□ A SYSTEM AND CONTROL THEORETIC PERSPECTIVE ON ARTIFICIAL INTELLIGENCE PLANNING SYSTEMS

K. M. PASSINO and P. J. ANTSAKLIS
Department of Electrical and Computer Engineering, University of Notre Dame, Notre Dame, Indiana 46556

In an artificial intelligence (AI) planning system the planner generates a sequence of actions to solve a problem. Similarly, the controller in a control system produces inputs to a dynamical system to solve a problem, namely the problem of changing a system's behavior into a desirable one. A mathematical theory of AI planning systems that operate in uncertain, dynamic, and time-critical environments is not nearly as well developed as the mathematical theory of systems and control. In this paper relationships and a detailed analogy between AI planning and control system architectures and concepts are developed and discussed. These results are fundamental to the development of a mathematical theory for the modeling, analysis, and design of AI planning systems for real-time environments.

INTRODUCTION

In an artificial intelligence (AI) planning system the planner generates a sequence of actions to solve a problem. It is a type of expert system since it emulates the way in which human experts represent and reason about abstract, uncertain information to solve a problem in a narrow field of expertise (Charniak and McDermott, 1985). The essential ideas in the theory of AI planning have been developed and reported in the literature. There is, however, a need to create a mathematical theory of AI planning systems that operate in dynamic, uncertain, and time-critical environments (real-time environments).

In a control system the controller produces inputs to a dynamical system to change its undesirable behavior to a desirable one. In contrast to AI planning, there exists a relatively well-developed mathematical systems and control theory for the study of properties of systems represented with, for instance, linear ordinary differential equations. The objective of this paper is to point out relationships and to develop and discuss an extensive analogy between AI planning and control system architectures and concepts. In the process, a foundation of fundamental concepts in AI planning systems based on their control theoretic counterparts is built. It is hoped that these discussions will help lead to the

This work was partially supported by McDonnell Aircraft Company's Artificial Intelligence Technology Group under a contract entitled ''Artificial Intelligence/Control Theory Relationships,'' reported in October 1987.

development of (1) quantitative systematic modeling, analysis, and design techniques for AI planning systems; (2) methods for analyzing planning system performance; and (3) empirical and/or analytical methods for AI planning system verification and validation.

Research from a wide variety of disciplines such as system and control theory, operations research, theoretical computer science, and artificial intelligence will aid in such studies. It is important to note that planners that operate in real-time environments *must* use feedback information to know where they are in their problem solving process. (This is discussed further later in this paper.) Feedback has been studied extensively in the field of system and control theory. Hence, in solving realistic problems it is logical that results from system and control theory will become particularly useful. An outline of the main results of this paper follows.

In the next section, distinctions are drawn between conventional (non-AI) and AI problem-solving systems, then between non-AI and AI planning systems. The relationship between expert, planning, and scheduling systems is discussed. Next, the elements of AI planning systems, their structure and functional components, are outlined. Issues and techniques from relevant literature on AI planning systems are highlighted. This gives an overview of planning ideas and sets the terminology for the paper.

After this overview, to begin the discussion of the relationships between AI planning and control theoretic ideas, it is shown how the AI planning domain is analogous to a physical plant in control theory. The plant model is analogous to the problem representation. The plant inputs, outputs, and disturbances are real-valued variables that are continuous or discrete in time, while the problem domain inputs, outputs, and disturbances are represented by symbols. The fact that disturbances will always occur in the problem domain is discussed at length.

A state of the problem domain is a ‘‘snapshot’’ of its behavior. It is shown how one can relate the mathematical models of the plant and problem domain. The ideas that are developed for planning theory are independent of both domain and representation. Controllability refers to the ability of the inputs of a problem domain to change its state. If a problem domain is uncontrollable, there does not exist a planner that can achieve arbitrary desired goals. We define two types of controllability, of which one is more restrictive than the other. Observability refers to the ability to determine the state of the problem domain from the inputs, outputs, and model of the problem domain. We also define two types of observability. If a problem domain is unobservable, there does not exist a situation assessor that can always determine its state. Minimality of a problem domain model refers to how well the system was modeled. It quantifies whether there are redundancies in the model.

A problem domain is said to be internally stable if, when the system begins in some particular set of states and is perturbed, it will always return to that set of states without control actions. It is said to be input-output stable if, for all "reasonable" inputs, the outputs are acceptable. A problem domain is stabilizable if there exists a planner that can make it stable. It is detectable if there exists a situation assessor that can determine if it is not stable. The rate of a problem domain specifies some measure of the speed of its response.

For the planner alone, the fundamental concepts borrowed from control theory have a special meaning. The state of the planner is the situation describing the planner's problem solving strategy at a particular instant. Planner observability refers to the ability to determine the planner state using the goal inputs, planner outputs, and model of the planner.

