KNOWLEDGL ACQUISITION FROM STRUCTURAL DESCRIPTIONS

I redei ick Hayes- Roth
The RAND Corporation
Santa Monica, CA 90106

and

John McDormott
Carnepie-Mellon University
Pittsburgh, PA I5213

Abstract. The representation of concepts and .»nlec eclent consequent pi odur linns is discussed and a method for indue inp knowled:',e by absh -at tin;; Mich i epi esontatinns from a sequence of training examples is desc i ibed. I ho proposed le,n ninp method, interference male limp,, induces abstractions by finding relational pioperties common to two or more exemplars.

1. INTRODUCTION

A numboi of distinct paradigm*"- for studying learning machines have emei \\f"(\ dminn the last twenty yeai s. Though each differs fiom the olhers in a variety of ways, the thice differences which most cleat ly demark each p.it adit'.m are (I) the types of knowledge which can be arqmied, *(?) the* way in WICh this knowledge is isplesea.ed and (3) the type of learning alponthm used. the leai ninp machine which we will desciibe in this paper .anquiies concepts i epi esenlalle as conjunctive forms of the predicate calculus and hehavjoi s repie osnntable as pioductions (antecedent consequent paii s of such conjunctive? forms); those concept', ,)\)(\ behavior rules are inferred from sequentially piosenlod pairs of examples by an alponthm that is provably effective foi a wide variety of problem',.

I eat ninp, is viewed bet e as a continual process of knowledge expansion, that is, as, the acquisition, in adaption h> traininp experiences, of higher-oi dor, more complex, and here e elaborate knowledge structures. One's knowledp.e at any point in line inc ludos those concepts and productions innately pi ovided or previously learned. I he concepts are paltrin template's; events which match a concept are i o< op,ni/ed as belnnpinp to the class delimited by that concept. | the productions are pairs of concepts; one of the concepts functions as a rocopnizer, the other specifies the form of an associated action. A production is interpreted as a behavior generator in the sense that (in some computing environment with an appropriate control .hue tore) the detection of a condition in the environment which matches the antecedent causes the consequent component to be instantiated and then evoked. Mere both The antecedent and the consequent are templates; the antecedent determines whether the pioduction is to be executed, and if so, what specific constants in the desciiplion of the event beinp, attended to are to be bound to variables in the consequent.

Within this fi amework, the machine learninp problem wild which we are concerned can be stated in the following way: Given a collection of concepts and productions constituting what is known at some time and a way of describing events in terms of their structure,

this lesoaich was performed while both authors were at Camege Mellon University. It was supported in part by the Defense Advanced Research Projects Agency (I44620 73 C 0074) and monitored by the Air Force Office of Scientific Research. A more detailed version of this research report can be obtained from the Computer Science Department, Carnegie-Mellon University, Pittsburgh, PA 15213.

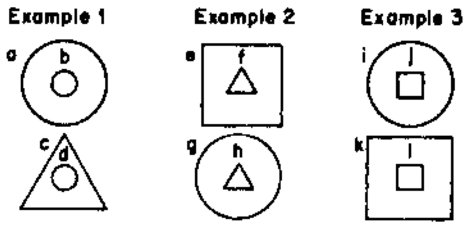

Figure 1

i onsh uc t a machine which is able to induce additional concepts or productions from tiaininp data. to make our teolmenl of this problem more concrete, we will use the amplest of the concept foimation tasks attempted by our machine as r)\) example throuphnut the paper. The task is find what the three exemplars in Figure 1 have in common. Our propram induces the following abstraction:

> There are three objects, including a small circle and a small square. The square is above the circle. The third objec t is larp„e.

This paper is divided into four sections. In the next section we discuss in perioral a way of describing events which facilitates hndinp what two of more events have in common and a matchinp alpoiillm which can be used to find these abstractions. we locale SPROE R, Our concept and production inducing program, within the broader context of out work. The third section describes SI 'ROUER'S interfeience matching (induction) algorithm in some detail; we indicate here more specifically how SPROUTCR makes use of structural representations of events to acquire and store knowledge. In the final section we conclude with a brief consideration of the strengths *and* weaknesses of SPROUTER and directions for future research.

II. STRUCTURAL REPRESENTATIONS

The problem which we are addressing is simply doseribed: Dosipn a propram which can infer concepts and produc lions fiom. illustrative instances. 1 he method we employ is correspondingly straightforward: Extract commonalities from the examples and attenuate their differences. Such an approach is like Galton's very primitive "composite pholopraph theory" of concept learning [8] and the "positive focusinp stratepy" for conjunctive concept learninp fust studied by Bruner, et al. |3|. While Gallon's contribution was simply to propose that unknown patterns could be inferred by overlaying homologous memory representations of related examples (as if one were forminp a composite of many photographs of the same subject), Bruner and his colleapues showed bow such a process could in fact be realized. Each presented object (exemplar) is described as a conjunction of specific feature values. To find the template which is matched by all of the presented objects, a feature vector containing only those features common to all of the exemplars is generated. This feature vector is the concept. Since that seminal worK, many computer scientists have produced increasingly practical and sophisticated feature-value concept learners based on i Haled techniques [1, JO, 16, ?0, 21, 29].

