. Define an abstraction of several exemplar propositional descriptions as any proposition which is directly implied (contained) by each of them. Abstractions are identified by computing conjunctive sets of propositions which are consistently present in (subsets of) the conjunctive sets of propositions describing several exemplars of the same pattern class. The plausibility of each such abstracted schema as a pattern characteristic is then evaluated by computing an appropriate measure which is an increasing function of its positive and negative frequencies in I_i and I_i', respectively (Hayes-Roth, 1973, 1974a; Stoffel, 1974). Of course, if the abstraction procedures produce a schema which is manifested (matched) by every exemplar in I_i and none in I_i' and if the underlying pattern is equivalent to a conjunctive concept (Bruner, et al., 1956), the abstraction is a plausible candidate for the class characteristic (see, for examples, the concepts learned in Winston, 1970 and Hayes-Roth, 1973).

The purposes of the present paper are three: first, to introduce a uniform representation for structural descriptions of events (e.g., visual or acoustic patterns, semantic or syntactic structures, etc.) which is designed to insure that all schemata which are manifested by each of a set of exemplars can be identified by any procedure which, in effect, can identify a common subgraph of several undirected, labelled graph representations associated with each of the exemplars; second, to generalize the previously published learning techniques to problems involving the induction of universally quantified rules of the predicate calculus like those of web grammar (Pfaltz & Rosenfeld, 1969) or transformational grammar (Friedman, 1971); and third, to illustrate this learning technique by applications to several current problems arising in the speech understanding research at Carnegie-Mellon University.

The remainder of the paper is organized as follows. Section 2 presents the essential details of the structural representation of patterns, first in terms of typical predicate calculus (list-based) forms and subsequently in terms of equivalent (set-based) relational representations. Difficulties arise in all such systems both from the necessity to abstract m-ary relations which are implicit in (contained by) n-ary relations ($n > m$) and from the desire to allow many-to-one object correspondence mappings from stimulus to template pattern representations; these problems are discussed and motivate the uniform representations which are then proposed. This scheme is then extended to cover the representation of grammatical rules (substitution productions). In Section 3, a formal statement of the learning problems addressed by the present paper is provided. A solution to any particular problem will require one of three distinct sorts of inference mechanisms depending on the nature of the unknown rule and the amount of information provided. Sections 4, 5, and 6 provide detailed solutions to problems of each of the three types. Section 7 briefly discusses methods for discovery of a particularly useful sort of unary predicate corresponding to a category, a set of mutually

exclusive elements which occur as alternatives in particular structural contexts within learned rules. Also, the correspondence between learned patterns and novel n-ary predicates is discussed. Section 8 illustrates some of the problems in our speech understanding research which are being approached with the proposed learning procedures. The last section discusses the relation of these learning problems, procedures, and results to related research, obstacles to the widescale implementation of such learning procedures, and directions for future research.

2. STRUCTURAL REPRESENTATIONS

Figure 1 illustrates two patterns representing (A) a triangle and (B) two lines. These patterns could be represented by typical conjunctive predicate calculus picture descriptions in terms of a binary symmetric predicate line as in A_1 and B_1, below.

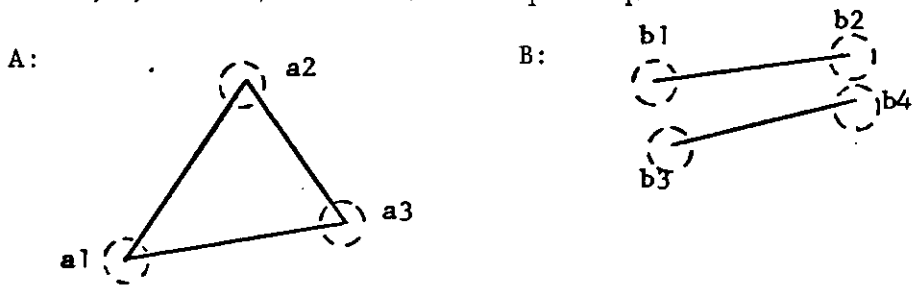


Figure 1. A triangle (A) and two lines (B) described in terms of relations on nodes in the line drawings.

A_1 = line(a1,a2) & line(a1,a3) & line(a2,a3) &
line(a2,a1) & line(a3,a1) & line(a3,a2) (1)

B_1 = line(b1,b2) & line(b2,b1) & line(b3,b4) & line(b4,b3) (2)

These representations, although extremely simple, suffice to illustrate most of the essential strengths and weaknesses of structural representations. Each term such as line(a1,a2) is an instantiated form (or instance) of the variable form line(x,y); the object names such as a1, a2, and a3 which name nodes (termini) in the line drawings are called parameters because, although they are equivalent to constants in the predicate calculus, only the alphabetic equality or inequality of two parameters is relevant to pattern description and recognition (Hayes-Roth, 1974d). If each parameter of A_1 or B_1 is considered to be a universally quantified variable, the associated quantified conjuctive variable form is a structural template which may be used for pattern recognition. Specifically, the structural representation S of some stimulus matches the structural representation T of some template if there is some correspondence between the parameters of S and T which insure that every form in the template T is also present in S. Symbolically, S matches T if there is a one-one (1-1) mapping

f (a parameter binding function) from the set of parameters of T, PS(T), to that of S, PS(S), (i.e., f(1-1): PS(T) → PS(S)) such that every form in T also occurs in S if the alphabetic differences between bound parameters are ignored (i.e., if it is assumed that (∀t ∈ PS(T)) t = f(t)) (Barrow, et al., 1972; Hayes-Roth, 1973, 1974b). When S matches T under f, this is denoted $S(*)_f T$ or simply S(*)T. Furthermore, any representation T which is matched by S is called an abstraction (subpattern, subrepresentation) of S. If both S(*)T and T(*)S, S = T.

Notice, however, that B_1, the representation of two lines, is not an abstraction of A_1, the representation of a triangle. This is because the only way the two lines of B can be placed in respective correspondences with two lines of A is if two distinct nodes in B are forced into correspondence with a single node in A. Only if some many-to-one binding function such as f' = {(b1,a1), (b2,a2), (b3,a3), (b4,a3)} were permissible, would it be true that $A_1(*)_{f'}B_1$. Furthermore, the reader should note that it is not obvious on a priori grounds whether A_1 should match B_1 under these circumstances. For example, if unrestricted many-to-one bindings are to be allowed, it would follow that a stimulus containing one line would match any template containing any number and any pattern of lines. A proposed solution to this multiple parameter correspondence problem will be momentarily deferred while an alternative representation scheme is introduced which facilitates the exposition of a second problem, that of implicit predicates.

A parameterized structural representation (PSR) Q is a two-tuple of the sort (PS(Q), body(Q)) where PS(Q) is the set of parameters used to refer to objects in Q and body(Q) is a set of M relations r_1, ..., r_M which correspond to the M predicate forms in an equivalent predicate calculus structural representation. For example, the PSRs A_2 and B_2 are representationally equivalent to the representations A_1 and B_1, respectively:

A_2 = ({a1,a2,a3},
{{p:line, node:a1, node:a2},
{p:line, node:a2, node:a3},
{p:line, node:a1, node:a3}}) (3)

B_2 = ({b1,b2,b3,b4},
{{p:line, node:b1, node:b2},
{p:line, node:b3, node:b4}}). (4)

Each instance of an n-th order predicate in a predicate calculus conjunctive form is represented by one predicate item of the form p:predicate and n object items of the form object $type_i$:object $value_i$ (i = 1, ..., n). Here, because line is a binary symmetric predicate, each of its two objects is of the same type, namely of type node. More details on PSRs and their comparative advantages can be found in Hayes-Roth (1974c, Ap. IV). Because they are

equivalent to the conjunctive variable forms of predicate calculus, they are only introduced here to facilitate the exposition. Note that in the framework of PSRs, $S(*)_f T$ <=> f(body(T)) $\subset$ body(S), where f(body(T)) is defined to be the body of T with every occurrence of each parameter $t \in PS(T)$ replaced by f(t), its correspondent under f in PS(S).

The foregoing definition of an abstraction as a subrepresentation which is contained by some "larger" representation fails, however, to capture the intuitive idea that each n-ary relation such as r_1 = {p:line, node:a1, node:a2} actually represents (is equivalent to) a set of predicate relations which are implicit in (implied by) it. For example, any PSR containing r_1 surely contains at least two distinct nodes and thus should match any template which asserts propositions about the existence of one or two nodes whether or not it also asserts the existence of a line joining both of them. One principal weakness of all previously proposed structural representations is just this failure to represent the fact that the n-ary relation r = {p:q, obj_1:x_1, ..., obj_n:x_n} should match any m-ary relation r' = {p:q, obj_{i_1}:x_{i_1}, ..., obj_{i_m}:x_{i_m}} where $\{i_1, ..., i_m\} \subset \{1, ..., n\}$. Simply stated, it is desirable that r(*)r' if r and r' are relations and r' is contained in r. For example, one would desire that both $A_2(*)C$ and $B_2(*)C$, if C represented "the dot and the line":

C = ({c1,c2,c3},
{{p:line, node:c1, node:c2},
{p:line, node:c3}}). (5)

Both the multiple parameter correspondence and the implicit relation problems have natural solutions if each PSR Q is converted before matching to an equivalent uniform representation U(Q) = (PS(U(Q)), body(U(Q))), as follows:

Step 1: Each object reference (every occurrence of a parameter) in body(Q) is replaced by a new, unique parameter symbol. The set of these symbols is the parameter set of U(Q), PS(U(Q)).

Step 2: Each object item type:x in a relation with predicate item p:q is used to generate a unary uniform relation {(q,type),x'} in the body of U(Q), where x' is the unique parameter in PS(U(Q)) generated from this occurrence of x in body(Q). The two-tuple (q,type) is called a property of x' and the predicate of the unary relation.

Step 3: If two distinct parameters, x' and x" in PS(U(Q)) were both generated from a single parameter x in PS(Q) and the intention of the PSR is to require that both x' and x" must be assigned a single correspondent in any matching pattern, the binary uniform same parameter (SP) relation {SP, x', x"} is added to body(U(Q)).

Step 4: Similarly, if two distinct parameters y' and z' in PS(U(Q)) were generated from two different parameters, y and z, in PS(Q) and it is the intention that any matching pattern must assign a distinct parameter correspondent to each of these, the binary uniform different parameter (DP) relation {DP, y', z'} is added to body(U(Q)).

Step 5: For each pair of parameters y' and z' in PS(U(Q)) which were generated from two object items occurring in the same relation in body(Q), the binary uniform same set (SS) relation {SS, y', z'} is added to body(U(Q)).