Open-loop planning systems are defined. They do not have a feedback connection; consequently, they will fail if there are disturbances in the problem domain. They cannot stabilize an unstable problem domain. Open-loop planning systems often require the use of more detailed models than are used in feedback planning. If controllability studies show the problem domain to be uncontrollable, there may not exist a planner capable of solving the problem. Although the rate of the system can be increased in both open- and closed-loop planning, open-loop planners are simpler and cheaper to implement than feedback planners.

In AI feedback planning systems the planner can sense the outputs of the problem domain and use them in its decision-making process. Feedback planners perform execution monitoring and replanning. Feedback planners can recover from plan failures that occurred because of disturbances in the problem domain. Controllability and observability studies can be used for actuator and sensor design guidelines. These guidelines show that there is a trade-off between expense of planning system implementation and planner complexity. Feedback planning systems are characterized as regulatory or goal following. Design issues such as stability, disturbance rejection, and rate are discussed.

For AI feedback planning systems that use situation assessment, observability studies can show the existence of a situation assessor for state determination and suggest its internal structure. An analogy between optimal situation assessment and Kalman filtering is shown, and the separation principle is discussed. We define adaptive planning systems that have the structure to implement metaplanning. They also allow for human interface to the problem solving process in the planner.

Certain concepts recently introduced in the field of intelligent control are helpful here. Planning systems can be viewed as having a three-level hierarchy: the execution level, the coordination level, and the management level. Models used at higher levels are more abstract. The time scale density and decision rate are higher at the execution level.

For the ideas just outlined to be useful for AI planning system analysis and design, there must be appropriate formalisms and methodologies for studying them. They must be able to tell if a particular planning system possesses the properties. For instance, we need a simple systematic method for determining whether a problem domain is controllable, observable, or stable. The concluding remarks discuss the importance of analyzing these properties of planning systems.

AI PLANNING SYSTEMS: CLASSIFICATION, FUNCTIONAL OPERATION, AND OVERVIEW

In this section the essential components and ideas of AI planning systems are outlined briefly and distinctions are drawn between AI planning systems and other similar problem-solving systems. This sets the terminology for this paper. A planning system reasons from the measured initial state of its problem domain and determines and then executes the sequence of actions that will achieve some final goal state in the problem domain. Before we discuss the essential elements of AI planning systems and describe their operation, the distinctive characteristics of AI planning systems that separate them from other planning systems, scheduling systems, expert systems, and control systems are outlined.

System Classification

In general, it is our view that we can classify problem-solving systems into two categories: conventional and AI. Several distinct characteristics distinguish these two classes of problem-solving systems. The conventional problem-solving system is numeric-algorithmic; it is somewhat inflexible; it is based on the well-developed theory of algorithms or differential/difference equations; and thus it can be studied using a variety of systematic modeling, analysis, and design techniques. Control systems are an example of conventional problem-solving systems.

An AI problem-solving system is a symbolic-decision maker. It is flexible with graceful performance degradation, and it is based on formalisms that are not well developed; actually there are very few methodical modeling, analysis, and design techniques for these systems. AI planning system are examples of AI problem-solving systems. When comparing the characteristics of AI and non-AI systems, one can make the following observations: The decision rate in conventional systems is typically higher than that of AI systems. The abstractness and generality of the models used in AI systems are high compared with the fine granularity of models used in conventional systems. Symbolic, rather than numeric, representations are used in AI systems. High-level decision making and learning capabilities similar to those of humans exists in AI systems to a much greater extent than in conventional systems. The result is that a higher degree of autonomy exists in AI systems than in conventional ones. A general discussion on problem solving is given in Shapiro (1987).

Although clear, distinct characteristics separate AI from non-AI planning systems, as planning systems evolve the distinction becomes less clear. Systems that were originally AI planners evolve to gain more character of non-AI planning systems. An example is a route planner. As problems like route planning become better understood, more conventional numeric-algorithmic solutions are developed. The AI approaches help to organize and synthesize approaches to problem solving, in addition to being problem-solving techniques themselves. AI techniques can be viewed as research vehicles for solving very complex problems. As the problem solution develops, purely algorithmic approaches, which have desirable implementation characteristics, substitute for AI techniques and play a greater role in the solution of the problem.

AI planning systems use models for the problem domain called problem representations. For instance, in the past, predicate or temporal logic has been used. A planner's reasoning methodology is modeled after the way a human expert planner would behave. Therefore, the planning systems use heuristics to reason under uncertainty. Conventional expert systems have many of the elements of planning; they use similar representations for knowledge and heuristic inference strategies. The planning systems that are studied here, however, are specifically designed to interface with the real world, whereas conventional expert systems typically exist in a tightly controlled computer environment. The planning system executes actions dynamically to cause changes in the problem domain state. The planner also monitors the problem domain for information that will be useful in deciding a course of action so that the goal state is reached. An explicit loop is traversed between planner-executed actions, the problem domain, the measured outputs, and the planner that uses the outputs to decide what control actions to execute to achieve the goal.