extending such learning models so that they can induce general elNational) classification and behavior rules is the goal of our work. In focusing on methods for for senerating relational abstractions which make possible the tecgnition of complex events, we encounter three problems"; not encounleiocl in previous work, first we must develop a formal scheme for desuihing complex events which fa< ilitatos the generation of absh actions. Second, given descriptions of two examples of the same concept or production, we must develop a method for comparing them so that their commonalities can be identified. Third, it is necessary to develop a way of storing The discovered abstractions to facilitate their subsequent use in either of two ways: they may be used as templates, for classification and behavior generation, or they may be used as knowledge represeetahons whose precision may later be improved by learning if new instances of the same concept or production are provided. These problems are referred to below as the description problem, the comparison problem, and the stot age problem. fach is considorod in more detail in the subsequent paragraphs,

I he dose Option problem entails providing a symbolic i epi esonlahon of each exemplar satisfying two demands. Inst, those attributes of the exemplar which are salient and potentially crilerial must be reflected in its description In iivane that the[1] classification till t * induced will be sufficiently discriminating. Note that since an exemplar may be composed of many objects, the description must distinguish each ohjec t and indicate cleai ly how it relates to I he others. Second, the descriptions should facilitate the identification of commonalities among the exemplars so that the abstraction being, sought can be found quickly. Since each object may exhibit a variety of charac teiistic s and par tic ipate in numerous relationships with other objects, finding commonalities between two or more examples will necessitate search. A representational scheme which helps direct this search is almost essential.

The method of description we employ is built on three central concepts, the property, the case frame, and The pai ameter. A propei ty is a feature or characteristic of a*n* object, for example, SQUAW: and SMAI. I. name two pioperties of small squares the pioperties AMOVE and W ar e used in our work to describe objects; which are above or below others in pictorial displays. To define the telationship of one object hemp, above another, a case frame of the sor t {ABOVE:', BHOW} is used. In general, case frames are sets of properties which are semantically related in some exogenenra'y determined manner. To produce descriptions of objects, events, or behaviors, case fr anes are parameterized (instantiated); that is, a name is given to each object in the event being described and this name is associated with each property of the object. Parameterized case frames are called case relations. For example, if b is the name of a square above a circle named c, this might be described by the following set of case telations: {{SQUARE: b), {CIRCI E : c }, {AMOVE:!.!, BEI OW: c}J. Such a set of case relations interpreted as a conjunction of valid propositions is called a parameterized structural representation or PSR [9, 12*j* 13]. In this example, {b, c) is the parameter set of the PSR.

A structural description of the first two exemplars in the concept formation task discussed in the introduction is given below.

E1:
{ {TRIANGLE:a, SQUARE:b, CIRCLE:c},
{LARGE:a, SMALL:b, SMALL:c},
{INNER:b, OUTER:a},
{ABOVE:a, ABOVE:b, BELOW:c},
{SAME!SIZE:b, SAME!SIZE:c}}

E2:
{ {SQUARE:d, TRIANGLE:e, CIRCLE:f},
{SMALL:d, LARGE:e, SMALL:f},
{INNER:f, OUTER:e},
{ABOVE:d, BELOW:e, BELOW:f},
{SAME!SIZE:d, SAME!SIZE:f}}

The deseription of (1 asserts that there is an event c omposed of three object'"., named a, b, rind c; that the object labeled a ha-. the properties of a tiangle, of a large object, and of containing the object labeled b; and so on.

PSRs provide a solution to the storage problem as well as to the description problem; that is, they can be used in storing discovered ahsti actions In the case of descriptions, pai ameler symbols are chosen to name each object so that if the same object is part of more than one case relation, it is refened to in a consistent way. If one alters the interpretation so that ea< h distinct parameter is considered as an unbound vni lable, the PSR can be consrdored a template for concept identification. Such templates have been used by several researchers [1,9, 12, *13, 15*>, 281 to specify what properties an object must have in oider to satisfy member ship in a pattern class. While the parametrrs in a description can be thought of as being e/istentiallv quantified, those in a PSR used as a template should bo thought of as being universally quantified. When used as a template for pattern classification, the PSR if, compaiod with an event (an existontially quantified P5R). If a mapping from the event to the template can be found which preserves., the par amotet bindings in the event desiiiphon and which makes each case relation of the template true, the event is said to match the template.

In addition to their role as classification ruler,, PSRs can be used as general hohavioi r ules. In this case two templates- are associated. One of them, the antecedent, is used to recognise a set of conditions (a context) which indicates that a particular ʳet of actions is appropriate; when the antecedent template is matehed by some event in the environment, the rule is invoked. The second template, The consequent, specifics, what actions are to be perfoimed. When the two templates share common pai ainftei s, each parameter in the- consequent is bound to the same value as the con responding parameter in the antecedent. These behavior rules may act, for example, as Post productions, transformational grammar rules, or the problem solving rules of STRIPS [7] In short, a rule with the antecedent A(X) and the consequent C(X) over the variables in the ret X is interpreted to mean (X) [A(X) «> C(X)]. In actual applications, A defines a precondition which can be true of the contents of some working memory, and C defines what is to be done if the precondition is satisfied. Note that any such production can be described by a PSR in which each case relation in the antecedent includes a term of the sort tEVENT:a, each case relation in the consequent includes a term of the sort EVLNT:c, and the PSR itself includes a case relation !ANIECDENBA. CEDEI :a, CONSQUENE. NI :c}.

