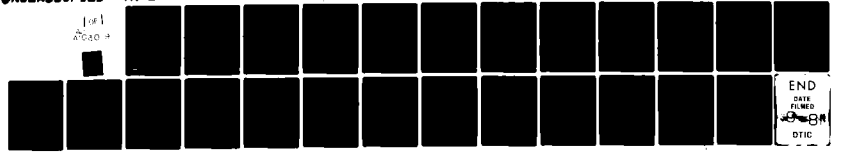
AD-A103 019 SRI INTERNATIONAL MENLO PARK CA ARTIFICIAL INTELLIGE--ETC F/G 9/2
AN INTERACTIVE PLANNING SYSTEM.(U)
JUL 81 D E WILKINS, A E ROBINSON F49620-79-C-0188
UNCLASSIFIED TN-245 NL



LEVEL II

12

AN INTERACTIVE PLANNING SYSTEM

Technical Note 245

July 1, 1981

By: David E. Wilkins
Computer Scientist

Ann E. Robinson
Senior Computer Scientist
Artificial Intelligence Center

The research reported here is supported by Air Force Office of Scientific Research Contract F49620-79-C-0188, (8871) and by Office of Naval Research Contrast N00014-80-C-0300 (1349).

SRI International
333 Ravenswood Ave. • Menlo Park, California 94025
(415) 326-6200 • Cable: SRI INTL MPK • TWX: 910-373-1246

81 7 20 041

ABSTRACT

A principal goal of our planning and plan execution research is to develop a computer system that interacts with a person planning some activity. The system, designed to be independent of the problem area in which the planning takes place, will allow the person to (1) represent the problem area and the actions that may be performed in it; (2) explore alternative plans for performing the activity; (3) monitor the execution of a plan so produced; and (4) modify the plan as needed during its execution. The system currently being tested allows a person to produce a plan interactively, suggesting alternative actions, showing the effects of actions on the situation, checking for problems in the plan, and (occasionally) suggesting corrections for such problems. The plan is represented as a hierarchy of actions linked together in a network, generally called a "procedural network".

Current areas of investigation include the following: (1) development of representations for encoding information about a given problem area, stressing the representation of actions that may be performed in it; (2) development of computational methods for identifying difficulties in a plan, such as the overallocation of a resource or the possible effect of one action on the successful performance of subsequent actions; (3) development of strategies for deciding which actions and action sequences should be included in a plan; (4) development of effective communication with the user, including determining which and how much information should be communicated, and how best to present it.

1.Introduction

A principal goal of our planning and plan execution research is to develop a computer system that interacts with a person planning some activity and monitoring the execution of a plan thus produced. The system is designed to be independent of the particular problem area or the activity being planned.[1] Our research builds directly on previous planning research [2] [6] [7] [8] and on research in natural-language dialogs about tasks [3] [4]. The basic approach to planning is to work within the hierarchical-planning paradigm, representing plans in procedural networks, as has been done in NOAH [6] and other systems.

In developing a planning system our approach has been to view it as an evolutionary process in which the computer plays an increasingly larger role in decision-making activities. The first tasks to be automated are those that have been the more traditional duties of computers: storing and giving back information. For a planning system such as described here, these include encoding information about what actions can be performed, when they are allowed, what their effects are, and keeping track of any actions planned and their effects. The next tasks to be automated include adding procedures to identify the decisions to be made, making the simpler ones, and asking the user to make the more difficult ones. As computational techniques for more sophisticated decision-making are developed and refined, more complex decision-making can be automated in the planning system which can then make decisions such as which steps to perform, what order to perform them in and how to accommodate to unanticipated changes or failures. The system described here is in the middle stage of development.

This evolutionary approach has several advantages. From the planning point of view it allows us to address larger, "real-world" problems, which may initially be beyond the capabilities of fully automatic planning techniques, but which could provide interesting

[1]The research reported here is supported by Air Force Office of Scientific Research Contract F49620-79-C-0188 and by Office of Naval Research Contract N00014-80-C-0300.

research problems. Development of an interactive planner also encourages us to deal with the issue of representing the planning problem in terms that can be easily communicated to a user. Clearly, ease of communication is crucial for an interactive system in which the person and computer must cooperate closely. It is also important for a system in which planning is automatic, because the machine must still be able to communicate about the planning that it has performed, and this requires that the machine have a representation of the problem in terms that are familiar to a person. An interactive system such as described here also provides an opportunity to further explore issues in human-machine interaction, such as deciding what information to communicate, how best to present it, and what medium to use (text, graphics, speech, etc.). In this note, we emphasize planning issues, particularly those that are important from the perspective of building an interactive system.

We have designed and implemented a system, SIPE, (System for Interactive Planning and Execution monitoring), that supports interactive planning. Unlike its predecessors, SIPE is designed to allow interaction with users throughout the planning and plan execution processes. The user is able to watch and, when desired, guide and/or control the planning process. Our long-range goal is to fully automate the processes of planning and execution-monitoring.

