INTRODUCTION

The research reported here is an investigation into the development of a computer program with general problem solving capabilities. This investigation involved the construction of one such computer program called the General Problem Solver (GPS, although more properly GPS-2-6) which was accomplished by modifying an existing program conceived in 1957 by A. Newell, J. C. Shaw, and H. A. Simon. (See references 1,2,3,4,5,6,7,8,9,10,11,12.)

The emphasis in this research is on the generality of GPS—on the variety of problems which GPS can attempt to solve. The quality of the problem solving exhibited by GPS is only a secondary consideration. Hence, the kind of problems for which GPS was designed are simple according to human standards. A typical problem is the missionaries and cannibals task in which there are three missionaries and three cannibals who want to cross a river. The only means of conveyance is a small boat with a capacity of two people, which all six know how to row. If, at any time, there are more cannibals than missionaries on either side of the river, those missionaries will be eaten by the cannibals. How can all six get across the river without any missionaries being eaten?

Another sample task is that of integrating, symbolically, a simple integral such as

\[ \int e^t \, dt. \]

This problem is apparently quite different from the missionaries and cannibals task, but GPS has the generality, as well as the ability, to solve both of these problems.

Although GPS-2-5 was designed to be general, it, together with its predecessors, only solved three different kinds of problems due mainly to inadequate facilities for representing tasks. The central problem of this research is to generalize GPS-2-5 so that it can attempt a wider variety of problems. We also demand that the formulation of problems for GPS requires no knowledge of the internal structure of the program. Underlying this specific objective is the desire to shed light on some of the issues involved in designing better representations for problem solvers.

This is a brief statement of the problem. Section A gives a more detailed description of the problem on which this research focuses. The organization of GPS is described in Section B. Section C describes the representation of problems used by GPS-2-5 and Section D describes the generalizations incorporated in GPS. Section E gives a concrete example of a task solved by GPS. The results are summarized in section F.

A. The approach

We may consider a problem solver to be a process that takes a problem specification as input and, if
Heuristic search

In a simplified form of the heuristic search paradigm, there are objects and operators, such that an operator can be applied to an object to produce either a new object or a signal that indicates inapplicability. A heuristic search problem is:

Given:

a. an initial situation represented as an object,
b. a desired situation represented as an object,
c. a set of operators.

Find: a sequence of operators that will transform the initial situation into the desired situation.

The first operator of the solution sequence is applied to the initial situation, the other operators are applied to the result of the application of the preceding operator, and the result of the application of the last operator in the sequence is the desired situation.

The operators are rules for generating objects and thus define a tree of objects. Each node of the tree represents an object, and each branch of a node represents the application of an operator to the object represented by the node. The node to which a branch leads represents the object produced by the application of the operator. A method for solving a heuristic search problem is searching the tree, defined by the initial situation and the operators, for a path to the desired situation.

For many problems we know of no obvious heuristic search formulation. Thus, in some sense adopting heuristic search limits the generality that can be achieved. However, heuristic search derives its appeal from its generality, demonstrated by its wide use in other research efforts into problem solving (discussed in Chapter II of 1).

The problem of generality

The power of a problem solver is indicated by the effectiveness of its problem solving techniques while its generality is indicated by the domain of problems that it can deal with.* The generality and the power of a problem solver are not independent because both depend strongly upon the internal representation. The internal representation is pulled in two directions: on the one hand, it must be general enough so that problems can be translated into it, and, on the

*This is an over-simplified statement since it qualifies a Turing Machine as a general problem solver. But its generality stems from the fact that the amount of information in the specification of a problem is not limited. For example, the problem of playing perfect chess can be given to a Turing Machine by listing all possible chess positions together with the best move for each position. But this specification, being impractical, does not qualify a Turing Machine as a chess player. We only point out the importance of the amount of information in the specification of a problem; it will not be dealt with in this paper.
other hand it must be specific enough so that the problem solving techniques can be applied.

To illustrate this interdependence, consider a heuristic search problem solver whose only technique is to generate objects by applying the operators in a fixed order and testing if any of the generated objects are identical to the desired situation. It would be easy to construct such a problem solver with a relatively high degree of generality even though it could only solve the most elementary problems. On the other hand, it would be difficult today to achieve even a slight degree of generality with a problem solver that discovered the terms in an evaluation function for determining the likelihood of the existence of a path from any object to the desired situation. Thus, there are many different problems of generality, one for each set of problem solving techniques, and the difficulty of achieving generality depends upon the variety and complexity of the techniques.

This research investigates a particular problem of generality—the problem of extending the generality of GPS while holding its power at a fixed level. This involved extending the internal representation of GPS in such a way that its problem solving methods remain applicable and in a way that increases the domain of problems that can be translated into its internal representation. Thus, this research is mainly concerned with representational issues. We would not expect the issues to be the same in generalizing the internal representation of a problem solver which employed markedly different techniques than GPS. In this respect, this research has the nature of a case study.

B. GPS

GPS attempts problems by tree search, as does any heuristic search program. But to guide the search GPS employs a general technique called means-ends analysis which involves subdividing a problem into easier sub-problems. Means-ends analysis is accomplished by taking differences between what is given and what is wanted, e.g., between two objects or between an object and the class of objects to which an operator can be applied. A difference designates some feature of an object which is undesirable. GPS uses the difference to select a desirable operator—one which is relevant to reducing the difference. For example, in attempting the original problem, GPS detects a difference, if one exists, between the initial situation and the desired situation. Assuming that a desirable operator exists and that it can be applied to the initial situation, GPS applies it and produces a new object. GPS rephrases the original problem by replacing the initial situation with the new object and then recycles. The problem is solved when an object is generated that is identical to the desired situation.

The problem solving techniques of GPS consist of a set of methods, which are applied by a problem solving executive. To solve a problem, the problem solving executive selects a relevant method and applies it. Subproblems may be generated by the method in an attempt to simplify the problem. In such cases, the main problem may temporarily be abandoned by the problem solving executive for the purpose of solving the subproblem. Subproblems are attempted in the same way that the main problem is attempted—by selecting and applying a relevant method.

A complete description of the problem solving executive and the methods is given in Chapter III. For the purposes of this paper, we will only illustrate the methods of GPS by describing how GPS approaches the missionaries and cannibals task. In this example INITIAL-OBJ* is the situation when 3 M (missionaries), 3 C (cannibals) and the BOAT are at the LEFT bank of the river. DESCRIBED-OBJ is the situation when 3 M, 3 C, and the BOAT are at the RIGHT. M-C-OPR, the only operator of this task, moves X missionaries, Y cannibals and the BOAT from the FROM-SIDE of the river to the TO-SIDE. We will ignore for the moment how the task, which includes the above objects and operator, is represented either externally or internally, assuming that there is some internal representation to which the problem solving methods of GPS can be applied.

