

April 1971

REASONING BY ANALOGY AS AN AID TO HEURISTIC THEOREM PROVING

by

Robert E. Kling

Paper accepted for presentation at IFIP Congress '71,  
Ljubljana, Yugoslavia, August 23-28, 1971

Artificial Intelligence Group

Technical Note 49R

SRI Project 8259

The research reported herein was sponsored by the Advanced  
Research Projects Agency and the National Aeronautics and  
Space Administration under Contract NAS12-2221.

# Report Documentation Page

Form Approved  
OMB No. 0704-0188

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

1. REPORT DATE <b>APR 1971</b>		2. REPORT TYPE		3. DATES COVERED <b>00-00-1971 to 00-00-1971</b>	
4. TITLE AND SUBTITLE <b>Reasoning by Analogy as an Aid to Heuristic Theorem Proving</b>				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S)				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) <b>SRI International, 333 Ravenswood Avenue, Menlo Park, CA, 94025</b>				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT <b>Approved for public release; distribution unlimited</b>					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT					
15. SUBJECT TERMS					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT <b>Same as Report (SAR)</b>	18. NUMBER OF PAGES <b>6</b>	19a. NAME OF RESPONSIBLE PERSON
a. REPORT <b>unclassified</b>	b. ABSTRACT <b>unclassified</b>	c. THIS PAGE <b>unclassified</b>			

# REASONING BY ANALOGY AS AN AID TO HEURISTIC THEOREM PROVING

ROBERT E. KLING  
Stanford Research Institute  
Menlo Park, California

When heuristic problem-solving programs are faced with large data bases that contain numbers of facts far in excess of those needed to solve any particular problem, their performance rapidly deteriorates. In this paper, the correspondence between a new unsolved problem and a previously solved analogous problem is computed and invoked to tailor large data bases to manageable sizes. This paper outlines the design of an algorithm for generating and exploiting analogies between theorems posed to a resolution-logic system. These algorithms are believed to be the first computationally feasible development of reasoning by analogy to be applied to heuristic theorem proving.

Any contemporary heuristic deductive theorem-proving system that proves theorems by applying some rules of inference to an explicit set of axioms must use a carefully tailored data base. Most search procedures will generate many irrelevant inferences when seeking the proof of some nontrivial theorem even when they are given a minimal set of axioms. Generally, the effective power of a search procedure is limited by the memory capacity of a particular system: most theorem provers run out of space (absorbed by irrelevant inferences) before they run out of time when they fail to prove a hard theorem.\*

Consider a particular theorem  $T_A$  that can be proved with a set of axioms  $D$ . Suppose that a theorem-prover  $S$  can prove  $T_A$  within its memory limitations. Suppose  $D$  is expanded to  $D'$  by adding axioms that include many of the same relations that appear in  $D$ . If  $S$  attempts  $T_A$  again, it will generate many new irrelevant inferences that are derived from the axioms in  $D' - D$ . In fact, the size of  $D'$  need not be too much larger than that of  $D$  to render  $T_A$  unprovable by  $S$ . Typical theorem provers work with a  $D'$  composed of less than 20 axioms. If  $T_A$  is hard for  $S$ , then just a few additional axioms may add a sufficient number of inferences to the search space to exhaust the memory before a solution is found. In the '60's, most research focused on the organization of  $S$  and the development of a variety of ever-more-efficient search procedures. Consequently, researchers could choose an optimal  $D'$  for each particular theorem without sacrificing their research goals. In contrast, as heuristic deductive systems are being proposed to solve real-world problems, such as robot manipulation [1], larger nonoptimal data bases are necessary.

Suppose we have a theorem  $T_A$  and a large data base  $D'$ . In general, there is no way to choose a small subset  $D$  of  $D'$ , such that  $D = T_A$ . Suppose we had previously solved some theorem  $T$  that is analogous to  $T_A$  in so far as analogs of the axioms used in the proof of  $T$  will be required in the proof of  $T_A$ . If we could generate the analogy

between  $T_A$  and  $T$  to find the set of axioms and use them as  $D$ , then we could let  $S$  attempt to prove  $T_A$  with greater hope of success. This paper describes a set of algorithms for generating an analogy between some given pair of analogous theorems and for exploiting this analogy to estimate  $D$ .

