A THEORY FOR THE COMPLETE MECHANIZATION OF A GPS-TYPE PROBLEM SOLVER

R. B. Banerji
Temple University
Philadelphia, Pa. 19122

C. W. Ernst
Case Western Reserve University
Cleveland, Oh. 44106

Abstract

The data structure that drives the General Problem Solver is the Connection Table. This paper describes the theoretical basis for the automatic construction of this table by computer programs. The programs for this purpose have been developed at the Case Western Reserve University. They basically isolate certain attributes of the problem states which are invariant under certain moves and then put those attributes together to "triangularize" the Connection Table.

Descriptive Terms

Theory of Heuristics, General Problem Solver

1. Introduction

According to our view of mechanical problem solving, there are a number of different problem solving methods each of which has problem dependent parameters. For each method there is a condition which specifies the properties the parameters should have in order for the method to "work". Hence, we view problem solving as the two phase process shown in Figure 1. In the first phase, the method's condition is used to generate "good" parameters for the method. The input to this phase is the problem specification, since the parameters are usually problem dependent. The output is either good parameters or an indication that this method should not be used on the given problem. The second phase attempts to solve the problem (as specified at the input to the first phase) using the method with the parameters generated in the first phase. Of course, there is a Dicture like Figure 1 for each method, and if the first method is not applicable, we merely move on to the next method and attempt to use it.

So far, all the methods studied this way [Coray(1970), Ernst(1969), Banerj1(1971)] seem to depend on the recognition of certain attributes of the problem states which remain invariant under some of the moves. We have previously published two reports on the design and implementation of a program which would isolate some of these attributes on the basis of the problem description [Ernst e£a_l(1974), >Oyen(1975)].

Our present effort deals with the combination of the invariant properties to yield the Connection Table of GPS [Newell & Simon(1963), Ernst & Newell(1969)]. Our efforts in using our previous theory [Ernst(1969)] for the purpose of mechanizing the heuristic were not successful, because a difference (good or bad) was a binary relation between states and sets of states, i.e., a subset of S x 2s where S is the set of problem states, which is a complicated concept.

In an attempt to simplify matters we said, "What if a difference were just a set of states?" In this case, a state s possesses difference D, if s *i* D. With this simple view we can visualize GPS's strategy as follows (Figure 2).

S is the set of all states. W is the set of goal states: $W \subset D' \subset D \subset S$, and s_0 is the initial state. GPS would attempt to solve the problem as follows:

1. Find a path from SO to some state *s* D.
2. Find a path from s to some state s_2 e D^1 but the path must be entirely inside D.
3. Find a path from s_2 to some state s e W but the path must be entirely inside D'.

In step 1 GPS is removing the most difficult difference D. In step 2 the second most difficult difference is being removed without reintroducing D. The easiest difference W is being removed in step 3 without reintroducing either D or D'.

A point ought to be made here about the original GPS which was a somewhat more general device than the one we are describing here in that,

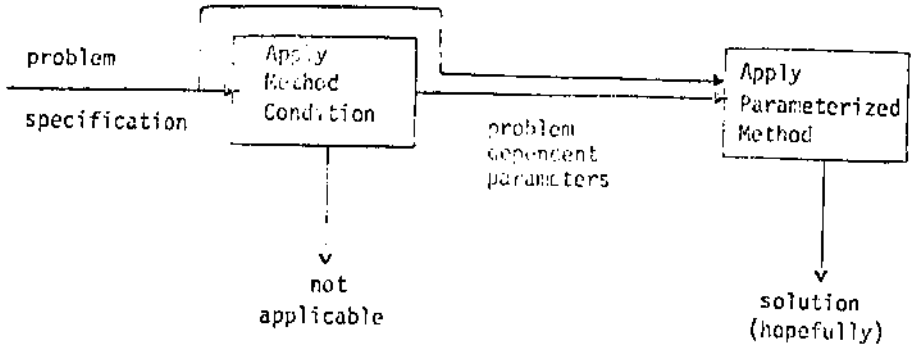
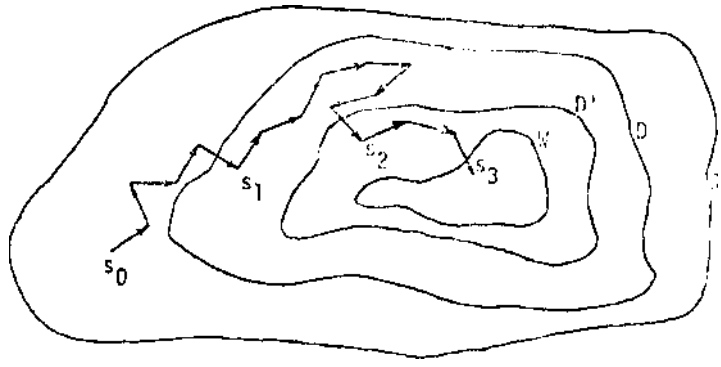