'the abstraction of the first and second examples in the sample concept formation task can be represented in the following way:

E1*E2:
{ {ABOVE:1,BELOW:2},
{SAME!SIZE:2,SAME!SIZE:1},
{SMALL:2},
{SQUARE:1},
{SMALL:1},
{CIRCLE:2},
{TRIANGLE:3},
{LARGE:3}}

Exemplar 1 is in fact an instance of this abstraction if the parameter 1 is replaced by the parameter b, the parameter 2 by c, and the parameter 3 by a. Likewise, exemplar 2 can be seen to match the abstraction if the parameter 1 is replaced by d, the parameter 2 by f, and the parameter 3 by e.

The comparison problem can be solved by using a technique called interference matching or IM [11-12, 15]. It is a process for identifying all of the common properties of two PSRs and extracting a third PSR which is a template matched by the two exemplars. When two events have N attributes in common, their descriptions will contain at most N case relations which are identical (except for alphabetic differences between the names of corresponding parameters). Figure 2 schematizes IM as a process for finding the intersection containing these case relations.

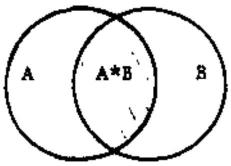

Figure 2. 'Interference matching.

"The circular are.is labelled A and B correspond to two PSRs, all of the case relations common to the two PSRs arc in the area labelled A+B (read "A star B"). Because any subset of this (conjunctive) set of common relations- also defines an abstraction of A *snc\ B*, it is important to be able to distinguish between the set and its proper subsets. We call *i\ny* abstraction of A and B which is properly contained in no other abstraction of A and B a maximal abstraction. More formally, if S (*) A denotes that A is a PSP matched by the PSR S, then a maximal abstraction, A, of two PSNs, S T, satisfies S(*)A and 1(+)A *and* (B) [B(*)A S(*)B T(*)B --> A(*)B].

It should be pointed out that for any two PSRs, there may be more than one abstraction which is maximal in the above sense. For example, given the following two exemplars,

E3: { {CIRCLE:a},
{RED:a}, {LARGE:a}}

E4: { {CIRCLE:b}, {CIRCLE:c}, {RED:b},
{GREEN:c}, {SMALL:b}, {LARGE:c}}

two innxim.nl abstractions oxid. If the parameters a and b are considered to be identical, the maximal abstraction is

E3*E4: { {CIRCLE:1}, {RED:1}}

If on the other hand, the parameters a and c are considered to be identical, the maximal abstraction is

E3*E4: { {CIRCLE:1}, {LARGE:1}}

To perform interference matching on reasonably complex representations, we *need* an algorithm which, operating within as small a search space as possible, can discover the best maximal abstractions as quickly as possible, two approaches to interference matching are Known: (I) In the bind-first appi oac h, each pai ameter in one PSR is associated with a parameter in the second PSR and then a maximal abstraction is found by extracting the c ase relations which are identical in the two PSRs (modulo Ibe parameter bindings). In this case, if the lesser number of parameters (in either PSR) is MP *and* the greater number is NP the. number of possible binding functions is combinatorial, (binomial coefficient of *NP* over MP) * MP!. *(?)* Alternatively, in the match-first approach, all instantiations of case frames of one type in one PSR are compared with all instantiations of the same type of case frame in the other PSR, and possible parameter bindings are identified by determining which parameter*; have corresponding properties in comparnl.>le relations. Here if *N\ and* MI are the numbers of case relations in the larger and smaller PSR (assuming only one type of case fiame), the number of possible way:, in which the relations can be forced into correspondence is similarly combinatorial. While it is true that if one were interested in computing abstraction*; of quite low-level event descriptions (such as undirected graphs) neither method would bo much pieforablo to the other, in most leal problems the number of instances of any particular case frame is quite small relative to the number of parameters in the PSR, and so the second method is usually prefei able to the first. It is this method which is used in our current work.

The actual algorithm we use has the following form: A randomly selected case relation from one of the exemplai PSRs is put into correspondence with a case relation (which is a parameterizalion of the same case frame) from a second exemplar PSR; parameters having identical properties are identified as equivalent and the insulting common case relation becomes the (primitive) absti ac lion associated with that set of parameter bindings. I hen other pairs of primitive case relations, one from each of the two exemplai PSRs, are put into correspondence. If a coinpaied pair of relations entails parameter bindings consistent with those already identified, the common relation is added to the ahsti action being produced. "Ibis now abstraction is the set union of the old absti action and the new case relation, and the new set of parameter bindings is the set union of those bindings entailed by the previous abstraction and the forced bindings of the parameters in the compared pair of case relations. If a pair of case relations entails parameter bindings inconsistent with those already identified, the common case relation becomes a new (primitive) abstraction.

Clearly, this algorithin may find a number of competing maximal abstractions. Our approach is to build as many distinct abstractions as possible, one relation at a time, until a limitation on the number of distinct abstractions which can be considered at one time is exceeded. At that point, only those abstractions which arc most significant in terms of the number >*M* type of case relations they include an* retained. These abstractions continue to be extended as other pairs of consistent relations are *iount* at the same time, the least significant absti actions are continually pruned from further consideration in order to keep the search space as small as possible.