Figure 2 shows the first few goals attempted by GPS. To solve TOP-GOAL, GPS matches the INITIAL-OBJ to the DESIRED-OBJ and detects that there are 3 too many cannibals at the LEFT. In attempting to alleviate this difference (GOAL 2) the M-C-OPR is applied with Y and FROM-SIDE specified to be 2 and LEFT, respectively. This operator application (GOAL 3) results in the OBJECT 1.

Since there are still too many cannibals at the LEFT, GPS attempts to move the remaining cannibal to the RIGHT (GOAL 4, GOAL 5, GOAL 6). However, first the BOAT must be moved back to the LEFT (GOAL 7, GOAL 8). The operator in GOAL 6 can be applied to OBJECT 2 which results in the old situation OBJECT 1. At this point GPS knows that it is in a loop and looks for something new to do. (To be continued in Section E.)

This example illustrates the kind of information that GPS must abstract from the internal representation in order to apply its problem solving

---

*We adopt the convention that words written in all capital letters correspond directly to IPL symbols inside the machine. These symbols are either defined in the IPL code* that comprises GPS or defined in the task specification.
TOP-GOAL: TRANSFORM the INITIAL-OBJ into the DESIRED-OBJ.

GOAL 2: REDUCE the number of C's at the LEFT bank of the river in the INITIAL-OBJ by 3.

GOAL 3: APPLY the M-C-OPR with Y = 2 and FROM-SIDE = LEFT, to INITIAL-OBJ. OBJECT 1: (LEFT (M 3 C 1) RIGHT (M 0 C 2 BOAT YES))

GOAL 4: TRANSFORM OBJECT 1 into the INITIAL-OBJ.

GOAL 5: REDUCE the number of C's at the LEFT bank of the river in OBJECT 1 by 1.

GOAL 6: APPLY the M-C-OPR with Y = 1 and FROM-SIDE = LEFT, to OBJECT 1.

GOAL 7: REDUCE the difference that the BOAT is not at the LEFT bank in OBJECT 1.

GOAL 8: APPLY the M-C-OPR with TO-SIDE = LEFT to OBJECT 1. OBJECT 2: (LEFT (M 3 C 2 BOAT YES) RIGHT (M O C 1))

GOAL 9: APPLY the M-C-OPR with Y = 1 and FROM-SIDE = LEFT, to OBJECT 2. OBJECT 1: (LEFT (M 3 C 1) RIGHT (M O C 2 BOAT YES))

GOAL 4: TRANSFORM OBJECT 1 into the INITIAL-OBJ.

Figure 2 – The first few goals generated by GPS in solving missionaries and cannibals

methods. Hence, these methods place large demands on the internal representation because processes that abstract the information must be feasible. Below we summarize the demands of the problem solving methods, cross-referencing each to the above example.

Each of these demands requires that GPS employ a process for abstracting certain information from the internal representation. These processes may be different for different representations, but the information abstracted does not depend on the representation.

Object-comparison. GPS must be able to compare two objects to determine if they represent the same situation. Object-comparison is used in attempting the TOP-GOAL in Figure 2.

Object-difference. If two objects do not represent the same situation, GPS must be able to detect differences between them that summarize their dissimilarity. In attempting TOP-GOAL, the object-difference process detects the difference that is used in the statement of GOAL 2.

Operator-application. GPS must be able to apply an operator to an object. The result of this process is either an object, or a signal that the application is not feasible. The operator-application is used to achieve GOAL 3.

Operator-difference. If it is infeasible to apply an operator to an object, GPS must be able to produce differences that summarize why the application is infeasible. In attempting GOAL 6, the operator-difference process detects the difference used in the state of GOAL 7.

Desirability-selection. For any difference GPS must be able to select from all operators of a task those operators that are relevant to reducing the difference. (Of course, this selection will not in general be perfect.)

The M-C-OPR is selected to reduce the difference in GOAL 2. But before generating GOAL 3 GPS specifies the variables Y and FROM-SIDE to insure that the operator performs the desired function. Such a specification of variables limits the number of different ways that the operator can be applied to a given object. Hence, it can be viewed as the selection of a few promising possibilities from the total number of possibilities.

Feasibility-selection. For any object GPS must be able to select from all the operators those that are applicable to the object. (Again, perfect selection is not necessary.) This is meant to cover the case where the internal representation permits several operators of limited range to be combined into a single operator of wider range, such that the application of the unified operator does not decompose simply to the sequential application of the sub-operators. Feasibility-selection is used in achieving GOAL 3 in Figure 2. Note that the operator, move 0 missionaries and 2 cannibals from LEFT to RIGHT, is a schema in the sense that it can be applied to many different objects to yield many different results. For example, in GOAL 3 it is applied to INITIAL-OBJ to yield OBJECT 1 but it can also be applied to the object,

LEFT (M O C 2 BOAT YES) RIGHT (M 3 C 1),
to yield DESIRED-OBJ. Feasibility-selection requires that GPS produce the result without generating the different instances of the operator schema.

Canonization. GPS must be able to find the canonical name of certain types of data structures. Canonization arises from GPS’s strategy for comparing two data structures. If they have canonical names, they are equivalent only if they have the same name. On the other hand, if two data structures do not have canonical names, they are equivalent only if all of their structure is equivalent. GOAL 9 in Figure 2 leads to the regeneration of GOAL 4. GPS recognizes that these two goals are identical because they have the same canonical name (even though they are generated in different contexts).

C. Internal representation of GPS-2-5

The current version of GPS was developed through the modification of an existing version, called GPS-2-5. That GPS-2-5, together with its predecessors, solved only three different kinds of problems was due mainly to inadequate facilities for representing
tasks. The internal representation of GPS-2-S will be described to clarify how the representation incorporated in GPS (described later) alleviated inadequacies in representation. The internal representation of a task for GPS (any version) consists of several different kinds of data structures:

a. objects
b. operators
c. differences
d. goals
e. TABLE-OF-CONNECTIONS
f. DIFF-ORDERING
g. details for matching objects
h. miscellaneous information

A complete description of the above types of data structures is given in Chapter IV. Here, we will only describe the representation of objects and operators, which provide the main representational issues. However, an example of each kind of data structure is given in Section E. Other than objects and operators, differences are the only other type of data structure whose representation in GPS is different from its representation in GPS-2-5. Their representation depends to a large extent on the representation of objects and operators, and will be discussed in more detail later.

