

A COMPARISON OF THREE PROBLEM-SOLVING METHODS*

Ranan B. Banerji
Temple University
Philadelphia, Pennsylvania 19122

and

George W. Ernst
Case Western Reserve University
Cleveland, Ohio 44106

Abstract

This paper is a comparison of ABSTRIPS, planning (as defined in Newell and Simon, 1972) and GPS. Each of these methods has parameters that contain heuristic information which is problem dependent. These parameters are used to guide the methods' search and usually cause them to be incomplete in the sense that they cannot solve some problems that have solutions. We show that the parameters of the methods serve the same function in the following sense: Given the parameters for one method we can formulate the parameters for the other two such that all three can solve the same class of problem; i.e. those which have totally ordered solutions. This result is somewhat surprising because the search spaces of the methods are different. The implications of this result to the efficiency of search is discussed at some length.

1. Introduction

The purpose of this paper is to compare 3 different problem solving methods: ABSTRIPS (Sacerdoti, 1974); planning (Newell and Simon, 1972) and GPS (Ernst and Newell, 1969)[†]. The relationship between planning and GPS is reasonably well understood, but their relationship to ABSTRIPS has been much less understood.

All three methods are incomplete in the sense that they cannot solve certain problems that have solutions. Hence, it makes sense to attempt to characterize the class of problems that they can solve. This should allow one to answer such questions as, "Under what conditions can GPS solve any problem that ABSTRIPS can solve?" To answer such questions we will build a formal model of ABSTRIPS

and planning. The model deviates slightly from the way ABSTRIPS really works but it is a close approximation to ABSTRIPS. Using this model and the one for GPS (Ernst, 1969) we can answer the above question.

The formalisms are not in themselves particularly interesting. However, they do reveal the relationship among parameters of the three methods. That is, each method has certain heuristic information as parameters to guide its search. In GPS, the parameters are the differences and difference ordering; in ABSTRIPS they are the criticality levels of the various predicates. Our analysis clearly shows the relationship between these two kinds of parameters. In addition, since there are formal conditions of "good" differences for GPS (Ernst, 1969), these conditions should place constraints on "good" parameters for ABSTRIPS and planning. Sacerdoti (1974) gives some informal rules that he uses in assigning criticality levels to predicates which are consistent with the conditions of good differences but the latter are considerably stronger than his rules.

We start off with a brief description of ABSTRIPS and planning. This is followed by a formalization and analysis of ABSTRIPS and planning. The last section contains a discussion of the three methods.

2. Problem Specification

A problem state of ABSTRIPS (and its predecessor STRIPS (Fikes and Nilsson, 1971)) is a set of formulae in first-order logic. For example, if the formula `Inroom (Robot, Room1)` were part of the state, that would indicate that the Robot was in Room 1 in the state. The set of desired states is also represented by a formula, `W`. Any state which implies that `W` is true is a desired state.

The operators are rules for transforming one state into another. For example, if the robot wanted to move `Box1` to `Box2` it would use the operator `Pushb(Box1, Box2)` where `Pushb` is given in Figure 1. For this operator to be applicable, the preconditions must be true of the current state, i.e., the preconditions give the domain of the operator. The current state is modified by deleting all of the formulae listed under deletions and adding the formulae under additions. For

* This research was supported by the National Science Foundation under grants GJ-1135 and MCS75-23412.

[†]Planning and GPS were conceived of by Al Newell, Cliff Shaw and Herb Simon in the 1950's. The references given here were chosen because they are more accessible than their original publications. This use of the word 'planning' is somewhat different than its use in robotics research.

example, if Nextto(Box1, Door1) was in the current state it would be one of the deletions and Nextto(Box1, Box2) would be one of the additions in forming the new state resulting from applying Pushb(Box1, Box2). The \$1 in Figure 1 is just a variable.

```

Pushb (x,y)
  Preconditions:
    Type(y, Object)
    Pushable(x)
    Nextto(Robot,x)
     $\exists r$  [Inroom(Robot,r) & Inroom(y,r)
        & Inroom(x,r)]
  Deletions:
    Nextto(Robot,$1)
    Nextto(x,$1)
    Nextto($1,x)
  Additions:
    Nextto(x,y)
    Nextto(y,x)
    Nextto(Robot,x)

```

Figure 1.

The ABSTRIPS operator that pushes box x to object y.

