

March 1971

FAILURE TESTS AND GOALS IN PLANS

by

Richard E. Fikes

Artificial Intelligence Group

Technical Note 53

SRI Project 8973

The research reported herein was sponsored by the Advanced Research Projects Agency and the National Aeronautics and Space Administration under Contract NASW-2164.

# FAILURE TESTS AND GOALS IN PLANS

by

Richard E. Fikes

February 3, 1971

## I INTRODUCTION

This note describes a proposal for the form that plans for the Stanford Research Institute mobile automaton might take and the rules an interpreter might use to execute these plans. We are particularly concerned here with adding tests to a plan that allow the executor to determine whether execution of a plan is succeeding and that specify what is to be done when a failure occurs.

We proceed by developing a syntax and semantics for plans in the context of STRIPS (Stanford Research Institute Problem Solver),<sup>1\*</sup> the program that acts as a planner for the automaton system. Subsequently, we present an algorithm that STRIPS can use to create the tests for the executor to use.

We assume that the reader has some familiarity with the capabilities of our robot vehicle and our means of modeling both the robot's external environment and its action routines (or operators). See Refs. 1 and 2 for this background material.

## II SYNTAX AND SEMANTICS OF A STRIPS PLAN

The role of the planner in our system is to determine what the robot should do to solve a given task. We may consider the output of

---

\*References are listed at the end of this note.

the planner to be a program each of whose steps is an operator to be executed or a test to be made on the world model. One simple form that a plan might have is the following:

```
BEGIN
  DO operator1;
  GOAL preconditions for operator2;
  DO operator2;
  GOAL preconditions for operator3;
  :
  :
  GOAL preconditions for operatork;
  DO operatork;
  GOAL task statement
END
```

The executor of the plan will call the operator routines contained in the DO statements and will employ a theorem prover such as QA3.5<sup>3,4</sup>

(a resolution-based deductive system) to determine whether the predicates

contained in the GOAL statements are true in the world model at each step.

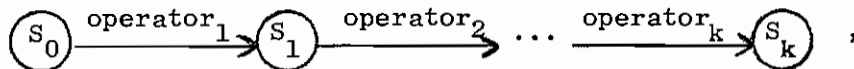
The GOAL statements in a plan provide a means of testing whether execution of the plan is proceeding successfully. We assign to the planner the responsibility of generating the information needed for these tests. In the form for a plan given above this information is merely the GOAL statements that determine whether the next operator in the plan can be applied, and a final GOAL statement to determine whether the task has been completed.

We would like the planner to provide statements in the plan that will allow the executor to determine as soon as possible when execution of the plan is failing. For example, if the success of some operator in a plan depends on a box B1 being at some location LA and a side effect of execution of some earlier operator in the plan is the determination that B1 is not located at LA, then we would like the plan to contain statements

that would allow the executor to realize that the new location of B1 is going to cause an eventual failure. Execution of the plan could then be discontinued without incurring the costs of doing the operations that precede the point in the plan where the actual failure would occur.

To see how STRIPS provides this type of information in its plans, we first note that it employs a means-ends analysis search strategy similar to that of GPS<sup>5</sup> to grow a search tree whose nodes specify world models and some of whose arcs represent planned operator applications. The means-ends analysis strategy directs search by creating subgoals and determining relevant operators. The theorem prover, QA3.5, is used to ask questions about world models, such as whether a goal is true in a model or whether an operator's preconditions are true in a model. A complete description of STRIPS can be found in Ref. 1.

STRIPS can create a plan when there is a path through its search tree of the following form:



where  $S_0$  is the initial world model, each  $S_i$  is the world model that would be produced by applying operator<sub>*i*</sub> to model  $S_{i-1}$ , and  $S_k$  is the model in which the task statement is satisfied. The operators on this path define the DO statements for the plan; each operator's preconditions and the task statement define the plan's GOAL statements.