Objects. In GPS-2-S objects are represented by tree structures encoded in IPL description lists. Each node of the tree structure can have an arbitrary number of branches leading from it to other nodes. In addition to branches, each node can have a local description given by an arbitrary number of attribute-value pairs. The tree structure in Figure 3, for example, represents the initial situation in the missionaries and cannibals task. In Figure 3 the node to which the LEFT branch leads represents the left bank of the river and the node to which the RIGHT branch leads represents the right bank of the river. The local description at the node which the LEFT branch leads to indicates that three missionaries, three cannibals, and the boat are at that bank of the river.

The use of variables in the tree structures described above allows a class of objects to be represented as a single data structure. For example, Figure 4 is the tree structure representation of \( e^u \). If \( u \) is a variable, this tree structure represents a large class of objects. All members of the class have the same form but different values for \( u \). GPS assumes that all tree structures may contain variables and it is prepared to process them as classes of objects.

Operators. In GPS-2-5 all operators were represented by representing the form of both the input and resultant objects. Assuming that \( u \) is a variable, Figure 4 is the tree structure representation of the operator, \( f \ e^u \), and Figure 5 is the tree structure of the output. Such an operator can only be applied to a member of the class of objects represented by the input form.

D. Representational issues

The representational issues that were investigated arose from various properties of tasks that could not adequately be dealt with by the existing program. Each of these issues will be discussed below. For some the representation was generalized, and the difficulty was removed. Other issues could not be dealt with within the framework of the existing program. However, attempting to alleviate these difficulties did clarify important aspects of the issues.

Desired situation
In many tasks the desired situation is a class of objects that could not be represented in GPS-2-5. In integration, for example, the desired situation is any expression that does not contain \( \int \). A tree structure cannot represent this class of objects because all of the members do not have the same form. For this
reason the representation of the desired situation had to be generalized.

In introducing a new representation for the desired situation, GPS must be given some new processes for abstracting information from the new representation: a new object-comparison process so that GPS can compare an object to the desired situation; and a new object-difference process so that GPS can detect differences between an object and the desired situation. Object-comparison and object-difference are the only demands (described above) of GPS's problem solving method that are affected by the introduction of a new representation for desired situation.

The generalization of the desired situation allowed it to be represented as a set of constraints called a DESCRIBED-OBJ. A set of constraints represents a class of objects, each of which satisfies all of the constraints. The desired object in the integration task can be represented by the single constraint:

No symbol in the expression is an \( f \).

Each constraint in a DESCRIBED-OBJ is a data structure, called a TEST, that consists of a RELATION, and several arguments (in most cases, two). In the previous example,

- a. the RELATION is NOT-EQUAL;
- b. the first argument is a symbol;
- c. the second argument is \( f \).

This constraint is quantified "for all" symbols. GPS recognizes NOT-EQUAL as a RELATION which it understands. (Currently GPS understands fifteen RELATIONS.) On the other hand, GPS only understands the generic form of the arguments, the arguments themselves being task dependent.

Using constraints to represent objects is convenient because each constraint is a simple data structure. Both the object-comparison process and the object-difference process analyze the structure of the constraints. The structure of many representations is too complex to permit such an analysis. For example, an alternate representation for the desired situation is a program whose input is an object and whose output is a signal indicating whether or not the object is a member of the class of objects that the program represents. The object-difference process for this representation would be extremely complex because it would require an analysis of the program.

**Operators**

The operators of many tasks, particularly mathematical calculi could be represented conveniently in GPS-2-5. However, the operators of other tasks could not, e.g., the operator of missionaries and cannibals. To alleviate this difficulty, in GPS an operator can be represented as a data structure, called a MOVE-OPERATOR, that consists of a group of TESTs and a group of TRANSFORMATIONs. The TESTs, which are the same as the TESTs in a DESCRIBED-OBJ, must be satisfied in order for the operator to be applicable and the TRANSFORMATIONs indicate how the resultant object differs from the input object.

A TRANSFORMATION is a data structure that consists of an OPERATION and several arguments. GPS knows the semantics of the OPERATIONS, but as in TESTs, only knows the generic form of the arguments, which are task dependent. Currently, GPS understands six OPERATIONS. A typical TRANSFORMATION (from the missionaries and cannibals operator that moves X missionaries, Y cannibals and the BOAT from LEFT to RIGHT) is:

- DECREASE the number of missionaries at the LEFT by X and increase the number of missionaries at the right by X.

In this TRANSFORMATION the OPERATION is DECREASE and the arguments are X, the number of missionaries at the LEFT, and the number of missionaries at the RIGHT.

Figure 6 illustrates how the operator that moves X missionaries and Y cannibals from LEFT to RIGHT can be represented as a MOVE-OPERATOR. The first two TESTs indicate that X and Y must be greater than O. The third TEST insures that at least one person is in the BOAT to operate it and that the capacity of the BOAT is not exceeded. The remaining TESTs prevent missionaries from being eaten.

**TESTs:**

1. \( X \in \{0, 1, 2\} \)
2. \( Y \in \{0, 1, 2\} \)
3. \( X + Y \leq 2 \)
4. Either
   a. the number of missionaries at the LEFT \( \geq \) the number of cannibals at the LEFT,
   or
   b. the number of missionaries at the LEFT \( \neq 0 \).
5. Either
   a. the number of missionaries at the RIGHT \( \leq \) the number of cannibals at the RIGHT,
   or
   b. the number of missionaries at the RIGHT \( = 0 \).

**TRANSFORMATIONs:**

1. DECREASE the number of missionaries at the LEFT by X and increase the number of missionaries at the RIGHT by X.
2. DECREASE the number of cannibals at the LEFT by Y and increase the number of cannibals at the RIGHT by Y.
3. MOVE the BOAT from the LEFT to the RIGHT.

Figure 6—The MOVE-OPERATOR representation of the operator that moves X missionaries, Y cannibals and the BOAT from the LEFT to the RIGHT.
The three TRANSFORMATIONs indicate how the application of the operator affects the number of missionaries, the number of cannibals and the BOAT, respectively. TRANSFORMATIONs can also implicitly test feasibility. For example, the BOAT must be at the LEFT in order for the third TRANSFORMATION to be applicable.

The introduction of MOVE-OPERATORS in GPS required the addition of new processes so that the problem solving methods could be applied to this new representation. New processes were needed for operator-application, operator-difference, desirability-selection, and feasibility-selection. Hence, the MOVE-OPERATOR representation was designed so as to make these processes simple. For many representations one or more of these processes would be too complex to implement.