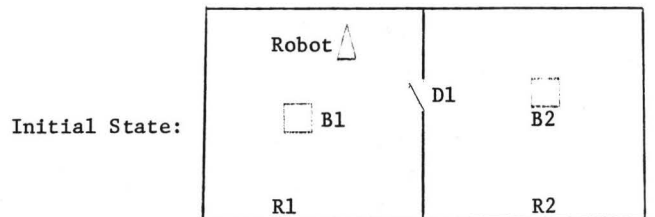
For the purposes of this report the details of the problem specification are not important. The important thing is that each problem has a state space, an initial state, a set of desired states, and a set of operators. Each operator is a partial function from states into states. The domain of the operator is given by its preconditions. It should be noted that Pushb is really a partial function schema; it becomes a partial function after values are specified for x and y, i.e., Pushb(Box1, Box2) is a partial function. The point is that ABSTRIPS is not dependent on the problem specification language but its underlying algebraic structure. Hence, ABSTRIPS could be applied to problems specified in the other languages such as the one given in Ernst, et al (1974) in which states are represented by arrays.

3. Description of ABSTRIPS

ABSTRIPS assumes that each predicate in the preconditions of operators has a criticality level assigned to it. (Actually Sacerdoti (1974) has a semi-automatic way of assigning criticality levels.) The intuitive idea behind criticality levels is that high level predicates are more difficult to change than low level predicates.

A simple problem is given in Figure 2 and ABSTRIPS' solution is depicted in Figure 3. It starts off by trying to transform the initial state in Figure 2 into one in which B1 is next to B2. (This is subproblem 1 in Figure 3.) ABSTRIPS notes that the operator Pushb(B1,B2) is relevant to this problem and applies it, which solves the

problem.



Desired states: Nextto(B1,B2)

Operators: Pushb(x,y)--Push box x to object y.
 PTD(x,y)--Push box x through door y.
 Gotob(x)--Go to box x.
 Pushd(x,y)--Push box x to door y.
 Other operators are not given here because they are not necessary to solve this problem.

Criticality: Level 6--Type, Pushable
 Level 5--Inroom
 Level 2--Nextto

Figure 2. A simple problem for ABSTRIPS.

The reader will quickly note that this is illegal because the initial state is not in the domain of the operator. However, this subproblem is being solved at criticality level 6 which means that all predicates of level 5 or less are ignored. Nextto has been designated as level 2 and Inroom is level 5. Hence, at level 6, ABSTRIPS considers the Preconditions of Pushb(B1, B2) (see Figure 1) to be Type(B2, Object) and Pushable(B1) which is true in the initial state and consequently the operator is applied. Normally, several operators will be necessary to solve the problem but in this case one suffices.

The next subproblem, #2 in Figure 3, is to get from the initial state to a state in which the level 5 and higher predicates of the preconditions of Pushb(B1, B2) are true. In Figure 3 we have only listed the predicates which are false in the initial state for purposes of exposition. (We are assuming that the variable r is assigned the value R2.) ABSTRIPS notes that PTD(B1, D1) is relevant to this subproblem and applies it which solves the subproblem. The level 2 predicates in the preconditions of PTD(B1, D1) are ignored because this subproblem is being solved at criticality level 5.

Next, Pushb(B1, B2) is applied to the result of this subproblem and this new state is transformed into a desired state. But is is already a desired state; i.e., applying (PTD(B1, D1), Pushb(B1, B2) to the initial state results in a state which satisfies Nextto(B1, B2). Hence, no subproblem

	Criticality level	Initial state	Solution	Desired states
1.	6	Fig. 2	Pushb(B1,B2)	Nexttto(B1,B2)
2.	5	Fig. 2	PTD(B1,D1)	Inroom(B1,R2) & Inroom(Robot,R2)
3.	2	Fig. 2	Gotob(B1), Pushd(B1,D1)	Nexttto(B1,D1) & Nexttto(Robot,D1)

Figure 3. Subproblems of the problem in Figure 2.

is created in this situation. This completes the activity at criticality level 5.