The result of the process is a set of best maximal alis tract ions, represented as PSRs. Any one of these absti actions (interpreted as exislentially quantified) can then be input to SPROUTER together with a third exemplar to produce a set of maximal abstractions of three exemplars, or the process may be repeated on as many additional exemplars as desired. Since a maximal abstraction is compared to an exemplar in the same way that an exemplar is compared to another exemptar, we find it desirable to store abstrac lions as PSRs, with the interpretation that their parametersm tors ropt esent exislentially quantified variables, derived from the correspondence of case lelations in the exemplars from which the P$R was induced.

The successive steps involved in producing the maximal abstraction of the first two examples in the concept formation task are shown below.

(1) {SMALL:1}
(2) ({SMALL:1}, {ABOVE:2, BELOW:1})
(3) (({SMALL:1}, {ABOVE:2, BELOW:1}),
    {SAME!SIZE:1, SAME!SIZE:2})
(4) ((({SMALL:1}, {ABOVE:2, BELOW:1}),
    {SAME!SIZE:1, SAME!SIZE:2}), {SMALL:2})
(5) (((({SMALL:1}, {ABOVE:2, BELOW:1}),
    {SAME!SIZE:1, SAME!SIZE:2}), {SMALL:2}),
    {SQUARE:2})
(6) ((((({SMALL:1}, {ABOVE:2, BELOW:1}),
    {SAME!SIZE:1, SAME!SIZE:2}), {SMALL:2}),
    {SQUARE:2}), {CIRCLE:1})

'The case relation {SMAI I :c } is selected at random from EJ and is then put into correspondence with the case relation SMALEF} from \2. 't he parameters r and f are identified as equivalent and so (since c and f are The first pair of parameters hound) the primitive abstraction {{SMALL:!}} is generated. Then the pair of c asr relations {ABOVE:b, 'BLLOW:C} and {ABOVL:d, BELO)W:f} are put into correspondence. Since the idenlificalion of c with f and of b with d is consistent with the already established binding, the piimitive abstraction {{AHOVE:2, BELOW:l}} is added to {•SMALI :J }(. II should be noted that our basic 1M algoiilhm actually finds only siy of the eight case relations c nnstituting the abstraction. This is because the? partial abstraction {{ FRIANGLE '.:3}, {l. ARG-.:3}} was pruned from consideration early in the match under the space limitation constraint, to insure that such complementary relations are not missed, our algorithm, after completing the process described above, searches for additional relations which can extend the abstractions produced. Any such relations which are found are conjoined to the abstraction to produce a maximal abstraction.

SPROUIER, the program which induces abstractions fi oin structural descriptions, is only one part of a classification and learning system which wo are developing. I be top-level progi am, called SI JM | 1, 10, 16J, is a general space limited inlei ference matching proceduie which builds abstractions from example-. and then uses these abstractions to classify test stimuli. While the abstraction of featrei evalue repesentations can be performed by simple bit vector operations (which SLIM itself is capable of), the gonei ation of abstractions bom PSRs requires the matching and paiamoler binding, detei minahons discussed above. The progi am, SPROUIIR, was treated for this purpose. Once an abstraction is computed from some PSRs, it is nearly as complex a problem lo use it for classification as it was to generate it originally [1, 11-13]. With this in mind, SPROU'TER was designed to produce two outputs: one of these is a PSR, which as we have indicated can be matched with subsequent exemplars to produce more refined abstractions; the other is a special purpose recognition network used to exploit an abstraction as a template.

the templates which SPROUTER generates for GLIM are automatically compilable recognition networks or ACORNs [13, 18]. An ACORN is a special data structure, equivalent in representational power to a PSR, but better adapted to serve as a template; it is essentially a Pandemonium pattern recognition system [27], generalised to handle patterns and data described as general pi operational formulae. Once an ACORN has been produced, SLIM can determine whether a descriptive PSR matches it by using The PSR to create an instance list at each of the lowest-level nodes in the ACORN and then allowing the relevant instances of subpatloins of interest lo percolate upward in the network. If any instances of the highest level node are found, the template is matched by The stimulus pattern. The lowest-level nodes of an ACORN correspond to the distinct case frames in a universally quantified PSR and are like the feature demons of a Pandemonium system. A feature demon, however, reports only the number of instances of its particular fealure to higher level demons, whereas the node in an ACORN actually passes its instances up to the higher-level nodes which it supports. The higher-level nodes look for instances of the particular conjunction of case, relations in which they are interested, just as higher-level "cognitive demons" in Pandemonium look for specific combinations of featuie values. the highest level node in an ACORN is instantiated if and only if the abstraction is matched by the PSR. I bus this highest-level node corresponds lo a Pandemonium's highest level cognitive demon which recognizes when a pattern of interest is matched. Because ACORNs have been developed to provide a means for sharing the results of the evaluation of subexpressions common to numerous templates, each conjunction of (Medicates or subtemplales is. associated with a single binai y bachcing node whose two descendants represent The conjoined prepositional formulae.