STRIPS forms additional tests for the plan by extracting information from the proofs produced by QA3.5. For the search-tree path we are considering, QA3.5 will have been used to prove that the task statement

is true in  $S_k$ , and that the preconditions for each operator <sub>$i$</sub>  are true in  $S_{i-1}$ . For each of these proofs, STRIPS determines which axioms from the world model were used. Any of these axioms from a given model that were not added to the model as one of the effects of the previous operator in the plan must also have been in the model before that operator was executed. This implies that STRIPS can add a test to the plan before the DO statement for that operator that will prevent the DO statement's execution if one of these axioms is missing from the model. This process can be iterated backwards through the plan so that tests can be inserted before each DO statement indicating what axioms the remainder of the plan assumes exist in the model at that point. Figure 1 shows an example plan into which test indicators have been inserted.

To include these new tests in the plan, we first surround each DO statement and the preceding preconditions GOAL statement by a labeled BEGIN statement and an END statement to form ALGOL-like blocks as follows:

```

BEGIN
  B1:BEGIN
    GOAL preconditions for operator1;
    DO operator1
  END;
  B2:BEGIN
    GOAL preconditions for operator2;
    DO operator2
  END;
  :
  :
  Bk:BEGIN
    GOAL preconditions for operatork;
    DO operatork
  END;
  GOAL task statement
END

```

```

BEGIN
  (Test for A1)
  (Test for A2 and A4)
  (Test for A6)
  (Test for A7 and A3)
  GOAL preconditions for op1;      (Proof used axioms A8 and A5)
  DO op1                          (Adds axiom A9)

  (Test for A1)
  (Test for A6)
  (Test for A7 and A3)
  GOAL preconditions for op2;      (Proof used axioms A9, A2, and A4)
  DO op2                          (Adds axioms A12 and A10)

  (Test for A1 and A12)
  (Test for A7 and A3)
  GOAL preconditions for op3;      (Proof used axioms A10 and A6)
  DO op3                          (Adds axiom A11)

  (Test for A1 and A12)
  GOAL preconditions for op4;      (Proof used axioms A11, A7, and A3)
  DO op4                          (Adds axioms A10 and A13)

  GOAL task statement              (Proof used axioms A1, A12, and A13)
END

```

FIGURE 1 ABSTRACT PLAN INDICATING  
POSSIBLE TESTS

Each new test in the plan is represented by a FAILTEST statement having the following form:

$$\text{FAILTEST } A_1 \wedge A_2 \wedge \dots \wedge A_n \text{ FOR } B_j, B_{j+1}, \dots, B_m;$$

where each  $A_i$  is an axiom used by QA3.5 in a proof and each  $B_i$  is the label of a BEGIN statement in the plan. When the executor encounters a FAILTEST statement it attempts to determine whether the conjunction of axioms is true in the current state. If the proof attempt succeeds, then execution continues with the next statement in the plan; if the proof attempt fails, then those blocks which begin with the labels listed in the right side of the FAILTEST statement are deleted from the plan before execution continues.

When deletions occur during the execution of a FAILTEST statement, the plan is no longer complete and its execution will almost certainly fail to satisfy some GOAL statement remaining in the plan. When the executor encounters an unsatisfied GOAL statement, it can recall the planner to create a plan that will achieve the unsatisfied goal. If the planner successfully produces such a plan, then the new plan can be executed to "get back onto the track" of the original plan. If the planner fails and the goal is the task-statement goal, then the system cannot accomplish the task. If the planner fails with any other goal, then the executor may still be able to retain some of the original plan by continuing execution at the next block.\*

---

\* At this point execution of the plan is in deep trouble, and in most cases FAILTEST statements will be encountered that will cause deletion of the remainder of the plan.

To determine which blocks should be listed in a FAILTEST statement, the planner assumes that if some axiom that it intends to be true at some point in a plan is not true at that point when the plan is being executed, then any proof in which that axiom participated is invalid. This assumption implies that the FAILTEST statement at that point in the plan can indicate deletion of any block whose only function in the plan is to add axioms used in the invalidated proofs.

Figure 2 shows an instantiation of the example plan from Figure 1 in the form that STRIPS would produce it. The task for the instantiated plan is to push the three boxes BOX1, BOX2, and BOX3 to the same location. The GO(x,y) operator used in the plan moves the robot from location x to location y where x and y are constrained to be in the same room. The PUSH(b,x,y) operator in the plan causes the robot to push an object b from location x to location y, where both the robot and the object are assumed to begin at location x.