If these five steps are applied to the PSRs A_2 (3) and B_2 (4), the following uniform representations are obtained:

$$
\begin{aligned}
A_3 = U(A_2) = (&\{a1',a1'',a2',a2'',a3',a3''\},\\
&\{\{(line,node),a1'\}, \{(line,node),a1''\},\\
&\{(line,node),a2'\}, \{(line,node),a2''\},\\
&\{(line,node),a3'\}, \{(line,node),a3''\},\\
&\{SS,a1',a2'\}, \{SS,a2'',a3'\}, \{SS,a3'',a1''\},\\
&\{SP,a1',a1''\}, \{SP,a2',a2''\}, \{SP,a3',a3''\}\} \cup\\
&\{\{DP,x_1,x_2\} : (\forall i=1,2)\ x_i \in \{a1',a1'',a2',a2'',a3',a3''\}\\
&\quad \& \{SP,x_1,x_2\} \notin body(A_3)\})
\end{aligned}
\tag{6}
$$

$$
\begin{aligned}
B_3 = U(B_2) = (&\{b1',b2',b3',b4'\},\\
&\{\{(line,node),b1'\}, \{(line,node),b2'\},\\
&\{(line,node),b3'\}, \{(line,node),b4'\},\\
&\{SS,b1',b2'\}, \{SS,b3',b4'\},\\
&\{DP,b1',b2'\}, \{DP,b1',b3'\}, \{DP,b1',b4'\},\\
&\{DP,b2',b3'\}, \{DP,b2',b4'\},\{DP,b3',b4'\}\}).
\end{aligned}
\tag{7}
$$

U(B2), of course, represents unambiguously two disjoint lines comprising four distinct nodes. If the actual intention for B in Figure 1 had been to represent two distinct but not necessarily disjoint lines, the appropriate uniform representation would be:

$$
\begin{aligned}
B_3' = (&\{b1',b2',b3',b4'\},\\
&\{\{(line,node),b1'\}, \{(line,node),b2'\},\\
&\{(line,node),b3'\}, \{(line,node),b4'\},\\
&\{SS,b1',b2'\}, \{SS,b3',b4'\},\\
&\{DP,b1',b2'\}, \{DP,b1',b3'\}, \{DP,b1',b4'\},\\
&\{DP,b2',b3'\}, \{DP,b3',b4'\}\}).
\end{aligned}
\tag{8}
$$

While it is still true that $(\nexists f)\ A_3(*)_f B_3$, nevertheless under the binding rule

$$f = \{(b1',a1'), (b2',a2'), (b3',a3'), (b4',a2'')\} \tag{9}$$

it can be seen that $A_3(*)_f B_3'$, because f(body(B_3')) $\subset$ body(A_3); i.e., B_3' = "two lines which may share at most one node in common" is a subpattern of A_3 = "a triangle."

Given any uniform PSR Q' = (PS(Q'), body(Q')) of a PSR Q, the corresponding uniform graph G(Q) = (X_Q,A_Q,P_Q) is produced in a straightforward way. The node set X_Q = PS(Q'). The arc (edge) set A_Q = $\{\{h,y',z'\} : \{h,y',z'\} \in$ body(Q') & $h \in \{SS,SP,DP\}$ & $y' \in X_Q$ & $z' \in X_Q\}$. Finally, the property set P_Q = $\{P_Q(y') : y' \in X_Q$ & $(q,type) \in P_Q(y')$ if and only if $\{(q,type),y'\} \in$ body(Q')$\}$. The interpretation of a uniform graph is as follows. Each original reference to an object y in a PSR relation, such as {p:q, type:y, ...}, results in a distinct node y' in the graph, and y' has a set of properties $P_Q(y')$ attached to it such that the property $(q,type) \in P_Q(y')$. Furthermore, each binary uniform relation of type $h \in \{SS,SP,DP\}$ between parameters y' and z' is reflected by an undirected arc connecting nodes y' and z' which is labelled h. If two PSRs S and T are such that U(S) $(*)_f$ U(T), it follows (Hayes-Roth, 1974c) that G(T) = (X_T,A_T,P_T) is a subgraph of G(S) = (X_S,A_S,P_S), meaning that: $(\forall t \in X_T)\ (\forall t' \in X_T)$ $f(t) \in X_S$ & $P_T(t) \subset P_S(f(t))$ & $[\{h,t,t'\} \in A_T] \Rightarrow [\{h, f(t),f(t')\} \in A_S]$. As an illustration, the two uniform graphs for A_3 and B_3' are shown superimposed in Fig. 2 according to the node bindings specified by f (9). Thus a uniform graph G(T) is a subgraph of another, G(S), if there is some 1-1 node binding function f for which it is true that all node properties and labelled edges in G(T) are also in G(S). This is denoted $G(T) \subset *_f G(S)$ or simply $G(T) \subset * G(S)$.

Because most of this paper is addressed to questions concerning the identification of rules of grammar (productions) through the extraction of abstractions of training exemplars, it is necessary to extend the notions of PSR, uniform PSR, and uniform graph to the representation of productions. Define a rule (production, transformation) as any formula of the form $F = [(\forall x_1, ...,x_n)\ C(x_1,...,x_n) \Rightarrow R(x_1,...,x_n)]$ where $C(x_1,...,x_n)$ (the contingency) and $R(x_1,...,x_n)$ (the response) of the rule F are both conjunctive products of variable forms over the variables $x_1, ..., x_n$. Thus, the rule F is like a production of a web grammar C' $\rightarrow$ R', where the conjuncts of $C(x_1,...,x_n)$ are represented by a web (graph) C' and those of $R(x_1,...,x_n)$ by the web R'. Specifically, the formalism of Pfaltz & Rosenfeld can be immediately generalized to allow such representations of rules by permitting labels on arcs, properties on nodes, and null embeddings. Then, the rule F can be represented by letting C' and R' be the uniform graphs of the PSR equivalents of $C(x_1,...,x_n)$ and $R(x_1,...,x_n)$, respectively.

It is possible, moreover, to obtain a single uniform representation of any rule F, and such a representation will greatly simplify the problem of abstracting rules from examples. Given F, it is clear from the foregoing how the uniform PSRs C' = (PS(C'),body(C')) and R' = (PS(R'),body(R')) corresponding to the conjunctive forms $C(x_1,...,x_n)$ and $R(x_1,...,x_n)$ are generated. Notice though that the parameter sets PS(C') and PS(R') are necessarily disjoint because every occurrence of each parameter x_i in $C(x_1,...,x_n)$ and $R(x_1,...,x_n)$ has been replaced

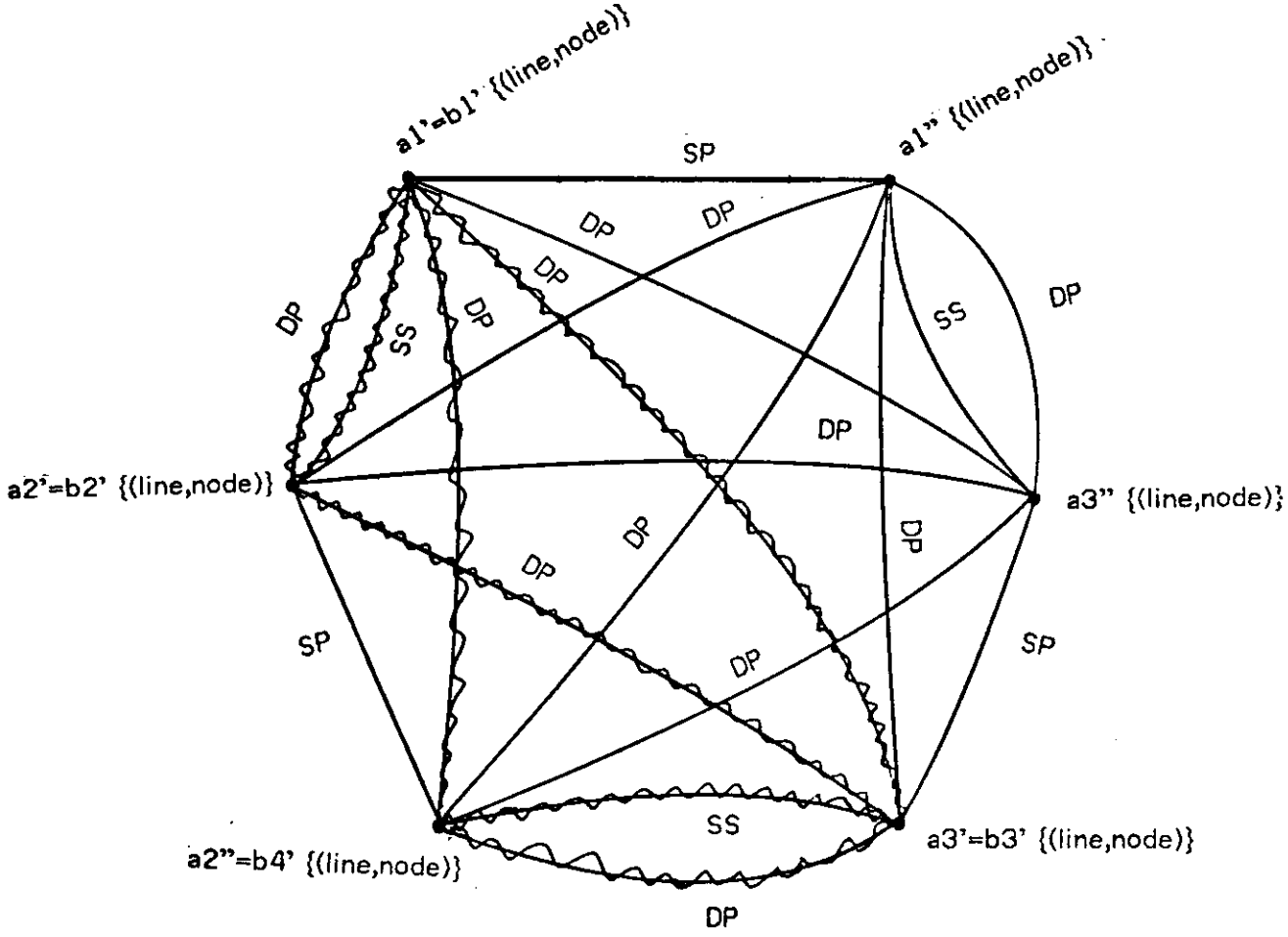
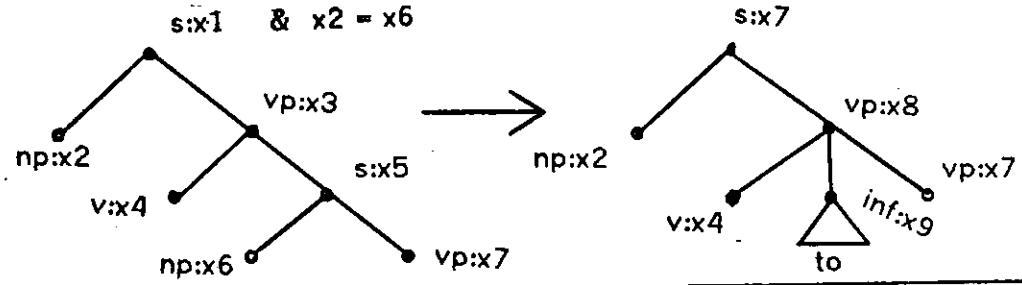