Once a set of best maximal abstractions is computed for two or more exemplars, all training exemplars (or a sample of them) may be examined to see if they match the infened hypothetical concept or rule. Only to the extent that exemplars of the same class match an abstraction and those of the other classes do not, do we find support for the inference that the abshr action is the criteria! concept underlying the training data [9-J0J. ACORNs greatly facilitate this examination process. One simply instantiates the terminal nodes of the ACORN whose highest nodes represent the abstractions of interest, and then iteratively computes all instances of each higher-level node from those pairs of instances of its subordinate' nodes which satisfy critcital tests on their values. If any instances of the abstraction are produced, the training exemplar matches the abstraction. Without ACORNs, it would be extremely difficult to determine which positive and negative training exemplars matched each abstraction.

111. THE INTERFERENCE MATCHING ALGORITHM

SPROUIPRY. function, as we have said, is to build ACORNs which can he used by SI IM for recognition, Before this construction process can begin, a set of piimilive (bottorn- level) nodes must be generated and then instantiated, Fo generate these nodes, SPROUTER reads in the set of caso frames which are relevant to the task it is facing, for each of these case frames, a primitive node is created which is essentially a universally quantified case i elation. SPROUTER then finds, in the descriptive PSRs of two exemplars, the set of distinct instances (case relations) which are instances of each of these nodes. Each node has two associated instance lists; each of these lists contains the instances of the case relation for one of the exemplars. Tor example, given the two case frames NI: {CIRCLE}, N2: {ABOVE, BLOW} and the two exemplars

E5: {{CIRCLE:a, CIRCLE:b},
    {ABOVE:a, BELOW:b}}

E6: {{CIRCLE:c}}

SPROUTER will create two nodes, NI and N2, **and then** produce four instance lists. Two of these lists, ([E5/a], |165/67) and ([("6/c]), are associalod with node NI. The other two, ([15/a, E5)/b]) and ( ), are amounted wilh node

When the primitive nodes have been instantiated, GPROUHR produce'., the set of maximal abstractions of the two PSRs by construeling, bottom-up, a binary-branching ACORN Fach higher-level node of this network is a conjunction of two nodes, one of which is always a primitive node. Before initiating the building process, SPROUII R deletes all of the primitive nodes that do not bave at least one instance from each exemplar. Then one exemplar, the one wilh fewer instances over the remaining nodes, is tagged Finto the other exemplar is tanged f-comp' And each instance-intro is marked unused. SPROUTER then begins the actual construe lion. An unused ' intro insance frrom a primitive node is chosen as one of thr two instances to be used in tbe construction; it is selected on the basis of tbe likelihood of its being an instance of a node which is a constituent of a best maximal abstraction. This instance is then paired with every instance from Einto of every node, farh of these pairs of instances is used to construct a candidate node which will accept instance pairs only if They are equivalent to The prototypic pair. If there is al least one such pair of instances in e_{comr}y the candidate node is , added to the network and all instances of the node (fiom both exemplars) are computed. Thus, each step in the abstraction building PROCess involves combining, itei atively, an unused instance from a primitve node wilh each other instance in the ACORN. After each of the resulting conjunctive nodes is generated for a pair of instances from into, all instances of that node, first from I comp, and then from einto, are computed, if no instances are found in I comp node represents an abstraction that is not true of the second exemplar and so the node is not added to the network. 'I tie process continue?, until all of the case relations that are common to both exemplars have been conjoined.

Of course, this algorithm, left unconstrained, would build a node for each subset of case relations in Einto for which there was an equivalent subset in '$_{COmn}$ Clearly, tbe size of tbe search space would increase exponentially. Thus, for even small problems, it is important to somehow reduce tbe number of nodes construe ted. We use two heuristics. The first of these enables us to keep the search space to a manageable size by providing for the automatic pruning of those conjunct ions least likely to be part of a best maximal abstraction. To determine which partial abstractions are least promising, a value is computed which we call the utility of a node. Basically, the utility of a node is an increasing function of the number of pioperties covered by tbe node and a decreasing function of the number of distinct parameters needed to instantiate the node. More specifically, our current utility measure adds 1.0 for each property of a case relation and subtracts 1.0 for each distinct parameter in the associated PSR. Our justification for this rather rough measure of utility is that it will yield as tbe highest valued nodes, those with the greatest scope and connectivity. Equivalently, tbe higher tbe utility of a node, tbe more informative and apparently "better" it is as an abstraction.

During the construction of the ACORN, a list of all nodes currently in the network is maintained. This list, which is ordered by tbe utility of its elements, has a stipulated maximum length. Whenever the number of total nodes in the ACORN exceeds this stipulated maximum, a primitive node which does not support any higher-order nodes is marked as removed from consideration. If all remaining primitive nodes support some higher-level node, then the least valued maximal abstraction (provided there is more than one maximal abstraction in the network) and all nodes supoi ting it (or supporting one of its supports, recursively) and not supporting some other higher valued maximal abstraction are deleted (or marked as removed from consideration if they are primitive nodes). Thus, the numbei of nodes in the network can ex< eed Ihe stipulated maximum only if just one maximal abstraction remains. While in some cases, it might be desirable to require that at least k (k>J) best maximal abstractions be maintained, we have not yet found a *\need* for this option.