Figure 1. Our view of mechanical problem solving.



]'i£i:re 2

while removing an easier difference, a more difficult difference could get reintroduced. The only search pruning involved in this general case was that involved in the relevance of moves to differences (vide ultra). The extra constraint we have introduced here (and one which also characterizes our previous work [Ernst(1969)]) constrains the search for greater efficiency, while at the same time it neglects a certain class of solutions. Our present analysis follows the same line.

It is probably somewhat counterintuitive that the most difficult difference contains all of the other differences as subsets. One would normally think that the larger the set of states, the easier it would be to "get" inside of it. Also, one does not normally think of W as a difference. But this somewhat unintuitive picture works quite well.

Consider, for example the Fool's Disk problem. Figure 3 gives the initial state of the Fool's Disk problem, in which there are 4 concentric disks each containing 8 numbers. These numbers line up so as to form 8 columns radiating from the center of the disks. A move consists of rotating one of the disks independently of the others. The desired state is one in which each of the 8 radial columns sums to 12.

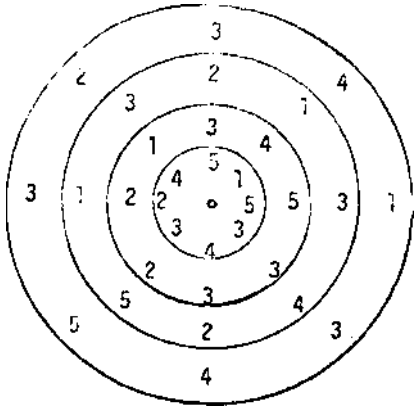


Figure 3
The initial state in the Fool's Disk problem

This problem fits the above picture exactly. D' is the set of states in which each diameter sums to 24, while D is the set of states in which the sum of the N, E, S, and W radii is 48. To keep the path from s_1 to s_2 in D, GPS only considers moves which rotate disks 90°. To get from s_1 to s_3, GPS rotates disks by 180° only.

One might be disturbed that each difference contains all of the easier differences. This is not a difficulty, because any set of differences not possessing this property can be converted to differences which have this property. (In fact, our theory [Banerji &. Ernst (1977)] does not require this "nesting" of differences.) Consider, for example, the 3 disk Tower of Hanoi in which we are trying to move all disks to peg P_3. Let D. be the set of states in which disk i is on P3 where disk 3 is the largest disk. Then, one might think of using D_1, D_2, and D_3 as differences for this problem. These are essentially the differences that were given to GPS for this problem. Certainly these sets are not perfectly nested. However, this set of differences can be converted to the above picture by intersecting them together, i.e., $D = D_1$, $D' = D_3 \cap D_2$, and $W = D_3 \cap D_2 \cap D_1$.

A more disturbing feature of this set of differences is that they are only useful when the set of desired states is W. In the original GPS (as well as In our previous work) the same set of differences served to characterize all subgoals - including "make such and such a move applicable." This is not the case anymore. If, for example, the set of desired states is the domain of the operator which moves disk 3 from P_1 to P_3, the D_1 seems to be a useless set of differences. The difficulty is that we have "built" W into the differences. We did this on purpose to simplify the differences to allow mechanization. Our original theory had differences as binary relations between states and sets of states. If we specify the latter to be W, then we are left with a monadic relation on states which is just a set of states. But how are we going to accommodate goals other than W?