Development of the basic planning system has led to several extensions of previous systems. These include the development of a perspicuous formalism for encoding descriptions of actions, the use of constraints to partially describe objects, the creation of mechanisms that permit concurrent exploration of alternative plans, the incorporation of heuristics for reasoning about resources, and mechanisms that make it possible to perform simple deductions. Section 2 describes each capability in more detail, while Sections 3, 4, and 5 present examples from the SIPE program to illustrate how problems are solved.

The execution-monitoring part of SIPE has not been fully implemented, although work on it is in progress. During execution of a plan, some person or computer system monitoring the execution can specify what actions have been performed and what changes have occurred in the domain being modeled. Based on this, the plan can be updated interactively to cope with unanticipated occurrences. Planning and plan execution can be intermixed by producing a plan for part of an activity and then executing some or all of that plan before elaborating on the remaining portion.

2.Extensions of Previous Research

We are extending planning research toward several major objectives; among them are the following:

•The development of more flexible and uniform representations in which partial descriptions of objects and their properties can be specified and the objects treated as resources.

•The ability to explore several alternatives in parallel.

•The introduction of deductive operators for making deductions about the system's representation of the state of the problem domain.

•The ability to interact with the user, communicating both in terms of text and graphical representations of the plans.

2.1 Representation of Actions and Objects

We have developed a formalism for representing the actions that can take place both in the domain and during the planning process. (See [5] for more details.) Action descriptions (often referred to as operators), procedural networks, and information about objects in the domain and their interrelationships are represented in the same formalism – a hierarchy

of nodes with attributes. This uniform representation makes it possible to encode partial descriptions of unspecified objects and of objects in the domain model. Thus, operators that refer to abstract (unbound) objects can be represented in the same formalism as procedural network nodes that refer to specific objects in the domain model.

2.1.1 Formalism for Describing Actions

Operators representing actions contain information about the objects that participate in the action (represented as resources and arguments of the action), what the action is attempting to achieve (its goal), the action's effects when it is performed, and the conditions necessary before the action can be performed (its preconditions). The action's effects, preconditions, and goal are all encoded as first-order predicates on variables and objects in the domain. Negated predicates that occur in the effects of a plan essentially remove from the model a fact that was true before but is no longer true. The current system makes the closed-world assumption: any negated predicate is true unless the unnegated form of the predicate is explicitly given in the model or in the effects of an action that has been performed. This is not critical; the system could be changed to assume that a predicate's truth-value is unknown unless an explicit mention of the predicate is found in either negated or unnegated form.

Operators also contain a plot that specifies how the action is to be performed. When used by the planning system, the plot can be viewed as instructions for expanding a node in the procedural network to a greater level of detail. The plot of an operator can be described either in terms of *goals* to be achieved (i.e., a predicate to make true), or in terms of *processes* to be invoked (i.e., an action to perform). (Most previous systems have represented a process as a goal with only a single choice of action.) Encoding a step as a process implies that only that action can be taken at that point, while encoding a step as a

goal implies that any action can be taken that will achieve the goal. Another less explicit difference between encoding a step as a goal or as a process is whether the emphasis is on the situation to be achieved or the actual action being performed.

During planning, an operator is applied to an already existing GOAL or PROCESS node in the procedural network to produce another procedural network at the next level of detail. In Figure 1, the SECURE PUMPBOLTS operator is applied to the GOAL node shown to produce the three-node expansion. Figure 1 shows only part of the information at each node.) Operators may specify preconditions that must obtain in the world state before the operator can be applied. (The operator in Figure 1 has no precondition.) Operators contain lists of resources and arguments to be matched with the resources and arguments of the node being expanded. In Figure 1, PUMP1 in the operator is matched with the PUMP in the goal node. The plot of the operator is not shown in the figure, but it contains a template for generating the three PROCESS nodes and their effects.

We have already encoded domain operators describing actions that can be performed in several problem areas; eventually planning operators will also be encoded. Domain operators provide the planning system with information required to produce a plan for some activity. Planning operators provide the planning system with information so it can reason about its own planning process (metaplanning). They also furnish a major portion of the interface between the planning system and the user, who will be able to direct the planning process by invoking various planning operators.

Uniformity of representation for domain information, specific plans of action, and all operators is expected to facilitate both the user's ability to interact with and control the planning system, and the system's ability to incorporate (learn) new operators from plans it has already produced. Examples of domain operators are given and discussed in Section 3; Section 4 discusses the advantages of the representation.