A key feature of the MOVE-OPERATOR representation is its transparent structure. Each of the new processes does an analysis of this structure in order to abstract the necessary information. Another good property of MOVE-OPERATORS is its structural similarity to DESCRIBED-OBJ. This similarity causes the MOVE-OPERATOR processes to be similar to the DESCRIBED-OBJ processes, and thus all of these processes use the same basic subroutines. For example, the operator-difference process for MOVE-OPERATORS and the object-difference process for DESCRIBED-OBJs are nearly identical.

Unordered sets
The representation of some tasks requires representation of an unordered set. Multiplication, for example, can be represented as an n-ary function of a set of arguments whose order is unimportant. Such an unordered set can be represented in GPS-2-5 as an object, representing an ordered set, and an operator for permuting the elements of the set. This representation has the drawback that discovering the identity of two sets may require several applications of a permutation operator. The permutation operator would be unnecessary if the identity test could implicitly take into consideration the unordered property of the two sets.

The objects of GPS-2-5 can implicitly represent unordered sets, provided that the nodes can be tagged either ordered or unordered. These tags designate the branches of a node to be either ordered or unordered. Although a seemingly simple generalization, it considerably complicates the object-comparison, the object-difference, the operator-application, the operator-difference, and the canonization processes. These processes were generalized for the integration task so that the nodes of objects and operators could be unordered. Although the generalized processes were more complex and did more processing, there was a savings due to an overall reduction in the problem space.

The main complicating feature of unordered sets is that in matching two unordered sets corresponding elements must be paired. A variable can be made identical to any element via substitution and thus can be paired with any element. However, to see the identity of two unordered sets may require that a particular variable be paired with a particular element. Chapter V discusses this issue in more detail and describes how the generalized processes (object-comparison, etc.) deal with this issue.

Large objects
GPS can only solve simple problems before its memory is exhausted. However, for some tasks the objects are so large that not even simple problems can be solved before its memory is exhausted. For example, the representation of a chessboard in GPS requires 1,000 memory locations and thus only several objects can be stored in memory.

There are two distinct difficulties with GPS's use of memory: (1) GPS saves in memory all objects generated during problem solving and (2) each object is a total situation, i.e., there is no provision for dealing with fragments of situations. These difficulties could not be dealt with in this research because they are too closely connected with the problem solving methods of GPS, which were held fixed.

Differences
In generalizing GPS, the representation of differences was degenerated. Each difference in GPS can only pertain to the value of an attribute of a node of an object. More global differences, such as the number of occurrences of a symbol, which could be represented in GPS-2-5, cannot be represented because they would introduce too much complexity in the operator-difference, the object-difference, and the desirability-selection processes. Thus, the generalization of these processes for MOVE-OPERATORS and DESCRIBED-OBJs was based on this simplified representation of differences.

Differences, although not part of the general heuristic search paradigm, are central to means-ends analysis, which is the main technique of GPS. Many tasks were not given to GPS, because the simple differences would not adequately guide GPS's search for a solution. For example, many of the logic tasks solved by GPS-2-5 cannot be solved by GPS due to the lack of direction provided by the degenerate differences. However, the representation of differences is adequate for the eleven tasks that were given to GPS.
E. An example

Since the emphasis of this research is on the internal representation of problems, it was designed without any consideration of how tasks might be communicated to the machine. The external representation was then designed so that a task expressed in it would be readable. Below we describe the external representation of the missionaries and cannibals task and how its corresponding internal representation is processed by GPS. It is hoped that the external representation of this task (and the other ten given to GPS) is sufficiently readable for the reader to decipher. If this is the case, he can determine precisely what information is contained in the specification of a task. As noted earlier, the amount of information in the specification of a task is related to the issue of generality.

Figure 7 shows the specification of the missionaries and cannibals task for GPS. The specification of any task is a string of words delimited by "spaces." Parentheses are used to group the words. The first part of the specification (the part before the occurrence of TASK-STRUCTURES) indicates how words that are not part of GPS's basic vocabulary, should be interpreted. For example, BOAT is a word peculiar to this task. The string

\[ \text{BOAT} = \text{ATTRIBUTE} \]

designates the BOAT to be an ATTRIBUTE of a node of an OBJECT-SCHEMA. This part of the task specification is analogous to declaration statements in ALGOL.

The remainder of the task specification defines the data structures that comprise the representation of a task. Each data structure definition has the form

\[ \langle \text{name} \rangle = ( \langle \text{body} \rangle ) \]

The first such definition in Figure 7 follows TASK-STRUCTURES. The name of this structure is INITIAL-OBJ and it designates the "initial situation" of the problem and DESIRED-OBJ to be the "desired situation." INITIAL-OBJ is the next data structure defined in Figure 7. This data structure, shown as a tree structure in Figure 3, represents the situation when the 3 missionaries, the 3 cannibals, and the BOAT are at the LEFT bank of the river. DESIRED-OBJ is a similar data structure representing the situation when the 3 missionaries, the 3 cannibals and the BOAT are at the RIGHT bank of the river.

The next four data structures are necessary only because they are used in the definition of M-C-OPR. X+Y is the name of a data structure representing the sum of X and Y. 1, 2 is the data structure representing the set that contains the two elements, 1 and 2. 0,1,2-SET and SIDE-SET are similar data structures that represent sets.

FROM-SIDE-TESTS is a data structure that represents a set of two TESTs. The first TEST is true if the number of missionaries at the FROM-SIDE is not less than the number of cannibals at that bank of the river. The second TEST is true if there are no missionaries at the FROM-SIDE. TO-SIDE-TESTS is a data structure similar to FROM-SIDE-TESTS. Both of these data structures are used in the definition of the next data structure, M-C-OPR.

M-C-OPR is the data structure that represents the only operator of this task. CREATION-OPERATOR indicates that the result of applying the M-C-OPR should be a new list structure instead of a modification of the list structure that represents the input object. The words enclosed in (’s are comments.

Following VAR-DOMAIN are four TESTs that constrain the legitimate values of variables. Incidentally, GPS knows which words are variables because they are listed in the data structure, LIST-OF-VAR, defined at the end of the task specification. The first two TESTs following VAR-DOMAIN require X and Y to be 0,1, or 2 and their sum to be no greater than 2.

The third and fourth TESTs indicate that both TO-SIDE and FROM-SIDE stand for different banks of the river.