As a result of the limitation on nodes in the ACORN, the typical behavior during construction is as follows: Instances are introduced one at-a -time from Einto and are conjoined with other Einto node instances to form PSRs leprespnling subsets of case lelalions of varying utility. As soon as the number of nodes corresponding to these nodes in the ACORN exceeds the stipulated maximum, the maximal node with the lowest utility together with all nodes which support only it are deleted from the network. Ibis construction and pruining cycle is lepoated until the sot of best maximal abstractions has been found.

The second heuristic provides the search with direction by indicating which one of the unused instances is to bo used in the next cycle of construction. Our search for the best maximal abstractions is essentially hill climbing,, but occurs on manv bills simultaneously. Since our pruning heuristic enables us to maintain a gradually deceasing number of maximal abstractions. the number of lulls under consideration is reduced as the search progresses. Clearly, if we could select first all of those instances from Einto which were instances of the best inaximal abstractions (Ihe highest hills), then our search, since it would take place in an essentially unimodal space, would be as efficient as possible. Of course it is impossible to cleteimine a priori which instances are instances of the best maximal absh actions. However, by using a variant of the utility function described above, it is possible to compute, fairly cheaply, the upper bound of the actual utility of any node which might be constructed. Using, the; strategy, we can, at relatively little cost, significantly increase the probability that the node cnnstiueted will be a constituent of a best maximal abstraction. "The selection procedure wo use is as follows: We set a sampling factor (currently 207] for the proportion of tbe unused instances from Einto which are to be examined. We select at random this percent of the unused instances (but at least three until there are fewer than three unused instances), for each of the instances in this sample, we determine an upper bound of the utility of all of the nodes which could bo constuctod by conjoining the sampled instance with the remaining instances of nodes still under consideration. The one instance which produces tbe node with the highest potential utility is constructed.

The actual construction of a node is a two step process. First SPROUTER creates a set of tests which are both necessary and sufficient to accept just those instances that are equivalent to the pair of instances used as a model in building the higher-level candidate node. It is possible to create such a set of tests working only with the 'ameness or difference of selected parameters. For example, to construct an ACORN node To accept tbe two instances {CIRCIFx} and {AHOVB :a, BILOWicj, a same parameter (SP) test is generated to insure that the first parameter of Ihe first case relation is the same as the second parameter of the second relation, and a different parameter (DP) test is generated to insure that no non-explicit SPs are accepted. If one thinks of this ACORN node as being constructed from a left and a right instance, where the parameter of the left instance is numbered 1, and tbe parameters of the right instance are numbered 2 and 3, then a minimal complete set of tests needed to exactly represent the same and different relation*; are (SP:1, SP:3} and {DP:I, DP:?}.

After tho sot of tests has been created, the candidate node is assoc iated with a |'generator set which speciture how the parameters of its instances are to be extracted from pair', of submchnte instance-, which satisfy the node''. SP and DP tests. Hoc an .r of tho implicit i equii ement for DP relatione to hold on all distinct parameters'., the order of thr new relation is exactly the number of distinct parametoi •. in the two relation instances need in building the node. In the ahove example, there would ho two parameter', in each instance of tho new node and these would correspond to pat amr lets] and 2 (since I and *3are* identical). I he genei atm list for this node would he just (J,?). IFrom the nature of The explicit SP and DP tests used, it follows thaT any two nodes having instances der ived front equivalent pairs of instances must he equivalent. Whenever such a duplicate node is consti ucted, it is removed fiom the ACORN.

It should he apparonl that an ACORN constructed in the fashion described ahove will not necessarily contain a maximal abstraction. Whether or not it will is partially dependent on what maximum has been stipulated for the number of nodes in thr- ACORN Hut even if the stipulated maximum is large enough so that thr- highest node in the ACORN is a constituent of a maximal abstraction, the ACORN may not he complete; that is, some of the case elations in the abstraction may have been lost. This can occur if one more primitive nodes whose instances are a part of the abstraction were o r omoved frorn consideiation early in the c o n s t i o n process. In such a case, howevei, it is always possible to extend the ACORN with conjunctions of these lost primitive node instances. This is done by successively • (introducing into the construct and prune c yc lo each instance in I-j$_n$|$_{(n}$ which does not support all of the instances of all of the highest nodes in the ACORN. Each Eache introduced instance is conjoined with each of the instances of each highest node In produce candidate nodes, If instances of any of these new abstractions are found in I Ecome$_|$.,, these new nodes ate retained; the ACORN is then extended further, in the same way, until the* best maximal absb actions have been found,

