AUTOMATIC PLANNING FROM A FRAMES POINT OF VIEW

Richard E. Fikes
SRI Artificial Intelligence Center

I. Introduction

In this note I would like to consider some of the ways in which Minsky's ideas concerning frames relate to automatic planning systems. I will proceed by making some observations about how knowledge is represented in planners and will relate some of the frame ideas to actual running systems. Hopefully, the presentation will increase the reader's understanding and intuition of both the systems and the ideas.

Typically, a planning system is one that assists in the accomplishment of tasks by generating plans and then monitoring their execution. The active agent that actually carries out the plans might be another part of the system (e.g. a mechanical manipulator), or a human that the system is providing instructions to. A task is usually specified to such a system by describing an initial situation and a desired (goal) situation. The system is aware of a collection of actions that the active agent can carry out, and the plan it produces is a sequence of these actions that are expected to transform the initial situation into the desired situation.

Taking a broader view, a planner can be thought of as being a constructor of scenarios or thematic frames for whatever reasons they are needed. For example, the system may be involved in understanding a natural language narrative that describes a sequence of actions, or it may be participating in an instructional dialogue with a human about some procedure, or it may be answering questions about how some task would be accomplished.

II. Action Frames as Plan Steps

The building blocks that a planner uses to construct a new plan can be usefully thought of as frames that describe some action. In order for a planner to make use of such existing "action" frames, they must include certain descriptive information about themselves. In particular, the planner needs to know under what conditions the action represented by a frame can be carried out. These are the action's "preconditions" or "applicability conditions". They provide a source of subgoals during the planning process and allow the planner to assure that an executable plan is being produced. In addition, the planner needs a description of the transformations that execution of an action is expected to make. This information allows the planner to simulate hypothetical action sequences and thereby investigate their results. Finally, the planner needs to know what kinds of tasks or subgoals each action is useful for achieving. This information indicates to the planner which actions to consider putting into the plan being constructed.

Typically, the action frames that are available to a planner as potential plan steps actually represent an entire class of actions by containing slots that must be filled by specific values. For example, a robot action for moving a box might include slots for the name of the box, the starting location of the move, and the final location of the move; hence, the action can be used to move any box from any location to any other location. An action frame will specify constraints on the values that can fill its slots such as type requirements and, for the box example, perhaps a restriction that the two locations be in the same room. A significant part of the planning activity is the determining of values for these slots that both satisfy the frame's constraints and produce an instance of the frame that is useful in the new plan. Hence, the planning process involves both selecting and "instantiating" a sequence of action frames that will achieve the desired goal situation.

III. Plans as Action Frames

One would like for the plans produced by a planner to be retained as new frames so that the system could be considered to be learning new scenarios and new themes. In some of our earlier work with the STRIPS planning system we made a reasonably successful attempt at developing such a learning mechanism [Fikes, Hart, and Nilsson, 1972]. The challange in such an effort is to have the system derive and store with the new frame the information required for its use by other parts of the system including the planner. This information includes all the kinds of information that is included with the original action frames, namely a set of preconditions for the plan, a description of the effects of the plan's execution, and indications of the class of tasks the plan is relevant to achieving. In addition, for the new frame to be useful in situations other than just reoccurrences of the one for which it was originally constructed, it must somehow be generalized so that, like the other action frames, there are slots at the frame's "terminals" that can be filled in with values suited to any of a class of situations and goals.

The basic goal of the STRIPS learning work was to generalize and save plans so that they could be used as single steps in future plans. Two aspects of this work seem worth mentioning here.

A. Kernels

First, the preconditions for the plan were directly available as a side effect of the determination of "kernels" preceding each step of the plan. The kernel preceding step i of a plan is a partial description of the situation that the planner expects to exist after execution of the first i-1 steps of the plan. In particular, it specifies a set of relational statements that must be true if the remainder of the plan is to

succeed. Hence, in effect, the kernel for each step is the preconditions for the remainder of the plan. The kernels for a plan can be computed in a direct manner by considering what relational statements were made true by each step in the plan and by knowing what statements were needed to assure the achievement of the final goal and each action's preconditions.