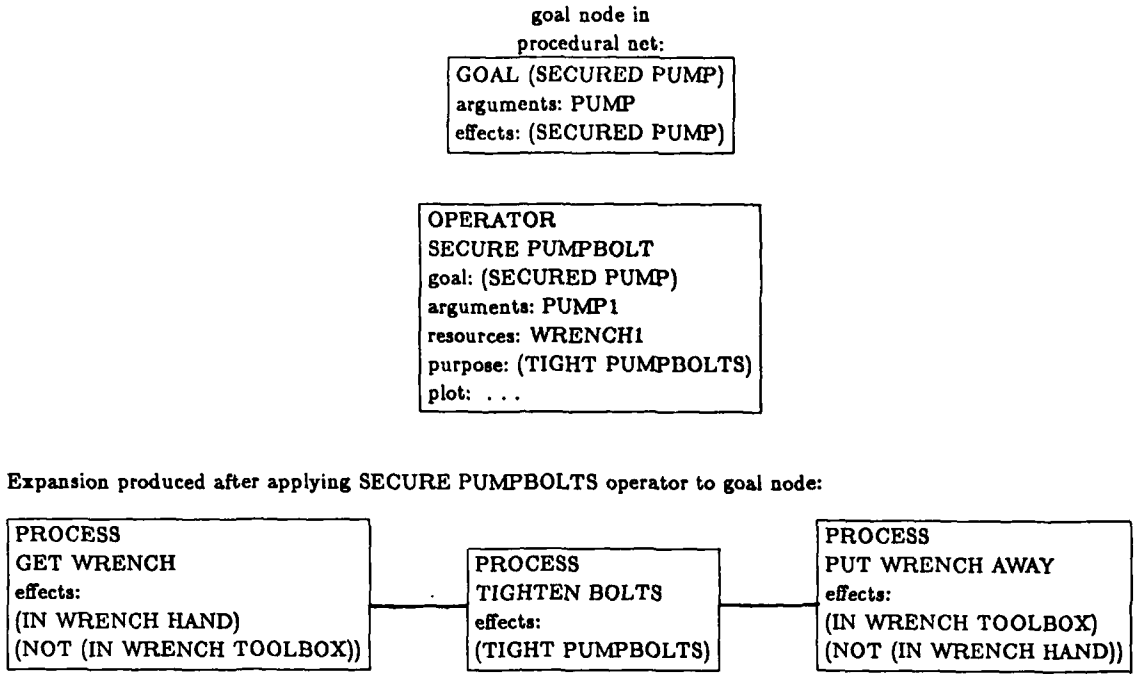


Figure 1
Using an Operator to Produce an Expansion

2.1.2 Partially Described Objects

Partial descriptions of unspecified objects can be viewed as setting constraints on the possible values of a variable representing an object. Partial descriptions include properties the object must have as well as relationships that must exist between that object and other objects (i.e., predicates that must be satisfied in a certain world state). SIPE provides a general language for expressing these constraints on variable bindings so they can be encoded as part of the operator. During planning, the system also generates constraints based on information in the operators, propagates them to variables in related parts of the network, and finds variable bindings that satisfy all constraints. Section 3 contains examples of constraints in operators; these are then discussed in Section 4.

Stefik [7] has used constraints to partially describe objects during planning in the domain of molecular genetics. Our system extends Stefik's approach in two ways. (1)

Constraints on variables can be evaluated before the variables are fully instantiated. For example, a set can be created that can be constrained to be only bolts, then to be longer than one inch and shorter than two inches, and then to have hex heads. This set can be used in planning before its members are identified in the domain. (2) Partial descriptions can vary with the context, thus permitting simultaneous consideration of alternative plans involving the same unidentified objects.

2.1.3 Objects Identified as Resources

Objects associated with an action can be characterized as *resources* that are to be used during a particular action and then released, e.g., a saw used during a cutting action. Since this is a common phenomenon and since it is often difficult or awkward to keep track of resources in current planning systems, we have included in the formalism a means of specifying the objects that serve as resources for an action. Declaration of a resource associated with an action connotes that one precondition of the action is that the resource be available. Mechanisms in the planning system, as they allocate and deallocate resources, automatically ensure that these preconditions will be satisfied. This form of reasoning about resources is a useful heuristic for quickly identifying and correcting problems in plans. Section 4 contains an example of the use of this heuristic and describes it in more detail.

2.1.4 Purposes

In the procedural networks that represent plans, PROCESS and GOAL nodes represent an action to be performed or a goal to be achieved. Associated with these nodes are predicates stating the expected effects of performing the action or achieving the goal. When a node is planned to a greater level of detail by applying an operator, the expansion

may consist of many nodes. Which node in the expansion is the main purpose of that sequence of steps must be determined, to ascertain when the effects of the higher level node become true in the more detailed expansion. (This may be the last in a series of nodes that, acting together, achieve the tacit "purpose" of the expansion.)

For example, let us consider the GOAL node in Figure 1 in which the effect is (SECURED PUMP). The SECURE PUMPBLOTS operator expands this node into three nodes at the next level of detail, as shown in the figure. The first node might be called a preparatory action, and the last a cleanup action. Somewhere must be encoded the fact that (SECURED PUMP) becomes true after the second node in the expansion. This is needed, for example, in answering user questions or determining the correct state at the "put wrench away" node (perhaps some operators for putting the wrench away may be effected by or depend upon the state of the pump).

In the example, the "tighten bolts" node accomplishes the repair of the pump, while the "put wrench away" node is a cleanup action that leaves objects in their normative states. Operators in SIPE have a PURPOSE attribute that specifies the main purpose of their expansion. Thus, the PURPOSE of the SECURE PUMPBOLTS operator is (TIGHT PUMPBOLTS). This is included in the effects of the "tighten bolts" node produced in the plan, so the (SECURED PUMP) effect is copied down to this node. In NOAH the assumption was that the last node of an expansion achieved the main purpose and so the effects were copied down to that node. In the example above, NOAH would incorrectly attach (SECURED PUMP) to the "put wrench away" node. SIPE allows flexibility in specifying purposes, so that situations like the one described above can be represented accurately.

