THE SUPERIORITY OF RELATIVE CRITERIA IN PARTIAL MATCHING AND GENERALIZATION

Paul J. Kline

Center for Cognitive Science
The University of Texas at Austin

ABSTRACT

Some familiar justifications for concluding that a data element D is an unacceptable match to a pattern element P are examined for their suitability for partial-matching applications. All of these absolute criteria are found to be unacceptable in that they rule out some plausible partial matches. A partial matching program, REIAX, is described which justifies the conclusion that a data element D is an unacceptable match to a pattern element P on the grounds that some other data element D_b is a better match to P. The use of such relative criteria makes it possible to compute all of the kinds of mappings that Kling [9] has claimed are required to account for intuitions about similarity. The ability to form unrestrictive mappings allows RELAX to construct generalizations that have eluded other approaches [5, 6].

I INTRODUCTION

The work described in this paper grows out of an attempt to simulate the behavior of junior-high school students learning geometry. An examination of the geometry textbook they were using [8] made it clear that students were expected to appreciate the similarity of their exercise problems to the examples provided in the text, and to use those examples as a guide in finding solutions. Protocol analyses of students using this text suggests that students do often (but not always) appreciate the similarities that are intended [3].

One way to model such behavior is to represent the worked-out examples as <$condition_1$,.....> => <$action_1$,...> rules, where the left-hand side describes the example problem and the right-hand side indicates the steps required for its solution. The description of an exercise problem can then be compared against the left-hand sides of these rules. Interpreting these rules as productions would imply that the description of the exercise problem would have to completely satisfy all conditions of one of the rules before any action could be taken towards formulating its solution. In other words, a full-match comparison using this representation could only account for a student's ability to remember the solutions to problems that have been solved previously. However, if instead of merely looking for a full match we are able to find a rule whose left-hand side "closely approximates" the description of the exercise problem, then we may be able to provide a computational account of what happens when a student recognizes the similarity of an exercise problem to a previous example.

Even if this can be done, however, the success is achieved at the cost of a considerable complication in the notion of executing the right-hand side of a rule. In those cases where it turns out that a full match is achieved to a rule's left-hand side, executing the right-hand side amounts only to a straightforward execution of a fully instantiated procedure. However, in cases where only a partial match has been achieved, the right-hand side actions will need to be modified to reflect the departures from a full match. This paper will only be concerned with the question of how to partial-match problem descriptions in a way that properly models students' abilities to detect similarities between problems.

II SOME ABSOLUTE CRITERIA

An approach which might be expected to yield a satisfactory partial matcher would be to examine the operating principles of a full matcher for a production system and to determine which principles could be retained and which principles would have to be discarded. Each condition in the left-hand side of a production rule is a proposition which can consist of constants, variables (existentially quantified), or other propositions. As an example we have (believes Vsubject (flat earth)), where (flat earth) is an embedded proposition. The V prefix indicates that Vsubject is a variable. We might choose to interpret the constants believe and flat as predicates and the constant earth as an argument of the predicate flat, but there is no need for the matcher to be concerned with these semantic issues. As the term proposition suggests, (believes Vsubject (flat earth)); can be true or false, but what is more important from the point of view of the matcher is whether there is some proposition in the data base like (believes Bill (flat earth)) which can serve as its instantiation.

A full matcher uses the following criteria to decide whether a set of instantiations constitutes

This work was sponsored by NSF grant no. IST-7918474 to Jane Perlrautter and by ONR contract no. N00014-78-C-0725 to John R. Anderson.

a satisfactory match to the left-hand side of a production rule:

1. There must be an instantiation of each condition proposition.

2. The constants contained in a condition proposition must also be contained in its instantiation.

3. There should be a one-one function that associates each variable with a unique binding so that all condition propositions containing that variable have instantiations containing its binding.

4. Each condition proposition should have the "same" structure as its instantiation. In the simplest case they should have the same number of terms and corresponding terms should be in the same order. In some production systems allowances are made for unequal numbers of arguments or symmetric predicates; however, the important point is that there is never any ambiguity about whether an instantiation has the proper structural requirements and those that don't are simply unacceptable.

A. Failure of the Absolute Criteria

We now examine each of these criteria in turn to determine their suitability for partial matching. We will find that while each constitutes a worthwhile goal for a partial matcher to attempt to achieve (so that a full match will be found if one is available), none of these criteria can be an absolute requirement if we want to be able to detect the same similarities that students can detect.