We have already soon the abstraction SPROUTER constructs given the first two exemplars in the first concept foimation task. "The set of case frames from which the primitive nodes were created, all three exemplars, and the best maximal absraction found by SPROUIER are given below.

```
CE:
 {  NE{CIRCEE},
   N2:( SQUARE},
   N:*:{T WANGLE},
   N1:|I ARGL!,
   Nb:{SMAI.I.},
   N6:{1NNFR, OUT ER},
   N7:{ AMOVE, BLUM},
   N8:{I.rf- '1, RICHI},
   N):JSAMt !SI IAPE, SAMEJSIIAPE},
   NI0:{SAME!SI7E, SAMI ISIzE),
   NI 1 bESIDE, nrsior.},
   N.I ?:{C0N1 1GU0US, CONTIGUOUS}}

TI:
 { {ER1ANGI L:a, SQUARE:!*, CIRCLE* },
   R.ARGE:a, SMALL:b, SMALL*},
   {INIMLR:b, OUTER:a},
   {AHOVha, AROVI :b, BELOW:c},
   {SAME!SIZE:b, SAME!SIZE:c}}

12:
 { {SQUARE:d, I RIANGEt m, CIRCLE-.I},
   {S.MAII :d, I.ARGI :e, SMAI I :f},
   ANNI R:f, OUII R.-eJ,
   ■JAHOVI :d, HI IOW:c, FFLOW:f},
   {SAMI'M/hd, SAME!SE/E:f}}

13:
 / (SOUAREr g, CIRCI f :h, CJRCI I :i},
   SMAI I :g, I AREI :h, SMAI I :i},
   ■1NNI R:i, OUR R:h},
   IAUOVI'.g, UELOW.h, BLLOW:i},
   {SAMI MIAPI :h, SAMI IShAPI :i|,
   {SAMI !SI/I :g, SAMI !SI/E:i}}

I- h i '/iI :):
 {  !NIO:{SAME!Si/E:I,SAMI !SJ7I :2},
   ;N/:{ AMOVE:),BELOW:?},
   (NOICIRCI h?},
   !N!-.:{S.MAI.I.:1 },
   •;N!>:.;SMAI L.?},
   ;N^V:{5;0UARE:I },
   {N/I:{t ARGI :3\\
 INSIANOES f ROM I XI MPI AR I 1 it ?
   (|r i-tt //;>,[ i iT/71,1 \H ;/3\)
 INSIANCI S I ROM I XI-MPI AR I 3
   <|I 3/}.si'Vi,L3/h|)
```

SPROUIER took b sc*conds of < pu time on a POP JO (model KA 10) to produce* I h i ? which it found after consti uc ling M nodes (/ moi e than necessaiy). SPROUTE R tool; *A* seconds and consh uc ted (> node-, (the* fewest po-.'.iblo) to prodme (F I it ?)il 3. 1 he ah'.traction SPRoU"II"R found, however, though it is the best absh action producible using our urate h frist melhod, is not maximal. It is missing two case relations. As we indicated in the* fust section of the paper, the abstraction SPROUTER induces is the following:

> There are throe objects, including a small circle and a small square, 1 he square is above? the circle. The third object is large.

The best maximal abstraction includes the specification that thr* lauge object contain:; another one which is one of the two small objects, SPROUTER is unable to find this abstraction for two reasons: (I) The grain sizo of The i epi esentabons used in describing the examples is bio big,; more atomic uniform representations ai e needed to make abstraction, which is a stuclly subh active process, inoie generally applicable | I I , Ib] (?) Many-one pai amojor correspondences must be allowed in order to iivuie that relevant cot respondences are not lost, these two problems, whose solution require*, methods of greater gonei ality than we have currently implemented, are discussed in detail in the CMU technical report from which this paper is taken.

IV. CONCLUDING REMARKS

SPROUIER has already solved learning problems of theoretical significance and of considerable complexity. Brcauso of the extensive size of the search spaces, such learning could not be done with simple enumerative matching algorithms. In essence, SPROUTER establishes the feasibility of induction from non-trivial exemplar descriptions. In many respects, however, SPROUTER is quite primitive. It is a purely syntactic matcher; it knows nothing at all about the underlying structure or significance of any of the predicate descriptions upon which it operates. Eor this reason, its utility function, and thus its heuristics, are very weak. One interesting aproach to improving the performance of SPROUTER would be to provide it with domain-specific utility functions. For example, if SPROUTER knew that concordance on antecedent or consequent relations was more important than concordance on most other relations, it would never attempt to match the antecedent part of an example with a consequent part. Similarly, if it knew that concordance of higher order grammatical constructs (e.g., a sentence) was more significant than concordance on lower-order ones, it could quickly zero in on the concordances of two sentence structures and then continue building abstractions in an essentially top-down fashion.

Even though SPROUTER's performance has been quite impressive on several tasks, there are three difficulties impeding the use of such a learning machine in general applications. First, an empirical question has been raised regarding the preferability of approaches to induction based on the one-one and many-one binding alternatives. If object integrity in representations is generally tenuous -- that is, if each object in one PSR can correspond to multiple, diverse objects in another PSR, as was the case in the concept formation task described above -- abstraction procedures based on the many-one approach will have to be developed. Second, one must identify which real-world problems can be solved by interference matching methods. Because the case frames which SPROUTER uses in inferring abstractions are assumed to be externally provided, the utility of our basic method depends upon the prior identification of useful properties. Third, while SPROUTER uses what is basically a common heuristic search strategy in AI (breadth first with pruning of low-valued paths), it is still too expensive to be widely applied in practical problems.