In an expert system there exists an analogous loop. The knowledge base is the problem domain and the inference strategy is the planner. For rule-based expert systems the premises of rules are matched to current working memory (outputs are measured and interpreted), and then a heuristic inference strategy decides which rule to fire, that is, what actions to take to change the state of working memory (the knowledge base) and so on. The expert system has an inherent goal of generating some diagnosis, configuring some computer system, etc. Some expert systems have more elements of planning than others. For instance, some consider what will happen several steps ahead, if certain actions are taken.

A further distinction must be made between AI planning and scheduling systems. It is the task of a planner to generate sequences of actions so that some goal is attained without expending too many resources. A scheduling system is concerned with *when* the action should be accomplished and uses the availability of resources to assign resources to times and actions (Kempf, 1987). In the next section we briefly describe the elements of AI planning systems.

Elements of AI Planning Systems

An *AI planning system* consists of the planner, the problem domain, their interconnections, and the exogenous inputs. The outputs of the planner are the inputs to the problem domain. They are the control actions taken on the domain. The outputs of the problem domain are inputs to the planner. They are measured by the planner and used to determine the progress in the problem-solving process. In addition, there are unmeasured exogenous inputs to the problem domain, which are called disturbances. They represent uncertainty in the problem domain. The measured exogenous input to the planner is the goal. It is the task of the planner to examine the problem domain outputs, compare them to the goal, and determine what actions to take so that the goal is met. Not all planners are completely autonomous. Some provide for human interface, through which goals may be generated, and allow varying degrees of human intervention in the planning process.

Problem Domain

The *problem domain* is the domain (environment) the planner reasons about and takes actions on. The problem domain is composed of a collection of *problems* that the planner desires to solve. The planner takes actions on the problem domain via the *inputs* to *solve* a particular problem. The planner measures the effect of these actions via the *outputs* of the problem domain. The *disturbances* represent uncertainty in the problem domain. The *solution* of a problem is composed of the sequence of inputs and outputs (possibly states) generated in achieving the goal.

One develops a *model* of the real problem domain to study planning systems. This is called the *problem representation.* The real problem domain is in some sense infinite; that is, no model could ever capture all the information in it. The problem representation is necessarily inaccurate. It may even be inaccurate by choice, such as when the planning system designer ignores certain problem domain characteristics in favor of using a simpler representation. Simpler models are desirable, since there is an inversely proportional relationship between modeling complexity and analysis power. The characteristics of the problem domain that are ignored or missed are collectively represented by disturbances in the model. The result is that disturbances in general have a nondeterministic character. Clearly, disturbances occur in *every* problem domain; they can be ignored when they are small, but their effect should always be studied to avoid erroneous planner designs.

AI Planner

In this section we describe the functional components of an AI planner. The AI planner's task is to solve problems. To do so, it coordinates several functions: *Plan generation* is the process of synthesizing a set of candidate plans to achieve the current goal. This can be done for the initial plan or for *replanning* if there is a plan failure. In plan generation, the system *projects* (simulates, with a model of the problem domain) into the future, to determine if a developed plan will succeed. The system then uses heuristic *plan decision rules* based on resource utilization, probability of success, and so forth to choose which plan to execute. The *plan executor* translates the chosen plan into physical actions to be taken on the problem domain. It may use scheduling techniques to do so. *Situation assessment* uses the problem domain inputs, outputs, and problem representation to determine the state of the domain. The estimated domain state is used to update the state of the model that is used for projection in plan generation. The term "situation" is used because of the abstract, global view of the system's state that is taken here. The term "assessment" is used since the value of the state is determined or estimated. *Execution monitoring* uses the estimated domain state and the problem domain inputs and outputs to determine if everything is going as planned. If it isn't—that is, if the plan has failed—the plan generator is notified that it must replan.

A *world modeler* produces an update to a world model or a completely new world model. The world modeler determines the structure of the problem domain rather than just the state of the problem domain, as is done by the situation assessor. It also determines what must be modeled for a problem to be solved; hence it partially determines what may be disturbances in the problem domain. The term "world modeler" is thus used to indicate that it must be cognizant of the entire modeling process. Its final output is a problem representation. A *planner designer* uses the problem representation produced by the world modeler and designs, or makes changes to, the planner so that it can achieve its goal even though there are structural changes in the problem domain. The planner designer may not need a new problem representation if there are not structural changes in the problem domain. It may decide to change the planner's strategy if some performance level is not being attained or if certain events occur in the problem domain.

Issues and Techniques in AI Planning Systems

In this section we briefly outline some of the issues and techniques in AI planning systems, giving reference to the relevant literature. General information is given in Charniak and McDermott (1985), Nilsson (1980), Barr and Feigenbaum (1981, 1982), Cohen and Feigenbaum (1982), Wilensky (1983), Shapiro (1987), Gevarter (1984), Fu et al. (1987), and Tanimoto (1987). Planning has its roots in the study of problem solving (Ernst and Newell, 1969). A relatively complete summary of planning ideas and an extensive bibliography on planning are given in Tate (1985). The goal of this section is to set the terminology of the report, not to create an extensive bibliography. As active research is progressing in many areas of planning, the terminology is also evolving; consequently, the definitions and terminology used here are tentative.