1. Every condition has an instantiation

Textbooks sometimes provide more information in the statement of example problems than is strictly required for their solution. This extra information would prevent the examples from being seen as relevant to the solution of exercise problems not sharing those inessential features if we were to retain the full-match criterion that all propositions in a problem description must receive an instantiation.

Many full matchers will allow two variables to be bound to the same constant; however, additional mechanisms are then required to deal with the inevitable cases where this is inappropriate. These additional mechanisms play no role in partial matching, so it simplifies the discussion to make the one-one assumption.

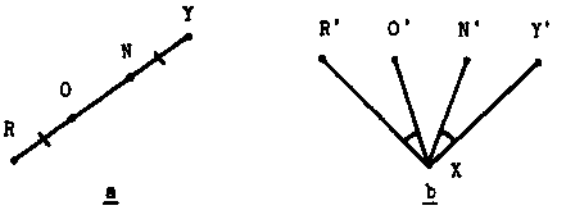
2. Constants

Students find problems involving the algebraic relation "<" very reminiscent of the ">" versions of those problems. In such cases partial matches between these problems would have to violate the full-match criterion that a condition and its instantiation contain the same constants. The unsuitability of full-match criteria (1) and (2) for partial matching is not controversial. All four algorithims reviewed in [4] for inducing a general description of a concept from examples can deal with these kinds of departures from a full match.

3. Variable Binding Requires 1-1 Functions

Imagine a person who has seen many National League baseball games, but is watching American League baseball for the first time. In the National League, besides playing in the field, a pitcher takes his turn at bat; in the American League, however, the pitcher is replaced in the batting lineup by the designated hitter. If the baseball fan tries to instantiate his pattern for National League baseball using the lineups for this game should he bind the variable V-NLpItcher to the pitcher or to the designated hitter? The correct answer seems to be both; to the pitcher if he is instantiating propositions describing the National League pitcher's role in the field, but to the designated hitter when instantiating propositions about the National League pitcher's role as a batter. On the other hand, an American League fan watching National League baseball for the first time has the problem of too few players rather than too many. He is in the position of having the same constant, NLpitcher, as the binding of two different variables, V-ALpitcher and V-ALdesighitter.

4. Only Match Propositions Having the Same Structure



1 b

Fig. 1: *&)* Prove RN * OY. *b)* Prove R'XN' - O'XY'. Marked objects are known to be congruent.

Fig. 1 shows two problems that some students can see as quite similar — Fig. 1b being just an angle version of the problem in Fig. 1a. This apparently requires that a partial matcher accept the proposition (angle VR' VX VO') as an instantiation of the proposition (segment VR VO). That segment can be matched by angle is just another example of matching one constant to another. What is more interesting is that the second argument, VO, of segment must be bound to the third argument, VO/, of angle rather than to the second argument, VX.

DP predicates. Also, when a variable such as V-NLpItcher receives a second binding, all of the propositions mentioning that variable, whether they already have an instantiation using the first binding or not, are checked to see if they have an instantiation using this new binding. Thus RELAX computes all of the kinds of mappings that Kling claims are necessary for capturing the correspondences between similar problems and does so using "best use" criteria which express preferences that are meaningful only to a program that is aware that it has these options regarding the form of its mappings.

A. Best Use of the Condition

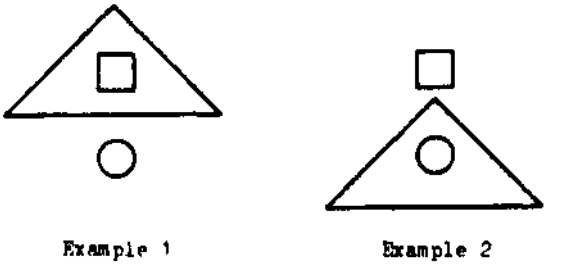
One use for the output from partial matching a set of conditions to a set of data is in computing a generalization, i.e., a new set of conditions which receives a full match when instantiated by either the original conditions or the original data* Since the original conditions and the original data play entirely parallel roles in the definition of generalization, one is led rather naturally (though not Inescapably) to the idea that the output of the partial match should not depend in any essential way on which set of propositions we call data and which set we call conditions. In fact, in implementing RELAX we found that there are computational advantages to viewing the matching process as searching for conditions to instantiate data at the same time as it is searching for data to instantiate conditions. For example, this viewpoint suggests that in addition to the requirement that each instantiation be the best available use of its data proposition, there should also be a complementary requirement that each instantiation be the best available use of its condition proposition. In this third (and last) of RELAX's relative criteria for partial matching, cost is again measured in terms of multiple assignments; but since in this context conditions are thought of as instantiations of data propositions, a multiple assignment means that a constant in the data is the binding for more than one condition variable. The two instantiations mentioned above for (male V-NLpItcher) are equally good in this respect since both ALpitcher and ALdesighitter are bindings of only one variable; thus both instantiations are permitted in the partial match. However, any attempt to get a second instantiation for (male V-KLcatcher) would be ruled out by this new criterion.