The three TRANSFORMATIONs following MOVES indicate how the values of the three ATTRIBUTES, BOAT, M, and C, are modified in applying the operator.
A General Problem Solver

B = L = FEATURE
B = R = FEATURE
C = L = FEATURE
C = R = FEATURE

DESIRED-OBJ = OBJECT-SCHEMA
FROM-SIDE = LOC-PROG
FROM-SIDE-TESTS = V-TESTS
INITIAL-OBJ = OBJECT-SCHEMA
M = ATTRIBUTE
M = OP = MOVE-OPERATOR
M = L = FEATURE
M = R = FEATURE
SIDE-SET = SET
TO-SIDE = LOC-PROG
TO-SIDE-TESTS = V-TESTS
X = CONSTANT
X = Y = EXPRES
Y = CONSTANT
0•1•2-SET = SET
1•2 = SET

TASK-STRUCTURES

TOP-GOAL = ( TRANSFORM THE INITIAL-OBJ INTO THE DESIRED-OBJ ; )

INITIAL-OBJ = ( LEFT ( M 3 C 3 BOAT YES )
RIGHT ( M 0 C 0 ) )

DESIRED-OBJ = ( LEFT ( M 0 C 0 )
RIGHT ( M 3 C 3 BOAT YES ) )

X = Y = ( X + Y )

1•2 = ( 1 2 )

0•1•2-SET = ( 0 1 2 )
SIDE-SET = ( LEFT RIGHT )

FROM-SIDE-TESTS = ( 1. THE M OF THE FROM-SIDE IS NOT-LESS-THAN
THE C OF THE FROM-SIDE.

2. THE M OF THE FROM-SIDE EQUALS 0 ,

From the collection of the Computer History Museum (www.computerhistory.org)
TO-SIDE-TESTS = ( 1. THE M OF THE TO-SIDE IS NOT-LESS-THAN
THE C OF THE TO-SIDE ,
 2. THE M OF THE TO-SIDE EQUALS: 0 .

M+C-OPR = ( CREATION-OPERATOR

$ MOVE X MISSIONARIES AND Y CANNIBALS FROM THE FROM-SIDE TO
THE TO-SIDE $

VAR-DOMAIN
1. Y IS A CONSTRAINED-MEMBER OF THE 0,1,2-SET ,
   THE CONSTRAINT IS X+Y IS IN-THE-SET 1,2 .
2. X IS A CONSTRAINED-MEMBER OF THE 0,1,2-SET ,
   THE CONSTRAINT IS X+Y IS IN-THE-SET 1,2 .
4. THE TO-SIDE IS AN EXCLUSIVE-MEMBER OF THE SIDE-SET .

MOVES
1. MOVE THE BOAT OF THE FROM-SIDE TO THE BOAT OF THE TO-SIDE ,
2. DECREASE BY THE AMOUNT X THE M AT THE FROM-SIDE AND ADD
   IT TO THE M AT THE TO-SIDE .
3. DECREASE BY THE AMOUNT Y THE C AT THE FROM-SIDE AND ADD
   IT TO THE C AT THE TO-SIDE .

POST-TESTS
1. ARE ANY OF THE FROM-SIDE-TESTS TRUE .
2. ARE ANY OF THE TO-SIDE-TESTS TRUE .

B-L = ( BOAT ON THE LEFT . )
B-R = ( BOAT ON THE RIGHT . )
C-L = ( C ON THE LEFT . )
C-R = ( C ON THE RIGHT . )
M-L = ( M ON THE LEFT . )
M-R = ( M ON THE RIGHT . )
DIFF-ORDERING = ( ( M=R M=L C-R C=L )
( B-R B-L ) )

TABLE-OF-CONNECTIONS = ( ( COMMON-DIFFERENCE M+C-OPR ) )

COMPARE-OBJECTS = "(BASIC-MATCH )
BASIC-MATCH = "( COMP-FEATURE LIST ( M-L C-L B-L ) )
OBJ-ATTRIB = "( M C BOAT )

LIST-OF-VAR = ( FROM-SIDE TO-SIDE X Y )

END

Figure 7 – The specification for GPS of the missionaries
and cannibals task
Following POST-TESTS are two TESTs that must be true of any object produced by an application of the operator. These constraints prevent missionaries from being eaten in the resultant object. We can assume that the object to which the operator is applied satisfies these constraints because INITIAL-OBJ and DESIRED-OBJ satisfy them and all other objects are produced by an application of the M-C-OPR. The first TEST following POST-TESTS requires one of the two TESTs in FROM-SIDE-TESTs to be true (of the resultant object). This prevents missionaries from being eaten at the FROM-SIDE. The second TEST following POST-TESTS is similar. It should be noted that conjunction is the implied logical connective of the TESTs in M-C-OPR or any other MOVE-OPERATOR. And the TESTs in POST-TESTS of the M-C-OPR illustrate how disjunction can be used.

The next six data structures defined in Figure 7 are the types of differences of this task. A difference consists of a type of difference and a value. For example, the difference

\[ C-L, -2 \]

indicates that there are two more cannibals at the LEFT in one object than in another object. DIFF-ORDERING is the data structure that designates M-R, M-L, C-R and C-L to be more difficult to reduce than B-L and B-R.

TABLE-OF-CONNECTIONS indicates that M-C-OPR is relevant to reducing any type of difference. COMPARE-OBJECTS and BASIC-MATCH causes GPS to look for the following types of differences in comparing two objects:

a. M-L
b. C-L
c. B-L

Hence, GPS only compares the LEFT banks of two objects because what is not at the LEFT must be at the RIGHT.

OBJ-ATTRIB is a list of the ATTRIBUTES and LIST-OF-V AR is a list of the variables.

Figure 8 shows the way that GPS attempts to solve the missionaries and cannibals task. In attempting TOP-GOAL, GPS detects that there are too many missionaries and cannibals at the LEFT. GOAL 2 is created in an attempt to reduce the number of cannibals at the LEFT by 3. GPS knows that there are 3 too many cannibals at the LEFT but the ‘3’ does not get printed. GPS attempts to reduce the most difficult differences first which eliminates B-L. C-L was selected instead of M-L because GPS detected C-L first.

Since the TABLE-OF-CONNECTIONS indicates the M-C-OPR is relevant to reducing this difference, GPS attempts to apply it to INITIAL-OBJ. Before creating GOAL 3, GPS specifies Y and FROM-SIDE to be 2 and LEFT, respectively, so that the operator will perform the desired function. GOAL 3 results in OBJECT 5 and GPS attempts to transform this new object into DESIRED-OBJ (GOAL 4).