Note that in the Figure 2 plan each pair of statements

```
GOAL preconditions for opi
DO opi
```

has been replaced by an IF statement. STRIPS produces these IF statements in the following form:

```
IF preconditions for opi THEN DO opi
ELSE GOAL relevant results of opi ,
```

where the relevant results of an operator are those axioms added by the operator that were used during the creation of the plan in the proof of some subsequent operator's preconditions or the task-statement goal. Hence the IF

```

BEGIN
B1:BEGIN
    FAILTEST AT (BOX1,LA) FOR B1,B2,B3,B4;
    FAILTEST AT (BOX2,LB)^SAMEROOM(LB,LA) FOR B1;
    IF AT (ROBOT,LD)^SAMEROOM(LD,LB) THEN DO GO (LD,LB)
        ELSE GOAL AT (ROBOT,LB)
    END;
B2:BEGIN
    FAILTEST AT (BOX1,LA) FOR B2,B3,B4;
    IF AT (ROBOT,LB)^AT (BOX2,LB)^SAMEROOM(LB,LA) THEN DO PUSH (BOX2,LB,LA)
        ELSE GOAL AT (ROBOT,LA)^AT (BOX2,LA)
    END;
B3:BEGIN
    FAILTEST AT (BOX1,LA)^AT (BOX2,LA) FOR B3,B4;
    FAILTEST AT (BOX3,LC)^SAMEROOM(LC,LA) FOR B3;
    IF AT (ROBOT,LA)^SAMEROOM(LA,LC) THEN DO GO (LA,LC)
        ELSE GOAL AT (ROBOT,LC)
    END;
B4:BEGIN
    FAILTEST AT (BOX1,LA)^AT (BOX2,LA) FOR B4;
    IF AT (ROBOT,LC)^AT (BOX3,LC)^SAMEROOM(LC,LA) THEN DO PUSH (BOX3,LC,LA)
        ELSE GOAL AT (BOX3,LA)
    END;
    GOAL (Ex) (AT (BOX1,x)^AT (BOX2,x)^AT (BOX3,x))
END

```

FIGURE 2 PLAN FOR THE THREE BOXES  
PROBLEM

statement indicates to the executor that if the operator can be applied then it should be, otherwise a new plan should be found and executed that will produce the same results that the operator was to achieve. For example, in the Figure 2 plan if the robot is not initially at location LD, then GO(LD,LB) will not be applied; instead a new plan will be created to move the robot to location LB, since that was the desired result of applying GO(LD,LB). Also, consider the case where the robot is at LD initially so that GO(LD,LB) is applied; if GO(LD,LB) fails to move the robot to LB, then the preconditions for PUSH(BOX2,LB,LA) will not be satisfied in block B2 and a new plan will be created to achieve the desired results of pushing BOX2 to LA.

Note also in Figure 2 the implications of the assumption that a single missing axiom invalidates any proofs in which the axiom participated. This assumption causes the first FAILTEST statement in the plan to indicate deletion of the entire plan if BOX1 is not at location LA. One might argue that when a particular axiom is not true as expected an attempt should be made to generate and execute a plan that would make the axiom true so that the original plan could still be used. That strategy can easily lead to nonoptimal plans. For example, if BOX1 is not initially at LA during execution of the plan for the three boxes problem, then this strategy would dictate that a new plan be constructed and executed to move BOX1 to LA; once this was accomplished, the original plan could be executed. This solution of the problem would require moving all three boxes to a new location; the strategy proposed here of deleting the entire original plan would cause creation and execution of a new plan involving the movement of two boxes to the third. Hence, this strategy prefers to expend greater replanning effort so that the resulting plan will require less execution effort.

Finally, note that some of the tests indicated in the Figure 1 plan have disappeared in the Figure 2 plan. The deleted tests were determined by STRIPS to be redundant. For example, a FAILTEST statement could have been inserted in block B1 to assure that BOX3 was at location LC; but since failure of that test would cause only block B3 to be deleted from the plan, the test need not be made in any of the blocks preceding block B3.

