STANFORD RESEARCH INSTITUTE
Menlo Park, California 94025 · U.S.A.

April 1971

MONITORED EXECUTION OF ROBOT PLANS PRODUCED BY STRIPS

by

Richard E. Fikes

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

Artificial Intelligence Group

Technical Note 55

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.

MONITORED EXECUTION OF ROBOT PLANS PRODUCED BY STRIPS

Richard E. Fikes
Stanford Research Institute
Menlo Park, California

We describe PLANEX1, a plan executor for the Stanford Research Institute robot system. The problem-solving program STRIPS creates a plan consisting of a sequence of actions, and the PLANEX1 program carries out the plan by executing the actions. PLANEX1 is designed so that it executes only that portion of the plan necessary for completing the task, reexecutes any portion of the plan that has failed to achieve the desired results, and initiates replanning in situations where the plan can no longer be effective in completing the task. The scenario for an example plan execution is given.

I INTRODUCTION

In this paper we describe PLANEX1, a program that functions as the plan executor for a robot system being developed at Stanford Research Institute. The particular robot device with which we work is a battery-powered wheeled vehicle connected to a PDP-10 computer via a radio link. Its capabilities include moving in rooms and hallways, manipulating objects with a push bar, and taking television pictures (all under computer control).

An appropriate task for this system might be "push the box to the next room." When given such a task, the system uses a planning program called STRIPS [1] to determine an appropriate sequence of executable actions (i.e., a plan). If STRIPS succeeds in producing a plan, PLANEX1 is called to accomplish the task. Since the plan must be executed in the real world by a mechanical device as opposed to being carried out in a mathematical space or by a simulator, consideration must be given by the executor to the possibility that operations in the plan may not accomplish what they were intended to, that data obtained from sensory devices may be inaccurate, and that mechanical tolerances may introduce errors as the plan is executed.

We are therefore interested in a plan execution scheme with the following properties: (1) When new information obtained during plan execution implies that some remaining portion of the plan need not be executed, the executor should recognize such information and omit the unneeded plan steps. (2) When execution of some portion of the plan fails to achieve the intended results, the executor should recognize the failure and either direct reexecution of some portion of the plan or call for a replanning activity.

II STRIPS PLANS

Before describing PLANEX1 we must first review some of the design features of our robot system and the planning program, STRIPS. The system maintains a collection of statements in the first-order predicate calculus to model the robot's real world environment. This model includes a large number of facts and relations describing the state of the robot and the attributes of various objects, rooms, doors, and walls. For example, the model might contain statements like the following:

$$\text{AT(ROBOT,3,5)};$$
$$(\forall \text{W,X1,Y1,X2,Y2})[\text{AT(W,X1,Y1)} \wedge \sim((\text{X1=X2}) \wedge (\text{Y1=Y2})) \supset \sim\text{AT(W,X2,Y2)}]$$

The first statement indicates the x-y coordinates (in feet) of the robot's location, and the second statement indicates that an object can be at only one location in any given model.

A task is also represented in the system as a statement in the first-order predicate calculus. For example, the task "collect together box1, box2, and box3" might be represented as NEXTTO(BOX1, BOX2) ∧ NEXTTO(BOX2,BOX3). The goal of the planner is to find a sequence of executable actions that will produce a state of the world in which the task statement can be shown to be true.

The executable actions from which plans are constructed are parameterized programs that cause the robot to perform some specific activity. For example, we might have actions that cause the robot to turn k degrees, move forward n feet, go to adjacent room rx, or find and follow an unobstructed path to location (x,y). In general, any capability that we expect will be frequently needed by the robot is a candidate for being programmed and included in its repertoire of actions.

STRIPS requires a description (or model) of each available executable action indicating under what conditions the action can be executed and what its expected effects are on the world model. In particular, each action description consists of the following components: (1) Action name and parameters. (2) Preconditions (A list of predicate calculus statements that must be true in the world before the action can be executed). (3) Deletions (A list of predicate calculus atoms indicating the statements that execution of the action is expected to remove from the world model). (4) Additions (A list of predicate calculus statements that execution of the action is expected to add to the world model). The information in the action descriptions provides STRIPS with the capability of computing the anticipated model that would result from executing a given action in the situation described by a given model.