Representation is fundamental to all issues and techniques in AI planning. It refers to the methods used to model the problem domain and the planner and it sets the framework and restrictions on the planning system. Often, it amounts to the specification of a formalism for representing the planner and problem domains in special structures in a computer language. Alternatively, it could constitute a mathematical formalism for studying planning problems. Different types of symbolic representations such as finite automata and predicate or temporal logics have been used. Some methods allow for the modeling of different characteristics. Some do not allow for the modeling of nondeterminism. Representational issues are examined in Charniak and McDermott (1985), Nilsson (1980), Barr and Feigenbaum (1981), Wilensky (1983), Tanimoto (1987), Fu et al. (1987), McDermott (1982, 1985), Warren (1974), Hayes-Roth (1979), Rosenschein (1981), Allen and Koomen (1983), Wilkins (1983), Firschein et al. (1986), Chapman (1987), Stephanopoulos et al. (1987), Hodgson (1987), Drummond et al. (1987), and many others. One should be very careful in the choice of how much detailed mathematical structure or modeling power is allowed, since too much modeling power can hinder the development of some functional components of the planner and of the analysis, verification, and validation of planning systems.

The generality of developed planning techniques depends heavily on whether the approach is *domain dependent* or *domain independent*. Techniques developed for one specific problem domain without concern for their applicability to other domains are domain dependent. An example of domain-dependent work is given in Stefik (1981), and Dudziak et al. (1987). An example of a more general domain-independent planner is given in Sacerdoti (1975), Fikes and Nilsson (1971), or McDermott (1985). Other work that examines domain dependence is given in Wilkins (1983, 1984). The results in the next section of this paper are both domain independent and problem representation independent.

Planners can be classified broadly as either *hierarchical* or *nonhierarchical*. A nonhierarchical planner makes all of its decisions at one level, while in a hierarchical planner there is delegation of duties to lower levels and a layered distribution of decision making. Their fundamental operation is explained in Tate (1985), Cohen and Feigenbaum (1982), and Nilsson (1980). Characteristics of and a comparison between these two types of planners are given in Stefik (1981). Some of the original work was done in Sacerdoti (1973). Other important work is given in Tate (1977), Adams (1985), Hayes-Roth (1979), Fikes (1971), Sacerdoti (1975), Charniak and McDermott (1985), and Stefik (1981).

Planners can also be classified as *linear* or *nonlinear*. A linear planner produces a strict sequence of actions as a plan, while a nonlinear planner produces a partially ordered sequence where coordination between the tasks and subtasks to be executed is carefully considered. The original work on nonlinear planning is given in the classic work of Sacerdoti (1975). Extensions to this work are given in Tate (1977), Allen and Koomen (1983), Vere (1983), and Wilkins (1983, 1984). A good summary is given in Tate (1985).

Several types of *interactions* can occur in planning. One is the interaction between subtasks or subplans that requires their proper coordination. Another is between different goals we might want to achieve, Waldinger (1975). Still another is between different planning modules or with the human interface. Important work in this area is found in Broverman and Croft (1987), Bruce and Newman (1978), and Hayes (1987). A nice categorization of types of interactions and summary of ideas is given in Tate (1985).

Search is used in planning systems, for instance, to find a feasible plan. There are many types of search, such as the *heuristic search* algorithms called A* and AO*. A good introduction to the topic is given in Pearl (1984), Nilsson (1980), or Barr and Feigenbaum (1981). A summary of search techniques used in planning is given in Tate (1985) or in Shapiro (1987). In Korf (1987) the author uses results from the theory of search to quantify some time and space complexities of planning. In Passino and Antsaklis (1988a, 1988b) planning problems are solved using heuristic search in a Petri net framework. An application to a robot problem is given in Graglia and Meystel (1987) and to mission control/decision support in Deutsch et al. (1985).

Skeletal plans, plan schemata, and *scripts* are all representations of plans with varying degrees of abstraction. Skeletal plans are plans that to some extent do not have all the details filled in. A script is a knowledge structure containing a stereotypic sequence of actions. Plan schemata are similar. Often planners that use these forms for plans store them in a *plan library. Hypothetical planning* is planning where the planner hypothesizes a goal, produces a subsequent plan, and stores it in a plan library for later use, all while the current plan is executing. Good explanations of some of these ideas are given in Shaprio (1987), Cohen and Feigenbaum (1982), and Adams et al. (1985).