### III THE STRIPS ALGORITHM FOR ADDING TESTS TO A PLAN

In this section we present and illustrate an algorithm for creating the FAILTEST statements and IF statements for a STRIPS plan. The input to the algorithm is a plan consisting of a sequence of blocks followed by a GOAL statement containing the task statement. Each block in the input plan has the following form:

```
Bi:BEGIN
      GOAL preconditions for operatori;
      DO operatori
      END
```

The algorithm also knows which axioms were used to prove each operator's preconditions and the task statement while the plan was being created by STRIPS. Finally, the algorithm knows from the operator descriptions the axioms added to the world model by each operator in the plan.

The algorithm is as follows:

1. For each GOAL statement in the plan do the following procedure:
  - 1.1 Create an axiom list, AXL, consisting of the axioms used by QA3.5 in the goal's proof during the creation of the plan.
  - 1.2 Proceed backwards through the plan from the GOAL statement to the initial BEGIN and execute the following procedure at each DO statement encountered:

- 1.2.1 Determine whether the DO statement's operator added to the model any of the axioms on the list, AXL. If it did not, then take no action at this DO statement. If it did, then continue at the next step.
  - 1.2.2 Delete from AXL those axioms added to the model by the operator and store at the DO statement a list of the deleted axioms.
  - 1.2.3 Mark the block in which the DO statement occurs as being relevant to the goal under consideration by storing at the block the goal and a copy of the list, AXL.
2. For each GOAL statement in the plan, do the following procedure:
    - 2.1 Create a null goal list, GL.
    - 2.2 Create a null block list, BL.
    - 2.3 Proceed backwards through the plan from the GOAL statement to the initial BEGIN and execute the following procedure at each block encountered:
      - 2.3.1 Determine whether the block is marked as being relevant to the goal under consideration. If it is then continue at the next step. If it is not, then continue at step 2.3.3.
      - 2.3.2 Set AXL to be the list of axioms stored at the block with the goal under consideration.
      - 2.3.3 Determine whether the block is marked relevant to any goal that is either not on the list GL or is not the goal under consideration. If it is, then take no further action at this block. If it is not, then continue at the next step.
      - 2.3.4 If there is a GOAL statement in the block under consideration, then add the goal to the list GL.
      - 2.3.5 Add the block under consideration to the list BL.
      - 2.3.6 Insert a FAILTEST statement at the beginning of the block under consideration to test for the conjunction of the axioms on AXL and to delete the blocks on list BL.
  3. For each block on the plan, do the following procedure:
    - 3.1 Form a wff, RELRESULTS, by conjoining all the axioms stored at the block's operator (in step 1.2.2 of the algorithm).

### 3.2 Replace the statements

```
GOAL preconditions for opi;  
DO operatori
```

in the block by the statement

```
IF preconditions for opi THEN DO operatori  
ELSE GOAL RELRESULTS .
```

We will illustrate the operation of this algorithm with the following input plan:

```
BEGIN  
B1:BEGIN  
    DO op1 (Adds axioms A7 and A10)  
    END;  
B2:BEGIN  
    GOAL preconditions for op2; (Proof used axioms A5 and A7)  
    DO op2 (Adds axioms A4 and A11)  
    END;  
B3:BEGIN  
    DO op3 (Adds axioms A1 and A12)  
    END;  
B4:BEGIN  
    DO op4 (Adds axioms A2, A6, and A13)  
    END;  
B5:BEGIN  
    GOAL preconditions for op5; (Proof used axioms A1, A6, and A8)  
    DO op5 (Adds axioms A3 and A14)  
    END;  
    GOAL task statement; (Proof used axioms A2, A3, A4, and A9)  
END
```

The parenthesized comments in the plan indicate the axioms that are added by each operator (as given in the operator descriptions) and the axioms used to prove each of the goals during creation of the plan. Operators  $op_1$ ,  $op_3$ , and  $op_4$  are assumed to have no preconditions.