STRIPS proceeds by conducting a heuristic search in a space whose objects are world models and whose operators are defined by the action descriptions. The QA3.5 theorem prover [2] is used by STRIPS to answer questions about individual models; for example, QA3.5 is used to determine whether an action's preconditions are true in a model or whether the task statement is true in a model. When STRIPS finds an acceptable plan, it has computed the world model anticipated before and after each action in the plan, has proven that the preconditions of each action in the plan are true in the model anticipated at the time of the action's execution, and has proven that the task statement is satisfied in the model anticipated after completion of the plan's execution. The information obtained by PLANEX1 from STRIPS when an acceptable plan has been found includes the sequence of actions that form the plan, the statements added to the world model by each action in the plan, and the world model statements used in proving each action's preconditions and the task statement.

III KERNEL MODELS

PLANEX1 extracts from STRIPS' output a kernel model for before and after each action in the plan. Each kernel model contains exactly those statements STRIPS anticipated would be true at that point in the plan's execution and that were used in the proof of some action's preconditions or in the proof of the task statement. These kernel models are the basic information needed for monitoring plan execution, since they indicate at each step in the plan a minimal set of requirements for ensuring a successful execution of the entire plan.* Figure 1 shows a sample output from STRIPS and the sequence of kernel models computed by PLANEX1.

The kernel models are computed in the following manner: Given an n-step plan, let K_i for $1 \leq i \leq n$ denote the kernel model preceding the ith action in the plan; let K_{n+1} denote the kernel model following the plan's nth action. Then: (1) K_{n+1} consists of the statements used in the

```
Action 1
  (Model statements used in preconditions proof:
   A01,A02)
  (Statements added to model:  A11,A12,...,A18)
Action 2
  (Model statements used in preconditions proof:
   A03,A04, A11,A12)
  (Statements added to model:  A21,A22,...,A26)
Action 3
  (Model statements used in preconditions proof:
   A05,A06, A13,A14, A21,A22)
  (Statements added to model:  A31,A32,A33,A34)
(Model statements used in task statement proof:
 A07,A08, A15,A16, A23,A24, A31,A32)
```

Fig. 1a. Sample STRIPS Output

*The kernel models are similar in spirit to the assertions used in Floyd's program verification scheme [3] in that they both define at a given point in a program the necessary and sufficient conditions for successful execution.

```
K1:  A01,A02
     A03,A04
     A05,A06
     A07,A08
Action 1
K2:  A03,A04, A11,A12
     A05,A06, A13,A14
     A07,A08, A15,A16
Action 2
K3:  A05,A06, A13,A14, A21,A22
     A07,A08, A15,A16, A23,A24
Action 3
K4:  A07,A08, A15,A16, A23,A24, A31,A32
```

Fig. 1b. Sample Kernel Models

proof of the task statement. (2) For $1 \leq i \leq n$, K_i consists of the statements used to prove the preconditions of the ith action in the plan plus those statements in K_{i+1} not added to the model by the ith action.

The design of STRIPS and the algorithm used to compute kernel models ensures that the kernels for a given n-step plan have the following property: If the axioms in a kernel model K_i are true in the robot's current environment and if the effects of action executions are as indicated in the action descriptions, then the sequence of plan steps i, i+1, ..., n is executable and its execution will complete the task.

IV EXECUTION STRATEGY OVERVIEW

We can now describe our basic strategy for plan execution. At each execution step we want PLANEX1 to find a kernel model whose statements can be proven in the system's current world model. Furthermore, we want this search to begin with the final kernel and proceed toward the initial kernel (i.e., in the order K_{n+1}, K_n, ..., K_1). If kernel K_{n+1} is found to be satisfied, then the task is completed, and a success exit can be taken. If any other kernel K_i is satisfied, then a step can be taken toward completing the task by executing the plan's ith action. If none of the kernels are satisfied, then a replanning activity is required to determine the next execution step.

This execution strategy has the desired properties mentioned in Sec. I above in that it will execute only that portion of the plan that is necessary for completing the task, it is free to reexecute any portion of the plan that has failed to achieve the desired results, and it can recognize situations in which the plan can no longer be effective in completing the task.

When replanning is necessary, PLANEX1 calls STRIPS with each of the kernel models as a separate task statement. STRIPS can then search for a plan whose execution will produce a state in which one of the kernels is true. If such a plan is found, PLANEX1 executes it and then continues execution of the original plan. This replanning scheme allows the system to react swiftly and intelligently in the frequently occurring situation where creation and execution of a new one- or two-step plan can get the robot "back onto the track" of the original plan.