2.2 Exploring Alternatives in Parallel

A context mechanism has been developed to allow constraints on a variable's value to be established relative to specific plan steps. Constraints on a variable's value, as well as the binding of a variable to a particular instance (possibly determined during the solution of a general constraint-satisfaction problem), can be retrieved only relative to a particular context. This permits the user to shift focus back and forth easily between alternatives.

SIPE accomplishes this in a hierarchical procedural-network paradigm by introducing CHOICE nodes in the procedural networks at each place an alternative can occur. Attributes of nodes and their values are stored relative to choice points. Thus, the constraints on a variable at a given point in a plan can be accessed by specifying the path of choices in the plan that is to be followed to reach that point. Different constraints can be retrieved by specifying a different plan (path of choices). This shifting of focus between alternatives cannot be done in systems using a backtracking algorithm, in which descriptions built up during expansion of one alternative are removed during the backtracking process before another alternative is investigated. Most other planning systems either do not allow alternatives (e.g., NOAH), or use a backtracking algorithm (e.g., MOLGEN [7], NONLIN [8]). An exception is the system described by Hayes-Roth et al. [2], in which a blackboard model is used to allow shifting focus between alternatives.

2.3 Deductive Operators

In addition to operators describing actions, SIPE allows specification of deductive operators that deduce facts from the current world state. As more complex domains are represented, it becomes increasingly important to deduce effects of actions from axioms about the world rather than explicitly representing these effects in operators. Deductive operators in SIPE may include both existential and universal quantifiers, and so provide a

rich formalism for deducing (possibly conditional) effects of an action.

Deductive operators are written in the same formalism as other operators in SIPE (see Section 3), permitting the system to control deduction with the same mechanisms it uses to control the application of operators. Deductive operators have no instructions for expanding a node to a greater level of detail. Instead, if the precondition of a deductive operator holds, its effects can be added to the world model (in the same context in which the precondition matched) without changing the existing plan. This may "achieve" some goal in the plan (by deducing that it has already been achieved), and avoid the need to plan actions to achieve it.

The ability to perform deductions is important in many domains. Consider the "blocks world" (described in [6]) that has been used as a test domain for many planners. In this world only one block may be on top of another, so that whenever a block is moved, the operator for the move action can be written to explicitly state the effect that the block underneath will be clear. In the more general case in which one block might have many blocks on top of it, there may or may not be another block on the underneath block so the effects of the action must be conditional on this. Since systems like NOAH and NONLIN must mention effects explicitly (universally or existentially quantified variables are not allowed in the description of effects), they cannot represent this more general case with a single move operator.

In SIPE, deductive operators may be used to deduce all the clearing and unclearing effects that occur, so the operators themselves do not need to represent these effects. As the domain grows to include many operators, this aspect becomes very convenient. Furthermore, in SIPE, existentially quantified variables can occur in any predicate; in particular, they can occur in both preconditions and effects of operators. Thus, by using existential variables in the precondition of a deductive operator, SIPE can handle the

general case of recognizing a clear block with one deductive operator.

2.4 Interaction With the User

A major goal has been to design and build a planning system that supports interactive planning. Among other features, the user must be able to invoke planning operations easily, but sHe must not be required to make tedious choices that could be performed automatically. In SIPE the user can either direct fairly low-level and specific planning operations or invoke higher-level operations that combine these lower-level ones. Some choices the system presently makes may not be optimal, so the user may want to make several of them. As the system's heuristics for selection improve, the user will presumably leave more choices to the system.

Planning operations the user can now invoke include: (1) determining which actions can be used to plan a given step in more detail; (2) testing to verify whether a given action can be used for further planning of a specific step; (3) testing the availability of resources for a given plan or subplan; (4) planning a step in greater detail (either with a user-supplied action or one chosen by the system); (5) indicating which object to use for an action; (6) instructing the system to select objects for an action; (7) instructing the system to find problems or conflicts in a plan or subplan; and (8) rearranging the order of certain plan steps when a conflict would arise from their parallel execution. In addition to supporting breadth-first and depth-first planning, these operations allow *islands* to be constructed in a plan (to arbitrary levels of detail), and then linked together later. The example in Section 5 shows a planning sequence in which some of the capabilities mentioned are invoked.