The algorithm begins by executing the step 1 procedure for each GOAL statement in the plan. Consider first the GOAL statement in block B2. We shall refer to the goal in this statement as G2. In step 1.1 AXL will be created as the list (A5,A7). Next the procedure of step 1.2 is executed

for the DO statement in block B1.  $Op_1$  added axiom A7 to the model, and A7 is an element of AXL. Hence, in step 1.2.2 list AXL becomes (A5) and the list (A7) is stored at the DO statement. In step 1.2.3 block B1 is marked relevant to goal G2 by storing the pair (G2;(A5)) at the block. This completes the step 1 processing for goal G2.

For the goal in block B5, which we shall refer to as G5, the list AXL is created as (A1 A6 A8). The first DO statement encountered is in block B4; at step 1.2.2 AXL becomes (A1 A8) and the list (A6) is stored at the DO statement. In step 1.2.3 the pair (G5;(A1 A8)) is stored at block B4. For the DO statement in block B3, AXL becomes (A8), the list (A1) is stored at the DO statement, and the pair (G5;(A8)) is stored at block B3. Since neither of the DO statements in blocks B2 and B1 add axiom A8 to the model, no further action is taken for goal G5.

For the task-statement goal, which we shall refer to as Gt, the list AXL is created as (A2 A3 A4 A9). The first DO statement encountered is in block B5; it causes AXL to become (A2 A4 A9), the list (A3) to be added to it, and the pair (Gt;(A2 A4 A9)) to be stored at block B5. For the DO statement in block B4, AXL becomes (A4 A9), the list (A2) is added to the statement, and the pair (Gt;(A9)) is stored at the block B2. Since the DO statement in block B1 does not add axiom A9, no further action is taken for goal Gt.

This completes the step 1 processing and leaves the plan in the following form:

```

BEGIN
B1:BEGIN                                (G2;(A5))
    DO op1;                             (Adds axioms A7 and A10) (A7)
    END;
B2:BEGIN                                (Gt;(A9))
    GOAL preconditions for op2;         (Proof used axioms A5 and A7)
    DO op2                               (Adds axioms A4 and A11) (A4)
    END;
B3:BEGIN                                (G5;(A8))
    DO op3                               (Adds axioms A1 and A12) (A1)
    END;
B4:BEGIN                                (G5;(A1 A8))(Gt;(A4 A9))
    DO op4                               (Adds axioms A2, A6, and A13) (A6) (A2)
    END;
B5:BEGIN                                (Gt;(A2 A4 A9))
    GOAL preconditions for op5;         (Proof used axioms A1, A6, and A8)
    GO op5                               (Adds axioms A3 and A14) (A3)
    END;
    GOAL task statement;                 (Proof used axioms A2, A3, A4, and A9)
END

```

Step 2 of the algorithm makes a second pass through the plan by again considering each GOAL statement. For goal G2, the step 2.3 process is executed once at block B1. At step 2.3.1 we determine that block B1 is marked relevant to G2. At step 2.3.2, AXL becomes (A5). The block passes the test at step 2.3.3, no action is taken at step 2.3.4, and list BL becomes (B1) at step 2.3.5. At step 2.3.6 the following statement is added to block B1:

```

FAILTEST A5 FOR B1 .

```

No further action is taken for goal G2.

For goal G5, B4 is the first block considered in the step 2.3 process. Since B4 is marked as being relevant to G5, step 2.3.2 is executed and sets AXL to be (A1 A8). Since block B4 is marked relevant to Gt and Gt is not an element of list GL, no further action is taken for block B4. For block B3, AXL becomes (A8) in step 2.3.2, BL becomes (B3) in step 2.3.5, and the following statement is added to block B3 in step 2.3.5:

```

FAILTEST A8 FOR B3 .

```

Since blocks B2 and B1 are not marked relevant to G5, no further action is taken for G5.