Since there are still too many cannibals at the LEFT (GOAL 5), GPS attempts to move the remaining cannibal to the RIGHT (GOAL 6). However, the operator cannot be applied because the BOAT is at the wrong bank of the river. In an attempt to bring the BOAT back to the LEFT (GOAL 7), GPS applies the M-C-OPR with the TO-SIDE equal to LEFT (GOAL 8) and OBJECT 6 is produced. GPS moves one cannibal across the river (GOAL 9); it does not realize that bringing the BOAT back to the LEFT also brought a cannibal with it. Since transforming the result of GOAL 9, which is an old object, into the DESIRED-OBJ is an old GOAL, GPS does not attempt it but looks for something else to do.

GOAL 11 is created in an attempt to transform all of the OBJECT-SCHEMAs which are derived from the INITIAL-OBJ, into the DESIRED-OBJ. (OBJECT 4, which is the SET of all OBJECT-SCHEMAs derived from the INITIAL-OBJ, is generated internally by GPS). GOAL 13 is created because OBJECT 6 has never appeared in a TRANSFORM type of GOAL. (This is the NEW-OBJ selection criterion.) Since there are too many cannibals at the LEFT in OBJECT 6 (GOAL 14), two are moved across the river (GOAL 15, OBJECT 7).

Everything goes smoothly until GOAL 27, which results in an old object, at which point GPS generates a GOAL identical to GOAL 22. GPS does not reattempt GOAL 22 but generates a new GOAL (GOAL 29) by selecting a NEW-OBJ (OBJECT 8). Attempting GOAL 29 quickly leads to the old object, OBJECT 7 (GOAL 31) and the old GOAL, GOAL 16.

GOAL 33 is generated by selecting another NEW-OBJECT and GPS does not run into trouble until GOAL 49 which results in an old object. Again a new GOAL is generated by selecting a NEW-OBJ. GOAL 52 is abandoned because attempting it creates a GOAL identical to GOAL 43. The generation of GOAL 54 quickly leads to success.

F. Results

The initial motivation for this research comes from the tasks themselves; they could not be expressed in the internal representation of GPS-2-5. But the majority of this research focuses on the processing
GOAL 26 APPLY M-C-OPR WITH TO-SIDE = LEFT; TO 0
  SET: Y = 1, X = 1, FROM-SIDE = RIGHT
  OBJECT 101 (LEFT: 2 C 2 BOAT YES) RIGHT(M 1: C 1)

GOAL 27 APPLY M-C-OPR WITH Y = 1, FROM-SIDE = LEFT; TO 10
  SET: X = 1, TO-SIDE = RIGHT
  OBJECT 91 (LEFT: 1 C 1) RIGHT(M 2 C 2 BOAT YES)

SELECT FROM 4 A/C NEW-OBJ OF DESIRED-OBJ
10 SELECTED

TRANSFORM 9 INTO DESIRED-OBJ

REDUCE C-L ON 8

GOAL 30 APPLY M-C-OPR WITH Y = 1, FROM-SIDE = LEFT; TO 0
  SET: X = 0, TO-SIDE = RIGHT
  OBJECT 71 (LEFT: 0 C 1) RIGHT(M 0 C 3 BOAT YES)

SELECT FROM 4 A/C NEW-OBJ OF DESIRED-OBJ
10 SELECTED

TRANSFORM 10 INTO DESIRED-OBJ

REDUCE C-L ON 10

GOAL 33 APPLY M-C-OPR WITH Y = 2, FROM-SIDE = LEFT; TO 10
  SET: X = 2, TO-SIDE = RIGHT

REDUCE M-U ON 10

GOAL 35 APPLY M-C-OPR WITH X = 2, FROM-SIDE = LEFT; TO 10
  SET: Y = 0, TO-SIDE = RIGHT
  OBJECT 111 (LEFT: 0 C 2) RIGHT(M 1 C 3 BOAT YES)

TRANSFORM 11 INTO DESIRED-OBJ

REDUCE C-L ON 11

GOAL 39 APPLY M-C-OPR WITH Y = 2, FROM-SIDE = LEFT; TO 11
  SET: X = 0, TO-SIDE = RIGHT

REDUCE B-L ON 11

GOAL 41 APPLY M-C-OPR WITH TO-SIDE = LEFT; TO 11
  SET: Y = 1, X = 0, FROM-SIDE = RIGHT
  OBJECT 121 (LEFT: 0 C 3) RIGHT(M 2 C 0)

TRANSFORM 13 INTO DESIRED-OBJ

REDUCE C-L ON 13

GOAL 45 APPLY M-C-OPR WITH Y = 2, FROM-SIDE = LEFT; TO 13
  SET: X = 0, TO-SIDE = RIGHT
  OBJECT 131 (LEFT: 0 C 1) RIGHT(M 3 C 0)

REDuce C-L ON 13

GOAL 47 APPLY M-C-OPR WITH Y = 1, FROM-SIDE = LEFT; TO 13
  SET: X = 0, TO-SIDE = RIGHT
  OBJECT 141 (LEFT: 0 C 2) RIGHT(M 3 C 1)

REDUCE B-L ON 13

GOAL 49 APPLY M-C-OPR WITH Y = 1, FROM-SIDE = LEFT; TO 14
  SET: X = 0, TO-SIDE = RIGHT
  OBJECT 141 (LEFT: 0 C 2) RIGHT(M 3 C 1)

From the collection of the Computer History Museum (www.computerhistory.org)
implication of several modes of representation. In generalizing GPS the trend appears to be toward richer representations whose structure is simple enough for GPS to "understand." For instance, GPS in some sense understands the M-C-OPR in Figure 7 because it can specify the pertinent variables so that the operator performs a desired function. On the other hand complex structures, such as IPL programs, were avoided in designing the internal representation of GPS.

The generalization of GPS has focussed on a particular group of tasks. If other tasks had been chosen, the generalization might have followed a quite different course. The tasks dealt with in this research were not chosen arbitrarily. Some categories of tasks, e.g., many optimization tasks, were deliberately avoided because we know of no obvious heuristic search formulation for them. Other categories of tasks were avoided because of deficiencies in GPS's problem solving methods. For example, games were avoided because GPS does not have a method for considering the opponent's moves and interests in addition to its own.

Of the tasks dealt with in this research, GPS can solve some, typified by the tasks described below. However, others cannot be solved by GPS due mainly to inadequacies in its representation. (See Chapter V' for a discussion of these inadequacies.)