The preceding discussion has been rather general and applies to any heuristic theorem prover such as LT [3] and resolution [4]. However, each paradigm will require slightly variant representations and methods for generating and using analogical information. Effective research demands working with a specific theorem prover; for reasons of convenience, I have chosen QA3, [5] a resolution-based theorem prover. QA3 and the algorithm ZORBA-I, described below, are implemented in LISP on a PDP-10 at Stanford Research Institute.

This paper is devoted to briefly motivating and outlining the ZORBA-I algorithms. Detailed explication requires considerably more space than is available here. I recommend that the reader whose interest is excited by this cursory account explore the lengthier accounts [6], [7] which supply all or some of the missing details.

Before describing ZORBA-I abstractly, I want to exemplify the kinds of theorems that it tackles. Briefly, they are theorem-pairs in domains which can be axiomatized without constants (e.g., mathematics) that have one-to-one analogies between their predicates. The theorems that follow are fairly hard for QA3 to solve even with an optimal memory. For example, ZORBA-I will be given the proof of the theorem

- T1. The intersection of two abelian groups is an abelian group,
- and is asked to generate an analogy with
- T2. The intersection of two commutative rings is a commutative ring;
- or, given
- T3. A factor group  $G/H$  is simple iff  $H$  is a maximal normal subgroup of  $G$ , and its proof, ZORBA-I is asked to generate an adequate analogy with
- T4. A quotient ring  $A/C$  is simple iff  $C$  is a maximal ideal in  $A$ .

\*This observation is based upon my own experience with resolution systems and is corroborated by other researchers using different paradigms [2], [3].

$T_1$  has a 35-step proof and  $T_3$  has a 50-step proof in a decent axiomatization. A good theorem prover (QA3) generates about 200 inferences in searching for either proof when its data base is minimized to the 13 axioms required for the proof of  $T_1$  or to the 12 axioms required for the proof of  $T_3$ . If the data base is increased to 30 reasonable axioms, the theorem prover may easily generate 600 clauses and run out of space before a proof is found. Note also that the predicates used in the problem statement of these theorems contain only a few of the predicates used in any proof. Thus,  $T_2$  can be stated using only the predicates {INTERSECTION; ABELIAN}, but a proof will use in addition {GROUP; IN; TIMES; SUBSET; SUBGROUP; COMMUTATIVE}. Thus, while the first set must map into {INTERSECTION, COMMUTATIVERING} which appear in the statement of  $T_2$ , the second set can map into anything.

ZORBA-1 considers an analogy to be a set of three maps:

- (1) A one-one map between predicates
- (2) A many-many map between the variable that appears in the statements of  $T$  and  $T_A$ .
- (3) A one-many map between the axioms in the proof of  $T$  and some of the clauses that appear in the large data base  $D'$ .

It begins with no a priori information about which particular predicates or clauses are to be analogous, and creates a mapping (analogy) in which it finds analogs for each predicate and clause used in the proof[ $T$ ].

Figure 1 shows a set  $P$  including all the predicates in the data base. Let  $P'_1$  and  $P'_2$  be the set of predicates in the statements of the new and old theorems,  $T_A$  and  $T$ . In addition, we know the predicates  $P_1$  in some proof of  $T$  (since we have a proof at hand). We need to find the set  $P_2$  that contains the predicates we expect in some proof of  $T_A$ , and we want a map\*  $G: G(P_1) = P_2$ . In our example on the last page  $P'_1 = \{\text{INTERSECTION, ABELIAN}\}$  and  $P'_2 = \{\text{INTERSECTION, COMMUTATIVERING}\}$ .

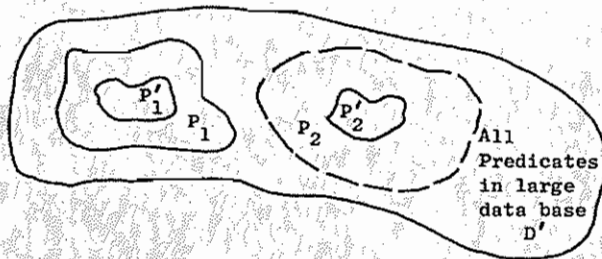


FIG. 1

VENN DIAGRAM OF RELATIONS IN STATEMENTS  $T$ ,  $T_A$  and  $D'$

\* Analogies will be denoted by script  $G$  (and possible subscripts) e.g.,  $G_3, G_k$ .