The key to answering this question is that not only W but also the domains of operators can be the goals of subproblems. Since the number of operators is usually quite small, we will use a dif-

ferent set of differences for the domain of each operator being the goal of a subproblem.

These modifications were introduced in our theory of GPS to make it easier for person or machine to discover "good" differences. An added advantage of the modified GPS is that it can easily handle problems in which the sets of differences for subproblems with different desired states are truly different.

The above discussions will, we hope, serve as a motivation for the changes we have introduced in the theory. We do not plan to give a formal counterpart to these motivations or exhibit a formal connection between the old and the new theories. Instead, we shall exhibit and motivate the new theory ab initio so that readers unacquainted with our previous work will find the discussion self-contained. We shall, of course, assume that the reader has had former acquaintance with GPS [Ernst & Newell(1969)1.

In the next section we give a formal definition of good differences. This is followed by an example of good differences and how they are used by GPS. Section 4 characterizes the class of solutions that GPS can find given the kind of differences described in Section 2.

2• Definition of Good Differences

Since GPS builds its solution to a problem by setting up subproblems, we cannot build this theory by defining what a problem is but rather by defining a larger structure in which a class of subproblems can be embedded. Also, this structure should contain the concepts which reflect the idea of differences and the connection table. We shall call this structure the problem domain, "domain" for short. As in the previous models, we start with a set S of states and a subset W of S, consisting of winning states. We also have a set C of partial functions (mapping subsets of S into S) which we shall call moves or operators. If f e G is a move, we shall denote by Sf its domain of definition, i.e., states where f is an applicable move. Since subgoals in GPS have the form "make move f applicable," these Sf, for various members f of G, serve as winning sets for subproblems just as W serves for a problem. The class of all these sets (W and Sf for various f) we shall call X. For each set in this class we also define the differences which allow GPS to work on them. That is, for each T e X (T being either W or S_f for some f e G) we define a class of sets Tj, $T_?$, T3 ..,T_n with the property that Tj n T n...T_n"= T. The actual number n of specified differences of course depends on the set T chosen. So, instead of writing n we shall write n(T) when there is any doubt as to which subproblem we are talking about. Also, for reasons of convenience of discussion we shall often give the name TQ to T and call S itself, $T_{n(T)+1}$.

It may be appropriate at this point to point out that the T^ catches the idea of difference in that when a state s *i* $T_{i\$}$ a difference exists between s and T. The higher i is, the larger the difference is.

The next important concept in GPS, of course, is that of relevance of a move to a difference. The major assumption on which GPS theory is based is that a solution can be obtained by removing the higher ("more difficult") differences before the lower differences and never reintroducing higher differences once they are removed. A difference Is considered higher than others, if fewer moves are available to remove it. Of course, S or $^{T}{}_{n}('T)+i^{are}$ the most difficult differences to remove, since no move changes a state to a non-state. Let H1 c G be the set of moves which, when applicable, affects the position of the state with respect to T1. Instead of making the very strong assumption that moves in H1 bring all states outside T1 into T,, we shall make the more realistic assumption that these moves remove the states from *Ti* when applied. This assumption seems "backward" to many, in spite of the fact that in most real problems, relevance of moves does appear that way and was used that way even in the original GPS. In our difference-finding program, a state is characterized by giving the values of certain attributes for the state. A winning state is characterized by specifying that some of the attributes should have specific unique values. To find mechanically that a certain move is relevant to a certain difference T., we test whether the move changes the values of those attributes which characterize T..

It is this "property-changing" characterization for moves which gives relevance the backward appearance. Of the various values to which the attribute can change, only one characterizes the win states. Hence, it is not to be expected that merely changing the value of a property yields a win value. On the other hand, if it already has a win value, changing it certainly changes it to a non-win value.

Another important characteristic we demand of the moves in $W_{\pm}$ (called triangularity of the difference table in the previous theory) is that Hi does not affect the differences higher than Ti, i.e., is irrelevant to Tj for j > I. Thus, once a state is in T_i, as long as we use moves in Hj with j < i, $T_{\pm}$ will not be reintroduced.