Replanning is the process by which plans are generated so that the system recovers after a plan has failed. There are two types of plan failures. One occurs in plan generation, where the planner fails to generate a plan. In this case, replanning can be successful only if a planner redesigner makes some changes to the planner strategy. The second type of plan failure occurs in the execution of the plan and is due to disturbances in the problem domain. This plan failure can be accommodated by replanning in the plan generation module, if the failure is not due to a significant structural change in the problem domain. If it is, then the world modeler will produce a new world model and the planner designer will make the appropriate changes to the planner so that it can replan. Some of these ideas are discussed in Charniak and McDermott (1985).

Projection is used in plan generation to look into the future so that the feasibility of a candidate plan can be decided. If it is assumed that there are no disturbances in the problem domain, and a plan can be generated, then it can be executed with complete confidence in plan success. Disturbances cannot be ignored in problem domains associated with real-world dynamic environments; therefore, complete projection is often futile. The chosen *projection length* (number of steps in symbolic simulation) depends on the type and frequency of disturbances and is therefore domain dependent. Notice that if the projection length is short, plan execution can be interleaved with planning and replanning. This sort of planning has been named *reactive planning* (Georgeff, 1987). A completely *proactive planner* always has a plan ready for execution (in a plan library) no matter what the situation is in the problem domain. These plans could be skeletal or scripts. Some mixture of proactive and reactive planning with varying projection length is probably most appropriate. These ideas are explained from a different point of view in Hayes-Roth (1979). There *opportunistic planning* is introduced; one forms a partial plan and then begins execution with the hope that opportunities will arise during execution that will allow for the complete specification of the plan and its ultimate success.

Planning with constraints is a planning methodology where certain constraints on the planning process are set and the planner must ensure that these constraints are not violated (Stefik, 1981; Winslett, 1987). A planning system that uses several planners to solve different parts of the problem is explained in Hayes-Roth (1979), Firschein (1986), and Georgeff (1984). Some issues in *human interface* to the planning process are discussed in Charniak and McDermott (1985), Barr (1981, 1982), and Cohen and Feigenbaum (1982).

Distributed planning occurs when a problem is solved by coordinating the results from several expert planners. It is also called *multiagent planning.* A relatively complete overview of distributed planning is contained in Tate (1985) and another bibliography is given in Kempf (1987).

Metaplanning is the process of reasoning about how a planner reasons. It is used with world modeling and changes the planning strategy. A domain-dependent example of metaplanning is given in Stefik (1981). A general explanation of metaplanning is given in Wilensky (1983). Other information on metaplanning is found in Wilensky (1981) and Hayes-Roth and Hayes-Roth (1979). Planners can also be made to *learn.* For example, a simple form of learning is to save successful plans in a plan library for later use in the same situation. Other forms of learning are described in Charniak (1985), Cohen and Feigenbaum (1982), and Fikes et al. (1972).

The objective of this paper is to begin the formulation of an approach for the quantitative study of AI planning systems that operate in uncertain, dynamic, and time-critical environments. One way to study planning system behavior is by the empirical method. Other more quantitative studies specify a computer language or formal model for the problem domain. (See all the representation research outlined above.) Formal modeling and analysis has used *propositional dynamic logic* for the verification of planning systems (Rosenschein, 1981). Several types of *temporal logics* have been used to model and analyze planning systems (McDermott, 1978, 1982, 1985; and Koomen, 1983). The work in Chapman (1987) using *modal logic* is also important. In Giordana and Saitta (1985) the authors model a production system's data and rules and check for its consistency using a type of *Petri net* formalism. Recently, a different Petri net has been introduced for the modeling and analysis of AI planning problems (Passino and Antsaklis, 1988a, 1988b). The growing body of literature on expert system verification and validation also may be relevant.

AI PLANNING AND CONTROL THEORY: ANALOGY AND RELATIONSHIPS

Relationships and an extensive analogy between AI planning and control system architectures and concepts are developed in this section. This is possible because both are problem solving systems (as described earlier) with different problem domains. It is useful to draw the analogy since conventional problem-solving systems, such as control systems, are very well studied. They have a well-developed set of fundamental concepts and formal mathematical modeling, analysis, and design techniques. The analogy is used to derive a corresponding foundation of fundamental concepts for AI planning systems that can be used to develop modeling, analysis, and design techniques.

The discussions below are meant to motivate the utility of using general systems theory for the study of AI planning systems. In particular, it is hoped that it is made clear that the general concepts of controllability, observability, stability, and so forth as defined in systems and control theory will be useful in the quantitative study of AI planning systems. The results here will probably need to be revised and expanded before a careful formulation of a mathematical theory of AI planning via control theory is possible.

After discussing some fundamental system theoretic concepts and the problem domain and plant analogy, open-loop planning is introduced. Feedback planning with and without situation assessment is introduced and discussed. Adaptive AI planning is introduced. The details of the internal architectures of the various planners introduced are given, but they are used only to discuss the concepts in this paper. A particular planner implementation may have a different functional architecture. This section closes by highlighting some recent ideas from intelligent autonomous control relevant to AI planning.