IV SOME EXAMPLES

A. A Generalization Example

Fig. 2 shows the generalization problem that led Hayes-Roth & McDermott to be concerned with multiple assignments for variables. The generalization that Hayes-Roth A McDerraott wanted but were unable to get SPROUTER to produce was:

> There is a small square above a small circle and one of these small figures is inside a large triangle.



Example 1 Example 2

Fig. 2: A generalization problem used by Hayes-Roth & McDermott.

When partial matching the description of Example 1 using the description of Example 2 as data, the variable Vsqr1 requires two bindings so that we can have both of the instantiations

$$(\text{above } V_{sqr_1}\ V_{crl_1}) \rightarrow (\text{above } V_{sqr_2}\ V_{crl_2})$$
and
$$(\text{inside } V_{sqr_1}\ V_{tri_1}) \rightarrow (\text{inside } V_{crl_2}\ V_{tri_2}).$$

These two instantiations are taken from Table 1 which shows the full set found by RELAX. The 15 instantiations in Table 1 are the survivors of the 185 instantiations that RELAX considered in computing this match. Of these 185 pairings, 100, or *54%* were rejected by the "best use" criteria without the need for any search at all. In 21 cases, pairings which satisfied these criteria when first made subsequently were rejected by them as the match proceeded and better uses were found for their conditions or their data.

Table 1: The set of instantiations found as the partial match of the examples in Fig. 2.

Example 1		Example 2
$(\text{square } V_{sqr_1})$	->	$(\text{square } V_{sqr_2})$
$(\text{circle } V_{crl_1})$	->	$(\text{circle } V_{crl_2})$
$(\text{triangle } V_{tri_1})$	->	$(\text{triangle } V_{tri_2})$
$(\text{size } V_{sqr_1}\ \text{small})$	->	$(\text{size } V_{sqr_2}\ \text{small})$
$(\text{size } V_{crl_1}\ \text{small})$	->	$(\text{size } V_{crl_2}\ \text{small})$
$(\text{size } V_{tri_1}\ \text{large})$	->	$(\text{size } V_{tri_2}\ \text{large})$
$(\text{samesize } V_{sqr_1}\ V_{crl_1})$	->	$(\text{samesize } V_{sqr_2}\ V_{crl_2})$
$(\text{above } V_{sqr_1}\ V_{crl_1})$	->	$(\text{above } V_{sqr_2}\ V_{crl_2})$
$(\text{below } V_{crl_1}\ V_{sqr_1})$	->	$(\text{below } V_{crl_2}\ V_{sqr_2})$
$(\text{inside } V_{sqr_1}\ V_{tri_1})$	->	$(\text{above } V_{sqr_2}\ V_{tri_2})$
" "	->	$(\text{inside } V_{crl_2}\ V_{tri_2})$
$(\text{below } V_{crl_1}\ V_{tri_1})$	->	" "
$(\text{above } V_{tri_1}\ V_{crl_1})$	->	$(\text{contains } V_{tri_2}\ V_{crl_2})$
$(\text{contains } V_{tri_1}\ V_{sqr_1})$	->	" "
" "	->	$(\text{below } V_{tri_2}\ V_{sqr_2})$

The demonstration that this partial match will lead to the correct generalization will have to be postponed until a later section which gives the details of the approach taken toward disjunctions. However, there are a number of things we can say at this point to defend the view that this is the

correct partial match for these examples. The fact that exactly the same correspondences are obtained when Example 2 is treated as the set of conditions and Example 1 is treated as the data provides a demonstration that RELAX at least functions as Intended. However, how can correspondences such as

$$(\text{inside } Vsqr_1\ Vtri_1) \rightarrow (\text{above } Vsqr_2\ Vtri_2)$$

and

$$(\text{below } Vcrl_1\ Vtri_1) \rightarrow (\text{inside } Vcrl_2\ Vtri_2)$$

be defended? The answer to this is that there is an obvious alternative to viewing Fig. 2 as two examples in need of a generalization. That alternative is to see Example 1 as the situation at time t_1 and Example 2 as the situation at a later time t2, where the task is to characterize the transformation that has occurred. One characterization of the transformation that changes Example 1 into Example 2 is:

> The triangle that originally contains the square moves down to contain the circle.