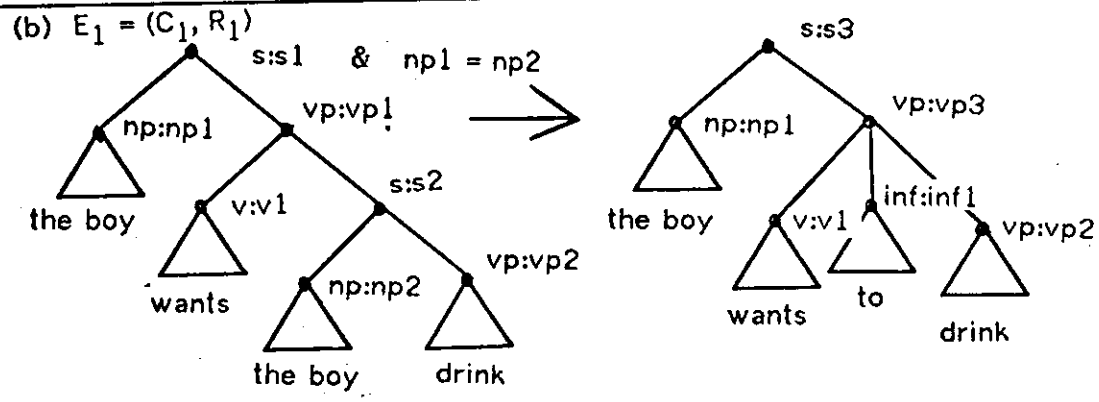
Figure 2. Two superimposed uniform graphs for a triangle ($G(A_3)$) and two lines possibly joined at one node($G(B_3')$)under the binding function f of equation (9). The arcs of $G(A_3)$ are drawn with smooth lines, while those of $G(B_3')$ are drawn with wavy lines.

by a new, unique parameter (see Step 1 of the procedure to generate uniform PSRs). Thus, a need exists for additional SP relations between any y' ∈ PS(C') and z' ∈ PS(R') where y' and z' are parameters which were generated from the same original parameter x_i occurring once in $C(x_1,...,x_n)$ and once in $R(x_1,...,x_n)$. Simply, the uniform PSR of the rule F is defined as U(F) = (PS(C') ∪ PS(R'), body(C') ∪ body(R') ∪ {{SP,y',z'} : ($\exists x_i$) y' is a parameter in C' and z' is a parameter in R' which replaced one occurrence of x_i in $C(x_1,...,x_n)$ and $R(x_1,...,x_n)$, respectively} ∪ {{"C",y'} : y' ∈ PS(C')} ∪ {{"R",z'} : z' ∈ PS(R')}. Only the last two sets in the union which constitutes the body of U(F) need explanation. Basically, these sets assign the additional property "C" to each "contingency" parameter y' ∈ PS(C') and "R" to each response parameter z' ∈ PS(R') and serve to distinguish the contingency and response parts of the integrated rule representation.

As an illustration of such a rule F, consider the equi-noun phrase deletion (END) rule of transformational grammar (TG) (Chomsky, 1967; Langendoen, 1969) shown in panel (a) of Figure 3 with two examples,

(a) $F = [(\forall x1,\ldots,x9)\ C(x1,\ldots,x9) \Rightarrow R(x1,\ldots,x7)]$





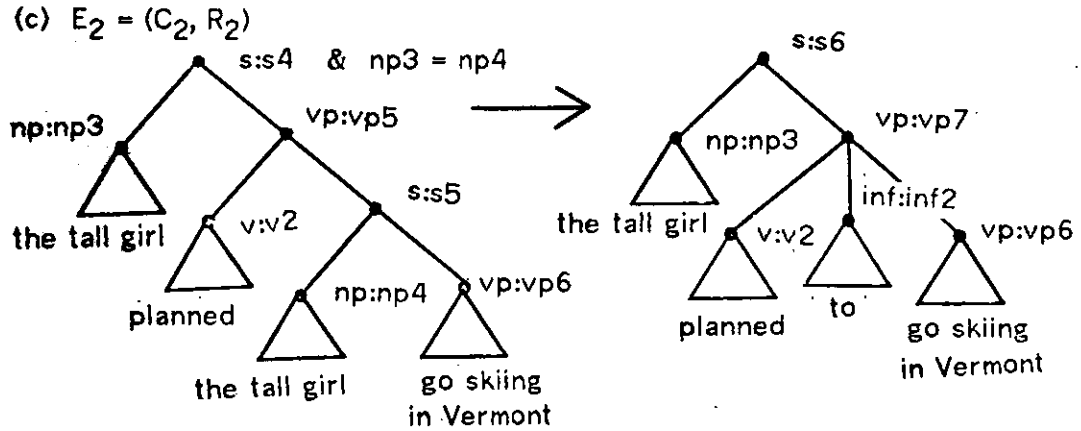


Figure 3. Two examples, $E_1 = (C_1,R_1)$ = ("The boy wants [that] the boy drink", "The boy wants to drink") and $E_2 = (C_2,R_2)$ = ("The tall girl planned [that] the tall girl go skiing in Vermont", "The tall girl planned to go skiing in

Vermont"), in panels (b) and (c), of the equi-noun phrase deletion rule F in panel (a).

$E_1 = (C_1,R_1)$ and $E_2 = (C_2,R_2)$, related by F shown in panels (b) and (c). Non-uniform PSR equivalents C', R', C_1', R_1', C_2', and R_2' corresponding to $C(x_1,...,x_n)$, $R(x_1,...,x_n)$, C_1, R_1, C_2, and R_2, respectively, are given below:

```
C' = ({x1,x2,x3,x4,x5,x6,x7,x8,x9},
        {{p:s(np,vp), name:x1, np:x2, vp:x3},
         {p:vp(v,s), name:x3, v:x4, s:x5},
         {p:s(np,vp), name:x5, np:x6, vp:x7},
         {p:equal, what:x2, what:x6}})                    (10)

R' = ({x1,x2,x3,x4,x5,x6,x7,x8,x9},
        {{p:s(np,vp), name:x7, np:x2, vp:x8},
         {p:vp(v,inf,vp), name:x8, v:x4, inf:x9, vp:x7},
         {p:"to", name:x9}})                              (11)

C1' = ({s1,np1,vp1,v1,s2,np2,vp2},
        {{p:s(np,vp), name:s1, np:np1, vp:vp1},
         {p:"the boy", name:np1},
         {p:vp(v,s), name:vp1, v:v1,s:s2},
         {p:"wants", name:v1},
         {p:s(np,vp), name:s2, np:np2, vp:vp2},
         {p:"the boy", name:np2},
         {p:"drink", name:vp2},
         {p:equal, what:np1, what:np2}})                  (12)

R1' = ({s3,np1,vp3,v1,inf1,vp2},
        {{p:s(np,vp), name:s3, np:np1, vp:vp3},
         {p:"the boy", name:np1},
         {p:vp(v,inf,vp), name:vp3, v:v1, inf:inf1, vp:vp2},
         {p:"wants", name:v1},
         {p:"to", name:inf1},
         {p:"drink", name:vp2}})                          (13)

C2' = ({s4,np3,vp5,v2,s5,np4,vp6},
        {{p:s(np,vp), name:s4, np:np3, vp:vp5},
         {p:"the tall girl", name:np3},
         {p:vp(v,s), name:vp5, v:v2, s:s5},
         {p:"planned", name:v2},
```

{p:s(np,vp), name:s5, np:np4, vp:vp6},
{p:"the tall girl", name:np4},
{p:"go skiing in Vermont", name:vp6},
{p:equal, what:np3, what:np4}}) (14)

R_2' = ({s6,np3,vp7,v2,inf2,vp6},
{{p:s(np,vp), name:s6, np:np3, vp:vp7},
{p:"the tall girl", name:np3},
{p:vp(v,inf,vp), name:vp7, v:v2, inf:inf2, vp:vp6},
{p:"planned", name:v2},
{p:"to", name:inf2},
{p:"go skiing in Vermont", name:vp6}}). (15)

Figure 4 illustrates the uniform graph G(F) for the rule F derived by converting C' (10) and R' (11) into uniform PSRs, combining these, and supplementing the union of their bodies by all appropriate SP relations establishing equivalences between parameters (e.g. x2' and x2"" for the parameter x2) occurring both as a contingency and a response variable.

Before proceeding to the next section, the reader should be convinced that the graph G(F) in Fig. 4 is a satisfactory (equivalent) representation of the rule F and, moreover, that the graphs $G(F_1)$ and $G(F_2)$ corresponding to the rules F_1 = [C_1'(s1,np1,vp1,v1,s2,np2,vp2,s3,vp3,inf1) => R_1'(s1,np1,vp1,v1,s2,np2,vp2,s3,vp3,inf1)] and F_2 = [C_2'(s4,np3,vp5,v2,s5,np4,vp6,s6,vp7,inf2) => R_2'(s4,np3,vp5,v2,s5,np4,vp6,s6,vp7,inf2)] satisfy the subgraph relations: G(F) $\subset^*$ $G(F_1)$ & G(F) $\subset^*$ $G(F_2)$. The fact that G(F) is a subgraph both of $G(F_1)$ and $G(F_2)$ or, equivalently, that F is a subrule (abstraction) of both F_1 and F_2 is the basis for the plausible inference that F is the common rule manifested by E_1 = (C_1,R_1) and E_2 = (C_2,R_2). This inference technique is fully clarified and exploited in solving general induction problems in the subsequent sections.