3.Examples of Operators

Several domains have been encoded in SIPE and problems solved in each of them, one

```
OPERATOR: MAKECLEAR
ARGUMENTS: OBJECT1,BLOCK1;
PURPOSE: (CLEARTOP OBJECT1);
PRECONDITION: (ON BLOCK1 OBJECT1);
PLOT:
    PROCESS
    ACTION: PUTON;
    ARGUMENTS: BLOCK1, OBJECT2 IS NOT OBJECT1;
    EFFECTS: (ON BLOCK1 OBJECT2);
END PLOT
END OPERATOR

OPERATOR: PUTON
ARGUMENTS: BLOCK1, OBJECT1 IS NOT BLOCK1;
PURPOSE: (ON BLOCK1 OBJECT1);
PLOT:
    PARALLEL
        BRANCH 1:
            GOAL
            GOALS: (CLEARTOP OBJECT1);
            ARGUMENTS: OBJECT1;
        BRANCH 2:
            GOAL
            GOALS: (CLEARTOP BLOCK1);
            ARGUMENTS: BLOCK1;
    END PARALLEL
    PROCESS
    ACTION: PUTON.PRIMITIVE;
    ARGUMENTS: OBJECT1;
    RESOURCES: BLOCK1;
    EFFECTS: (ON BLOCK1 OBJECT1);
END PLOT
END OPERATOR
```

Figure 2
SIPE's Blocks World Operators

is the blocks world described in [6]. Many domain-independent planning systems (e.g., NONLIN and NOAH) have presented their solutions to the same problems in this domain. To facilitate comparison with other systems, operators for the blocks world in SIPE are shown in Figure 2.

On the basis of their names, the arguments and resources in an operator are automatically constrained by the system to be objects of a certain type, i.e., BLOCK1 must be a

block while OBJECT1 may be any object. The MAKECLEAR operator has the precondition (ON BLOCK1 OBJECT1), which means that it will not be applied unless there is a block on OBJECT1. Matching this precondition will constrain BLOCK1 to be one of the blocks on OBJECT1. In this encoding of the blocks world there will be only one block on OBJECT1, so this matching will instantiate BLOCK1. The more general case in which many blocks can be on one block is also handled. This case would result in a constraint that would limit possible instantiations for BLOCK1 without instantiating BLOCK1. When MAKECLEAR is applied to expand a plan node, its plot generates only one node, a PROCESS node. This new node represents the plan for putting BLOCK1 on OBJECT2 where OBJECT2 is constrained to be something other than OBJECT1. OBJECT2 is not instantiated, merely constrained, and can later be chosen to be a table, another block, or any other object in the domain (unless it is the instantiation of OBJECT1).

The plot of the PUTON operator represents the plan of first clearing BLOCK1 and OBJECT1 in parallel, then of putting BLOCK1 on OBJECT1 with a primitive action. The effects of the PUTON.PRIMITIVE process node in the plot include a predicate that matches the predicate stating the purpose of the PUTON operator. The PUTON.PRIMITIVE node in the plan accomplishes the purpose of the PUTON operator and therefore inherits higher-level effects attached to the node being expanded. SIPE assumes that the two CLEARTOP goals produced in the plan must be kept true until the PUTON.PRIMITIVE process is executed, since it accomplishes the purpose of the PUTON operator. (SIPE allows for overriding this in the specification of operators.) If one of the CLEARTOP goals is achieved and then later becomes untrue before the PUTON.PRIMITIVE process is performed, the system will recognize the problem and attempt to correct it. The system will not complain if the CLEARTOP goal becomes false after the PUTON.PRIMITIVE operation.

The effect encodes as part of the PUTON.PRIMITIVE process is only one predicate because the system deduces all the other effects with deductive operators. For example, suppose that BLOCK1 was on BLOCK2 when the PUTON operator was applied to put BLOCK1 on OBJECT1. The blocks world deductive operators would be triggered when (ON BLOCK1 OBJECT1) is posted as an effect and they would (in a typical case) deduce (CLEARTOP BLOCK2), (NOT (ON BLOCK1 BLOCK2)), and (NOT (CLEARTOP OBJECT1)). These effects therefore do not have to be listed as effects in the PUTON operator.

4.The Resource Heuristic

Blocks world operators (those of SIPE are in Figure 2) are used most frequently in the literature to get block A on block B on block C. In a procedural network, this goal is given as two parallel GOAL nodes – one for (ON A B) and one for (ON B C). Let us suppose that initially A, B, and C are all on the table. (Other popular configurations include having A on either B or C initially, but in both cases A must be moved to the table first so that B can be moved onto C. Thus, having all blocks on the table is a central subproblem in all these configurations.)

In NOAH, NONLIN, and SIPE, both original GOAL nodes are expanded with the PUTON operator or its equivalent. The central problem is to notice that B must be put on C before A is put on B (otherwise B will not be clear when it is to be moved onto C). NOAH and NONLIN both build up a table of multiple effects (TOME) that tabulates every predicate instance asserted and denied in the parallel expansions of the two GOAL nodes. Using this table, the programs detect that (CLEARTOP B) is asserted in the expansion of (ON B C), but is denied in the (ON A B) expansion. Both programs then solve this

problem by doing (ON B C) first.

SIPE uses its resource heuristic to detect this problem and propose the solution without having to generate a TOME. (SIPE will eventually generate TOMEs to detect interactions that do not fit into the resource-reasoning paradigm.) When some object is listed in a plan node as a resource, the resource heuristic then prevents that particular object from being mentioned as either a resource or an argument in any plan node that is in parallel. In the example above, BLOCK1 is is listed as a resource in PUTON.PRIMITIVE since it is being physically moved. Therefore, nothing in a parallel branch should try to move it, or even be dependent on its current location. Thus, as soon as the expansion of (ON B C) with the PUTON operator is accomplished, even before the expansion of (ON A B), SIPE recognizes that the plan is not valid; because B (matched to BLOCK1 in PUTON) is a resource in the PUTON.PRIMITIVE process and an argument in (ON A B).