This effort shows up nicely in the difference transformation tables of GPS. If we arrange the Ti's from top to bottom in decreasing order of i and the H. from left to right in decreasing order, and mark the (i,j) cell with a 1 if moves in Hj are relevant to $T_{\pm t}$ then the upper right half of the table will be blank. Tables of this nature we call triangular tables, and differences which give rise to triangular tables we call good differences .

We define the maximum difference between T and s, M(s, T), to be i if s *4* $T_{\pm}$ and s t Tj for all j greater than i.

Implicit in the above definitions is an ordering of the $T_{\pm}$ (and the Hi) which corresponds to the difference ordering of GPS. The most difficult difference is T_n, while the easiest difference is $T_{\pm}$. GPS's basic problem solving strategy is to work on hard differences first and easy differences last. GPS accomplishes this (as discussed in Section 4) by using the following to guide its search:

51 To reduce the maximum difference T-j, use only operators in H^.

52 Suppose a subproblem were generated to reduce difference T_{i4} Then do not use the operators in Hj, i < j < n to solve the subproblem.

Rule SI was in our previous theory. Note that there may be many other operators besides H. which are relevant to T^ because we have placed no conditions on Hj for j > i. SI causes GPS to ignore such Hj even though some of its operators may be relevant to Tj_.

The purpose of S2 is to require subproblems to be easier than the problem for which they are created. In our previous theory this was accomplished by requiring the differences of a problem to be harder than the differences of its subproblems. This is no longer possible, because we cannot compare subproblem differences to problem differences because they will have different goals and hence different differences. However, S2 can be used, because all differences are reduced by the same operators. Note that S2 is applied recursively. That is, suppose Fl and F2 are the sets of operators according to S2 that cannot be used on subproblems SP1 and SP2, respectively. If SP2 is a subproblem of SP1, then GPS will not use any operator in Fl u F2 to solve SP2, because, the restrictions on SPI are passed down to all of its subproblems.

3. An Example of Good Differences

The definitions above appear quite formidable and somewhat unlike GPS. A simple example will clarify things. For our example we have chosen that old chestnut about the monkey and the bananas, a formulation of which is given in Figure 4. We have chosen this example because it has (non-trivial) good differences, subproblems are created in solving it, and it is simple.

One way to formalize the differences above is by positing that there is a separate table of connections for each goal which is either W or the domain of an operator. Figure 5 illustrates Monkey and Bananas this way. The I's indicate which operators are relevant to which differences. The O's indicate irrelevance. A move is neither relevant nor irrelevant - we use a question mark. Note that the bottom row heading of each table is just the goal and that each row is a subset of the row above it. Although our theory does not require these properties, they make things easier to visualize as discussed at the beginning of

A state is a 3 place vector whose components are the monkey's position, the box's position, and the contents of the monkey's hand.

A win is a state in which the bananas are in the monkey's hand.

(Walk, Climb, Push, Grab)

Walk The monkey walks to someplace in the room.

Climb The monkey climbs onto the box, i.e., the monkey's position becomes ONBOX. Climb is applicable only when the monkey's position equals the box's position.

Push The monkey pushes the box to some in the room. Push is applicable only when the monkey's position equals the box's position.

Grab The monkey grabs the bananas. Grab is applicable only when the monkey is on the box, and the box is under the bananas.

Figure 4

A Formulation of the Monkey and Bananas Problem

Section 2.

The row headings are the Tj_ in the definitions of Section 2, and the column headings are the Hi. The definitions of the T_A and the Hi require that the tables of connection are triangular in the sense that the main diagonal and all entires above it are 0. In addition, the subdiagonal (the diagonal immediately below the main diagonal) contains all l's.

Walk is a total function on S, hence its domain is S. We do not need a table of connections for such an operator, because a subproblem of getting into its domain will never be created. We included the table of connection for Walk in Figure 4, because the degenerate case of a definition often helps one understand the definition.

If a column of an operator is all 0's, then that operator will never remove a state from the goal set and will never transform a state outside the goal set into the goal set. An all 0 row indicates that no operator will add or remove a state to the T^ which labels the row.