3. A GENERAL LEARNING PROBLEM

The type of learning problem to be solved in this paper is formally defined as follows. Let I = {(C_1,R_1),...,(C_N,R_N)} be training information comprising N contingency-response (input-output) pairs of structured representations which manifest some (unknown) rule F = [($\forall x_1$,..., x_n) $C(x_1,...,x_n)$ => $R(x_1,...,x_n)$], where $C(x_1,...,x_n)$ and $R(x_1,...,x_n)$ are conjunctive variable forms over the variables x_1, ..., x_n. Each E_i = (C_i,R_i) manifests F in the sense that ($\forall i$ = 1, ..., N) ($\exists f_i$) C_i $(*)_{f_i}$ $C(x_1,...,x_n)$ & R_i $(*)_{f_i}$ $R(x_1,...,x_n)$. The interpretation of this condition is that although the training pairs might contain much superfluous information (relations not present in $C(x_1,...,x_n)$ or $R(x_1,...,x_n)$ and therefore not criterial), they must contain relations corresponding to each

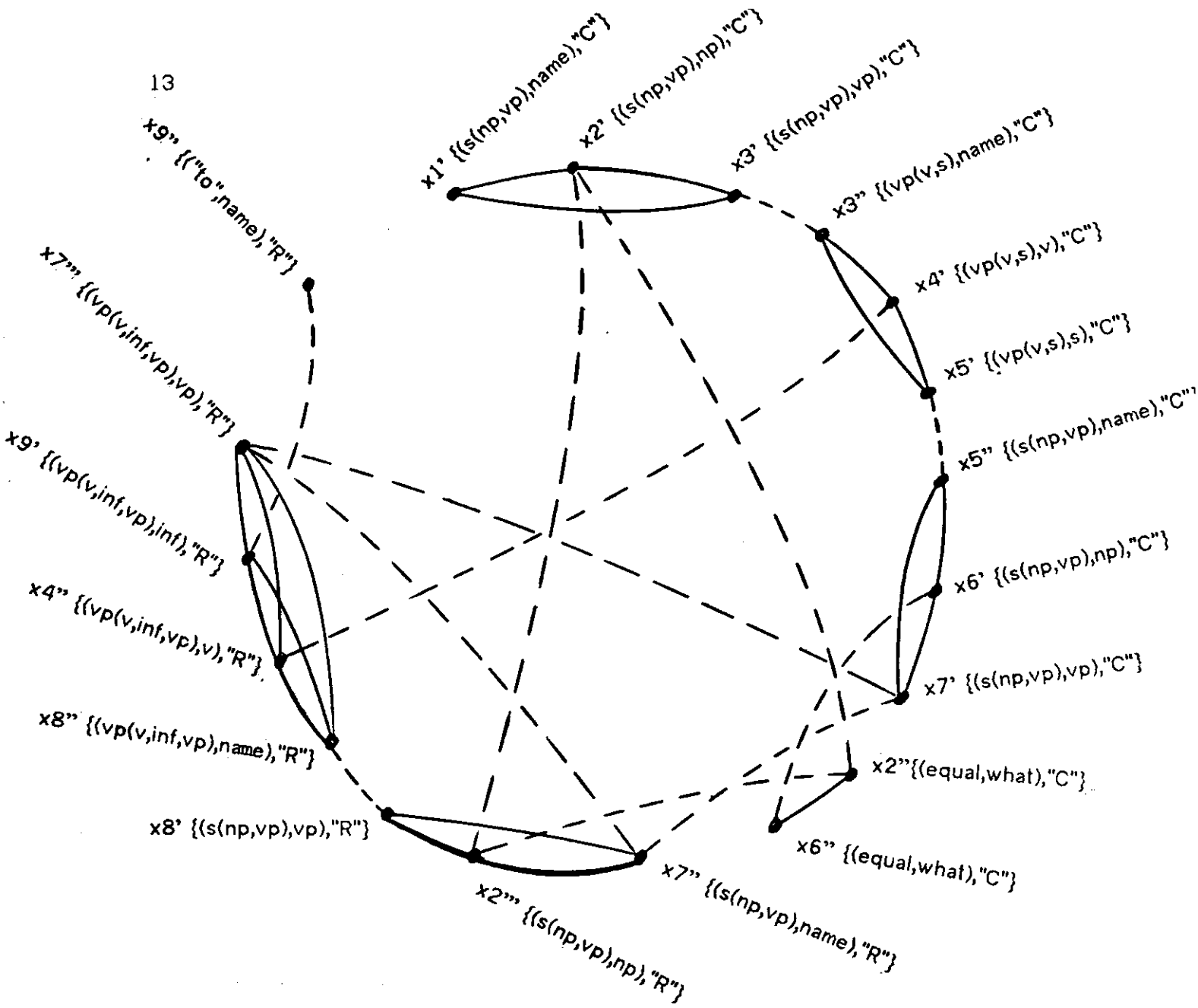


Figure 4. The uniform graph G(F) of the END rule of TG corresponding to Fig. 3 (a) and equations (10-11). Here SP arcs appear as broken lines while SS arcs appear as solid lines. No DP arcs are represented since all pairs of nodes not connected by SP arcs presumably have DP arcs between them.

criterial term in the conjunctive sets $C(x_1,...,x_n)$ and $R(x_1,...,x_n)$. The learning problem is to define an effective procedure for computing the rule F given I.

Two additional points are now made to motivate the proposed approach. First, the context in which this learning problem originally occurred was the following: Suppose a learner is provided with structural representations S_t (t=1,2,...) of the successive internal

states (a data base of semantic, syntactic, perceptual, and response-producing predicates) of some machine M to be modelled. Furthermore, suppose the behavior of M is determined by a finite set of independent productions which correspond to k rules, F_1,..., F_k of the same sort as F above. At any point in time, t, the transition from S_t to S_{t+1} may be accounted for in terms of a set of rules F(t) which, because their contingencies were satisfied (matched) by S_t, were invoked and caused specific response structures to enter S_{t+1}. In such a case (fully general, of course), an effective algorithm for identifying rules in F(t) would be adequate for the task of inducing a behavior model of M.

The second point to notice is that, while there can be no solution which is error free after any finite time to grammatical induction problems which require the inference of the relevant underlying phrase structures (Gold, 1967), the current restricted statement of the learning problem is one which does afford, with appropriate training information and acceptability criteria, certain solutions. This certainty is obtained at the expense of requiring each training input-output exemplar to manifest directly the underlying rule. Whether or not this requirement can be considered a serious weakness of the proposed approach depends principally on the power of this restricted procedure to solve important and general problems. This point is considered in some detail in the last section. At this point, however, the reader should realize that the proposed learning mechanisms are primarily relevant to problems where the space of plausible rules is implicitly determined by a predefined set of predicates all of whose relevant instances are provided directly to the learner. The learner's task--find the relevant conjunctive predicate structures--is then adequately constrained so that an effective procedure is straightforward.

Given the training information $I = \{E_i = (C_i,R_i) : i = 1, ..., N\}$, any F manifested by each E_i is, by definition, a plausible hypothetical rule underlying I. In order to further usefully constrain the space of plausible rules, one would also frequently like to exploit the idea of negative instances or counterexamples of a rule. When provided to a learner, the negative training information $I' = \{E_i' = (C_i',R_i') : i = 1, ..., N'\}$ for any F satisfies the condition: $(\forall i)\ (\exists f_i$ such that $C_i'\ (*)_{f_i}\ C(x_1,...,x_n))\ \text{not}[R_i'\ (*)_{f_i}\ R(x_1,...,x_n)]$. That is, E_i' is a counterexample of F if C_i' matches the contingency structure of F but R_i' does not match the response structure of F. An example would be the pair (C_1',R_1') where C_1' is the structural description of some input sentence which matches the contingency pattern of the END rule, but where R_1' is a related output sentence representation which fails to match the response pattern of that rule. In other papers (Hayes-Roth, 1974a, 1974c), the quantitative use of such negative information in the evaluation of alternative plausible rules is formally considered. For the present purposes however, the following assumption will necessarily suffice: given the training sets I and I' for an unknown rule F and any hypothesized rule F', the function $U(F', |I/F'|, |I'/F'|) = V(F')$ will be taken to be the performance (utility, value, plausibility) of the rule F', where V(F') necessarily

is an increasing function of the positive frequency of F' in I (|I/F'|) and a decreasing function of the frequency of negative instances of F' in I' (|I'/F'|). The functions to compute these frequencies and V(F') will be assumed to be exogenously provided. Note that, as stated above, the learning problem requires that |I/F'| = |I| = N for all hypothesized rules F'. The generalization of this case to those where |I/F'| < N is discussed in detail in Hayes-Roth (1974a) and briefly in the last section of this paper.

The three special cases of the rule learning problem are defined as follows. Case 1 - Categorical Rule Learning: The response $R(x_1,..., x_n)$ of the unknown rule F is categorical, i.e., it is unchanged regardless of variations among inputs C_i. For example, the response might be "red" or "class 2" or any constant semantic structure. Case 2 - Substitution Rule Learning Given Exogenously Determined Contingency and Response Parameter Correspondences: The rules to be learned allow for general substitutions (forced equivalences) between any contingency and response parameters, and these are manifested by SP relations supplied by the trainer. The repetition of several parameters (e.g. np1, v1, vp2) in both C_1' (12) and R_1' (13) for the training example $E_1 = (C_1,R_1)$ of the END rule is an example of such exogenously provided input-output parameter correspondences, and the rule F described by (10-11) and Figs. 3(a) and 4 is an example of such an inferrable substitution rule. Case 3 - Substitution Rule Learning Without Exogenously Supplied Input-Output Parameter Correspondences: This case refers to problems where training examples like those in the preceding case are supplied except that parameters in each C_i are necessarily distinct from those in R_i. The learner in this case must, in addition to hypothesizing rules, also hypothesize equivalences (substitutability) between contingency and response parameters. Each of these cases is treated in turn in the subsequent sections.

The remainder of this section briefly addresses the issue of feature extraction or predicate coding, the production (computation) of the predicate instances upon which inference of rules by the learner is to be based. It should be clear that actually encoding a stimulus pattern as a structural representation is a non-trivial task. Moreover the current statement of the learning problem requires every (C_i,R_i) exemplar pair to manifest the unknown rule F. Thus, if the trainer (as well as the learner) is "in the dark" as to what predicates are criterial to the rule F, the learning procedures defined herein will be guaranteed to succeed only if the trainer provides the actual values of all potentially relevant predicates and features as input to the learner. While it is generally understood how instances of unary predicates (feature values) are computable, the goodness of alternative procedures for computing instances of n-ary predicates seems to be heavily dependent upon the actual nature of the patterns involved (see, for examples, Evans, 1968; Barrrow, et al.,1972; Newell, 1972; Rulifson, et al., 1972; Hayes-Roth, 1974c; Hayes-Roth, 1974d; Hayes-Roth & Mostow, 1975). All that will be said here is that the learning procedures operate by abstracting commonalities from sufficiently

described exemplars and, as a result, all relevant predicate instances need to be supplied to the learner by the trainer. Later, in section 7, possible methods are discussed for expanding the set of predicates upon which rule learning may occur by the discovery of novel unary and n-ary predicates.

