July 1972

# LEARNING AND EXECUTING GENERALIZED ROBOT PLANS

By

Richard E. Fikes
Peter E. Hart
Nils J. Nilsson

Artificial Intelligence Center

Technical Note 70

SRI Project 1530

The research reported herein was supported at SRI by the Advanced Research Projects Agency of the Department of Defense, monitored by the U.S. Army Research Office-Durham under Contract DAHC04 72 C 0008.

## II SUMMARY OF STRIPS

### A. Description

Because STRIPS is basic to our discussion, let us briefly outline its operation. (For a complete discussion and additional examples, see Ref. 1.) The primitive actions available to the robot vehicle are precoded in a set of action routines. For example, execution of the routine GOTHRU(D1,R1,R2) causes the robot vehicle actually to go through the doorway, D1, from room R1 to room R2. The robot system keeps track of where the robot vehicle is and stores its other knowledge of the world in a model composed of well-formed formulas (wffs) in the predicate calculus. Thus, the system knows that there is a doorway D1 between rooms R1 and R2 by the presence of the wff CONNECTSROOMS(D1,R1,R2) in the model.

Tasks are given to the system in the form of predicate calculus wffs. To direct the robot to go to room R2, we pose for it the goal wff INROOM(ROBOT,R2). The planning system, STRIPS, then attempts to find a sequence of primitive actions that would change the world in such a way that the goal wff is true in the correspondingly changed model. In order to generate a plan of actions, STRIPS needs to know about the effects of these actions; that is, STRIPS must have a model of each action. The model actions are called operators and, just as the actions change the world, the operators transform one model into another. By applying a sequence of operators to the initial world model, STRIPS can produce a sequence of models (representing hypothetical worlds)

ultimately ending in a model in which the goal wff is true. Presumably then, execution of the sequence of actions corresponding to these operators would change the world to accomplish the task.

Each STRIPS operator must be described in some convenient way. We characterize each operator in the repertoire by three entities: an add function, a delete function, and a precondition wff. The meanings of these entities are straightforward. An operator is applicable to a given model only if its precondition wff is satisfied in that model. The effect of applying an (assumed applicable) operator to a given model is to delete from the model all those clauses specified by the delete function and to add to the model all those clauses specified by the add function. Hence, the add and delete functions prescribe how an operator transforms one state into another; the add and delete functions are defined simply by lists of clauses that should be added and deleted.

Within this basic framework STRIPS operates in a GPS-like manner[6]. First, it tries to establish that a goal wff is satisfied by a model. (STRIPS uses the QA3 resolution-based theorem prover[3] in its attempts to prove goal wffs.) If the goal wff cannot be proved, STRIPS selects a "relevant" operator that is likely to produce a model in which the goal wff is "more nearly" satisfied. In order to apply a selected operator the precondition wff of that operator must of course be satisfied: This precondition becomes a new subgoal and the process is repeated. At some point

we expect to find that the precondition of a relevant operator is already satisfied in the current model. When this happens the operator is applied; the initial model is transformed on the basis of the add and delete functions of the operator, and the model thus created is treated in effect as a new initial model of the world.

To complete our review of STRIPS we must indicate how relevant operators are selected. An operator is needed only if a subgoal cannot be proved from the wffs defining a model. In this case the operators are scanned to find one whose effects would allow the proof attempt to continue. Specifically, STRIPS searches for an operator whose add function specifies clauses that would allow the proof to be successfully continued (if not completed). When an add function is found whose clauses do in fact permit an adequate continuation of the proof, then the associated operator is declared relevant; moreover, the substitutions used in the proof continuation serve to instantiate at least partially the arguments of the operator. Typically, more than one relevant operator instance will be found. Thus, the entire STRIPS planning process takes the form of a tree search so that the consequences of considering different relevant operators can be explored. In summary, then, the "inner loop" of STRIPS works as follows:

(1) Select a subgoal and try to establish that it is true in the appropriate model. If it is, go to Step 4. Otherwise,

(2) Choose as a relevant operator one whose add function specifies clauses that allow the incomplete proof of Step 1 to be continued.

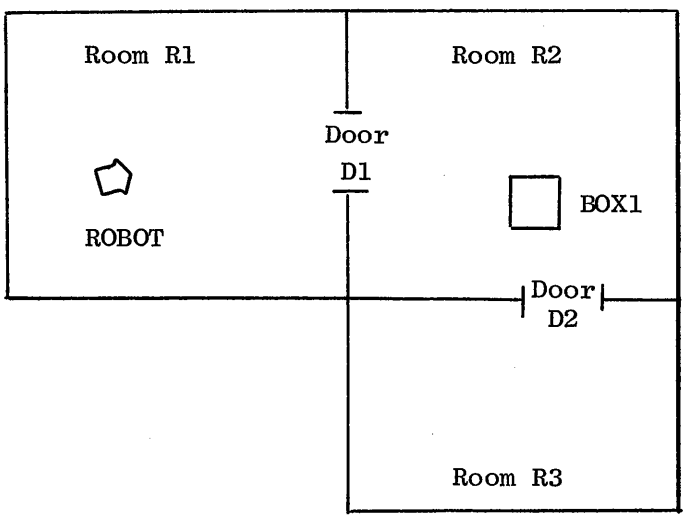
(3) The appropriately instantiated precondition wff of the selected operator constitutes a new subgoal. Go to Step 1.