The next discrepancy shows up at level 2 because the preconditions of PTD at this level are not satisfied. ABSTRIPS sets up subproblem 3 in Figure 3, and notes that Pushb(B1, D1) is relevant. However, this operator is not directly applicable because the level 2 predicates in its precondition are not satisfied. Gotob(B1) rectifies this situation and hence (Gotob(B1), Pushd(B1, D1)) is applied to the initial state which results in a state *s* in the domain of PTD(D1). Since this subproblem is solved, ABSTRIPS applies PTD(D1) to *s* which yields a new state *t* and then attempts to transform *t* into the domain of Pushb(B1, B2). All of the level 2 predicates in the preconditions of Pushb(B1, B2) are considered because this subproblem is being solved at criticality level 2. But, since *t* is already in the domain of Pushb(B1, B2), this subproblem is trivially solved. Hence, Pushb(B1, B2) is applied to *t* and ABSTRIPS attempts to transform the result into the set of desired states. But the result is already a desired state and thus this subproblem also has a trivial solution. Since criticality level 2 is the smallest level, the solutions of the subproblems comprise a solution of the main problem.

To summarize, ABSTRIPS sets out to solve a problem at a given criticality level by essentially removing from the preconditions of all operators, predicates whose level are less than the given level. After finding a solution it goes to the next level down and adds the predicates at this level back to the preconditions of the appropriate operators. This gives rise to subproblems that ABSTRIPS attempts to solve at the current criticality level, i.e., with lower level predicates removed from operator preconditions. This process is repeated until the subproblems at the smallest criticality level, which is 2 in the above example, are solved. Of course, there is

search involved at all levels because at any given level a problem may have several different solutions some of which may generate unsolvable subproblems.

4. Planning

ABSTRIPS is quite similar to what Newell & Simon (1972) call planning. ABSTRIPS and planning differ in 2 major respects:

- P1. Planning uses an entirely new problem solving space--both states and operators.
- P2. Planning uses only 1 level of abstraction while ABSTRIPS has one for each criticality level except the lowest which is the problem space itself.

Although planning does not presuppose criticality levels, they can be used to define the planning space. We will apply planning to the example of the previous section to contrast the two methods.

In the planning space of the problem in Figure 2 all of the level 2 predicates are removed. (In this example Nexttto is the only level 2 predicate.) Thus, a state in the planning space will be described exclusively by level 5 and level 6 predicates. All level 2 predicates are removed from operators to get the planning space operators. This includes the level 2 predicates in the addition and deletion lists as well as in the preconditions of the operators. Of course, removing these predicates causes some operators to become identity maps, e.g. Pushb, in which case they are not used in the planning space. All level 2 predicates are removed from the formula that describes the desired states to get its analogue in the planning space. Hence, the planning desired states must be given by Inroom(B1, R2) & Inroom(B2, R2). Removing the level 2 predicates from the description of the desired states in Figure 2, gives the null description which represents the set of all states.

The solution to the problem in Figure 2 in the planning space is the single operator PTD(B1, D1). This generates 2 subproblems in the original problem space: transforming the initial state into the domain of PTD (B1, D1), and transforming the result of applying PTD (B1, D1) to the solution of the first subproblem, into a desired state.

In general, planning is a two phase process: the first phase is to solve the problem in the planning space. The second phase is to elaborate the solution so that it works in the original problem space. This is done by inserting operators into the solution that was found in the planning space. The operators to be inserted are found by solving subproblems of transforming the appropriate states into the domains of the appropriate operators.

5. Formalization of ABSTRIPS and Planning

Our formal model of ABSTRIPS is based on two assumptions in addition to the above description.

- A1. The set of desired states is processed in the same way as the preconditions of operators.
- A2. When solving a subproblem at criticality level 1, ABSTRIPS will reject any subproblems of criticality level greater than 1.

By the first assumption we mean that at criticality level 1, ABSTRIPS will ignore all predicates less than level 1 in the description of the desired states. This is what planning does. For example, the Nextto(B1, B2) in the goal states of Figure 3 would be considered at level 2 but not at level 6. We are not suggesting that this is a good idea, but without it and A2 we could not characterize the solutions that ABSTRIPS can find. This will be discussed further in section 7.

The second assumption has to do with efficiency of search. Suppose ABSTRIPS is working at criticality level 2 on a Nextto predicate and an incorrect Inroom predicate is generated by attempting to apply an operator. "Fixing" the Inroom predicate is a level 5 subproblem, but when it is considered at level 2 it is much more difficult because all of the level 2 and level 5 predicates are considered to be part of the operator preconditions. Assumption 2 above specifies that all such subproblems are rejected as being too difficult.