We conclude this paper by briefly discussing the eleven tasks that were actually given to GPS. One of the instructive aspects of this research is the light shed upon the structure of these tasks. In addition, they serve as concrete examples of the level of generality achieved by GPS.

Missionaries and cannibals
GPS and one of its predecessors, GPS-2-2, both solved the missionaries and cannibals task. The representation of the task in GPS was quite different from that used by GPS-2-2. The latter contains information about the nature of operators which the current GPS discovers for itself. GPS-2-2 was given ten operators: Move one missionary from left to right; move two missionaries from left to right; move one missionary and one cannibal from left to right, etc. The desirability of these operators for reducing the various types of differences was given to GPS-2-2, exogenously (in the TABLE-OF-CONNECTIONS). GPS is only given a single operator which moves X missionaries and Y cannibals across the river. In applying this operator GPS specifies the variables (X, Y, and the direction of the boat) so that the operator performs a desirable function.

GPS-2-2 was given a desirability filter for operators. This filter prevented GPS-2-2 from attempting to move more missionaries and cannibals across the river than there were on the side from which they were being moved. Such a separate filter is unnecessary in GPS because GPS never considers applying such an operator. Each operator in the GPS-2-2 formulation consisted of an IPL routine with its parameters (described on page 30'). The operator filter was also encoded in IPL. Not only is it tedious to construct IPL routines but the construction of these routines

---

Figure 8—The performance of GPS on the missionaries and cannibals task
requires some knowledge of the internal structure of GPS-2-2. The construction of the single operators given to GPS is much less tedious and requires no knowledge of the internal structure of GPS.

Integration

GPS symbolically integrated \( \int te^2dt \) and \( \int (sin^2 (ct)cos(ct) + t^2) dt \).

In the integration, multiplication and addition are represented as n-ary functions whose arguments are represented as an unordered set. Thus, the commutativity and associativity of multiplication and addition are expressed implicitly instead of representing them explicitly as operators. If they were explicitly given to GPS as operators, there would be an overall increase in the problem space which would prevent GPS from solving some trivial integrals.

SAINT, a program that is quite proficient at symbolic integration, also represents the commutativity and associativity of multiplication and addition implicitly. Other similarities and some dissimilarities between SAINT and GPS are discussed in Chapter V.

Tower of Hanoi

In the Tower of Hanoi, which is a classical puzzle, there are three pegs and a number of disks, each of whose diameter is different from all of the others. Initially, all of the disks are stacked on the first peg in order of descending size. The problem is to discover a sequence of moves that will transfer all of the disks to the third peg. Each move consists of removing the top disk on any peg and placing it on top of the disks on another peg, but never placing a disk on top of one smaller than itself. GPS solved the 4-disk Tower of Hanoi task.

The Tower of Hanoi is an example of a task for which means-ends analysis is very effective. GPS never makes a mistake on this task, mainly because the differences and the DIFF-ORDERING are in some sense optimal. For many tasks, it is difficult to find good differences and a good DIFF-ORDERING. For a different treatment of this task see reference 18.

Proving theorem in the predicate calculus

GPS proved the following simple theorem in the first order predicate calculus:

\[
(3u) (3y) (Vz) ((P(u,y) \supset (P(y,z) \& P(z,z))) \&
((P(u,y) \& Q(u,y)) \supset (Q(u,z) \& Q(z,z))))
\]

\( \exists \) is the existential quantifier; \( \forall \) is the universal quantifier; \( P \) and \( Q \) are predicates; \( u \), \( y \) and \( z \) are variables; \( \supset \), \( \& \), and \( V \) are implication, conjunction and disjunction, respectively. The formulation of this problem is basically the same as that used by Robinson.

Perhaps the most instructive part of this example is the light it cast upon the evolution of problem solving programs. In LT, a theorem proving program for the propositional calculus, which is the predecessor of GPS, it was noted that the match routine was the source of most of the power of the program over a brute force search. GPS may be considered as an attempt to generalize the match routine, based on that experience. The first predicate calculus theorem prover did in fact use brute force search. From an efficiency point of view the main effect of the resolution principle of Robinson was to reintroduce the possibility of matching (gaining, thereby, a vast increase in power). And it is this feature that allows GPS to use the resolution principle in a natural way.

Father and sons task

GPS solved the task in which a father and his two sons want to cross a river. The only means of conveyance is a small boat whose capacity is 200 pounds. Each son weighs 100 pounds while the father weighs 200 pounds. Assuming that the father and either son can operate the boat, how can they all reach the other side of the river? Of course, there is no way to cross the river except by boat.

This task is very similar to the missionaries and cannibals task. Both tasks involved moving two different kinds of people across a river in a small boat. But their formulations for GPS are quite different, in that none of the operators, objects, or differences are the same. Many of the earlier publications on GPS (e.g., "and") make the distinction between a task and a task environment—the common part of a group of similar tasks. The father and sons task and the missionaries and cannibals task muddies this distinction. On the one hand, they should both have the same task environment because of their similarity. In fact they have different task environments because none of their objects, operators and differences are the same.

Monkey task

This task, which GPS solved, was invented by McCarthy as a typical problem for the Advice Taker program. In a room is a monkey, a box, and some bananas hanging from the ceiling. The monkey wants to eat the bananas, but he cannot reach them unless he is standing on the box when it is sitting under the bananas. How can the monkey get the bananas? The answer is that the monkey must move the box under the bananas and climb on the box before he can reach the bananas. The problem originates in the study of the problem solving ability of primates; its interest lies not in its difficulty, but in its being an example of a problem subject to common sense reasoning.
It is interesting to compare GPS's formulation of this task to the formulation for a typical Advice Taker program. In GPS the objects are models of room configurations whereas in the objects are linguistic expressions that describe certain aspects of the room configurations. Both representations have advantages. For example, linguistic expressions are useful for representing imperfect information such as the monkey is in one of two places. However, models can represent implicit information that has to be represented explicitly when linguistic expressions are used, e.g., the monkey can only be in one place at a time.

Three coins puzzle
In this task there are three coins sitting on a table. Both the first and third coins show tails, while the second coin shows heads. The problem is to make all three coins the same—either heads or tails—in precisely three moves. Each move consists of "turning over" any two of the three coins. For example, if the first move consisted of turning over the first and third coins, all of the coins would be heads in the resulting situation. But the task is not solved because only one move was taken instead of the required three.

The peculiarity of the three coins task is in the solution being constrained to a fixed number of operator applications. This constraint was handled in GPS by expanding the representation of objects to include a counter that indicates the number of operator applications involved in producing the objects. In the desired situation the counter must have a particular value.