The above is an example of "difference information" which satisfies our definition of good in Section 2. The most important feature of the tables in Figure 5 is that the triangularity constraint orders the rows (and the columns). This row ordering is the difference ordering - difficult differences are at the top of a table, and easy differences are at the bottom. Of course, there may be several different row orderings

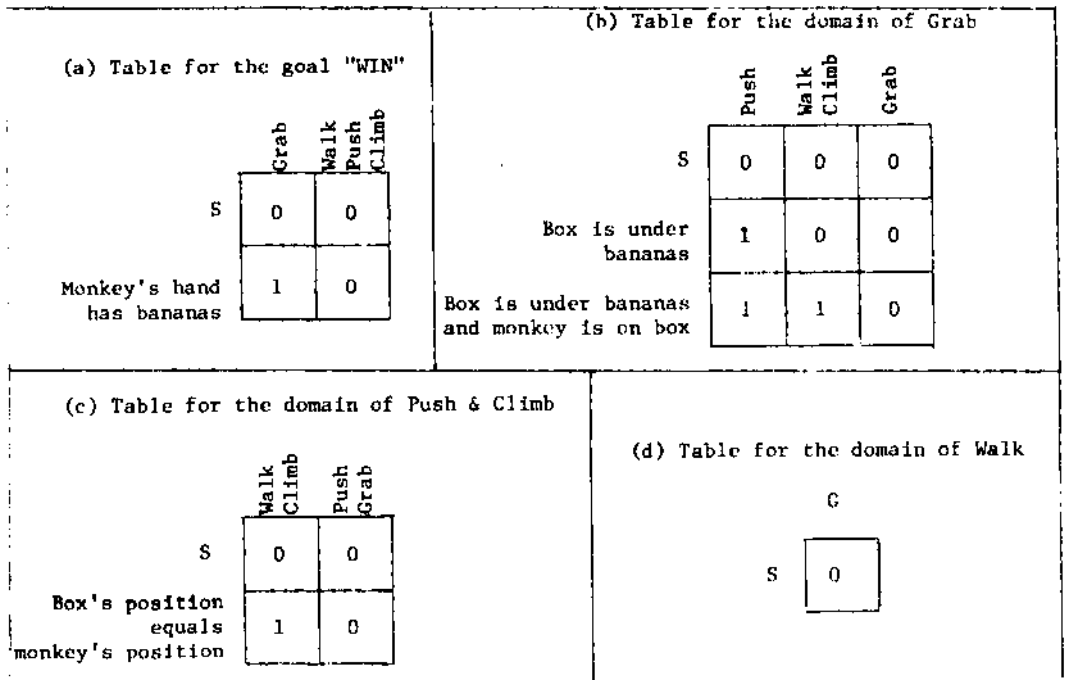


Figure 5
The Table of Connections for
each goal in Monkey and Bananas

which gives rise to a triangular table, in which case any one of them will satisfy our formal definition of good.

Now we can describe how GPS solves Monkey and Bananas using the difference information in Figure 5. Suppose that in the initial state S0 the monkey's hand is empty and the box is not under the bananas. Then the largest difference, M(S0,W), is that the monkey's hand is empty, hence GPS attempts to apply Grab. But so *i* Sgrab, hence GPS sets up the subprobiem of transforming S0 into Sgrab, b^{ut} Grab cannot be used in solving the subproblem because of rule S2.

To solve the subprobiem, GPS attempts to reduce the difference that the box is not under the bananas since this is M(so, SG_{rat}>)• Hence, GPS attempts to apply Push which is not applicable, and the subprobiem of transforming SQ into Sp_{ush} is generated, but S2 restricts the solution of this subprobiem to the operators Walk & Climb. The remaining part of solving this problem is quite straightforward and similar to the way the usual GPS works.

4. Totally-Ordered Solutions

The above discussion raises the question, "Can GPS solve all problems which have a solution?" The answer is no (which can be shown quite easily), because'the differences, together with rules SI and S2, prevent GPS from looking at sequences of operators that may be necessary to find a solution. Hence, the question becomes, "Can we somehow characterize the class of problems which GPS can solve?" The purpose of this section is to show that GPS can solve any problem that has a totally-ordered solution which is our characterization of the class of problems that GPS can solve. Intuitively a totally-ordered solution is one in which one never introduces a difference more difficult than the current differences at any point in solving the problem. This applies to all subproblems at all levels. We will indicate in Theorem 1 that using rules Sl and S2 with good differences produces the class of totally-ordered solutions.