Clearly, a wise method would be to find some  $G_1$ , a restriction of  $G$  to  $P'_1$  such that  $G_1(P'_1) = P_2$ , and then incrementally extend  $G_1$  to  $G_2, G_3, \dots$ , each on larger domains until some  $G_n(P_n) = P_2$ . Each incremental extension  $G_j$  differs from its predecessor  $G_{j-1}$  in that it includes the analog of at least one predicate and one axiom which does not appear on  $G_{j-1}$ . Each of these successive mappings is called a "partial analogy" in contrast to a "complete analogy" which includes the analog of every predicate and every clause used in the proof of  $T$ . ZORBA-1 performs in such a way that each incremental extension picks up new clauses that could be used in a proof of  $T_A$ . ZORBA-1 first computes  $G_1$  which maps  $P'_1$  onto  $P_2$  by using a special program called INITIAL-MAP. This stage is designed to use the syntax of the statements of  $T$  and  $T_A$  to assist in associating atoms (and hence predicates). It then passes this starting analogy (usually only one) to a program called EXTENDER which is designed to develop a series of extensions  $G_j$  to the initial  $G_1$ . When it attempts to increment a partial analogy  $G_j$ , EXTENDER sweeps the larger memory  $D'$  searching for the analogs of selected clauses from proof[ $T$ ] according to  $G_j$ . If it finds new clauses, it has information for adding new predicate associations to  $G_j$ , and create an  $G_{j+1}$ . In addition, these analog clauses are likely to be used in the proof of  $T_A$ . If we get no new clauses from an extended  $G_j$ , that may be reason to believe that  $G_j$  is faulty. I now will describe the generation algorithm in more detail.

The user presents ZORBA-1 with the following information:

- (1) A new theorem,  $T_A$ , to prove.
- (2) An analogous theorem,  $T$  (chosen by the user), that has already been proved.
- (3) The proof of theorem  $T$ , proof[ $T$ ], which is an ordered set of clauses  $\{c_k\}$  such that  $\forall k, c_k$  is either
  - (a) A clause in  $\neg T$
  - (b) An axiom
  - (c) Derived by resolution from two clauses  $c_i$  and  $c_j$   $j < k$  and  $i < k$ .

These three items of information are problem-dependent. ZORBA-1 accesses a large data base which includes more axioms than it needs for  $T$  or  $T_A$  and is, in this sense, problem independent. In addition, the user specifies a "semantic template" for each predicate in his language. This template associates a (semantic) type with each predicate and predicate-place and is used to help constrain the predicate mappings to be meaningful. For example, (STRUCTURE SET OPERATOR) is associated with the predicate "group." Thus, ZORBA-1 knows that "group" is a structure, "A" is a set, and "\*" is an operator when it has seen group[A;\*]. ZORBA-1 allows only associations between predicates of the same type. The predicate-place types are used to assist associating the arguments of atoms of different order with each other. Currently, the predicate types (for algebra) are STRUCTURE, RELATIONS, MAP, and RELSTRUCTURE; the variable types are SET, OPERATOR, FUNCTION and

OBJECT. These semantic templates are critical for the operation of INITIAL-MAP and only incidentally by EXTENDER. Instead, EXTENDER uses an entity called a "clause description."

A description,  $\text{descr}[c]$ , can be made for any clause  $c$  according to the following rules:

- (1)  $\forall p$  If both  $p$  and  $\neg p$  appear in  $c$ , then  $\text{impcnd}[p] \in \text{descr}[c]$ .
- (2)  $\forall p$  If  $p$  appears in  $c$ , then  $\text{pos}[p] \in \text{descr}[c]$ .
- (3)  $\forall p$  If  $\neg p$  appears in  $c$ , then  $\text{neg}[p] \in \text{descr}[c]$ .