At this point it is convenient to formalize the notion of the difficulty of a subproblem. A subproblem with initial state s and desired states T has difficulty i if there is a level i predicate in T which is not satisfied by s and all predicates of higher criticality level in T are satisfied by s . We define $D_i(s, T)$ to be true if and only if the subproblem defined by s and T has difficulty i .

Only certain operators will be used to solve the subproblems of difficulty i ; these operators will be denoted by H_i . To define H_i we first define the notion of relevance and irrelevance. The operator f is irrelevant to D_i if for all s and T , $D_i(s, T)$ iff $D_i(f(s), T)$. That is, f is irrelevant to D_i if it never changes the difficulty level of a subproblem of difficulty i . All operators f that are not irrelevant to D_i are relevant to D_i . We can now define H_i to be the set of operators which are relevant to D_i but irrelevant to D_j for $j > i$.

The reason for these definitions is that all subproblems that ABSTRIPS attempts to solve at criticality level 1 have difficulty 1; i.e., $D_1(s, T)$ where s is the initial state and T is the desired states. In fact the reason for A1 and A2 is to make this statement true. ABSTRIPS actually violates A1 and hence considers some level 2 pro-

blems at level 6 as shown by the example in section 3. It is not clear from the description in Sacerdoti (1974) whether or not ABSTRIPS adheres to A2. In any case our formal model assumes both A1 and A2. The only operators that ABSTRIPS uses at criticality level 1 are in H_1 . Of course, this is true because all subproblems at this level have difficulty 1.

We formalize ABSTRIPS as a process which searches exhaustively, for well stratified solutions as defined below. Let $(f_1, \dots, f_k)$ be the solution of a subproblem whose initial state is s and whose desired states are given by T . That is, $f_k(\dots f_1(s) \dots) \in T$ and each intermediate state is in the domain of the appropriate operator. Below we will denote a subproblem by a pair (s, T) in which s is the current state and T is the set of desired states and S_f will denote the domain of the operator f . Define a non-empty subsequence $(f_{i_1}, f_{i_2}, \dots, f_{i_t})$ of the solution and an integer m such that

- i) $f_{i_q} \in H_m$ for all q ($1 \leq q \leq t$);
- ii) $f_j \notin H_p$ for $1 \leq j \leq k$ & $p > m$;
- iii) $f_j \notin H_m$ unless $j = i_q$ where $1 \leq q \leq t$.

This subsequence defines the following set of subproblems and a corresponding solution for each.

1. If $i_1 > 1$, define $(f_1, \dots, f_{i_1-1})$ to be the solution of $(s, S_{f_{i_1}})$.
2. For all j when $1 \leq j \leq t-1$ and $i_j + 1 < i_{j+1}$ define $(f_{i_j+1}, f_{i_j+2}, \dots, f_{i_{j+1}-1})$ to be the solution of the problem $(f_{i_j}(\dots f_1(s) \dots), S_{f_{i_{j+1}}})$.
3. If $i_t < k$ then define $(f_{i_t+1}, f_{i_t+2}, \dots, f_k)$ to be the solution of $(f_{i_t}(\dots f_1(s) \dots), T)$.

Such a subsequence of a solution is called the stratification of the solution and the subproblems it defines are called its stratified subproblems.

This definition is conceptually much simpler than it appears to be, as shown by the following example. Suppose that (f_3, f_5, f_6, f_{10}) is the stratification of the solution $(f_1, f_2, \dots, f_{10})$ to the problem (s, W) . This stratification "groups"

the solution as $(f_1 f_2) f_3 (f_4) f_5 f_6 (f_7 f_8 f_9) f_{10}$. Each of the parenthesized quantities are solutions corresponding to stratified subproblems. For example, (f_7, f_8, f_9) is the corresponding solution to $(f_6(\dots f_1(s)\dots), S_{f_{10}})$. Note that there is no subproblem between f_5 and f_6 . In addition, f_3, f_5, f_6 and f_{10} are in some H_m and all the other f 's are in "smaller" H 's.