The questionable correspondences mentioned above have an obvious relevance for characterizing this transformation.

B. A Transformation Example

The next example comes from observations of two subjects learning to program in LISP made by John R. Anderson and his collegues at CMU. As an exercise in their textbook [10] these subjects had to write a LISP function to compute the powerset (i.e., the set of all subsets) of the set (YALL COME BACK). This problem is quite difficult for novices and both subjects had little success until they hit upon the representation for the problem shown in Table 2. Once in this form, it appeared that both subjects simply did some pattern matching to arrive at the following solution: POWERSET(YALL COME BACK) requires two copies of POWERSET(COME BACK) and the second copy must have YALL CONSed onto the front of all of its sublists.

Table 2: The results of successive calls to the function POWERSET(L).

L = (COME BACK)	L = (YALL COME BACK)
( )	( )
(COME)	(COME)
(BACK)	(BACK)
(COME BACK)	(COME BACK)
	(YALL)
	(YALL COME)
	(YALL BACK)
	(YALL COME BACK)

In pattern matching terms, the need for one copy is trivial — it results from a full match of the sets in POWERSET(COME BACK) using the sets in POWERSET(YALL COME BACK) as data. The need for the second copy, however, requires a partial match of POWERSET(COME BACK) to the remaining sets in POVERSET(YALL COME BACK).

When the problem of finding this partial match

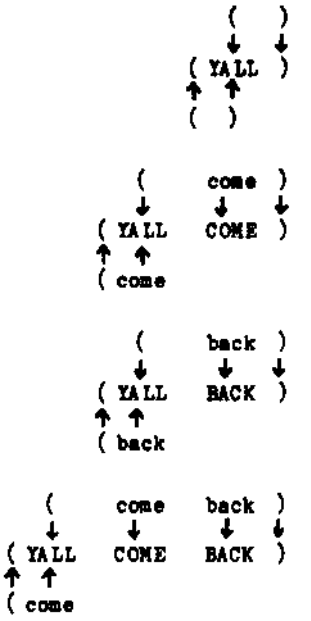


Fig. 3: The partial match obtained for the lists in Table 2. Lowercase entries are from POWERSET (COME BACK); uppercase from POWERSET (YALL COME BACK).

was given to RELAX the results were as shown in Fig. 3. Here each sublist was successfully matched starting from the left end up until the point where the YALL was encountered and also was successfully matched starting from the right end until the YALL was encountered. Extending the match from the left end any farther was ruled out by the best-use-of-the-data and best-use-of-the-condition criteria.

If the results of this partial match could be used as data by other rules, then we could construct a rule that would recognize that an atom had been CONSed onto each sublist because the characteristic transformation produced by CONS is found in each case. In general, however, without some notion of a constituent, REIAX would not be able to detect other cases where a list (rather than an atom) has been CONSed onto each sublist. Thus, while REIAX falls short of accounting for all of the pattern matching that subjects can display in an exercise like this, it does appear to be a promising beginning.

V DISJUNCTIONS

Exclusive disjunctions arise in two different ways in the process of forming generalizations. What we might call external disjunctions have their source in the fact that for any set of examples $e^1, \ldots, e$ there is the logically correct, but unillumlnating, generalization e_1 or e_2 or .•. e_n The goal of generalization programs that attempt to account for external disjunctions [7, 11] is to merge as many of the e_i as possible into conjunctive generalizations so as to minimize the number of disjuncts in the resulting disjunctive normal form generalization.

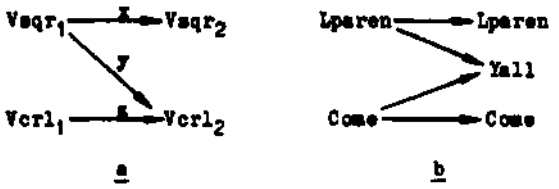
Internal disjunctions, by contrast, do not emerge until a partial match has been computed between two examples e_i and e_j. Then each instantiation in the partial match

$$(\underline{rel}\ \underline{a}_1 \ \ldots\ \underline{a}_m) \rightarrow (\underline{rel'}\ \underline{a}_1' \ \ldots\ \underline{a}_m')$$

yields the again logically correct, but unillumInating, generalization

$$((\underline{rel}\ \text{or}\ \underline{rel'})\ (\underline{a}_1\ \text{or}\ \underline{a}_1')\ \ldots\ (\underline{a}_m\ \text{or}\ \underline{a}_m').$$

Just as in the case of external disjunctions, the way to produce pleasing generalizations seems to be to remove as many disjunctions as possible. The simplest case of this is the reduction of all disjunctions of the form (c or *c)* to the single constant £. Less obvious cases where disjunctions can be removed are best understood by reference to Fig. 4a which shows a portion of the full network of bindings that can be extracted from Table 1.



a b

Fig. 4: A summary of selected variable bindings ja) from Table 1 and b) from Fig. 3.