A plan's kernels provide extremely valuable information for many uses of the plan, and therefore are prime candidates for inclusion in the plan's action frame. First of all, the kernel preceeding the first step of the plan specifies the preconditions for the plan. Second, the kernels specify what tests the execution monitor should make after each action to determine whether the plan is proceeding as expected. Third, the kernels are useful for modifying or extending the plan to accomodate a new situation or new task goals. That is, instead of forming a completely new plan to accomplish some task goal, the planner has the option of modifying an existing frame by adding and/or changing some of its steps. The basic information that is needed to allow such modification to occur is contained in the kernels since each kernel indicates what must remain true between any two adjacent steps in a plan. Hence, the planner can add any number of steps between step i-1 and step i of an existing plan as long as the kernel between those two steps is not violated.

B. Plan Generalization

The second comment to be made about the STRIPS learning work relates to the generalization that was done on the plans before they were stored. The goal was to "unbind" the slots in each of the plan's steps so that, whenever possible, they became unvalued slots in the frame for the entire plan. The logical structure of the plan imposes restrictions on this unbinding process so that some pairs of slots are required to take on the same value (i.e., they become a single slot in the new frame), and others must retain their binding (i.e., they lose their status as slots in the new frame). For example, if one step of the plan causes the robot to go to a door, and the next step causes the robot to go through a door, then the door in those two steps must be the same door.

This generalization process essentially parameterizes the plan and thereby provides it with enough generality to make it worthwhile to save. For example, consider a plan that takes a robot from room R1 through door D1 into room R2 and then has the robot bring box B1 from room R2 back into room R1 through door D1. That plan would be generalized so that it would take the robot from any room into any adjacent room through any connecting door and then take any box in the room into any adjacent room through any connecting door. The generalization removes all bindings to the particular rooms, doors, and box, and allows the room into which the box is taken to be different from the one in which the robot was initially located.

Hence, we can expand our characterization of the activities of a planner to include the instantiation and modification of plans that have been previously produced, generalized, and stored. This type of activity corresponds to Minsky's frame matching processes, where values are being assigned to frame terminals to produce an instance of a frame that matches the situation of interest.

IV. Subplanners vs Action Frames

The question arises as to whether thematic or action frames in the spirit of the STRIPS generalized plans are a useful way of storing knowledge about plans and planning strategies. It is essential that there be some means of communicating to our planning systems knowledge about scenarios and strategies for accomplishing tasks in the domain of interest. For example, in an equipment maintenance domain, the planning strategy for assembling any given device might be to create a plan to install each component of the device, and the scenario for connecting two components to each other might be to first position the components with respect to each other and then fasten them together.

In most planning systems, however, such planning strategies and domain specific scenarios are not stored in frame-like data structures, but are represented procedurally as "subplanners" (e.g. consequent theorems, if-needed methods, goal team members) that are applicable to highly specific situations. In many cases, such procedural representations seem to be necessary, primarily because of the complexity of the strategies involved. For example, in Fahlman's BUILD program [Fahlman, 1974], the planner has the task of building complicated towers out of toy blocks of several different shapes. The basic planning strategy is to build the tower from the bottom up by adding a block to the tower only after all the blocks that are to support it are already in place. This strategy is embedded in a procedure that computes the support relationships for the entire tower and then uses those relationships to determine a block placement ordering. Such a procedural representation suffers from the standard difficulty that the strategy is "hidden" inside the procedure so that the system cannot answer questions about it, modify it, investigate its properties, etc. However, it is difficult to imagine how such a strategy could be represented as an action frame or thematic frame since the number and ordering of the steps depends on the particular tower being constructed. Similar arguments would hold for robot route finding situations and the equipment assembly example given above.

There do seem to be interesting planning domains where it is desirable to input to the system thematic frames containing scenarios with all of their steps explicitly included. In such cases, the system is being told that the identity and order of the steps in the plan are not in

question. The planner's task is one of determining values for the scenario's slots (i.e. determining an instantiation) that both satisfy the scenario's constraints and match the situation and task goal that the planner is considering.

For example, we have found such explicit scenarios useful in a management support system currently being developed at SRI. In that work, we wish to allow a manager to specify to the system various operational procedures that he uses in his organization. The system can then act as an administrative assistant by planning and monitoring the execution of these procedures at the manager's request. The planning activity in this system is primarily one of scheduling the individual steps of an operation, and this scheduling is concerned with assuring the availability of resourses and personnel at specific times.