A solution $(f_1, \dots, f_k)$ with a stratification $(f_{i_1}, \dots, f_{i_t})$ is called well stratified if $D_m(s, T)$ and $D_m(f_{i_j}(\dots f_1(s)\dots), T)$ for $1 \leq j \leq t$,

and the corresponding solution of each of the stratified subproblems is well stratified. That is, each subproblem is of difficulty m . (Note that i-iii above only refer to the H 's and not the D 's.) The definition of well stratified is recursive since it requires all of its subproblems to be well-stratified also. By the way the stratification subsequence is defined, m decreases with each level of recursion. Continuing with the above example, the solution is well stratified if all the difficulty of all top level subproblems is m ; e.g., $D_m(f_3(f_2(f_1(s))), W)$. In addition all of the stratified subproblems must be well stratified. For example, (f_8) might be the stratification of (f_7, f_8, f_9) which groups it as $(f_7) f_8 (f_9)$. This indicates that f_7 is the corresponding solution of $(f_6(\dots f_1(s)\dots), S_{f_8})$. The

difficulty of the subproblems in this stratification is m' where $f_8 \in H_{m'}$, and $m' < m$; e.g., $D_{m'}(f_8(\dots f_1(s)\dots), S_{f_{10}})$.

To summarize then our formal model of ABSTRIPS is an exhaustive search procedure for well stratified solutions. Of course, ABSTRIPS uses a specific search strategy; i.e. first it looks for the stratification subsequence from "left to right" and then for subproblem solutions and backtracks in its own special way. We have not modeled this search order. In addition, our formal model is only approximate since it has A1 and A2 built into it.

Our formal model of planning is an exhaustive search procedure for well stratified solutions that have 2 levels. Again we do not model the order of search. Since planning uses A1, the only approximation used by our model is A2.

6. Totally-Ordered Solutions

GPS uses the strategy of working on difficult subproblems first, saving the easier subproblems for later. The difference ordering is used to specify the difficulty of a subproblem. This strategy is not complete in the sense that some problems that have solutions cannot be solved using this

strategy. Ernst (1969) has characterized the class of solutions that GPS can find as totally-ordered solutions. Intuitively, a totally ordered solution is one which generates a sequence of states in which the difficulty is monotonically nonincreasing. In addition, all subproblem solutions are also totally ordered. The subproblems of a solution are defined in terms of m_0 and i_0 . Given a solution $(f_1, \dots, f_k)$ of a subproblem (s, T) , let m_0 and i_0 be defined as follows:

- i) $f_j \notin H_p$ for $p > m_0$ and $1 \leq j \leq k$;
- ii) $1 \leq i_0 \leq k$
- iii) $f_{i_0} \in H_{m_0}$
- iv) $f_j \notin H_{m_0}$ for $1 \leq j < i_0$

That is m_0 is the "highest level" of the operators in the solution and f_{i_0} is the first of such

operators occurring in the solution. Let $M(s, T)$ denote the difficulty of the subproblem. That is, $M(s, T) = c$ if $D_c(s, T)$ and not $D_j(s, T)$ for $j > c$. Then $(f_1, \dots, f_k)$ is a totally ordered solution to the subproblem (s, T) if

- i) $M(f_j(\dots f_1(s)\dots), T) \geq M(f_{j+1}(\dots f_1(s)\dots), T)$ for $1 \leq j < k$;
- ii) if $i_0 > 1$ then $(f_1, \dots, f_{i_0})$ is a totally ordered solution to the subproblem $(s, S_{f_{i_0}})$.
- iii) if $k > i_0$ then $(f_{i_0+1}, \dots, f_k)$ is a totally ordered solution to the subproblem $(f_{i_0}(\dots f_1(s)\dots), T)$.

Totally-ordered is not only a characterization of the solutions that GPS can find but is also a characterization of the solutions that either ABSTRIPS or planning can find.

Theorem: Any well stratified solution is totally ordered. The proof will not be given in this paper; the proof of this theorem, stated in a slightly different formalism, is given in Banerji and Ernst (1977). Although we have not proven the converse of the theorem we believe that it is true when "good" difference informations is used. If so, then the set of totally ordered solutions is a characterization of the class of solutions

that ABSTRIPS or planning or any other process that can find all and only well stratified solutions can find.

In Ernst (1969) a triangular table of connections was shown to be good difference information for GPS and it appears that this also specifies good parameters for ABSTRIPS and planning. Rather than giving a formal definition of a triangular table of connections we will give an example of it using the Fool's Disks problem. Figure 4 gives the initial state of the Fool's Disks problem, in which there are 4 concentric disks each containing eight 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 independent of the others. The desired state is one in which each of the 8 radial columns sums to 12.