Thus, the axiom, "every abelian group is a group," e.g.,

$$\forall(x*) \text{abelian}[x;*] \Rightarrow \text{group}[x;*],$$

is expressed by the clause

$$e_1: \neg \text{abelian}[x;*] \vee \text{group}[x;*],$$

which is described by

$$d_1: \text{neg}[\text{abelian}], \text{pos}[\text{group}].$$

The theorem, "the homomorphic image of a group is a group," e.g.,

$$\forall(x y *1 *2 \varphi)$$

$$\text{hom}[\varphi;x;y] \wedge \text{group}[x;*1] \\ \Rightarrow \text{group}[y;*2],$$

is expressed by the clause

$$c_2: \neg \text{hom}[\varphi;x;y] \vee \neg \text{group}[x;*1] \vee \text{group}[y;*2]$$

and is described by

$$d_2: \text{neg}[\text{hom}], \text{impcnd}[\text{group}].$$

These clause descriptions are used by EXTENDER when it is seeking the analog of a particular axiom. When it wants to find the analog of an axiom  $ax_k$ , in an attempt to extend an analogy  $G_j$ , then it searches  $D'$  for some clause which "satisfies" the description  $G_j[\text{descr}[ax_k]]$ . A clause  $c$  is said to satisfy a description  $d$  iff  $d \subseteq \text{descr}[c]$ . Thus,  $c_1$ , but not  $c_2$  satisfies  $\text{pos}[\text{group}]$ . The analog of the description of  $ax_k$  under  $G_j$ ,  $G_j[\text{descr}(ax_k)]$ , is created by substituting each predicate in  $\text{descr}[c]$  with its analog that appears in  $G_j$ . If an analogy  $G_j$  associates:

hom	hom
group	ring
abelian	commutativering

then,

$$G_1[\text{descr}[c_1]] = \text{neg}[\text{commutativering}], \text{pos}[\text{ring}]$$

and

$$G_1[\text{descr}[c_2]] = \text{neg}[\text{hom}], \text{impcnd}[\text{ring}].$$

ZORBA-I operates in two stages. INITIAL-MAP is applied to the statements of  $T$  and  $T_A$  to create an  $G_1$  which is used by EXTENDER to start its sequence of partial analogies  $G_i$  which terminates in a complete analogy  $G$ . INITIAL-MAP

starts without a priori information about the analogy it is asked to help create. It utilizes the syntax of the wffs which express  $T$  and  $T_A$  as well as the semantic templates to generate  $G_1$  which includes all the predicates (and variables) which appear in the statement of  $T$ . For example, the statements of  $T_1 - T_2$  can contain three of the nine predicates used in  $\text{proof}[T]$ , and the statements of  $T_3 - T_4$  can contain five of the nine predicates used in  $\text{proof}[T_3]$ . The INITIAL-MAP uses a rule of inference called ATOMMATCH [ $\text{atom}_1; \text{atom}_2; G$ ] which extends analogy by adding the predicates and mapped variables of  $\text{atom}_1$  and  $\text{atom}_2$  to analogy  $G$ . ATOMMATCH now limits ZORBA-I to analogies where atoms\* map one-to-one.

INITIAL-MAP is a sophisticated search program that sweeps ATOMMATCH over likely pairs of atoms, one from the statement of  $T$ , the other from the statement of  $T_A$ . Alternative analogies are kept in parallel (no backup), and INITIAL-MAP terminates when it has found some analogy that includes all the predicates in theorem statements. Only one analogy is output.†

EXTENDER accepts the initial analogy generated by INITIAL-MAP and uses it as the first term in a sequence of successive analogies  $G_j$ . In addition it accesses both to the large data base  $D'$  used by the theorem prover and the (unordered) set of axioms used in  $\text{proof}[T]$ . The axioms used in  $\text{proof}[T]$  are called AXSET and are few in comparison to the size of the data base  $D'$  and comprise the "domain" for a complete  $G$ . For each axiom in AXSET, we want to find a clause from  $D'$  which is analogous to it. Now, EXTENDER treats AXSET in a special way by partitioning it into three disjoint subsets called ALL, SOME and NONE.

If all the predicates on an axiom  $ax_k$  are in  $G_j$ ,  $ax_k \in \text{ALL}$ , if some of its predicates are in  $G_j$ ,  $ax_k \in \text{SOME}$ , and if none of its predicates are in  $G_j$ ,  $ax_k \in \text{NONE}$ . This partition is trivial to compute, and initially none or a few  $ax_k \in \text{ALL}$ , and most  $ax_k$  belong to SOME and NONE. We want to develop a sequence of analogies  $G_j$  that contain an increasingly larger set of predicates and their analogs. If an axiom is contained in ALL, then by definition we know the analogs of each of its predicates. It can't assist us in learning about new predicate associations. In contrast, we know nothing about the analogs of any of the predicates used in axioms contained in NONE. Analog clauses for these axioms are hard to deduce since we have no relevant information to start a search. Unlike these two extreme cases, the axioms in SOME are especially helpful and will become the focus of our attention. For each such axiom we know the analogs of some of its predicates from  $G_j$ . These

\* Atoms, not predicates.