The Problem Domain–Plant Analogy

In this section we shall give the structural analogy (functional analogy) between the problem domain and the plant. In conventional control, the *plant* is a dynamical system whose behavior we wish to study and alter. It is generally described with linear or nonlinear differential/difference equations and is either deterministic or nondeterministic. The *problem domain* is the domain (environment) the AI planner reasons about and takes actions on. It can be modeled using predicate or temporal logic or other symbolic techniques such as finite automata. We develop the analogy further using Fig. 1.

As it is often done, we adopt the convention that actuators and sensors are part of the plant and thus part of the problem domain description. Plant *actuators* are hardware devices (transducers) that translate commands $u(t)$ from the controller into actions taken on the physical system. The variable t represents time.

In a problem domain, we take a more macroscopic view of an actuator, a view that depends on available hardware and the type of inputs generated by the planner. For example, in a robotic system a manipulator may be quite dexterous;

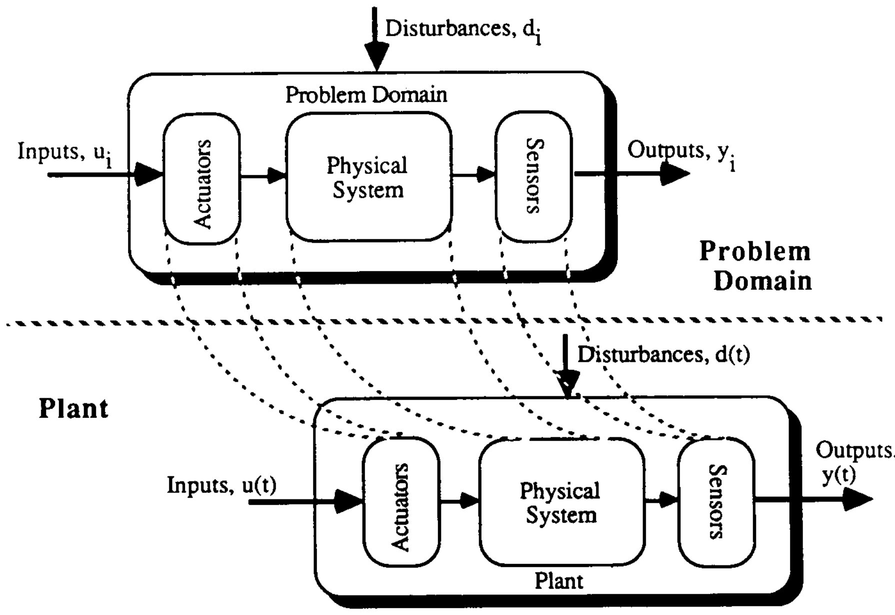

FIGURE 1. Problem domain/plant structure.

one may be able to send the simple command "pick up object," and it will know how to move to the object, grip it, and pick it up. Such a manipulator can be seen as an actuator, although simpler ones also exist. Clearly, the inputs to the problem domain can be more abstract than those of a plant; consequently, we describe them with symbols u_i rather than numbers. The index i represents time in the problem domain. The symbols are quite general and allow for the representation of all possible actions that *any* planner can take on the problem domain. For example, u_1 = "pick up object" or u_2 = "move manipulator from position 3 to position 7." Rather than an input $u(t)$ for the plant, the problem domain input u_i is a time sequence of symbols.

The *physical system* for both the problem domain and the plant is some portion of the real world that we wish to study and alter. The difference between the two is in the types of systems that are normally considered and thus the modeling techniques that are used (see discussion in the section entitled Problem Domain). Aspects of the dynamical behavior of plants such as cars, antennas, satellites, or submarines can be modeled by differential equations. Problem domains studied in the AI planning literature include simple robot problems (Charniak and McDermott, 1985), experiments in molecular genetics (Stefik, 1981), or running errands (Hayes-Roth and Hayes-Roth, 1979). Notice that problem domains cannot always be described by differential equations. Consequently, conventional control techniques are inappropriate for AI planning problems.

The *sensors* in the plant and problem domain are used to measure variables of the physical system and translate this information to $y(t)$ for the controller and y_i for the planner. The symbols y_i provide for the representation of all possible measured values of outputs of the problem domain. As with the actuators in the problem domain, we take a more macroscopic view of sensors. They can combine various data to form an aggregate representation of dynamic problem domain information. This necessitates the use of symbolic representations of the measured outputs; consequently, y_i is a time sequence of symbols. For example, in the robot problem the positions of some of the objects to be moved could be represented with y_i. The outputs could be y_1 = "object 1 in position 5" and y_2 = "object 1 in position 3." The inputs u_i can affect the physical system so that the outputs y_i can change over time.