REIAX operates on this network in order to determine a way of realizing each of these bindings in a generalisation. The goal is to choose a different label for each link in this network, where possible labels are the terms at either end of a link or the disjunction of those terms. Link *x* in Fig. 4a receives the label $\underline{Vsqr_1}$ because three "conditions are satisfied: 1) $\underline{Vaq_2}$ has only one connection to the network, 2) $\underline{Vaqr_1}$ can have the binding $\underline{Vsqr_2}$ in a full match, and 3) the label $Vsqr_1$ is not already used as the label for some other link. Link z receives the label Verlg because these conditions are also satisfied there. Link y is then forced to have the label ($\underline{Vsqr_1}$ or $VcjJ_3$)[f] because both of its endpoints have already been used as labels.

Table 3: The generalization resulting from the partial match in Table 1.

```
(inside (Vsqr1 or Vcrl2) Vtri)
(contains Vtri (Vaqr1 or Vcrl2))
((below or inside) Vcrl2 Vtri)
((inside or above) Vaqr1 Vtri)
(samesize Vaqr1 Vcrl2)
(above Vaqr1 Vcrl2)    (below Vcrl2 Vaqr1)
(size Vaqr1 small)     (size Vcrl2 small)
(size Vtri large)      (triangle Vtri)
(square Vaqr1)         (circle Vcrl2)
```

REIAX produces the generalization in Table 3 from the instantiations in Table 1. It can be seen that Table 3 contains propositions expressing the generalization that Hayes-Roth & NcDermott wanted for the examples of Fig. 2.

Example 1 from Fig. 2 producea a full match to this generalization by matching the flrat member of every disjunction, while Example 2 produces a full match by matching the second member of every disjunction. In fact, there is a new full-match criterion requiring that the same position be matched in all disjunctions in a rule. Now it should come as no suprise that when partial matching we will want to violate this criterion and match different positions or even both positions of some disjunctions. However, this anticipates an approach to partial matching disjunctions that there is insufficient space to elaborate on here.

Although the natural way of viewing the partial match found in the POWERSET example presented above is as a characterization of the tranaformations required to go from POWERSET(COME BACK) to POWERSET(YALL COME BACK), it is also informative to look at the generalization that would result from this partial match. Fig. 4b shows the pattern of bindings found for the partial match of the sublist (COME) to the sublist (YALL COME) and indicated convergences that should lead to disjunctions. The generalization obtained for this sublist (ignoring the need for type-token distinctions) is:

(before Lparen (Come or Yall))
(before (Lparen or Yall) Come)
(before Come Rparen)

This generalization provides a disjunctive representation of the fact that YALL ia an optional firat element of these lists. In the case where the YALL's are present, the second position of every disjunction is matched and the firat two propositions of this generalization are inatantiated by (before Lparen Yall) and (before Yall Come), respectively. On the other hand, when the YALL's are absent, the first position of every disjunction is matched so in this case these two propositions are both instantiated by the single proposition (before Lparen Come).

A• Implications for Semantic Approaches

Obtaining a sensible generalization for this problem is important because it helps bolster our confidence in the unorthodox correspondences made by REIAX in partial matching these lists. If a left parenthesis must be matched to the atom YALL to get the correct generalization, then what about partial-matching programs that compare the semantic categorization of terms before making assignments? Surely these two terms are sufficiently different semantically that such programs should be reluctant to see them aa corresponding.

How serious a problem this poses depends largely on whether these partial matchers use semantic relatedness as an abeolute criterion or aa a relative one. Programs such as Kling's ZORBA [9] which make semantic relatedness an absolute criterion will either reject the assignement of LPAREN to YALL or will have to decrease the amount of semantic relatedneas required to the point where this criterion has little ability left to discriminate useful from useless matches. On the other hand, in Winston's program [12] the closer

two terms are semantically the more "points" that assignment contributes to an overall match score which determines the best partial match. By making semantic relatedness a relative criterion in this manner, Winston allows for the possibility that other factors can compensate for the semantic divergence of two terms. (Winston's partial matcher relies totally on one-one mappings, so presumably there is no way for his program to produce the correct generalization for this particular example, however.)