V TESTING THE KERNEL MODELS

PLANEX1 begins each step in the execution process by searching for a satisfied kernel model. Since the testing of each kernel model requires a search by QA3.5 (our theorem prover) for a proof, this process can require a significant amount of computing time and therefore introduce a major inefficiency into the system. We have responded to this difficulty by transforming the set of kernel models for a plan into a single triangular execution table, and by defining a search algorithm on the execution table that minimizes the theorem-proving requirements.

The basic observation that motivates the execution table and its associated search algorithm is that some statements appear in more than one of a plan's kernel models and therefore the theorem prover should not be called upon to prove these statements more than once. For example, note in the kernel models shown in Fig. 1b that statements A07 and A08 appear in all four kernels, and statements A15 and A16 appear in kernels 2-4. Our search algorithm takes maximal advantage of these redundancies. In the Fig. 1 example, if statement A15 is found to be unprovable during the test of kernel 4, then the tests for kernels 2 and 3 are omitted since those kernels also contain statement A15 and therefore cannot be satisfied.

The structure of the execution table for an n-step plan is determined by decomposing the plan's kernel models as shown in Fig. 2. In this decomposition, each P_{ij} is a set of statements defined as follows: If we consider the statements in K_1 to have been added to the model by a fictitious initializing program, action 0, and if we consider the task statement to be the preconditions of a fictitious success program, action n+1; then P_{ij} is the set of statements that were added to the model by action j and that were used in the proof of the preconditions of action i. Given this decomposition, an execution table of the form shown in Fig. 3 is constructed. Figure 4 shows the table that would be constructed for the example plan from Fig. 1.

$P_{i,0}$	$P_{i,1}$	···	$P_{i,i-1}$
$P_{i+1,0}$	$P_{i+1,1}$	···	$P_{i+1,i-1}$
⋮	⋮		⋮
$P_{n+1,0}$	$P_{n+1,1}$	···	$P_{n+1,i-1}$

Fig. 2. Decomposition of kernel model i for an n-step plan

Note the following properties of the execution table for a plan: (1) The entries in the row of the table that ends with action i are exactly those axioms used in the proof of the preconditions

fail					
$P_{1,0}$	action 1				
$P_{2,0}$	$P_{2,1}$	action 2			
$P_{3,0}$	$P_{3,1}$	$P_{3,2}$	action 3		
⋮	⋮	⋮			
$P_{n+1,0}$	$P_{n+1,1}$	$P_{n+1,2}$	···	$P_{n+1,n}$	success

Fig. 3. Execution table for an n-step plan

fail				
A01,A02	action 1			
A03,A04	A11,A12	action 2		
A05,A06	A13,A14	A21,A22	action 3	
A07,A08	A15,A16	A23,A24	A31,A32	success

Fig. 4. Execution table for the plan shown in Fig. 1

for action i. (2) The entries in the column of the table headed by action i are exactly those axioms that were added to the model by action i and that were used in some later precondition proof or in the task statement proof. (3) The kernel model preceding action i in the plan is the rectangular portion of the table consisting of columns 0 through i-1 of rows i through n+1.

Once the execution table has been constructed for a plan, the algorithm shown in Fig. 5 is employed to execute the plan. The algorithm proceeds by searching from left-to-right and bottom-to-top that portion of the execution table that is the kth kernel. The value of k is initialized to n+1 at the beginning of each search, and when a table entry is found that cannot be proven, the value of k is set to the highest numbered kernel that does not contain that entry. The search is designed so that no table entry is tested more

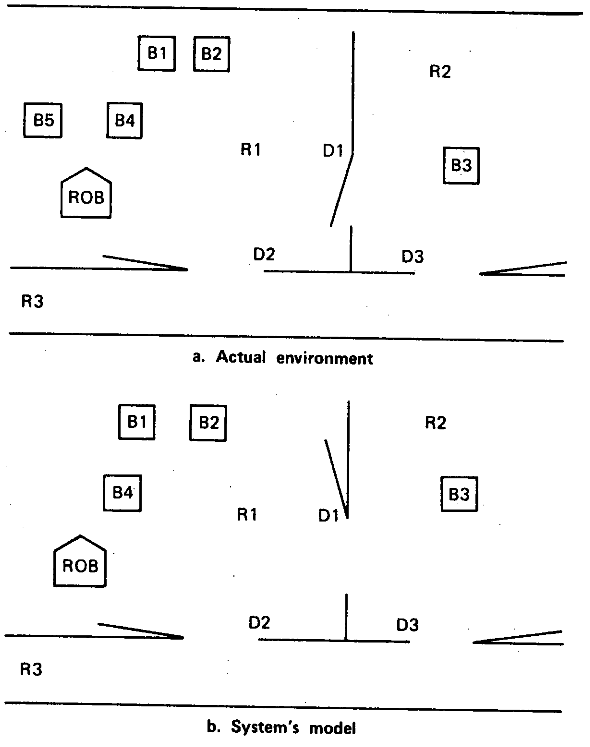
```
1.   base ← n+1; k ← n+1.
2.   j ← 0.
3.   i ← base.
4.   Test $P_{ij}$; if not true, then go to step 9.
5.   i ← i-1; if $i \geq k$, then go to step 4.
6.   j ← j+1; if $j < k$, then go to step 3.
7.   If k = n+1, then return success.
8.   Execute action k; go to step 1.
9.   If j = 0, then replan.
10.  base ← k-1; k ← j; go to step 2.
```