The *state* of the plant or problem domain (or any dynamical system) is the information necessary to predict the future behavior of the system given the present and future system inputs. A particular state is a snapshot of the system's behavior. The *initial state* is the initial condition on the differential/difference equation that describes the plant, or the initial situation in the problem domain prior to the first time a plan is executed. We shall denote the state of the plant with $x(t)$ and the problem domain with x_i. The set of all possible states is loosely referred to as the *state space.* In our robot problem domain, the initial sate can be the initial positions of the manipulator and objects. For two objects, the initial state might be x_0 = "object 1 in position 3 and object 2 in position 7 and manipulator in position 5." Notice that part of the state is directly contained in the output for our example. The state describes the current actuation and sensing situation in addition to the physical system, since the sensors and actuators are considered part of the problem domain.

The plant and problem domain are necessarily affected by *disturbances* $d(t)$ or symbols d_i respectively (see discussion under Problem Domain). These can appear as modeling inaccuracies, parameter variations, or noise in the actuators, physical system, and sensors. In our robotics problem domain a disturbance might be some external, unmodeled agent, who also moves the objects. Next we show how the functional analogy between the plant and problem domain extends to a mathematical analogy.

The Plant–Problem Domain Model Analogy

Because of their strong structural similarities it is not surprising that we can develop an analogy between the models that we use for the plant and the problem domain and between fundamental systems concepts. Essentially this involves a discussion of the application of a general systems theory described in Kalman et al. (1969) to planning systems. We extract the essential control theoretic ideas and adapt them to planning theory, without providing lengthly explanations of conventional control theory. The interested reader can find the relevant control theoretical ideas presented below in Kalman et al. (1969), D'Azzo and Houpis (1981), Chen (1984), Miller and Michel (1982), Goodwin and Sin (1984), and Astrom and Wittenmark (1984). We assume that the plant is described by a set of stochastic, possibly nonlinear, differential equations called the state equation and the output equation. They describe the dynamics of the plant, its structure, and its connections. We assume that we can describe the dynamics of the problem domain by a set of symbolic equations such as those used to describe finite-state automata (Hopcroft and Ullman, 1979). For systems described with, for instance, a Moore machine, there exist analogous state and output equations. These equations describe the dynamics of a system such as the problem domain (or the planner or planning system), its structure, and its connections.

The mathematical analogy continues by studying certain properties of systems that have been found to be of utmost importance in conventional control theory.

Controllability

In control theory, and thus in planning theory, *controllability* refers to the ability of a system's inputs to change the state of the system. It is convenient to consider a deterministic system for the discussion. A sequence of inputs u_i can *transfer* or *steer* a state from one value to another. In the robot example, a sequence of input actions transfers the state from x_0 = "object 1 in position 3 and object 2 in position 7 and manipulator in position 5" to x_7 = "object 1 in position 5 and object 2 in position 10 and manipulator in position 1."

A system is said to be *completely controllable* at time i if there exists a finite time $j > i$ such that for any state x_i and any state x there exists an input sequence $u_i, \ldots, u_j$ that will transfer the state x_i to the state x at time j, that is, $x_j = x$.

Intuitively, this means that a problem domain is completely controllable at some time if and only if, for every two state values of the state space of the problem representation, there exists a finite sequence of inputs (that some planner could produce) that will move the state from one value to the other (one state to the other). Also notice that the time $j - i$ is not necessarily the minimum possible. There might be another sequence of inputs that will bring one state to the other in fewer steps. In the robot example, the problem domain is completely controllable if, for any position of the manipulator and objects, there exist actions (inputs) that can change to any other position of the objects and manipulator.

If a problem domain is completely controllable, then for *any* state there exists a planner that can achieve *any* specified goal state. Sometimes complete controllability is not a property of the system, but it may possess a weaker form of controllability, which we discuss next. To discuss a more realistic, weaker form of controllability we assume that the state space can be partitioned into disjoint sets of controllable and uncontrollable states.

A system is said to be *weakly controllable* at time i if there exists a finite time $j > i$ such that for any states x_i and x, both in the set of controllable states, there exists an input sequence $u_i, \ldots, u_j$ that will transfer the state x_i to the state x at time j, that is, $x_j = x$.

If the initial state and the goal state are given and contained in the set of controllable states and the problem representation is weakly controllable, then there exists a planner that can move the initial state to the goal state. That is, there exists a planner that can solve the problem. In the robot example, if the problem representation is weakly controllable and the initial state begins in the set of controllable states, then there are actions (inputs) that can move the manipulator and objects to a certain set of positions in the set of controllable states, the ones one might want to move them to.

A problem representation that is not completely controllable may still be weakly controllable. If a problem representation is not weakly controllable, then it is not completely controllable. Note that there are corresponding definitions for *output controllability.*

Observability

In control theory, and thus in planning theory, *observability* of the problem domain refers to the ability to determine the state of a system from the inputs, outputs, and model of the system.

A system is said to be *completely observable* at time i if there exists a finite time $j > i$, such that for any state x_i the problem representation, the sequence of inputs, and the corresponding sequence of outputs over the time interval $[i, j]$ uniquely determine the state x_i.