Not being able to refer to a resource in another branch is sometimes too stronga restriction. SIPE also permits specification of shared resources, which will eventually allow the same object to be a shared resource or an argument in a parallel branch if certain conditions for the sharing are met (the sharing conditions have not been implemented yet).

SIPE's heuristic for solving a resource-argument conflict (as distinct from a resource-resource conflict) is to put the branch using the object as a resource before the parallel branch using the object as an argument. (This can be prevented by the user interacting with the system.) In this way SIPE decides that (ON B C) must come before (ON A B) without generating a TOME and without expanding both nodes. The assumption is that an object used as a resource will have its state or location changed by such a use, so the associated action must be done first to ensure that it will be "in place" when later actions occur that use it as an argument.

Resources are a useful heuristic, but they must be used properly by the person writing

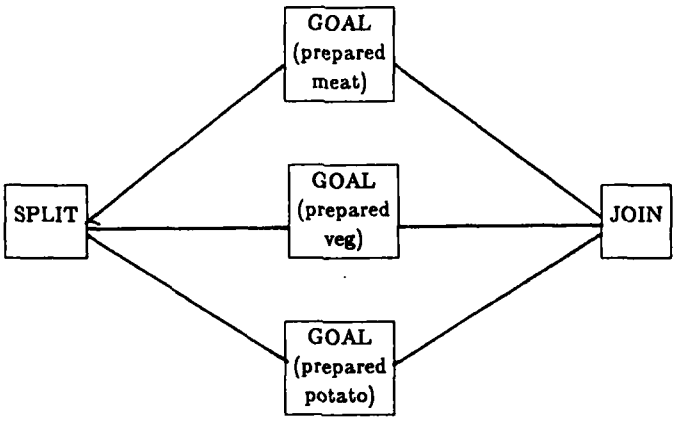


Figure 3
First Level of Cooking Plan

operators. They enable the system to identify incorrect plans more quickly. The PUTON operator could have listed BLOCK1 as a resource of the operator as a whole. In this case, SIPE would detect that a plan with a conflict would be produced as soon as the decision was made to expand (ON B C) with PUTON (before the expansion was actually done). Such early identification of conflicts can help prune undesirable plans from the search space even before they are generated. The authors have found resources generally helpful in encoding several domains in SIPE. We also find reasoning about resources to be natural (especially in scheduling tasks), and easy to communicate to the user.

5.An Example of User Interaction

An early version of the planner that supports graphical interaction with a user was implemented on the Dolphin, a research computer developed at Xerox-PARC [1]. The Dolphin has a high-resolution black-and-white bitmap display terminal that is used for both graphics and text. More recently, a color-graphics terminal connected to a DEC-2060 has been used for interacting with the user.

Both the procedural networks (plans) produced and, for certain domains, the domain

configuration (principally the location of objects) are presented graphically. As new plans or partial plans are developed, they are also displayed on the terminal. The user can also choose to view different portions of the plan, at different levels of detail, and can look at any alternative plans. Steps in the plan (nodes in the network) can either be referred to by name or by pointing to them.

The user can also see a graphic representation of either the actual domain configuration or the one that will exist after some sequence of planned steps. The configuration is generally shown together with a plan or partial plan, and the configuration corresponds to the expected state following execution of that plan. The use, however, can vary this.

The planning choices available to the user appear in a menu. The user selects one of these actions by pointing to the relevant box. Current choices include the following:

- Planning the details of an action.
- Testing the resource requirements of a plan or plan-segment.
- Testing resource requirements and automatically fixing any problems, if possible.
- Identifying an object to be used in an action.
- Linearizing a parallel plan-segment.
- Viewing other parts of the plan, other plans, etc.
- Changing to another plan.
- Labeling a plan or plan-segment for future reference.

In addition, some plan-execution steps are available including:

- Record that a step has been executed.
- Record that several steps have been executed in parallel.

•Record that a previously unknown situation now holds.

The first two update the system's model of the situation; the third will also cause the system to determine if the reported situation effects any planned actions and if so, will inform the user so that the plan can be modified accordingly.

Whenever a node or domain object is to be specified, the user can either point to it or type its name. Currently, the user may make most of the planning choices, although we have developed mechanisms that automate more decisions, such as the allocation of resources and the selection of objects to be used in an action.

Two problem domains in addition to the blocks world have been encoded and are being used to test the planner. They are launching planes from the deck of an aircraft carrier (including the prelaunch movement of planes) and preparing a meal. We will use the meal-preparation domain to illustrate the system's interaction with a user.

Figure 3 shows a preliminary plan for preparing three dishes–a vegetable, a meat, and a potato dish. These are represented by the three GOAL nodes in the plan. At this level of abstraction, they are assumed to be performed in parallel, as indicated.