Parsing sentences
GPS parsed the sentence,

Free variables cause confusion,

according to a simple context-free grammar. (It contains ten productions or rewrite rules.) A great deal of effort has been devoted to the construction of efficient parsing algorithms for simple phase structure grammars. The point of this example is not GPS's proficiency as a parser, but to illustrate the kinship between heuristic search and syntactic analysis.

Bridges of Konigsberg
In the German town of Konigsberg ran the river Pregel. In the river were two islands connected with the mainland and with each other by seven bridges as shown in Figure 9. How can a person walk from some point in the town and return to the same point after crossing each of the seven bridges once and only once? In 1736 Euler proved that this task is impossible, and his proof stands as one of the early efforts in topology.

This is the only impossible task that was given to GPS. Although GPS's behavior is not aimless (it crosses six bridges in two different ways), GPS cannot see the impossibility because it lies in the topological properties of the bridges. GPS only attempts to cross bridges and has no way of viewing the problem as a whole.

Water jug
Given a five gallon jug and an eight gallon jug, how can precisely two gallons be put into the five gallon jug? Since there is a sink nearby, a jug can be filled from the tap and can be emptied by pouring its contents down the drain. Water can be poured from one jug into another, but no measuring devices are available other than the jugs themselves.

For the water jug task, means-ends analysis seems to be a rather ineffective heuristic as demonstrated by the fact that GPS stumbled onto the solution. Better types of differences would improve GPS's performance on this task. Better differences do exist although they are quite complex (involving modular arithmetic, naturally).

Letter series completion
This task, which is found in aptitude tests, is to add the next few letters to a series of letters. GPS correctly completed the series,

B C B D B E . .

The letter series completion task is the only task whose solution requires inductive reasoning. The formulation is quite clumsy, but this example demonstrates how the problem can be approached by searching for a suitable description in a space of descriptions. The binary choice task, another task requiring inductive reasoning, was formulated in a similar way by Feldman, et al. (Simon and Kotovsky use a different formulation of letter series tasks.)
Generality of GPS's basic processes

The appeal of the heuristic search paradigm lies in its generality. Hence, it is important that GPS's problem solving techniques have general applicability to heuristic search problems.

GPS uses all of its basic processes (described above) except for operator-difference, on each of the eleven tasks. In solving the predicate calculus, three coins and letter series tasks, GPS never attempts to apply an infeasible operator, and thus the operator-difference process is never evoked. If each of the basic processes were specialized to a particular aspect of one task, it would be necessary to add new basic processes in order to give GPS a new task. However, the applicability of the basic process to all eleven tasks implies that they would also be applicable to other tasks.

REFERENCES

1 G W ERNST A NEWELL
Generality and GPS
Carnegie Institute of Technology 1966
2 A NEWELL
Self-Organizing Systems
Some problems of basic organization in problem-solving programs
Yovits M C Jacobi G T and Goldstein G D eds
3 A NEWELL
A guide to the general problem-solver program GPS-2-2
RAND Corporation Santa Monica Calif RM-3337-PR 1963
4 A NEWELL
Proceedings of IFIP congress 62
Learning, generality and problem-solving
North Holland Publishing Amsterdam Holland 1962
5 A NEWELL J C SHAW H SIMON
Preliminary description of general problem-solving program
–I GPS-1
CIP Working Paper No 7 Graduate School of Industrial Administration Carnegie Institute of Technology Pittsburgh Pa Dec 1957
6 A NEWELL H C SHAW H A SIMON
Information processing: proceedings of the international conference on information processing
Report on a general problem-solving program for a computer
UNESCO Paris 1960 p. 256-264
7 A NEWELL H A SIMON J C SHAW
Self-organizing systems
A variety of intelligent learning in a general problem-solver
Yovits M C and Cameron S eds
8 A NEWELL H A SIMON
Science
Computer simulation of human thought 134
P 2011-2017 December 1961
9 A NEWELL H SIMON
Lernende automaten
GPS a program that simulates human thought
Munich Germany 1961
10 H A SIMON A NEWELL
Current trends in psychological theory
The simulation of human thought
University of Pittsburgh Press Pittsburgh Pa p 152-179 1961
11 H A SIMON A NEWELL
Management and the computer of the future
Computer simulation of human thinking and problem solving
Greenberger M ed
John Wiley and Sons New York 1962
12 H A SIMON A NEWELL J SHAW
Contemporary approaches to creative thinking
The processes of creative thinking
Gruber H A Terell G and Wertheimer M eds
13 D G BOBROW
Natural language input for a computer problem solving system
Doctoral Dissertation Mathematics Dept Massachusetts Institute of Technology Cambridge Mass 1964
14 B RALPHEd
SIR: A computer program for semantic information retrieval
Doctoral Dissertation Mathematics Dept Massachusetts Institute of Technology Cambridge Mass 1964
15 A NEWELL G ERNST
Proceedings of IFIP congress 65
The search for generality
Kalench W A ed
16 A NEWELL ed
Information processing language-v manual
17 J R SLAGLE
J ACM 10
A heuristic program that solves symbolic integration problems
in freshman calculus
p 335-337 Oct 1963
18 HORST BOBROW
Behav sci 10
Gaku: an artificial student
p 88-107 Jan 1965
19 J A ROBINSON
J ACM 12
A machine-oriented logic based on the resolution principle
p 23-41 Jan 1965
20 A NEWELL J C SHAW H SIMON
Empirical explorations with the logic theory machine
Proceedings of the Western Joint Computer Conference p 218-239 1957
21 P C GILMORE
IBM j res develop
A proof method for quantification theory
p 28-35 Jan 1960
22 J McCARTHY
Situations actions and causal laws
Stanford Artificial Intelligence Project Memo No 2
July 1963
23 J McCARTHY
Proc symposium on mech of thought processes
Programs of common sense
Her Majesty's Stationery Office London p 75-84 1959

From the collection of the Computer History Museum (www.computerhistory.org)
24 FISHER BLACK
A deductive question answering system
Doctoral Dissertation Division of Engineering and Applied Physics
Harvard University Cambridge Mass 1964

25 J FELDMAN F TONGE H KANTER
Symposium on simulation models
Empirical explorations of a hypothesis-testing model of binary choice behavior
Hogott A C and Balderston F Eds
South-Western Publishing Co Cincinnati Ohio p 55-100 1963

26 H A SIMON K KOTOVSKY
Psychological review 70
Human acquisition of concepts for sequential patterns
p 534-546 June 1963