REFERENCES

1. Barrow, H. G., Ambler, A. P., and Burstall, R. M. Some techniques for recognizing structures in pictures. In **Frontiers of Pattern Recognition**, S. Watanabe (Ed.), Academic Press, New York, 1972.
2. Bobrow, D., and Winograd, T. A knowledge representation language. **Cognitive Science**, in press.
3. Bruner, J. S., Goodnow, J. J., and Austin, G. A. **A Study of Thinking**. Wiley, New York, 1956.
4. Burge, J., and Hayes-Roth, F. A novel pattern learning and classification procedure applied to the learning of vowels. **Proc. of the 1976 I.E.E.E. Intl. Conf. Acoustics, Speech, and Signal Processing**. Philadelphia, 1976.
5. Ernst, G. W., and Newell, A. **GPS: A Case Study in Generality and Problem Solving**. Academic Press, New York, 1969.
6. Evans, T. G. A program for the solution of geometric-analogy test questions. In M. Minsky (Ed.), **Semantic Information Processing**. MIT Press, Cambridge, 1968.
7. Fikes, R. E., and Nilsson, N. J. STRIPS: A new approach to the application of theorem proving to problem solving. **Artificial Intelligence** 2, (1971), 189-208.
8. Galton, F. **Inquiries into Human Faculty and its Development**. Dent, London, 1907.
9. Hayes-Roth, F. A structural approach to pattern learning and the acquisition of classificatory power. **Proc. First Intl. Jt. Conf. Pattern Recognition**, 1973.
10. Hayes-Roth, F. Schematic classification problems and their solution. **Pattern Recognition 6**, 2 (Oct. 1974), 105-114.
11. Hayes-Roth, F. Fundamental mechanisms of intelligent behavior: the representation, organization, acquisition, and use of structural knowledge in perception and cognition. **Doctoral Dissertation, The University of Michigan, Ann Arbor**, 1974.
12. Hayes-Roth, F. An optimal network representation and other mechanisms for the recognition of structured events. **Proc. Second Intl. Jt. Conf. Pattern Recognition**, 1974.
13. Hayes-Roth, F. Representation of structured events and efficient procedures for their recognition. **Pattern Recognition, 8**, 3 (Aug. 1976), 141-150.
14. Hayes-Roth, F. Patterns of induction and associated knowledge acquisition algorithms. In **Pattern Recognition and Artificial Intelligence**, C. Chen (Ed.), Academic Press, New York, in press.
15. Hayes-Roth, F. Uniform representations of structured patterns and an algorithm for the induction of contingency-response rules. **Information and Control**, 32, (1976), in press.
16. Hayes-Roth, F., and Burge, J. Characterizing syllables as sequences of machine-generated labelled segments of connected speech: a study in symbolic pattern learning using a conjunctive feature learning and classification system. **Proc. Third Intl. Jt. Conf. Pattern Recognition**, 1976.
17. Hayes-Roth, F., Erman, L. D., Fox, M., and Mostow, D. J. Syntactic processing in Hearsay-II. In Speech understanding systems: summary of results of the five-year research effort. Department of Computer Science, Carnegie-Mellon University, Pittsburgh, 1976.
18. Hayes-Roth, F., and Mostow, D. J. An automatically compilable recognition network for structured patterns. **Proc. Fourth Intl. Jt. Conf. Artificial Intelligence**, 1975.
19. Hirschman, L., Grishman, R., and Sager, N. Grammatically-based automatic word class formation. **Information Processing and Management, 11**, 1-2 (Mar. 1975), 39-57.
20. Hunt, E. B. **Concept Formation: An Information Processing Problem**. Wiley, New York, 1962.
21. Michalski, R. S. AQVAL/1 Computer implementation of a variable valued logic system VL1 and examples of its application to pattern recognition. **Proc. First Intl. Jt. Conf. Pattern Recognition**, 1973.
22. Minsky, M. A framework for representing knowledge. In **The Psychology of Computer Vision**, P. H. Winston (Ed.), McGraw-Hill, New York, 1975.
23. Moore, J., and Newell, A. How can Merlin understand? In **Knowledge and Cognition**, L. W. Gregg (Ed.), Erlbaum, New York, 1974.
24. Plotkin, G. D. A note on inductive generalization. In **Machine Intelligence**, vol. 5, B. Meltzer and D. Michie (Eds.), American Elsevier, New York, 1970.
25. Plotkin, G. D. A further note on inductive generalization. In **Machine Intelligence**, vol. 6, B. Meltzer and D. Michie (Eds.), American Elsevier, New York, 1971.
26. Reed, S. K., Ernst, G. W., and Banerji, R. The role of analogy in transfer between problem states. **Cognitive Psychology 6**, 3 (Nov. 1974), 436-450.
27. Selfridge, O. G. Pandemonium: a paradigm for learning. Symposium on the Mechanization of Thought Processes. H. M. Stationery Office, 1959.
28. Shaw, A. C. Picture graphs, grammars, and parsing. In **Frontiers of Pattern Recognition**, S. Watanbe (Ed.), Academic Press, New York, 1972.
29. Stoffel, J. C. A classifier design technique for discrete variable pattern recognition problems. **IEEE Trans. on Computers C-23**, 4 (Apr. 1974), 428-441.
30. Trebakoff, A. Computer-generated word classes and sentence structures. **Information Processing 1974, Proc. IFIP Congress 74**, 1974.
31. Vere, S. A. Induction of concepts in the predicate calculus. **Proc. Fourth Intl. Jt. Conf. Artificial Intelligence**, 1975.
32. Winston, P.H. Learning structural descriptions from examples. In **The Psychology of Computer Vision**, P. H. Winston (Ed.), Mc-Graw-Hill, New York, 1975.