Figure 4 shows the plan at a later stage of development. Each goal node has been expanded to one more level of detail. Here the choice was made to prepare fried meat, steamed broccoli, and fried potatoes–so far still in parallel. The plan has been elaborated to the point that the meat and potatoes are in pans. A display of the expected domain configuration at this point would show the meat and potatoes in the pans.

Since burners and pots are critical resources, the user now decides to select the objects to be used and to test for possible resource conflicts. An automatic resource allocation mechanism could be invoked, or, as in the example, the user can make the choice by typing the names of objects or by pointing to them and then sHe can execute a resources-testing action to test for conflicts. Each cooking step in this plan requires a burner; since there

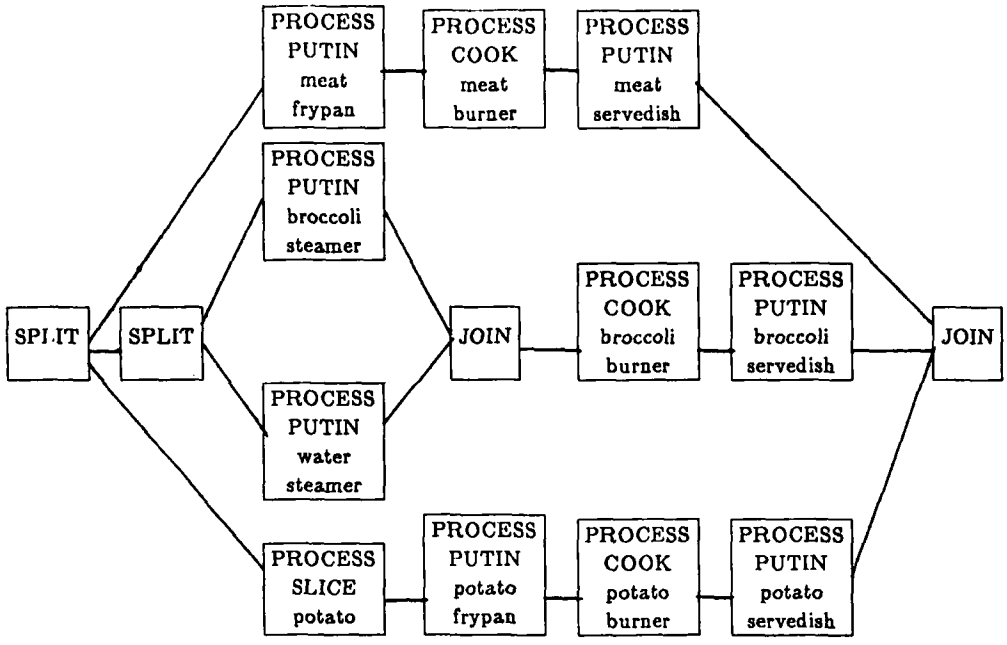


Figure 4
Expanded Cooking Plan

are only two, the system, using the resource heuristic described in Section 4, detects a conflict. The conflict has many solutions; one is linearizing the parallel branches, another is choosing an alternative expansion for one of the branches. The choice here (made by the user) was to linearize – frying the potato before frying the meat. The resulting plan is shown in Figure 5.

If an alternative action were chosen to resolve the conflict, a CHOICE node would be created in the plan, and the decisions made in the context of this new choice would be stored relative to the choice. Each alternative plan so created is named, either by the user if sHe desires, or by the system – and at any time the user can elect to view and/or extend any of the plans by identifying the desired plan. Other parts of a plan can be labeled and/or displayed.

Planning will continue in this manner until each node in the network has been expanded

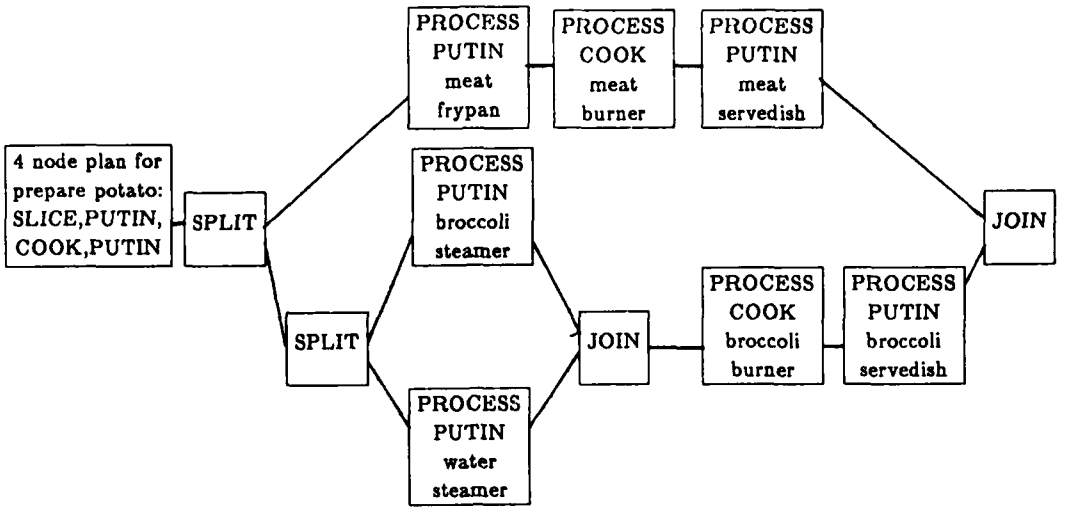


Figure 5
Reordered Cooking Plan

to primitive actions and no resource or other conflicts exist. The user could have elected to start execution of the first part of the plan before fully elaborating it, using some of the execution-monitoring mechanisms currently being developed.