(4) If the subgoal is the main goal, terminate. Otherwise, create a new model by applying the operator whose precondition is the subgoal just established. Go to Step 1.

The final output of STRIPS, then, is a list of instantiated operators whose corresponding actions will achieve the goal.

B. An Example

An understanding of STRIPS is greatly aided by an elementary example. The following example considers the simple task of fetching a box from an adjacent room. Let us suppose that the initial state of the world is as shown below:



Initial Model

Mo: INROOM(ROBOT,R1)

CONNECTS(D1,R1,R2)

CONNECTS(D2,R2,R3)

BOX(BOX1)

INROOM(BOX1,R2)

$\vdots$

$(\forall x\ \forall y\ \forall z)[\text{CONNECTS}(x,y,z) \Rightarrow \text{CONNECTS}(x,z,y)]$

Goal wff

Go: $(\exists x)[\text{BOX}(x) \wedge \text{INROOM}(x,R1)]$

We assume for this example that models can be transformed by two operators GOTHRU and PUSHTHRU, having the descriptions given below. Each description specifies an operator schema indexed by schema variables. We will call schema variables parameters, and denote them by strings beginning with lower-case letters. A particular member of an operator schema is obtained by instantiating all the parameters in its description to constants. It is a straightforward matter to modify a resolution theorem prover to handle wffs containing parameters,[1] but for present purposes we need only know that the modification ensures that each parameter can be bound only to one constant; hence, the operator arguments (which may be parameters) can assume unique values. (In all of the following we denote constants by

strings beginning with capital letters and quantified variables by x, y, or z):

GOTHRU(d,r1,r2)

(Robot goes through Door d from Room r1 into Room r2.)

Precondition wff

INROOM(ROBOT,r1) ∧ CONNECTS(d,r1,r2)

Delete List

INROOM(ROBOT,$)

Our convention here is to delete any clause containing a predicate of the form INROOM(ROBOT,$) for any value of $.

Add List

INROOM(ROBOT,r2)

PUSHTHRU(b,d,r1,r2)

(Robot pushes Object b through Door d from Room r1 into Room r2.)

Precondition wff

INROOM(b,r1) ∧ INROOM(ROBOT,r1) ∧ CONNECTS(d,r1,r2)

Delete List

INROOM(ROBOT,$)

INROOM(b,$)

Add List

INROOM(ROBOT,r2)

INROOM(b,r2).

When STRIPS is given the problem it first attempts to prove the goal $G_0$ from the initial model $M_0$. This proof cannot be completed; however, were the model to contain other clauses, such as INROOM(BOX1,R1), the proof attempt could continue. STRIPS determines that the operator PUSHTHRU can provide the desired clause; in particular, the partial instance PUSHTHRU(BOX1,d,r1,R1) provides the wff INROOM(BOX1,R1).

The precondition $G_1$ for this instance of PUSHTHRU is

$G_1$: INROOM(BOX1,r1)

$\wedge$ INROOM(ROBOT,r1)

$\wedge$ CONNECTS(d,r1,R1).

This precondition is set up as a subgoal and STRIPS tries to prove it from $M_0$.

Although no proof for $G_1$ can be found, STRIPS determines that if r1 = R2 and d = D1, then the proof of $G_1$ could continue were the model to contain INROOM(ROBOT,R2). Again STRIPS checks operators for one whose effects could continue the proof and settles on the instance GOTHRU(d,r1,R2). Its precondition is the next subgoal, namely:

$G_2$: INROOM(ROBOT,r1)

$\wedge$ CONNECTS(d,r1,R2).

STRIPS is able to prove $G_2$ from $M_0$, using the substitutions r1 = R1 and d = D1. It therefore applies GOTHRU(D1,R1,R2) to $M_0$ to yield:

$M_1$: INROOM(ROBOT,R2)

CONNECTS(D1,R1,R2)

CONNECTS(D2,R2,R3)

BOX(BOX1)

INROOM(BOX1,R2)

⋮

$(\forall x\ \forall y\ \forall z)$[CONNECTS(x,y,z) $\Rightarrow$ CONNECTS(x,z,y)].

Now STRIPS attempts to prove the subgoal $G_1$ from the new model $M_1$. The proof is successful with the instantiations r1 = R2, d = D1. These substitutions yield the operator instance PUSHTHRU(BOX1,D1,R2,R1), which applied to $M_1$ yields

$M_2$: INROOM(ROBOT,R1)

CONNECTS(D1,R1,R2)

CONNECTS(D1,R2,R3)

BOX(BOX1)

INROOM(BOX1,R1)

⋮

$(\forall x\ \forall y\ \forall z)$[CONNECTS(x,y,z) $\Rightarrow$ CONNECTS(x,z,y)].

Next, STRIPS attempts to prove the original goal, $G_0$, from $M_2$. This attempt is successful and the final operator sequence is

GOTHRU(D1,R1,R2)

PUSHTHRU(BOX1,D1,R2,R1).