† In addition, it is rather fast. It generates the analogy for  $T_1 - T_2$  with about 2 seconds of PDP-10 CPU time.

provide sufficient information to begin a search for clauses which are analogous to them. Thus, the analogs of axioms in SOME provide a bridge between the known and the unknown, between the current  $G_j$  and a descendant  $G_{j+1}$ . When EXTENDER has satisfactorily terminated, ALL = AXSET SOME = NONE =  $\emptyset$ . So the game becomes finding some way to systematically move axioms from NONE to SOME to All in a way that for each  $ax_k$  moved, some image  $G_j(ax_k) = ax'_k$  is found that can be used in the proof of  $T_A$ . Moreover, each new association of clauses should help us extend  $G_j \rightarrow G_{j+1}$  by providing information about predicates not contained in  $G_j$ . When we finally associate an axiom with its analog we can match their respective descriptions and associate the predicates of each that don't appear on  $G_j$ . We can extend  $G_j$  to  $G_{j+1}$  by seeking the analogs of axioms on SOME.

When image clauses are sought, all the clauses that satisfy a particular description are sieved out of the data base. EXTENDER creates the analog of the description of each clause in SOME under the current analogy  $G_j$ . Usually [6] this description is restricted to include only those predicates that appear on  $G_j$ . For example, if our analogy contained only

$$G_j: \text{group} \Leftarrow \text{ring}$$

and we were seeking an analog of clause  $c_2$ , we would look for any clause that satisfies the description "impcnd[ring]". If  $G_j$  were more complete, for example,

$$G_j: \text{group} \Leftarrow \text{ring} \\ \text{hom} \Leftarrow \text{hom}$$

we would look for clauses that satisfy neg[hom], impcnd[ring]. In the first case EXTENDER would find several clauses, while in the latter case, certainly fewer would be selected. The actual number depends upon the data base  $D'$ . Often, there will be some axiom  $ax_k \in \text{SOME}$  which has but one candidate image under  $G_j$ . EXTENDER will attempt to extend  $G_j$  by mapping just this one pair of clauses, and then iterate with  $G_{j+1}$  as its active analogy. When no clause returns only one candidate, EXTENDER uses an ordering relation to select the most likely image out of the set of candidates.

Theorem  $T_1$  described above required the axiom:

$$c_3: \neg \text{int}[x;y;z] \vee \text{subset}[x;y] ,$$

which is described by: pos[subset], neg[int]. When the system searches memory for all clauses that satisfy this description, it finds, in addition

$$c_4: \neg \text{int}[x;y;z] \vee \text{subset}[x;z] ,$$

which has an identical description. ZORBA-1 discriminates clauses only in terms of their descriptions and does not discriminate between these two clauses. For most clauses  $c$ ,  $c$  is the only clause that satisfies descr[ $c$ ]. But some clauses, such as  $c_3$  above, may have two or more "description equivalents." In practice, we find few such clauses.

Given a clause  $ax_k \in \text{SOME}$  with description  $d_k$ , its image set  $ax'_k$ , and the partial analogy  $G_j$  developed at this point, EXTENDER picks up the analog information regarding the new predicates appearing in  $ax_k$  and  $ax'_k$  by deleting from  $d_k$  and  $d'_k$  all the terms referencing the predicates  $G_j$ . If there is one term left in  $d_k$  and  $d'_k$ , the corresponding predicates are mapped by default. If more terms are left, the predicates are mapped in a way that preserves (1) description features (e.g., pos terms are associated with pos terms) and (2) semantic types of predicates.

If the system knows that

$$\text{abelian} \Leftarrow \text{cring}$$

and wants to associate the clause (from AXSET)

$$\neg \text{abelian}[x;*] \vee \text{commutative}[*;x]$$

with the clause

$$\neg \text{cring}[x;*;+] \vee \text{commutative}[*;x]$$

it compares the description

$$\text{neg}[\text{abelian}], \text{pos}[\text{commutative}]$$

with

$$\text{neg}[\text{cring}], \text{pos}[\text{commutative}]$$

and extends the analogy to include commutative  $\Leftarrow$  commutative.