4. CASE 1: CATEGORICAL RULE LEARNING

Rules to be learned in this case are characterized by constant The solution to the general categorical rule learning problem requires three basic operations: (1) a mechanism to compute C^*, the set of plausible contingency patterns for the unknown rule F; (2) a mechanism to identify R^*, the set of all plausible response patterns; and (3) a mechanism to evaluate the performance of each plausible rule $F' = [C \Rightarrow R]$, where $C \in C^*$ and $R \in R^*$. In the current case, steps (1) and (2) may be completely separated, because in categorical rules no substitution of the name of an input object which is found to correspond to a contingency parameter is made to an associated response parameter; i.e. the contingency and response structures are independent. An example of such a learning problem would be that of abstracting the pattern characteristic "three lines and three angles" (A_2 in (3)) as the contingency of a rule whose response pattern is "triangle."

The essential step in this problem then is: Given $I = \{E_i = (C_i, R_i) : i = 1, ..., N\}$ and $I' = \{E_i' = (C_i', R_i') : i = 1, ..., N'\}$, compute $C^* = C_1 * ... * C_N$ and $R^* = R_1 * ... * R_N$, the set of abstractions $\{C_i : i = 1, ..., N\}$ and $\{R_i : i = 1, ..., N\}$, respectively. By the definition of the learning problem, each E_i manifests the unknown rule F and, therefore, $C(x_1,...,x_n)$ and $R(x_1,...,x_n)$ must correspond to common abstractions of $\{C_1,...,C_N\}$ and $\{R_1,...,R_N\}$, respectively. The operator * (star) is called the interference product or partial matching operator (Hayes-Roth, 1974a, 1974b). The set of abstractions of a set of (presumably uniform) PSRs $\{C_i : i = 1, ..., N\}$ with corresponding uniform graphs $\{G(C_i)\}$ is the set of PSRs $\{A_j : j = 1, 2, ...\}$ with graphs $\{G_j\}$ such that $(\forall j)\ [G_j \subset^* G(C_1)\ \&\ ...\ \&\ G_j \subset^* G(C_N)]$. A previously published interference matching (IM) procedure (Hayes-Roth, 1974b) effectively and efficiently computes all abstractions $\{A_j\}$ of the set $\{C_i\}$.

The IM procedure is best understood in terms of the concepts of one-one binding relations and models. A one-one correspondence binding relation B on the cross-product space $P = PS(C_1) \times ... \times PS(C_N)$ of the parameter sets of $\{C_i\}$ is defined to be any subset of P which assigns to each parameter $x_i \in PS(C_i)$ at most one correspondent x_j from any other $PS(C_j)$. Symbolically, $B \subset P$ is a one-one binding relation if $[(\forall i,j)(\forall b \in B)(\forall b' \in B)\ (b)_i = (b')_i = x_i\ \&\ (b)_j = x_j\ \&\ (b')_j = x_j'] \Rightarrow x_j = x_j'$, where $(b)_i$ is the i-th element (a_i) of the N-tuple $b = (a_1,...,a_N)$. Given any one-one binding relation B, a model M of $\{C_i : i = 1, ..., N\}$ comprises an abstraction A and a set of residuals $\{R(C_i) : i = 1, ..., N\}$, where A contains all relations common

to each C_i when the alphabetic differences between bound correspondent parameters (all x_i, x_j such that $(\exists b \in B)\ (b)_i = x_i\ \&\ (b)_j = x_j$) are ignored and $R(C_i)$ contains all relations in the body of C_i which are not accounted for in A. What the IM procedure does is efficiently enumerate all distinct binding relations and models which entail a non-null abstraction.

In short, the IM procedure is effective for the task of computing all possible abstractions of any set of uniform PSRs, including C^* and R^*. Thus, each rule $F' = [C \Rightarrow R]$, where $C \in C^*$ and $R \in R^*$, which is a plausible solution to this induction problem may be effectively computed. Of course, the performance measure V induces an ordering on the plausible rules: F' is preferred to F" if V(F') > V(F"). On the other hand, it is clear that at any time t during the training program, a less preferred rule F" may be identical to the unknown correct rule, and only additional counterexamples to momentarily more preferred rules F' will tend to eliminate them by reducing their plausibilities. A program for performing the computations required for categorical rule learning in limited space (and time) exists and is described in detail in Hayes-Roth (1974a).

One of the advantages of uniform representations, in addition to their providing assurance that the IM procedure will be effective for categorical rule learning (formally proved in Hayes-Roth, 1974c), is that all possible rules $F' = [C \Rightarrow R]$ may be computed at the same time if the abstraction operator (*) is applied directly to the uniform representations $U(D_i)$ of the rules $D_i = [C_i \Rightarrow R_i]$ (i=1,...,N) representing causal inferences that each training exemplar contingency pattern (prior event) caused or predicts the related response pattern (subsequent event). In the rules D_i, there are no substitution SP relations connecting any of the parameters of C_i with those of R_i, because there are no substitutions in categorical rules.

If the training information I is simply converted into the uniform rule representations $\{U(D_i)\}$, it follows that the unknown rule F which is manifested by each $E_i \in I$ is an abstraction (subrule) of each D_i, i.e., the uniform representation $U(F) \in U(C_1 \Rightarrow R_1) * \ldots * U(C_N \Rightarrow R_N)$. While the ability to associate each plausible rule F' with a particular element of the set of abstractions

$$I^* = U(D_1) * \ldots * U(D_N) \tag{16}$$

and to assert that

$$(\exists F' \in I^*)\ F' = F \tag{17}$$

may seem to be of only incidental interest for rule learning of the first case, under certain interpretations the truth of equations (16-17) holds for each of the other cases too and, thus, these equations can be considered a complete solution to the learning problem. Because case 2 can be considered to subsume case 1 and because of the generality of (16-17), a more detailed illustration of rule learning will be deferred until the next section.

5. CASE 2: SUBSTITUTION RULE LEARNING GIVEN INPUT-OUTPUT PARAMETER CORRESPONDENCES

Rules to be learned in this case reflect the substitution of arguments from contingency to response patterns by the use of the same parameter in both parts of the rule. In the current case, the training information I will necessarily exhibit the repetition of the same parameters in C_i and R_i wherever the same object occurs in both the input and output patterns. For example, each training exemplar (C_i, R_i) for learning the TG END rule F (Fig. 3) would necessarily repeat the parameter symbols (such as np1, v1, vp2 in C_1 and R_1) which name the noun phrases, verbs, and verb phrases, respectively, corresponding to the phrase structures named x2, x4, and x7 in F which are substituted from C_i to R_i when F is applied. Thus, the training set $I = \{E_1, E_2\}$ comprising the representations (12-15) for the sentences in panels b and c of Fig. 3 manifest F in the way required for case 2.

On the other hand, it is useful to consider the training information for case 2 rule learning problems to be redefined as $I = \{E_i = (C_i,R_i,Q_i)\}$ where C_i and R_i are uniform PSRs whose parameter sets are completely disjoint but where Q_i is a set of exogenously provided substitution SP relations, such that Q_i contains one relation {SP, x', x"} for every contingency parameter x' from C_i and response parameter x" from R_i for which it is true that x' and x" name the same object occurring both in C_i and R_i. Under this condition, each exemplar E_i in I must manifest F in the following sense: There must exist a one-one parameter binding function f_i from the parameters of the uniform representation U(F) to those of $U(C_i{=>}R_i)$ such that every relation in U(F) occurs in $U(C_i{=>}R_i)$ or Q_i if the alphabetic differences between bound correspondent parameters x and f(x) is ignored. That is, if $U(D_i)$ is defined to be

$$U(D_i) = (PS(C_i) \cup PS(R_i), body(U(C_i{=>}R_i)) \cup Q_i) \tag{18}$$

E_i manifests F if and only if $(\exists f_i(1\text{-}1) : PS(U(F)) \rightarrow PS(U(D_i)))\ (\forall r_j \in body(U(F)))\ f_i(r_j) \in body(U(D_i))$, where $f_i(r_j)$ is the relation r_j with each parameter x replaced by $f_i(x)$.

As a result of the fact that each $U(D_i)$ contains U(F) as an abstraction, the same techniques that were applicable for case 1 (16-17) are equally applicable for the current case. However, while it is true that $(\exists F' \in H^* = U(D_1) * \ldots * U(D_N))\ F' = F$, it is not true that every F' $\in H^*$ is a <u>well-formed</u> rule. Thus, while H^* may be computed by the IM (or other subgraph extraction) procedure, the set I^* of plausible well-formed rules is a particular subset of H^*. A hypothetical rule F' $\in H^*$ with uniform representation U(F') is <u>well-formed</u> if every set A of <u>SP-connected</u> response parameters is <u>SP-connected</u> to at most one set B of <u>SP-connected</u> contingency parameters and no parameter in A is <u>SP-connected</u> to a contingency parameter which is not in B. Two parameters, x and y, are <u>SP-connected</u> in U(F') if {SP,x,y} $\in$ body(U(F')); a set of parameters is <u>SP-connected</u> if every pair of parameters in the set is SP-connected; two sets are <u>SP-connected</u> if their union is SP-connected.

Thus, the solution to the case 2 learning problem is provided by the following program:

Step 1: Compute $U(D_i)$ by adding the set Q_i of substitution SP relations to the body of $U(C_i \Rightarrow R_i)$.

Step 2: Compute the set of plausible subrules $H^* = U(D_1) * \ldots * U(D_N)$ by the IM procedure (or other subgraph extraction procedure).

Step 3: Compute the set of all plausible and well-formed rules $I^* = \{U(F') : U(F') \subset H^* \ \& \ F'$ is a well-formed rule$\}$. Each such rule $F' \subset I^*$ is consistent with the training information I and since I manifests F, $(\exists\, U(F') \subset I^*)\ F' = F$. A simple, formal proof of the effectiveness of such a program is provided elsewhere (Hayes-Roth, 1974c).

As an example, consider again the illustration in Fig. 3 of the END rule F of TG and the examples E_1 and E_2 of its application. The exogenously determined input-output parameter correspondences Q_i appear as identities between some parameters (e.g. np1) which occur both in C_i and R_i. Thus,

$$(\forall i=1,2)\ U(D_i) = U(C_i \Rightarrow R_i)\ (*)_{f_i}\ U(F), \tag{19}$$

where

$$f_1 = \{(x1,s1), (x2,np1), (x3,vp1), (x4,v1), (x5,s2), (x6,np2), (x7,vp2), (x8,vp3), (x9,inf1)\} \tag{20}$$

and

$$f_2 = \{(x1,s4), (x2,np3), (x3,vp5), (x4,v2), (x5,s5), (x6,np4), (x7,vp6), (x8,vp7), (x9,inf2)\}. \tag{21}$$

In other words, if the IM procedure were applied to the set $\{U(D_1), U(D_2)\}$, one abstraction that would be produced would be F, under the parameter correspondences in the one-one relation $B = \{(s1,s4), (np1,np3), (vp1,vp5), (v1,v2), (s2,s5), (np2,np4), (vp2,vp6), (vp3,vp7), (inf1,inf2)\} \subset PS(D_1) \times PS(D_2)$ (or the equivalent but more numerous 1-1 correspondences which would obtain among the appropriately expanded parameter sets of the uniform representations $U(D_1)$ and $U(D_2)$).