In particular, we have been considering management problems on board a Navy aircraft carrier. One operation of interest in that domain is the flying of training missions. Such a mission involves steps such as preflight and postflight maintenance of the aircraft, fueling the aircraft, briefing and debriefing the pilot, launching and recovering the aircraft, etc. These steps and the order in which they must occur do not vary from mission to mission, hence they can be included in a thematic frame that is part of the definition of a mission.

The mission scenario or action frame has many terminals with unassigned values, and the planner's basic task is to find an acceptable set of values for them. Most of these slots specify the start or the end time of some step in the plan; others specify the identity of the pilot and aircraft, the amount of fuel to be carried, bearings and locations to indicate the flight route, etc.

We include at each terminal of the mission's action frame a set of constraints on the value to be assigned there. The most common constraints are those that are derived from the temporal partial ordering of the steps (e.g. fueling of the aircraft must occur after the preflight maintenance and before the launch). Default values are included at the terminals, usually expressed as functions of other slot values in the frame. For example, the default value for the start of preflight maintenance is expressed as the time of launch minus constant. These default values allow the planner to make feasibility estimates and to make scheduling decisions before all of the constraints from the other schedulers have been determined.

Also included at each terminal is a specification of what part of the system can be called upon to determine a value for the terminal. These are typically calls on planners and schedulers that are responsible for other operational areas on the carrier such as maintenance, pilot assignment, aircraft assignment, the flight deck, etc. Hence, the mission planner engages in a negotiation dialogue with these other planners in an attempt to reach agreement on an acceptable schedule. In this domain, Minsky's idea of action frames as a representation of the system's understanding of various procedures seems a natural and useful one.

We conclude, then, that action frames are a useful representation as opposed to completely procedural subplanners in situations where the identity and ordering of the steps in a scenario do not vary from situation to situation. When that is not the case, a more traditional planning activity is required to construct an appropriate scenario; the strategy information needed to perform such scenario construction is often complex and can apparently be most naturally represented procedurally rather than in a frame-like structure.

V. Subplanners as Action Frames

Even though subplanners are procedural in nature, they can usefully be embedded in frame-like structures as if they were actions to be included as single steps in a plan. That is, we can talk about the preconditions, effects, task relevance, etc. of subplanners such as the tower builder, the route finder, and the device assembler as if they were single actions. These "meta-action" frames can provide the system with descriptive information to allow such activities as planning, question answering, and discourse about the "scenario" even though the actual steps in the scenario for any particular situation have not been determined. The subplanner acts as an implicit scenario and it can be called whenever the scenario needs to be made explicit.

The availability of such meta-action frames allows the system to do "hierarchical planning", in the sense that a meta-action can be included in a plan without calling its subplanner to determine the plan steps that it represents. This approach to planning has been carefully explored by Sacerdoti in two systems [Sacerdoti, 1974,1975], and found to provide significant advantages both during the generation and the execution of plans. In these systems, plans tend to grow in a top-down "breadth-first" manner in that typically a complete plan will be constructed using high level meta-actions before the detailed steps of any of the meta-actions are determined. Since a meta-action's subplanner may itself produce a plan containing meta-actions, a multiple level plan hierarchy can be formed, and the planner can make independent decisions for each meta-action as to whether its subplanner should be called.

The advantages of hierarchical planning derive basically from the fact that by using meta-actions the system can ignore detailed actions until it is confident that they will become part of the final plan. Indeed, in many situations the detailed steps may not be needed at all. For example, the system

may be giving instructions to a human who is skilled in the task domain and therefore does not need detailed instructions; or the system may be only answering a specific question about a task such as "Can the robot fetch the box?" or "How long will it take to replace the pump?".

REFERENCES

[1] Fahlman, S. E., "A Planning System for Robot Construction Tasks", Artificial Intelligence Journal, Vol. 5 (1974), pp 1-49.

[2] Fikes, R. E., Hart, P. E., and Nilsson, N. J., "Learning and Executing Generalized Robot Plans", Artificial Intelligence Journal, Vol. 3 (1972), pp 251-288.

[3] Sacerdoti, E. D., "Planning in a Hierarchy of Abstraction Spaces", Artificial Intelligence Journal, Vol. 5 (1974), pp 115-135.

[4] Sacerdoti, E. D., "A Structure for Plans and Behavior", Ph.D. Thesis, forthcoming.