Fig. 5. Execution Algorithm for an n-Step Plan

than once, the kernels are tested in the desired order (i.e., K_{n+1}, ..., K_1), and no test is ever made on a table entry that occurs only in kernels already shown to be unsatisfiable. When all of kernel n+1 has been proven, the algorithm returns "success." When all of any other kernel has been proven, an action is executed and the search is begun again. When a proof attempt fails and the value of k is to be set to 0, the replanning process described in Sec. IV above is initiated.

VI AN EXAMPLE PLAN EXECUTION

We will now trace through an anticipated scenario of the solution to a task by our robot system to indicate the characteristics of the plan execution scheme we have described. Assume we want the system to push three boxes together and that we give it the task statement NEXTTO(B1,B2)∧ NEXTTO(B2,B3). Assume that the system's world model of the robot's environment is as diagrammed in Fig. 6b, but that the environment is actually as diagrammed in Fig. 6a. Note that the model has incorrectly located box B1 so that the system does not know that B1 is already next to box B2, the model does not indicate the existence of box B5, and the model incorrectly indicates that door D1 is open.



Boxes: B1,B2,B3,B4,B5	Doors: D1,D2,D3
Rooms: R1,R2,R3	Robot: ROB

Fig. 6. Environment for example task

STRIPS creates a plan with the following action steps: GOTO(B1), PUSHTO(B1,B2), GOADJRM(R2), GOTO(B3), PUSHADJRM(B3,R1), PUSHTO(B3,B2). The descriptions used by STRIPS for the actions in this plan can be translated as follows:

GOTO(BX)
<u>Preconditions</u>: BX is an object in the same room as the robot. <u>Effects</u>: Determine a route to move the robot next to BX and then move the robot along that route.

PUSHTO(BX,BY)
<u>Preconditions</u>: BX is a pushable object, BY is an object in the same room as BX, and the robot is next to BX. <u>Effects</u>: Determine a route to push BX next to BY and then push BX along that route.

GOADJRM(RX)
<u>Preconditions</u>: RX is a room and there is an open door connecting RX with the room in which the robot is located. <u>Effects</u>: Determine a route to move the robot to the open door, move the robot along that route, and move the robot through the door.

PUSHADJRM(BX,RX)
<u>Preconditions</u>: BX is a pushable object, the robot is next to BX, RX is a room, and there is an open door connecting RX with the room in which the robot is located. <u>Effects</u>: Determine a route to push BX to the open door, push BX along that route, and push BX through the door.

The execution table for the plan is constructed as shown in Fig. 7.

When plan execution begins, the values of both k and base are set to 7. The first statement to be proved is NEXTTO(B1,B2), and no proof is found for it. This implies that only kernels 1 and 2 can possibly be satisfied; hence, the value of k is set to 2. The value of base is set to 6 to indicate that the rows below row 6 in the table have already been tested. The next unprovable statement encountered is NEXTTO(ROBOT,B1) in row 2 of the table. This failure causes both base and k to be set to 1. The proof attempt for the row 1 entry succeeds and the action GOTO(B1) is executed. The GOTO program determines a route for the robot to where it believes box B1 to be, and attempts to move the robot through that route. The route causes the robot to unexpectedly bump into box B5. The GOTO program responds by moving the robot a short distance backwards, taking a picture of the bumped object, entering a description of the object into the model, and returning control to PLANEX1.