To exhibit this result we have to give definitions of what a problem is and what a solution is. Given a domain as defined in Section 2, a problem is defined by a state $s \in S$, a subset F of G (the set of moves), and a goal $T \subset X$. A solution of a problem defined by the triple <s, F, T> is a sequence of moves $(f_1,\ldots,f_k)$, each f_i $(1 \le i \le k)$ being an element of F, and such that $sf_1,\ldots f_i$ is defined for all i $(1 \le i \le k)$ and $sf_1\ldots f_k \in T$.

At this point we invoke the partition $H_0(T)$, $H_1(T),\ldots,H_{n(T)}(T)$ and recall that each f_i in the above solution is an element of some $H_j(T)$. This yields a sequence $j_1, j_2,\ldots,j_k$ of integers such that for each i, $f_i \in H_{j_i}(T)$. Among them will now occur an integer which is greater than all the integers before it and no less than any element after it, i.e., the "leftmost peak" below, where we have plotted j_i against i.

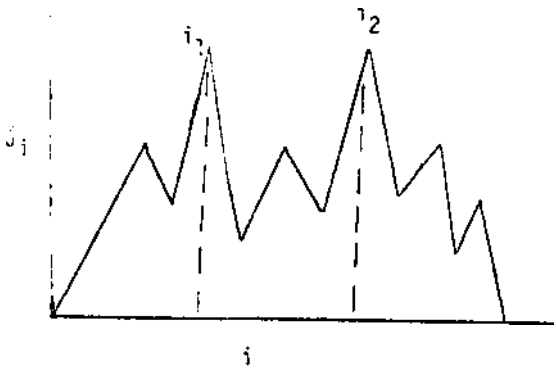


Figure 6

We will initially show that the solution can be interpreted to mean that, at this point, the first effort was made to remove the highest difference between the goal and the initial state s. The moves before this peak are attempts to make it possible to apply the move used at the peak. This sequence (from the start to the peak i_1) will be called the solution to the subproblem, and the sequence between it and the end will be called the solution to the pseudoproblem. The rest of the definitions follow from these considerations.

The triple $<(f_1,\ldots f_{i_1-1}),\ f_{i_1},\ (f_{i_1+1},\ldots f_k)>$ is called the parse of the solution $f_1,\ldots f_k$. It is obvious that given a domain and a solution of any problem <s, F, T>, the parse exists and is unique.

Given the parse as above, we define two problems. The first, which is defined only in the case that $i_1 > 1$, is

$$<s,\ F\text{-}DR(p),\ S_{\bigcup_{p=1}^{n(T)} i_1}>$$

and will be called the subproblem of <s, F, T> corresponding to the solution $(f_1,\ldots f_k)$. The second, which is defined only when $i_1 < k$, is

$$<sf,\ldots f_{i_1},\ F,\ T>$$

and will be called the pseudo-problem of <s, F, T> corresponding to the solution.

Once again, it is obvious that $f_1,\ldots f_{i_1-1}$ is a solution of the subproblem corresponding to the original solution.

Thus, one can say that any solution can be interpreted, i.e., parsed, to be one in which one seeks to apply moves in $H_m(T)$ with "the highest m," making such a move applicable by using moves only in $H_p(T)$ with $p < m$. However, such an interpretation could be given with any ordered partition, having nothing to do with a difference ordering. The reader will recall that in our original definition of a domain the partition on F was so done as to be in keeping with the differences.

To bring our discussion back to the difference orderings rather than with the partitions on the moves, we introduce another definition. Given a problem <s, F, T> and a solution $(f_1,\ldots,f_k)$ and a parse, the solution is called totally ordered, if

ti) $M(sf_1\ldots f_i, T) \geq M(sf_1\ldots f_{i+1}, T)$ for all i $(1 \leq i < k)$, and

tii) each subproblem and pseudoproblem has its corresponding solution totally ordered.