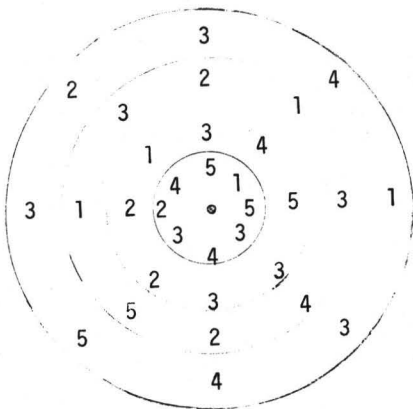


Figure 4.

The initial state in the Fool's Disks problem.

	1/8, 3/8, 5/8, 7/8 revolution	1/4, 3/4 revolution	1/2 revolution
The vertical and horizontal diameters does not sum to 48.	1	0	0
Some diameter does not sum to 24.	1	1	0
Some radius does not sum to 12.	1	1	1

Figure 5.

A triangular table of connections for Fool's Disks.

A triangular table of connections for this problem is given in Figure 5. A 0 entry in the table indicates that a difference is invariant over the operators. For example, the 0 in the upper right corner of Figure 5 indicates that turning a disk 1/2 revolution does not change the sum of the horizontal and vertical diameters. Note that Figure 3 has all 0's above the main diagonal and no 0's on it. The entries below the main diagonal are unimportant. The difference ordering is just the row order; i.e., the top row heading is the most difficult difference while the bottom is the easiest.

GPS only uses the entries on the main diagonal to solve the problem. Hence, first 1/8, 3/8, 5/8 and 7/8 revolution of disks are used to remove the most difficult difference, i.e. used to get the horizontal and vertical diameters to sum to 48. Next, 1/4 and 3/4 revolutions are used to get each diameters to sum to 24. And finally the 180° moves are used to get a desired state. Notice that once a difference is removed it is never reintroduced because of the triangularity of the table in Figure 5.

7. Discussion

In the previous 2 sections we made a number of highly technical definitions ending with a somewhat cryptic theorem, but what can be concluded from all of this? There are 3 major points, listed below; they are followed by a more general discussion.

Point 1. Given the "same" heuristic parameters planning, ABSTRIPS and GPS (or at least our formal models which approximate them) can all solve precisely the same class of problems, i.e. those that have totally ordered solutions. The relationship of parameters of the different methods is described below but note that the D_i of Section 5 are just differences for GPS. This result is somewhat surprising because the search space of GPS is quite different than that of planning or ABSTRIPS. For example, for all states s generated by GPS there is a path from the initial state to s . This is not true for planning or ABSTRIPS because many of the subproblems may not be solvable. For any given problem probably one of the three methods is more efficient than the others, but none is best for all problems.

Point 2. Selecting heuristic parameters can have a much more drastic effect on the search efficiency than selecting one of the three methods, as shown by the Fool's Disks example above. The reason for this is that all of the methods are designed to search efficiently for totally-ordered solutions; they will not even consider subproblems that give rise to unordered solutions. Totally-ordered is defined in terms of the parameters which are problem dependent, and hence the parameters determine whether the class of totally ordered solutions is the same as the class of all solutions or whether the former is much smaller than the latter. In fact, one can view

the problem of selecting good parameters as the problem of making the class of totally ordered solutions as small as possible without eliminating all solutions to problems of interest.

Point 3. Any parameters which are good (in the sense of guiding search) for one method will have analogues which will be good for the other methods. A triangular table of connections (see Section 6) is a property of good difference information for GPS. Another property of good differences is given in Banerji and Ernst (1977). These properties, after suitable translation, are also properties of good parameters of planning and ABSTRIPS. Currently, we are implementing a program to discover difference information for GPS which satisfies these properties. Hence, this program effectively will discover good parameters for planning and ABSTRIPS also.

We feel that point 2 is the most important of the three points. Not much is known about the process of selecting good parameters. Some initial work on this topic is outlined under point 3 but more research is needed on this topic.