VI THE LAST ABSOLUTE CRITERION

Now RELAX does obey one absolute criterion, namely, that a proposition and its instantiation must have the same structure. However, as was discussed above, the problem in Fig. 1β may be seen as a version of the problem in Fig. 1a using angles instead of segments. Detecting the similarity of these problems requires that the condition (segment VR VO) have the instantiation (angle VR' VX VO'). To enable RELAX to find the correct partial match for this example we followed the lead of Hayes-Roth & McDermott (cf. their use of SCR's) and switched to a more fine-grained representation. Finer grain was obtained by "exploding" each proposition in the description of these problems in such a way that every link in the semantic network representation of that proposition itself becomes a full proposition in the exploded version. Thus (segment VR VO) becomes the set of propositions

(Rel segment VsegRO)
(Arg1 VR VsegRO)
(Arg2 VO VsegRO).

For these condition propositions, RELAX finds the instantiations

(Rel segment VsegRO) -> (Rel angle VangR'XO')
(Arg1 VR VsegRO) -> (Arg1 VR' VangR'XO')
(Arg2 VO VsegRO) -> (Arg3 VO' VangR'XO').

No use is made of the data proposition (Arg2 VX VangR'XO').

RELAX was able to calculate the correct partial match for the two problems in Fig. 1 when they were described in this detail. In the process, Arg2 was matched to Arg3 five times and to Arg2 three times. This success suggests that the relative criteria outlined in this paper are also adequate, at least in principle, for matching different structures. However, finding the 36 instantiations in the final match required searches to evaluate 230 candidate instantiations. Thus it was clear that by recoding whole problems into this fine-grained representation we had placed a large computational burden on REIAX. Work is in progress on defining principles that would permit the program to switch dynamically to the more fine-grained representation for just those portions of the problem description where it is required. This capability would effectively free RELAX from the need to rely on any absolute criteria whatsoever.

ACKNOWLEDGEMENTS

The author would like to acknowledge the hundreds of hours of discussion that he has had with John R. Anderson about these topics. The lack of a shared consensus on these issues has, if anything, only made these discussions more valuable. Helpful comments on a previous draft of this paper were also provided by G. Iba, J. Perlmutter, M. Schustack, and the IJCAI reviewers.

REFERENCES

[1] Anderson, J.R.
Language, memory, and thought.
Lawrence Erlbaum Associates, Hillsdale, N.J., 1976.

[2] Anderson, J.R. and Kline, P.J.
A Learning System and its Psychological Implications.
Proc. of 6th IJCAI :16-21, 1979.

[3] Anderson, J.R., Greeno, J.G., Kline, P.J., and Neves, D.M.
Acquisition of Problem Solving Skill.
In Anderson, J.R., editor, Cognitive Skills and their Acquisition. Lawrence Erlbaum Assoc, Hillsdale, N.J., 1981.

[4] Dietterich, T.G. and Michalski, R.S.
Learning and Generalisation of Characteristic Descriptions: Evaluation Criteria and Comparative Review of Selected Methods.
Proc. of 6th IJCAI :223-231, 1979.

[5] Hayes-Roth, F. and McDermott, J.
Knowledge Acquisition from Structural Descriptions.
Proc. of 5th IJCAI :356-362, 1977.

[6] Hayes-Roth, F. and McDermott, J.
An Interference Matching Technique for Inducing Abstractions.
CACM 21(5):401-410, 1978.

[7] Iba, G.A.
Learning Disjunctive Concepts from Examples.
A.I. Memo 548, M.I.T., Sept., 1979.

[8] Jurgenson, R.C., Donnelly, A.J., Maier, J.E., and Rising, G.R.
Geometry*
Houghton-Mifflin, Boston, 1975.

[9] Kling, R.E.
A Paradigm for Reasoning by Analogy.
Artificial Intelligence 2:147-178, 1971.

[10] Siklossy, L.
Let's talk LISP.
Prentice-Hall, Ehglewood Cliffs, N.J., 1976.

[11] Vere, S.A.
Inductive Learning of Relational Productions.
In Waterman, D.A. & Hayes-Roth, F., editor, Pattern-Directed Inference Systems.
Academic Press, 1978.

[12] Winston, P.H.
Learning and Reasoning by Analogy.
Communications of the ACM 23(12):689-703, 1980.