The second search through the execution table leads again to the execution of the GOTO(B1) action, since the truth values of the statements in the table were not changed by the first execution step. Hence, PLANEX1 responds to the failure of an action by noting that the appropriate conditions exist for trying the action again and by reexecuting the action. The GOTO program determines a route that avoids both boxes B4 and B5, moves the robot to where it believes box B1 to be, and notes that no contact was made with the box. This failure causes the GOTO program to take a picture of the area where box B1 was modeled to be and to correct the location of the box in the model. Control is then returned to PLANEX1.

During the next iteration through the execution table a proof is found for NEXTTO(B1,B2), and

Replan							
Type(B1,object); Inroom(B1,R1);* Inroom(ROBOT,R1)*	GOTO(B1)						
Inroom(B1,R1); Inroom(B2,R1); Pushable(B1); Type(B2,object)*	Nextto(ROBOT, B1)	PUSHTO(B1,B2)					
Inroom(ROBOT,R1); Connectsrooms (D1,R1,R2);* Doorstatus (D1,open)*			GOADJRM(R2)				
Inroom(B3,R2); Type(B3,object)			Inroom(ROBOT, R2)*	GOTO(B3)			
Connectsrooms (D1,R2,R1); Doorstatus (D1,open); Pushable(B3)*			Inroom(ROBOT, R2)	Nextto(ROBOT, B3)	PUSHADJRM (B3,R1)		
Type(B2,object); Pushable(B3); Inroom(B2,R1)					Inroom(B3,R1); Nextto (ROBOT,B3)	PUSHTO(B3,B2)	
		Nextto(B1,B2)				Nextto(B2,B3)	Task Completed

*These axioms can be deleted from the table by an editing routine since they occur previously in the same column of the table.

Fig. 7. Execution table for example plan

the first nonprovable statement encountered is NEXTTO(B2,B3). The unsatisfied statements encountered as the search continues include NEXTTO(ROBOT,B3) and INROOM(B3,R1) in the last column of row 6 and INROOM(ROBOT,R2) in row 5. These unsatisfied entries leave k set to 3 and base set to 4. The search successfully completes kernel 3, and the action GOADJRM(R2) is executed. Note that PLANEX1 has made use of the new information in the model that box B1 is next to B2 and has omitted execution of the PUSHTO(B1,B2) action. The GOADJRM program moves the robot to door D1, discovers that the door is closed, corrects the model, and returns control to PLANEX1.

The unsatisfied statements encountered during the next search of the execution table include NEXTTO(B2,B3) in row 7, INROOM(B3,R1) and NEXTTO(ROBOT,B3) in row 6, and DOORSTATUS(D1,OPEN) in row 5. The row 5 unsatisfied entry would cause k to be set to 0 and therefore demands a replanning effort. STRIPS is given each of the kernel models as a separate planning goal and constructs the following plan to achieve the kernel that precedes the final step in the original plan: GOADJRM(R3), GOADJRM(R2), GOTO(B3), PUSHADJRM(B3,R3), PUSHADJRM(B3,R1). This new plan uses doors D2 and D3 to get the robot to box B3 and to push B3 into room R1.

PLANEX1 creates an execution table for the new plan and continues execution with the new table. Assume that execution of the new plan proceeds normally so that the robot and box B3 end up next to each other in room R1. Then PLANEX1 returns to the execution table for the original plan and proceeds as before. The only unsatisfied statement found during the next search through the table is NEXTTO(B2,B3) in the bottom row. All the statements in row 6 are satisfied and therefore the action PUSHTO(B3,B2) is executed. If the PUSHTO program succeeds in pushing box B3 to box B2, then in the next search of the execution table the statements in the bottom row will be satisfied, and PLANEX1 will therefore return "success."

We have seen in this example how PLANEX1 can reexecute a portion of a plan that has failed, can omit execution of a portion of the plan whose effects have already been realized, and can use replanning to respond to situations unaccounted for in the original plan.

ACKNOWLEDGMENT

The design of the robot system described in this paper is a group effort of the Stanford Research Institute Artificial Intelligence Group. My special thanks for contributions to the design of PLANEX1 go to Peter Hart, John Munson, and Nils Nilsson.

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

REFERENCES

[1] R. Fikes and N. Nilsson, STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving, Second IJCAI (September 1971).

[2] C. Green, Theorem Proving by Resolution as a Basis for Question-Answering Systems, in Machine Intelligence 4 (American Elsevier Publishing Co., Inc., New York, 1969).

[3] R. W. Floyd, Assigning Meanings to Programs, Proc. Symp. Appl. Math., Vol. 19 (American Mathematical Society, 1967).