To understand the above points, we need to know the relationship among the parameters of the 3 methods. The D_i of Section 5 are just the differences of GPS. That is, subproblem (s, T) possesses difference d if $D_i(s, T)$ where d corresponds to D_i . The subscripts on the D 's give the difference ordering, e.g., the difference corresponding to D_2 in the example in Figure 2 is the easiest (smallest) difference because 2 is the smallest criticality level. The table of connections indicate that the operators in H_1 should be used to reduce D_1 . Note that in our formal models of both ABSTRIPS and GPS an operator f may be relevant to D_1 but not in H_1 . This will happen when $f \in H_j$ for some $j > i$ which indicates that f is relevant to the more difficult difference D_j and hence is used solely for the purpose of reducing D_j . Including f in H_1 would give rise to more search but not to more totally-ordered solutions.

Planning, in many respects, looks like ABSTRIPS with 2 criticality levels. In fact our formal model of the previous section is really a hybrid of ABSTRIPS and planning; it has many levels of abstraction like ABSTRIPS, and like planning assumes A1. The heuristic parameters of planning are basically the difference information of GPS. Although planning requires a entire planning space as input, this information is really derived from the difference ordering. In fact one of the innovations in ABSTRIPS is its ability to automatically generate the various planning or abstracted spaces from the criticality levels of the predicates.

Our formal model of ABSTRIPS uses assumptions A1 and A2 (Section 5) which is a deviation from the

real ABSTRIPS. Why might ABSTRIPS want to violate A1 and/or A2? The answer lies in the fact that the class of totally-ordered solutions may be a little too restrictive because some solutions may not be totally-ordered. Violating A1 and/or A2 allows ABSTRIPS to find solutions that are "almost but not quite" totally-ordered. We do not know how to measure the degree of unorderedness or what these violations due to the efficiency of search. To see how an unordered solution is generated note that ABSTRIPS' solution to the problem in Figure 2 is unordered because the difficulty of the original problem is level 2 whereas the difficulty of some of the subproblems is level 5. This unordered solution is a result of violating A1. To have planning solve the problem in Figure 2 the desired states had to be more completely specified as described in Section 4 because planning adheres to A1.

What is the relationship of the properties of good differences in (Ernst, 1969) to the parameters of ABSTRIPS? As mentioned above the criticality levels of predicates basically specify the differences and difference ordering. If the table of connections is generated as defined in Section 5 (i.e. use operators in H_i to reduce D_i), then the properties of good differences are automatically satisfied except that some of the H_i may be empty. What this means is that if all solutions of a problem contain subproblems of difficulty i , then the problem has no totally-ordered solution. In other words the criticality level assigned to predicates is not good. A better approach is to ask what assignment of criticality levels results in no empty H 's. This is very similar to the approach in Eavarone and Ernst (1970).

We feel that the biggest restriction in ABSTRIPS is its use of a very restrictive class of differences, i.e., those defined by the predicates in the problem specification. Most of the good ideas in ABSTRIPS apply to a more general class of differences; and we hope that the restriction is removed in the future. As an example of a difference not defined by a predicate in the problem specification, the logic task in Newell and Simon (1972) uses the number of different proposition letters in an expression as a difference. Figure 5 also contains some more complex differences. We believe that ABSTRIPS can be extended to a wider class of differences such as these.

References

1. Banerji, R. B. and Ernst, G. W., Some Properties of GPS-Type Problem Solvers, Report #1179, Computer Engineering and Information Sciences Dept., Case Western Reserve University, 1977.
2. Eavarone, D. S. and Ernst, G. W., A Program that Discovers Good Difference Orderings and Table of Connections for GPS, 1970 IEEE Systems Science and Cybernetics Conference Record, 1970.

3. Ernst, G. W., Banerji, R. B., Hookway, R. J., Oyen, R. A. and Shaffer, D. E., Mechanical Discovery of Certain Heuristics, Report #1136, Computer Engineering Department, Case Western Reserve University, 1974.
4. Ernst, G. W., "Sufficient Conditions for the Success of GPS," Journal of the ACM, October, 1969.
5. Ernst, G. W. and Newell, A., GPS: A Case Study in Generality and Problem Solving, Academic Press, 1969.
6. Fikes, R. E. and Nilsson, N. J., "STRIPS: A New approach to the Application of Theorem Proving to Problem Solving," Artificial Intelligence, 1971, pp. 189-208.
7. Newell, A. and Simon, H. A., Human Problem Solving, Prentice-Hall, 1972.
8. Sacerdoti, E. D., "Planning in a Hierarchy of Abstraction Spaces," Artificial Intelligence, 1974, pp. 115-135.