For goal Gt, block B5 is marked relevant and therefore AXL is set to be (A2 A4 A9). The test is passed at step 2.3.3, list GL is set to be (G5) at step 2.3.4, list BL is set to be (B5) at step 2.3.5, and the following statement is added to block B5 at step 2.3.6:

```
FAILTEST A2/A4/A9 FOR B5 .
```

At block B4, AXL becomes (A4 A9), the test in step 2.3.3 is passed since G5 is on list GL, BL becomes (B4 B5), and the following statement is added to B4:

```
FAILTEST A4/A9 FOR B4,B5 .
```

At block B3, the test in step 2.3.1 causes step 2.3.2 to be skipped, BL becomes (B3 B4 B5), and the following statement is added:

```
FAILTEST A4/A9 FOR B3,B4,B5 .
```

At block B2, AXL becomes (A9), GL becomes (G2 G5), BL becomes (B2 B3 B4 B5), and the following statement is added:

```
FAILTEST A9 FOR B2,B3,B4,B5 .
```

At block B1, the test in step 2.3.1 causes step 2.3.2 to be skipped, BL becomes (B1 B2 B3 B4 B5), and the following statement is added:

```
FAILTEST A9 FOR B1,B2,B3,B4,B5 .
```

No further action is taken for goal Gt.

The algorithm's final pass through the plan occurs in step 3. At that time IF statements are added to blocks B2 and B5. Note that in

blocks B1, B3, and B4 the operators have no preconditions, so that the IF statements for those blocks collapse into DO statements and leave the blocks unchanged.

This completes the algorithm and produces the following plan:

BEGIN	
B1:BEGIN	(G2;(A5))
FAILTEST A9 FOR B1,B2,B3,B4,B5;	
FAILTEST A5 FOR B1;	
DO op <sub>1</sub>	(Op <sub>1</sub> adds axioms A7 and A10)
END;	(RELRESULTS for op <sub>1</sub> is (A7))
B2:BEGIN	(Gt;(A9))
FAILTEST A9 FOR B2,B3,B4,B5;	
IF preconditions for op. THEN DO op <sub>2</sub>	(Preconditions proof used
ELSE GOAL A4	axioms A5 and A7)
END;	(Op <sub>2</sub> adds axioms A4 and A11)
	(RELRESULTS for op <sub>2</sub> is (A4))
B3:BEGIN	(G5;(A8))
FAILTEST A4^A9 FOR B3,B4,B5;	
FAILTEST A8 FOR B3;	
DO op <sub>3</sub>	(Op <sub>3</sub> adds axioms A1 and A12)
END;	(RELRESULTS for op <sub>3</sub> is A1)
B4:BEGIN	
FAILTEST A4^A9 FOR B4,B5;	
DO op <sub>4</sub>	(Op <sub>4</sub> adds axioms A2,A6, and A13)
END;	(RELRESULTS for op <sub>4</sub> is (A2 A6))
B5:BEGIN	
FAILTEST A2^A4^A9 FOR B5;	
IF preconditions for op <sub>5</sub> THEN DO op <sub>5</sub>	(Preconditions proof used
ELSE GOAL A3	axioms A1, A6, and A8)
END;	(Op <sub>5</sub> adds axioms A3 and A14)
	(RELRESULTS for op <sub>5</sub> is (A3))
GOAL task statement;	(Proof of task statement
END	used axioms A2,A3,A4, and A9)

## REFERENCES

1. R. E. Fikes and N. J. Nilsson, "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving," Technical Note 43R, Artificial Intelligence Group, Stanford Research Institute, Menlo Park, California (January 1971).
2. J. H. Munson, "Robot Planning, Execution, and Monitoring in an Uncertain Environment," draft paper submitted for presentation to the Second International Joint Conference on Artificial Intelligence, London, England, September 1-3, 1971.
3. C. Green, "Theorem Proving by Resolution as a Basis for Question-Answering Systems," in Machine Intelligence 4, B. Meltzer and D. Michie, eds., pp. 183-205 (American Elsevier Publishing Co., New York, 1969).
4. C. Green, "Application of Theorem Proving to Problem Solving," Proc. Intl. Joint Conf. on Artificial Intelligence, pp. 219-239 (The Mitre Corporation, Bedford, Massachusetts, 1969).
5. G. Ernst and A. Newell, GPS: A Case Study in Generality and Problem Solving, ACM Monograph Series (Academic Press, New York, 1969).