The preceding discussion provides an introduction to the ZORBA-1 algorithm. Figure 2 describes the relationship between ZORBA-1 and QA3. While EXTENDER iterates through the partitions

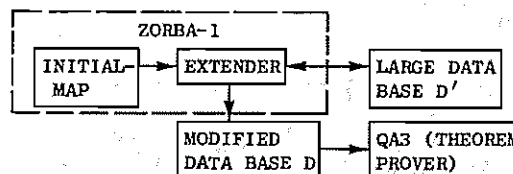


FIG. 2

RELATIONSHIP BETWEEN SECTION OF ZORBA-1 and QA3

of AXSET to create a final analogy, it accesses  $D'$  and builds up a small set of images of the clauses on AXSET. When it terminates (all clauses have images), it passes this image set into the memory of QA3, which then attempts to prove  $T_A$  using the restricted data base.

At this time, the INITIAL-MAP and EXTENDER run on problem pairs in algebra such that  $T_1-T_2, T_3-T_4$ . A large data base of 250 clauses includes the axioms needed for these proofs but is much too large for QA3 to use in any effective way. In effect, without ZORBA-1, QA3 cannot prove any of these theorems using the full data base.

Theorems  $T_1$  and  $T_2$  each require 13 axioms, whereas  $T_3$  and  $T_4$  require 12. When ZORBA-I is asked to find an axiom set for  $T_2$  given the proof of  $T_1$  and the 250 clause algebraic data base, it finds 16 axioms, which include the necessary 13 in about 16.5 seconds. When it is applied to  $T_4$  given a proof of  $T_3$ , it finds 15 axioms, including all the necessary 12. In both cases, the QA3 is able to prove the new theorems ( $T_3$  and  $T_4$ ) with little more search than a humanly selected optimal data base would generate.

#### SUMMARY

The preceding sections described a specific (implemented) algorithm for generating the analogy between a new and an old problem, extracting pragmatically important information from this analogy to aid a problem solver in its search for the solution to the new problem. The system knows none of the associations that constitute the analogy in advance, although it does have a description of some of the semantics (templates) of the language. It can generate analogies that involve many relations (predicates) but is implemented to meet several severe restrictions. In particular

- (1) The problem solver is a resolution-logic based system with one rule of inference.
- (2) The extracted information takes only the form of a problem-dependent data base.
- (3) The analogies involve one to one between predicates.

None of the restrictions is necessary, and weakening is quite possible. In general, ZORBA-I restricts the environment on which its associated problem solver (QA3) operates. This approach circumvents the need for a sequential planning language and detailed information specifying exactly how each (analog) axiom is to be used. Nevertheless, the analogy generator is nontrivial and needs only a simple semantic type theory represented by templates to supplement the problem-solving language (first-order predicate calculus). Although the analogies are not generated by a formal inference system, they provide information which helps a highly formal theorem-proving system prove theorems more efficiently.

#### ACKNOWLEDGEMENT

The research reported here was sponsored by the Advanced Research Projects Agency and the National Aeronautics and Space Administration under Contract NAS 12-2221.

#### REFERENCES

- [1] C. Green, Application of Theorem Proving to Problem Solving, Proc. Int'l. Joint Conf. on Artificial Intelligence, Washington, D.C. (May 1969).
- [2] G. Ernst and A. Newell, GPS: A Case Study in Generality and Problem Solving, ACM Monograph Series (Academic Press 1969).

- [3] A. Newell, J. C. Shaw and H. Simon, Empirical Explorations with the Logic Theory Machine, in Computer and Thought, E. Feigenbaum and J. Feldman, eds. (McGraw-Hill, New York, 1963).
- [4] N. J. Nilsson, Problem-Solving Methods in Artificial Intelligence (McGraw-Hill Book Company, 1971).
- [5] C. Green, Theorem Proving by Resolution as a Basis for Question-Answering Systems, in Machine Intelligence 4, B. Meltzer and D. Michie, eds. (American Elsevier Publishing Co., Inc., New York, 1969).
- [6] R. Kling, A Paradigm for Reasoning by Analogy, to be presented at the 2nd International Joint Conference on Artificial Intelligence, London, England (September 1-3, 1971).
- [7] R. Kling, Reasoning by Analogy with Applications to Heuristic Problem Solving: A Case Study, Stanford University Ph.D. Thesis, forthcoming.