6.Summary of the Advantages of SIPE

A major difference between SIPE and previous planners is that SIPE is interactive. Its interactive capabilities help the user guide and direct the planning process, allowing alternative plans to be explored concurrently by means of the context mechanism. Thus the user can shift focus as sHe pleases without being required to understand the program's search strategy or backtracking algorithm.

Enabling graceful interaction is one reason the operator description language (illustrated in Section 3) was designed to be straightforward and easy to understand. SIPE's deductive operators also contribute to this. They allow quantified information to be encoded and therefore can be used to make fairly sophisticated deductions, thus eliminating the

need to express effects in operators when they can be deduced. Perspicuous operators are of primary importance in facilitating interaction with nonexperts planning tasks using operators written by an expert. In addition, it is hoped that nonexperts will also be able to encode operators for their own domains without undue effort. (It is interesting to compare the readability of the operators in this paper with the SOUP code operators in NOAH for the same problem.)

SIPE has developed a resource heuristic (described in Section 4) that recognizes conflicts more effectively and earlier in the planning process than did previous planning systems that employed a table of multiple effects. It has been beneficial in user interaction to have reasoning about resources as a central part of the system, because resources seem to be a natural and intuitive way to think about objects in many domains.

Purposes in SIPE operators are used to coordinate higher level effects with lower-level plans. Being able to mention purposes explicitly provides the flexibility needed to represent many domains. In some domains (such as the meal-preparation one encoded in SIPE operators), operators have cleanup actions to perform after accomplishing the main purpose. SIPE has the flexibility to represent this, whereas in NOAH, higher-level effects would improperly be attached to the last cleanup action. SIPE also permits deductive operators, which provide even more flexibility for representing new domains.

One of the most important features of SIPE is the ability to constrain the possible values of variables with partial descriptions. In addition to the constraints described in PUTON and MAKECLEAR, an operator could require certain values for certain attributes of an object (e.g., that a block be colored red, or be bigger than 5 inches on a side). It is well known that this allows more efficient planning, since choices can be delayed until information has been accumulated.

Other advantages of constraints, however, are also critical. A key consideration is that

constraints allow expression of a much wider range of problems. For example, SIPE can be given the problem of placing one red block on top of one blue block without being told specifically which blocks to use (assuming that there are many red and blue blocks in the world). Because they do not permit constraints, most domain-independent planners (e.g., NOAH and NONLIN) cannot express such a problem. The ability to express such problems is vital to many real-world applications. For example, consider the problem of scheduling airline flights in San Francisco. For the SFO-JFK flight, the scheduler wants to plan for the availability of a wide-bodied jet (without caring which one it is) and a pilot qualified to fly that type of aircraft (again sHe doesn't care which pilot, as long as the selected pilot meets this constraint). A planning system requiring the user to state exactly which plane and pilot sHe wanted to make available before allowing the formulation of plans for their availablity would not be acceptable.

REFERENCES

1. Burton, Richard R., Kaplan, R. M., Masinter, L. M., Sheil, B. A., Bell, A., Bobrow, D. G., Deutsch, L. P., and Haugeland, W. S., "Papers on INTERLISP-D," *Report CIS-5*, XEROX Palo Alto Research Center, Palo Alto, California (September 1980).

2. Hayes-Roth, B., Hayes-Roth, F., Rosenschein, S., and Cammarata, S., "Modeling Planning as an Incremental, Opportunistic Process," *Proceedings IJCAI–79*, Tokyo, Japan (August 1979), pp. 375–383.

3. Hendrix, G., "Encoding Knowledge in Partitioned Networks," in *Associative Networks-The Representation and Use of Knowledge in Computers*, N.V. Findler (Ed.), Academic Press, New York, New York (1979).

4. Robinson, A.E., Appelt, D., Grosz, B., Hendrix, G., and Robinson, J., "Interpreting Natural-Language Utterances in Dialogs About Tasks," *Artificial Intelligence Center Technical Note 210,* SRI International, Menlo Park, California (March 1980).

5. Robinson, A.E., and Wilkins D.E., "Representing Knowledge in an Interactive Planner," *Proceedings of the First Annual Conference of the AAAI,* Stanford, California (August 1980), pp. 148–150.

6. Sacerdoti, E., *A Structure for Plans and Behavior,* Elsevier, North-Holland, New York, 1977.

7. Stefik, M., "Planning With Constraints," *Report STAN-CS-80-784,* Computer Science Department, Stanford University, Ph.D. Dissertation (1980).

8. Tate, A., "Generating Project Networks," *Proceedings IJCAI–77,* Cambridge, Mass. (August 1977), pp. 888–893.