Because of the relatively weak nature of the constraints on plausible rules, F will in all probability be just one of many plausible and well-formed rules which are thus generated. Examples of reasonable (but heuristic) constraints which might be exploited successfully in some task environments are: (1) "Consider any F' which has one or more counterexamples to be implausible" or (2) "Find only the one rule F' which has the maximal similarity to (the most relations in common with) each $U(D_i)$." In general, however, the practical application of learning procedures will be in real problem contexts where neither of these heuristics will produce

consistently desirable effects. Further discussion of the nature of and selection among probabilistically valid rules can be found in Hayes-Roth (1974a, 1974c). For the present purposes, all considerations related to the relative desirabilities of alternative rules with various combinations of positive and negative support are relegated to the "exogenously provided" performance measure V.

To conclude this section, it seems desirable to characterize briefly the type of environment in which problems of case 2 (and the subsumed case 1) will occur. The essential character of case 2 learning problems is that training information completely describes the correspondences (identities) between objects in input-output pattern pairs. Such information is frequently known in the sorts of problems to which one would like to apply practical abstraction procedures. Several of these problems are considered in detail in Section 8. For the present, however, a useful test which may be employed to decide if a particular learning problem falls into case 2 and can be solved by the program above is this: If an unknown rule F is operative in transforming one configuration of a specific set of identifiable objects into another and if the identities of the objects are known, the rule may be found by the methods of case 2. All of the rules of TG (Langendoen, 1969), for example, are thus inferrable from training examples like those in Fig. 3. To date, it is the author's experience that every problem of practical interest satisfies the preceding test. Nonetheless, it is conceivable that other researchers may encounter problems where the identities of objects are not known. For example, one might wish to induce the nature of a rule which presumably accounts for the transformation of each of several (input) chemical structures into corresponding (output) structures, where the identities of every one of numerous hydrogen atoms in both structures are confusable with one another. To accommodate such possibilities, the solution of problems of this sort is considered in the next section.

6. CASE 3: SUBSTITUTION RULE LEARNING WITHOUT EXOGENOUSLY SUPPLIED INPUT-OUTPUT PARAMETER CORRESPONDENCES

Rules to be learned in this case are of the same sort as in the previous case but less information is provided to the learner. Rule learning problems of the third case are the most general and, in terms of the amount of necessary computation, potentially the most difficult. An example of such unrestricted rule learning would be the problem of inducing the END rule in Fig. 3 (a) from the examples in panels (b) and (c) if each parameter q in C_i and r in R_i had been distinctively renamed C_i.q and R_i.r, respectively. That is, any correspondences between contingency and response parameters would necessarily be inferred by the learner at the same moment at which it inferred the contingency and response patterns of the unknown rule F.

One possible brute-force solution to the problem would be as follows. Let $U(D_i) = U(C_i \Rightarrow R_i)$ and compute I^* as in (16). Then for any abstracted rule F', the learner is free to add any desired substitution SP relations which establish contingency-response parameter correspondences as long as the modified rule remains well-formed. Such a procedure, if exhaustively applied, will surely identify F.

On the other hand, the brute-force method fails to exploit the potentially useful structural redundancies between each C_i and R_i. A structural redundancy between C_i and R_i is a common abstraction (substructure) of C_i and R_i. In general, such redundancies are highly indicative of substitutability of contingency and response parameters. For example, consider the sentences C_1 and R_1 in Fig. 3 (b) and the corresponding PSRs C_1' (12) and R_1' (13). Again, remember that in the current case all parameter symbols have been prefixed by "C_1." or "R_1." and, therefore, all response and contingency parameters are distinct.

Now, consider the set of abstractions $E_1^* = C_1 * R_1$ and, in particular, the abstraction $A \in E_1^*$ associated with the correspondence bindings in $B = \{b_1 = (C_1.np1, R_1.np1), b_2 = (C_1.v1, R_1.v1), b_3 = (C_1.vp2, R_1.vp2)\}$ where

$$A = (\{b_1, b_2, b_3\}, \{\{p:\text{"the boy"}, name:b_1\}, \{p:\text{"wants"}, name:b_2\}, \{p:\text{"drink"}, name:b_3\}\}). \quad (22)$$

The abstraction A identifies plausible loci of substitutions in F by locating phrase structures which are repeated in both C_1 and R_1. Thus, it would be reasonable to hypothesize substitution SP relations for each set of related bindings in B; i.e. one could let $Q_1 = \{\{SP, C_1.x, R_1.y\} : (C_1.x, R_1.y) \in B\}$ and then proceed to induce F as if this were a problem in case 2.

Such an idea must be generalized, however, before this suggested procedure constitutes a guaranteed solution to case 3 learning problems. The central question here is how to enumerate all possible ways in which substitutions might have occurred between C_i and R_i, and an answer to this question requires a discussion of the semantics of substitutability, which now follows. In any given problem context, the structural representations used will permit interpretations of substitutability between contingency and response parameters in possibly varying and numerous ways. In TG rules of the sort with which this paper has been concerned, a substitution between $C_i.q$ and $R_i.r$ is plausible if the entire phrase tree structures descendant from root nodes $C_i.q$ and $R_i.r$ are identical. Thus any procedure, including the IM procedure, which can identify such equivalences is a satisfactory mechanism for enumerating plausible contingency-response parameter substitutions. On the other hand, the semantics of substitutability may be different in other knowledge representations (other systems of predicates) or other problem domains.

In each case, however, an SP relation S = {SP, C_i.q, R_j.r} is called plausible if the PSR T with body body(U(C_i=>R_j)) ∪ {S} is not impossible in the sense that it represents the simultaneous attribution of two mutually exclusive attributes (category elements) to the same object (a pair of SP-connected parameters). In the case of TG rule learning for example, every relation of the sort {p:"...", name:q} which associates a specific word sequence with the object named q is a mutually exclusive possibility; i.e. any parameter can name at most one distinct word sequence. Similarly, at a higher level of abstraction, no parameter q which occurs in the object item name:q of a noun phrase relation can simultaneously occur as the name of any other phrase type (or, for that matter, in the object item name:q of any relation whose predicate item is different).

In the previously mentioned example of inducing a rule which transforms one chemical structure into another, no SP relation is plausible which identifies a contingency parameter identifying a carbon atom with a response naming a hydrogen atom. On the other hand, each distinct SP relation S = {SP, C_i.q, R_j.r} where C_i.q and R_j.r are identical chemical elements or compounds is plausible.

It is apparent, however, that the plausibility of hypothetical substitutions spans a range of values between the extremes of "totally plausible" and "totally implausible." For example, objects may change in shape, color, structure, or other attributes under some transformations, and while these attributes may for some purposes constitute categories of exclusive possibilities, it does not follow that such parameter substitutions are impossible, only, in general, that they are apparently less likely than others. The behaviors of a magician, for example, are frequently of just such an improbable sort: {{p:red, name:x}, {p:handkerchief, name:x}} → {{p:green, name:x}, {p:scarf, name:x}}. Thus, while one might reasonably hope to infer from "x is red at time t" and "y is green at t+1" that x and y are different objects, this is only probably valid.

The semantics of substitutability is thus seen to be dependent on what is known about the problem domain, including especially the sorts of inferences that may be correctly made about the probable identicalness of two objects occurring in different structural representations. Thus, while the learning procedures for cases 1 and 2 and the proposed brute-force method for case 3 were entirely syntactic in nature (were explainable without reference to reality), any efficient solutions to case 3 problems which significantly reduce the set of plausible substitution SP relations which are considered must necessarily exploit what is known about the likelihood that two differently named objects having attributes which are more or less similar are in fact the same.

In sum, a plausible SP relation is any which the real-word semantics do not disallow.

For example, if there is some category Y = $\{(q_k,type_k) : k=1, \ldots\}$ such that the unary relation $\{(q_k,type_k),x\}$ implies not$[\{(q_{k'},type_{k'}),x\}]$ for all $k' \neq k$ and $(q_{k'},type_{k'}) \in Y$, the subsitution defined by {SP,x,y} would not be plausible if x and y were parameters of C_i and R_i, respectively, and $\{\{(q_k,type_k),x\}, \{(q_{k'},type_{k'}),y\}\} \subset body(U(C_i \Rightarrow R_i))$. On the other hand, while a consideration of degrees of plausibility and a strategy which prefers rules which are "more plausible" to those which are less may improve the performance of procedures which generate plausible rules by a best-first sort of search technique, a total enumeration of plausible rules is required if a certain identification of F is desired. If Q_i is defined to be the set of all substitution SP relations which are plausible for the transformation of C_i into R_i, then the solution to the case 3 learning problem is provided by exactly the same three step rule induction program presented in the preceding section. Thus, while induction of a magician's rules of behavior, where object identicalness is highly uncertain, may take significantly more computing time that the induction of the rules of TG, where every property of the sort (q,name) which is true of an object x precludes any substitution SP relation which would entail the simultaneous assignment of a different property (q',name) to x, both inductions are accomplishable by the same effective procedure.

7. PREDICATE AND CATEGORY LEARNING

The purpose of this section is to indicate briefly some possible avenues of approach to the problems of predicate and category learning. In the case of predicate learning, two points will be made. First, as Winston (1970) notes, once any pattern over n parameters is learned as the contingency of an induced rule, it may be used as an n-ary predicate to describe patterns in a novel way. For example, if the template A_2 (3) for a triangle (over the parameters a1, a2, a3) were induced in response to training, A_2 would define a ternary predicate and the relation $\{p{:}A_2$, a1:x, a2:y, a3:z$\}$ could then be used to augment the description of a scene in which the nodes x, y, and z were found to satisfy the relations required of the three nodes of a triangle. Second, because heuristic methods may later be used to generate and test the simplest, most plausible rules first, an encoding of training information patterns in terms of such higher-order predicates may be useful (exactly as the use of learned, higher-order phrase structure defining predicates like np or vp seem to be useful in representing and inducing TG grammar rules). That is, while the number of possible rules which may be inferrable from a set I of input-output line drawings described in terms of individual lines might be very large, the number of rules which are inferrable from the same data when encoded in terms of higher-order figure-descriptions is probably much smaller. Again, the desirability of such approaches appears to be general but is obviously dependent upon the real semantics (i.e., the nature of the rules to be learned).