Totally ordered solutions are of importance in that they characterize the kind of solution that can be found by the technique used by GPS. So far we have not formally defined this technique. The statement of the following theorem formalizes the technique as well as characterizes solutions that can be found by it. However, the statement of the theorem needs the following definition.

Given a problem <s, F, T> in a domain and a solution $f_1,\ldots,f_k$ for it, the problem-set for this solution consists of the problem and the members of the problem set for the solutions to the sub- and pseudo-problem of the original problem, if they exist.

We are now ready to state the major theorem of this paper.

Theorem 1: Given a solution $(f_1, f_2,\ldots f_k)$ of a problem <s, F, T> in a domain, the solution is totally ordered, if and only if for each i $(1 \leq i < k)$, f_i is the second element of the parse of the solution of some problem <s', F', T'> in the problem set, and $f_i \in H_m(T')$ implies $M(s', T') = m$.

Intuitively, this theorem says that, if the solution $(f_1,\ldots f_k)$ were found by a search process guided by rules S1 and S2, then it is totally ordered. In addition, such a search process is capable of finding any totally ordered solution.

We shall not include the proof, because it is a long "walk along the definitions" given above and needs some more inessential pedantry like transfinite induction (on a finite space at that!).

5. Conclusion

We now have a working program [Goldstein (1977)] which, using invariant attributes of the problem [Oyen(1975)], would construct a set of properties T_i and partitions H_i as given in Section 2. These would yield what we have previously called triangular connection tables.

It will be noticed that, as previously warned, Theorem 1 of Section 4 does not assure us of

a solution whenever a triangular difference table exists; one has to be blessed with a totally ordered solution - totally ordered by the ordering mechanically or otherwise chosen in the connection table. We have had various problems in which more than one triangular connection table exist, and yet one can prove that some of the connection tables would not yield a solution. This problem has appeared in other, seemingly closely related, garbs in planning programs for Robots, leading to the work on Non Linear Plans [Sacerdoti(1975)]. The analogous problem in our formalization would be the detection of the nonexistence of totally ordered solutions. One approach, that of the detection of "factorable subproblems" [Goldstein 1977)] will be reported on at a future date.

6. Acknowledgements

The work described in this paper was supported by the National Science Foundation under grant MCS 75-23412 to the Case Western Reserve University. The preparation of the paper was partially supported by them under grant MCS76-0-200 to Temple University.

References

1. Banerji, R. B. "Similarities in Games and their use on Strategy Construction," Proceedings of the Symposium on Computers and Automata, pp. 337-359, Polytechnic Press of the Polytechnic Institute of Brooklyn (1971).

2. Banerji, R. B. and Ernst, G. W., "Strategy Construction Using Homomorphisms Between Games," Artificial Intelligence, Vol. 3, pp. 223-248 (1972).

3. Banerji, R. B. and Ernst, G. W. Ernst, "Some Properties of GPS-type Problem Solvers," Report #1179, Jennings Computing Center, Case Western Reserve University, (January, 1971).

4. Coray, G., "An Algebraic Representation of a Puzzle Solving Heuristic," Publication 67, Dept. d'Informatique, University of Montreal (1970).

5. Ernst, G. W., "Sufficient Conditions for the Success of GPS," Journal of the ACM (October, 1969).

6. Ernst, G. W. and Newell, A. GPS: A Case S udy in Generality and Problem Solving, Academic Press (1969).

7. Ernst, G. W. et. al., "Mechanical Discovery of Certain Heuristics," Report #1136-A, Jennings Computer Center, Case Western Reserve University (January, 1974).

8. Goldstein, M., Unpublished Research (1977).

9. Newell, A. and Simon, H. A., "GPS, A Program that Simulates Human Thought," Computers and Thought, E. Feigenbaum & J. Feldman, eds., pp. 279-293 (McGraw-Hill, 1963).

10. Oyen, R. A., "Mechanical Discovery of Invariances for Problem Solving," Report //1168 Jennings Computing Center, Case Western Reserve University (June 1975).

11. Sacerdoti, E. D., "The Non Linear Nature of Plans," Proceedings of the Fourth International Joint Conference on Artificial Intelligence, pp. 206-214 (1975).