With respect to category learning, a similar type of observation can be made. A

category, defined to be a set of mutually exclusive alternatives, may be hypothesized to comprise each of the various properties (unary uniform relation predicates) which are found to occur systematically in pattern representations C_i matching some rule contingency $C(x_1,...,x_n)$ but which are not criteral (not contained in $C(x_1,...,x_n)$). For example, when the END rule F is inferred from the training examples in Fig. 3 (b,c), any model of E_1 and E_2 comprising the abstraction $A \subset E_1 * E_2$ which matches F will necessarly bind the parameters v1 and v2 as correspondents. As a result of this binding, the same model that contains A will also contain the unary relations {p:"wants",name:v1} and {p:"planned", name:v2} in the associated residuals of E_1 and E_2. Thus, while "wants" and "planned" are not the same predicate and cannot therefore be included in this abstraction of E_1 and E_2, they are explicitly comparable because they are alternative properties of otherwise corresponding objects (v1 and v2).

As a result, one could reasonably hypothesize a new category, say v', and the relationship that {"wants", "planned"} $\subset$ v'. Note that v' is the name of the particular "place" in the common abstraction of E_1 and E_2 where v1 and v2 were bound but had different properties; this place is identified by the name x2 in the END rule F. Now, suppose several more examples of this rule F were provided and the possible members of v' were found to include "wants", "planned", "desired", ..., and "hoped." It would seem reasonable to canonize this category, say as "verb of intention," and subsequently to produce the predicate instance {p:verb of intention, name:x} whenever a word x occurred which was a member of v'. Moreover, the reader should note that such a predicate is in fact criterial to the contingency of the END rule, because the restriction of x4 (input verbs) to the category of verbs in general is too weak to prevent inappropriate applications of the rule. Thus, while the possibilities here are only minimally understood, it appears that such a category learning mechanism is potentially very powerful. Aside from providing a basis for refinement of induced rules by adding such additionally restrictive predicates to overgeneralized contingencies, the learning of such categories of mutually exclusive elements plays the additionally useful role of providing semantic cues related to the plausibility of potential substitutions, as previously described.

8. APPLICATIONS IN SPEECH UNDERSTANDING

While a need for brevity precludes a thorough discussion of the applications of these learning procedures to many diverse problems, an adequate appreciation may be gained by consideration of several induction problems in the speech understanding domain whose solutions are presently or will in the immediate future be attempted by procedures of the sort presented in this paper. In order to expedite the presentation, each problem will be described in terms of the concepts of knowledge model, training data, induction algorithm, and solution acceptability criterion. While the meaning of the last three of these terms is self-

evident, some explanation for the first is necessary. A problem knowledge model is a loose description of the theoretical relationship betwen the unknown rule and the training data. In particular, a knowledge model is a specification of a set of predicates and features the conjunctive products of all instances of which will necessarily (in theory) manifest the unknown rule. That is, the model comprises a set of tests to be performed on training data which, when exhaustively applied, will identify every instance of each n-ary predicate which may be criterial to an unknown rule.

Given these four aspects of a learning problem, it is now easy to describe several prototypic problems in speech research. The problems described are actual cases arising in the continuing development of HEARSAY (Reddy, et al., 1973; Lesser, et al., 1974). The three problems considered in turn are phonological rule, word phonetic spelling, and predictive syntax rule learning.

Problem 1. Phonological Rule Learning.[3] It is well known in speech work (e.g., Barnett, 1974; Cohen & Mercer, 1974) that some phones spoken in the temporal context of (preceded or succeeded by) some phones tend to be transformed into sounds which appear to be other phones. This process might be represented as $C_1PC_2 \rightarrow C_1P'C_2$ where the phone P in the temporal context of phones C_1 and C_2 is converted to (perceived as) the phone P'. The learning problem is to induce rules of the sort[4] $F = [C_1P'C_2 \rightarrow C_1PC_2]$ from training data which can reliably predict when a perceived sequence like $C_1P'C_2$ may be reinterpreted as C_1PC_2.

Knowledge Model: The predicates currently employed in HEARSAY define the space of all possibly induced phonological rules and are typical of those used to represent phonetic data. These include temporal contiguity of phones, phone labels, membership in classes of phones sharing particular articulatory features (e.g. vocalic, fricative), indicators of the presence or absence of articulatory features, presence of local extrema in the amplitude function, indicators of relatively short, medium, or long duration, etc. Every phonological rule currently employed in HEARSAY is representable in terms of conjunctions of these predicates. Training data: Since each rule to be induced is of the form $F = [C_1P'C_2 \rightarrow C_1PC_2]$, the learning program is to be applied to all exemplars of perceived (automatically phonetically classified) phone sequences ($C_1P'C_2$) which actually occurred when the "true" phonetic sequence was C_1PC_2. That is, the positive exemplars of the class defined by the categorical response pattern C_1PC_2 are the actual data observed when that pattern was theoretically correct. For any hypothetical abstracted rule, negative instances are occurrences in the data base of all utterances which match the inferred contingency $C_1P'C_2$ but should not in fact be rewritten as C_1PC_2. Induction Algorithm: Because the relational structure of all such rule contingencies is rigidly constrained to descriptions of the prior (first) context phone (C_1), the transformed

(second) phone (P') and the subsequent (third) context phone (C_2), a simple feature-value representation ($f_{1,1}$,...,$f_{1,n}$,$f_{2,1}$,...,$f_{2,n}$,$f_{3,1}$, ...,$f_{3,n}$) of each training exemplar may be used where $f_{i,j}$ is the value of the j-th feature of the i-th phone. Then the induction of plausible contingency patterns may be performed by interference matching of the feature representations. This simplification of the representations insures that any rules which are reliable and satisfy the structure of the associated knowledge model can be found without even performing the permutations required in graph matching (Hayes-Roth, 1974a). Rule Acceptability Criterion: Given the limited computing space and time for our speech system, we would like to find any rule F which is correctly applicable to no fewer than .05 percent of all utterances and for which the performance measure V(F) = [3(|I/F|) - 1(|I'/F|)]/[3(|I/F|) + 1(|I'/F|)], which counts each positive instance of F three times as much as a negative instance of F in the training data, exceeds .80.

Problem 2. Word Phonetic Spelling. It has been shown elsewhere (Hayes-Roth, 1974d) that extremely efficient parallel procedures exist for recognizing words (with tolerability for insertions and deletions) in sequences of hypothesized phones if the words are represented by templates which are equivalent to finite state transition networks with the interpretaton that each state is reachable by a number of alternative sequences of phones. Such a rule F for recognizing word w might be represented F = [(P_1, P_2, ..., P_n) → w], where each P_i = {$p_{i,1}$, ..., p_{i,m_i}} is a class of alternative phones which may occur in the i-th temporal position.

Knowledge Model: The contingency of any such categorical rule F may be represented as a conjunction of features $F_{1,1}$, ..., $F_{1,N}$,..., $F_{n,1}$, ..., $F_{n,N}$ where $F_{i,k}$ is true only if an element of the k-th class P_k of phones is recognized at time t = i. The classes {P_k} of confusable phones are defined exogenously: for example, P_1 = {AX(as in "but"), AH(as in "ma")}, P_2 = {AX, IH(as in "it")} are the only two classes needed to capture the eight alternative pronunciations of "America" as in F = [(P_1, {M}, {EH}, {R}, P_2, {K}, P_1) → "America"] (Hayes-Roth, 1974d). Training Data: Each instance of the word w to be learned in the sample of training utterances is represented as a sequence of sets A_1, A_2, ..., A_M of hypothesized phones, where A_i = {$a_{i,1}$, ..., a_{i,n_i}} is the set of possibly occurring phones in the i-th temporal position. This sequence yields the feature representation E = ($F_{1,1}$, ..., $F_{n,N}$) where $F_{i,k}$ is a boolean variable which is true only if some $a_{i,s}$ in A_i is also in P_k. A negative instance of any hypothesized rule F' is any temporal sequence of sets of hypothesized phones which match the contingency of F' but should not be recognized as an instance of the word w. Rule Induction Algorithm and Acceptability Criterion: As in problem 1.

Problem 3: Predictive Syntax Rule Learning. A distinction between formal and predictive syntax rules is an important pragmatic consideration in the design of real-time speech understanding systems. While it may be desirable for academic purposes to build an

understanding system which can fully and formally analyze an utterance, it is probably preferable in some contexts to exploit syntactic rules which, on the basis of some information, predict (hypothesize) values of other important variables and even, sometimes, suppress related concurrent but unfinished understanding processes. For example, in the HEARSAY news story retrieval task (keyword retrieval of wire service news stories) , a potentially useful predictive syntactic rule is F = [begin(t_0) & fetch-word(t_1,t_2) & re-word(t_3,t_4) & topic(t_4, t_5,expr) & end(t_5) & ($\forall i<j$) $t_i<t_j$ => retrieve(expr)] which is interpreted as follows. The rule contingency is: (1) the beginning of the input utterance is at time t_0, (2) any word which can mean fetch (e.g. "tell", "retrieve", "get", "fetch", "seek") has been recognized beginning at t_1 and ending at t_2, (3) a word meaning re (e.g., "about", "concerning", "mention", "cites", "re") has been recognized between times t_3 and t_4, (4) a keyword expression (topic) coded as expr has been recognized between t_4 and t_5 (e.g. expr = "Nixon" & "Agnew") where (5) the end of the utterance was at t_5 and (6) $t_i < t_j$ for all i < j. The associated response of the rule requires the understanding system to assume the sentence is parsed and respond by retrieving the news stories which satisfy expression expr (e.g. all news stories mentioning "Nixon" and "Agnew"). This rule is predictive because it hypothesizes that nothing that occurs in the intervals t_0 to t_1 or t_2 to t_3 adds anything essential to the understanding of the utterance. For example, it would not distinguish between "Please tell me about Nixon and Agnew" and "Would you kindly get for us any news stories which mention Nixon and Agnew" (or "Tell me nothing about Nixon and Agnew").

Knowledge Model: At the outset, the rule learning effort will be a modest one, due primarly to limited resources and specific goals. Thus, we will be seeking rules like F above which are essentially reliable (high performing) subrules of complete parse-to-response training examples. That is, the contingency F might have been abstracted from larger matching representations which include numerous other predicates occurring in a complete parse for each example such as words(t_0,t_1), silence(t_2,t_3), verb(t_2,t_3), etc. From training pairs of such complete syntactic analyses of the utterances and related responses, abstractable subrules will be sought which, when invoked over the training data, produce reliably desirable results. Training Data: Examples of utterance analyses for each distinct sort of response. Rule Induction Algorithm: The IM algorithm will be used and the training data will be structured as in case 2. Rule Acceptability Criterion: Because the specifications of the speech project require 95 percent semantic accuracy, at the outset only those rules which satisfy that criterion will be accepted.

9. CONCLUSIONS

This section attempts to provide a perspective on the paper's contibution in three ways: (1) the current learning problem and solution are related to other previously studied ones; (2)

the principal potential obstacle to widescale use of the proposed solution, a combinatorial explosion of abstractable rules, is considered; and (3) the possible value of heuristics in rule learning as alternatives to exhaustive methods of abstraction is briefly discussed.

The proposed approach to grammatical inference differs from previous approaches (Gold, 1967; Biermann & Feldman, 1973; Wharton, 1974) in the emphasis placed here on inducing rule representations by extracting commonalities from (deep) structural descriptions of events (sentences, scenes, behaviors) rather than using general heuristics to generate productions whose behaviors, hopefully, converge (statistically) toward the same sequential constraints exhibited by the surface descriptions provided for training. The current approach essentially eliminates the generation phase by requiring that every (C_i, R_i) training exemplar manifest directly the rule of interest. In this case, simple algorithms for identifying common subgraphs (abstractions) can be used to generate hypothetical productions which may be statistically evaluated, if necessary, to determine their reliability and utility vis-a-vis counterexamples. On the other hand, the proposed representational framework is computationally universal (Hayes-Roth, 1974c) and the approach is capable of inducing highly complex rules (those with a large number of relations in their uniform representations) in the same way as very simple ones. In both cases, the goal is to produce all abstractions manifested by the training information so that each may be considered as a hypothetical solution to the rule learning problem.

The validity of such an approach to learning must depend not only on its demonstrated ability to solve previously unsolvable induction problems, like that of inferring from examples the rules of TG, but also on the power of the approach to solve real-word problems of obvious value. While empirical testing and refinement of the proposed techniques will probably go on for years, intuitive sorts of evidence may be adduced to support the suggested approach. Two observations of this sort will be made. First, each of the major knowledge sources in our current speech understanding system is naturally representable as a set of rules of the sort considered here and, moreover, the current framework provides a much needed scheme for evaluating the performance of these rules alone and as well as in comparison to machine-generated plausible alternatives. Second, the proposed approach provides a basis for decomposition of knowledge and for step-by-step training of one rule after another. With a helpful trainer, the proposed algorithms can surely succeed in learning to imitate any behavior. This is the first learning technique known to this author for which this is true.

(2) There are two chief obstacles impeding the widescale use of the proposed rule learning techniques which require the interference matching (IM) procedure. First, although it is logically simple, the IM procedure may require enormous amounts of temporary storage to

compute and store alternative models because of combinatorial possibilities. Second, the learning problem as described in this paper required that all training exemplars (C_i, R_i) in I manifest the desired rule F. If this requirement is relaxed, so that at least one (C_i, R_i) fails to manifest F, then the assumption must be made that each distinct subset of I may be sufficient for induction of the unknown F. This greatly expands the number of possibilities (abstractions) that must be considered. Such a situation would arise, for instance, if a machine were supposed to learn rules of TG and the input contained errors or incomplete structural descriptions.

It appears that there can be no cheap and robust solution which eliminates all combinatorial problems and the attendant requirement for large amounts of storage. However, one desirable technique for reducing computing time and storage space is provided by the space limited interference matching (SLIM) procedure (Hayes-Roth, 1974a). As currently programmed (in PL/1 and in SAIL), SLIM is limited to non-relational event descriptions (feature-value descriptions), but the basic technique is immediately generalizable (see Hayes-Roth, 1974c, Ap. I for details). SLIM is chiefly constrained by the number of models it may maintain in working memory. Within the limitation imposed on memory space, the procedure performs the following actions in sequence: it successively introduces exemplars from the training set and partial-matches them (using the IM procedure to generate abstractions) with previously introduced exemplars and previously produced abstractions to generate new maximally informative abstractions; it evaluates the performance or utility of each inferred rule on the basis of the related positive and negative instances; it reduces (conditionalizes) the utility of rules which are subrules of better performing rules already abstracted; and it eliminates from overcrowded storage rules with lowest conditional utilities. As a result, the procedure dynamically optimizes the expected overall performance of all rules in storage. (If sufficient storage is provided, all inferrable rules are computed and retained.)

(3) Apparently reasonable heuristics might be introduced into the plausible rule induction procedure. These might be of a best-first sort, where the best hypothetical rule is taken to mean the one whose uniform representaion is a maximal abstraction of the training exemplars. Search algorithms of this sort are described in Hayes-Roth (1973) and Anderson (1975). On the other hand, for every situation where a heuristic can be shown to be desirable, there is usually another where it causes demonstrably undesirable effects. At this point, it is the author's intention to experiment with a variety of heuristics in each of our currently active learning projects so that the costs and benefits of each will hopefully come to be better understood.

In sum, many researchers have made useful application of structural representations of patterns and rules. In the current paper, a general procedure capable of inducing such rules

from appropriate training data and methods for comparing alternative hypothesized rules were discussed. While the prospects for fruitful application of these techniques seem bright, problems of combinatorics loom large. For the present, however, it is the author's opinion that even very large amounts of off-line computing dedicated to the discovery of reliable rules will be justified by a significant gain in knowledge or an improvement in the performance of the rules which are actually used in important real-time applications like speech understanding.

FOOTNOTES

1. This work was supported in part by National Institutes of Health Grant GM-01231 to The University of Michigan and Advanced Research Projects Agency Contract F44620-73-C-0074 to Carnegie-Mellon University.

2. I would like to acknowledge the valuable assistance of many people who have read various versions of this paper and the dissertation on which much of it is based: Lee Erman, John Holland, Dave Krantz, Walt Reitman, Elaine Rich, and J. E. Keith Smith.

3. No distinction is made here between phonological and acoustic-phonetic rules, although the HEARSAY system does reflect such a distinction. The current method is applicable to both cases.

4. This formalization does not reflect the full generality of actions such rules can perform (e.g., in HEARSAY, some rules modify validity ratings and time boundaries of hypothesized structures). Consideration of these actions and more general rule contingencies is beyond the scope of this paper.

REFERENCES

Anderson, J. Computer simulation of a language acquisition system: a first report. In R. L. Solso (Ed.), Informaton processing and cognition: The Loyola Symposium. Washington: Lawrence Erlbaum, 1975 (in press).

Barnett, J. A. A phonological rule compiler. Proceedings IEEE Symposium on Speech Recognition, 1974.

Barrow, H.G., Ambler, A.P., & Burstall, R.M. Some techniques for recognising structures in pictures. In S. Watanabe (Ed.), Frontiers of pattern recognition. New York: Academic Press, 1972.

Biermann, A. W., & Feldman, J. A. A survey of results in grammatical inference. In S. Watanabe (Ed.), Frontiers of pattern recognition. New York: Academic Press, 1972.

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

Cohen, P. S., & Mercer, R. L. The phonological component of an automatic speech-recognition system. Proceedings IEEE Symposium on Speech Recognition, 1974.

Chomsky, N. Syntactic structures. The Hague: Mouton, 1967.

Eden, M., & Halle, M. Handwriting and pattern recognition. In E. E. David (Ed.), Special issue on sensory information processing, IRE Transactions on Information Theory, 1962, IT-8, 74-191.

Evans, T. G. A grammar-controlled pattern analyzer. Proceedings IFIP Congress 68, 1968.

Evans, T.G. Descriptive pattern-analysis techniques. In A. Grasselli (Ed.), Automatic interpretation and classification of images. New York: Academic Press, 1969.

Friedman, J. A computer model of transformational grammar. New York: American Elsevier, 1971.

Gold, M. Language identification in the limit. Information and Control, 1967, 10, 447-474.

Hayes-Roth, F. A structural approach to pattern learning and the acquisition of classificatory power. Proceedings of the First International Joint Conference on Pattern Recognition, 1973.

Hayes-Roth, F. Schematic classification problems and their solution. Pattern Recognition, 1974a, 6, 105-114.

Hayes-Roth, F. An optimal network representation and other mechanisms for the recognition of structured events. Proceedings of the Second International Joint Conference on Pattern Recognition, 1974b.

Hayes-Roth, F. Fundamental mechanisms of intelligent behavior: the representation, organization, acquisition, and use of structured knowledge in perception and cognition. Unpublished doctoral dissertation. Ann Arbor: The University of Michigan, 1974c.

Hayes-Roth, F. The representation of structured events and efficient procedures for their recognition. Pittsburgh: Department of Computer Science, Carnegie-Mellon University, 1974d.

Hayes-Roth, F., & Mostow, D. J. An automatically compilable recognition network for structured patterns. Pittsburgh: Department of Computer Science, Carnegie-Mellon University, 1975.

Hopcroft, J. E., & Ullman, D. D. Formal languages and their relation to automata. Reading, Massachusetts: Addison-Wesley, 1969.

Langendoen, D. T. The study of syntax. New York: Holt, Rinehart, Winston, 1969.

Lesser, V. R., Fennel, R. D., Erman, L. D., & Reddy, D. R. Organization of the HEARSAY II speech understanding system. Proceedings IEEE Symposium on Speech Understanding, 1974.

Michalski, R. S. AQVAL/1--computer implementation of a variable-valued logic system VL_1 and examples of its application to pattern recognition. Proceedings of the First International Joint Conference on Pattern Recognition, 1973.

Newell, A. A theoretical exploration of mechanisms for encoding the stimulus. In A. W. Melton & E. Martin (Eds.), Coding processes in human memory. Washington: Winston, 1972.

Pfaltz, J.L., & Rosenfeld, A. Web grammars. Proceedings of the First International Joint Conference on Artificial Intelligence, 1969.

Reddy, D. R., Erman, L. D., Fennell, R. D., & Neely, R. B. The HEARSAY speech understanding system: an example of the recognition process. Proceedings Third Interational Joint Conference on Artificial Intelligence, 1973.

Rulifson, J. F., Derksen, J. A., & Waldinger, R. J. QA4: a procedural calculus for intuitive reasoning. Menlo Park: Stanford Research Institute, 1972.

Shaw, A. C. Picture graphs, grammars, and parsing. In S. Watanabe (Ed.), Frontiers of pattern recognition. New York: Academic Press, 1972.

Stoffel, J. C. A classifier design technique for discrete variable pattern recognition problems. IEEE Transactions on Computers, 1974, C-23, 428-441.

Watanabe, S. Subspace method in pattern recognition. Proceedings of the First International Joint Conference on Pattern Recognition, 1973.

Wharton, R. M. Approximate language identification. Information and Control, 1974, 26, 236-255.

Winograd, T. A program for understanding natural language. Cognitive Psychology, 1972, 3, 1-191.

Winston, P.H. Learning structural descriptions from examples. AI-TR-76. Cambridge: MIT Artificial Intelligence Laboratory, 1970.