Intuitively, this means that a problem domain is completely observable at some time if and only if, for every sequence of domain inputs and their corresponding outputs, the model of the domain and the input and output sequences are all that is necessary to determine the state that the domain began in. A problem domain that is completely observable on some long time interval may not be completely observable on a shorter interval. It may take a longer sequence of inputs and outputs to determine the state.

In the robot example, if the problem domain is completely observable, then for every sequence of actions (inputs) there exists a situation assessor that can determine the position of the objects and manipulator from the input sequence, output sequence, and model of the problem domain.

If a problem domain is completely observable, then for *any* initial state, there exists a situation assessor that can determine the state of the problem domain. This situation assessor needs both the inputs and the outputs of the problem domain, and there is the assumption that there are no disturbances in the domain. Sometimes complete observability is not a property of systems, but they may possess a weaker form of observability, which is defined next. To discuss a more realistic, weaker form of observability, we will assume that the state space can be partitioned into disjoint sets of observable and unobservable states.

A system is said to be *weakly observable* at time i if there exists a finite time $j > i$ such that, for any state x_i in the set of observable states, the problem representation, the sequence of inputs, and the corresponding sequence of outputs over the interval $[i, j]$ uniquely determine the state x_i.

If the problem domain is weakly observable, there exists a situation assessor that can determine the state of the problem domain given that the system state begins in the set of observable states. In the robot example, if the problem domain is weakly observable, then for any initial observable state and every sequence of actions (inputs) that any planner can produce, there exists a situation assessor that can determine the position of the objects and manipulator from the planner input sequence, output sequence, and model of the problem domain.

If a problem domain is not completely observable, it may still be weakly observable. If it is completely observable, it is weakly observable. Like controllability, observability is a property of systems in general; therefore it has meaning for the problem domain, planner, and planning system.

In control, and thus planning, theory, a model of a system is *minimal* or *irreducible* if it uses the least number of state variables to describe the dynamical behavior. That is, it is minimal if there are no redundancies in the model. If a system is not minimal, then there exists a different system representation whose state space is of smaller dimension (size). The minimality property quantifies how well the problem domain was modeled. Minimality is also a property of the planner and the whole planning system.

Stability

In control, and thus in planning theory, we say that a system is *internally stable* if with no inputs, when the system begins in some particular set of states and the state is perturbed, it will always return to that set of states. For the discussion we partition the state space into disjoint sets of "good" states and "bad" states. Also, we define the *null input* for all problem domains as the input that has no effect on the problem domain. Assume that the input to the system is the null input for all time. A system is said to be *internally stable* if, when it begins in a good state and is perturbed into any other state, it will always return to a good state.

To clarify the definition, a specific example is given. Suppose that we have the robot manipulator described above. Suppose further that the set of positions the manipulator can be in can be broken into two sets, the good positions and the bad positions. A good position might be one in some envelope of its reach, while a bad one might be where it would be dangerously close to some human operator. If such a system was internally stable, then if the manipulator was in the good envelope and was bumped by something, it might come dangerously close to the human operator, but it would resituate itself back in the good envelope without any external intervention.

We make the following definitions to produce one more definition of stability. We assume that we can partition the set of possible input and output symbols into disjoint sets of good and bad inputs and outputs. A system is said to be *input-output stable* if for all good input sequences the corresponding output sequences are good.

In the robot example, suppose the inputs to the manipulator can be broken into two sets, the good ones and the bad ones. A bad input might be one that takes a lot of resources or time to execute, or it might be an input that takes some unreasonable action on the problem domain. Let the output of the robot problem domain be the position of the objects that the manipulator is to move. A bad output position would be to have an object obstruct the operation of some other machine or to have the objects stacked so that one would crush the other.

The robot problem domain is input-output stable if for all reasonable actions (good inputs) the manipulator is asked to perform, it produces a good positioning of the objects (good outputs) in the problem domain. These stability definitions and ideas also apply to the planner and the planning system. We shall expand on this in a later section.

Stabilizability refers to the ability to make a system stable. For a planning system it may, for instance, refer to the ability of any planner to stabilize the problem domain. A system is said to be *stabilizable* if the set of controllable states contains the bad states. For the robot example, the problem domain is stabilizable if, for all states that represent bad positions of the manipulator arm, there are inputs that can move the arm to its good (state) operating envelope. Detectability refers to the ability to detect instabilities in a system. For a planning system it may, for instance, refer to the ability of the situation assessor to determine if there are instabilities in the problem domain. A system is said to be *detectable* if the set of observable states contains the bad states. For the robot example, if the problem domain is detectable, then for all input sequences that place the manipulator arm in a bad position, there exists a situation assessor that can determine the state. These definitions also apply to the planner and the planning system.

Rate of a System

The *rate* of a system in conventional control theory quantifies how quickly the system will react to its inputs or how fast the outputs will change for a given set of inputs. In the time domain, other terms used include time constant and rise time. For linear control theory, *bandwidth* is used in the frequency domain. In an AI planning system, rate is defined similarly. However, it cannot be properly defined mathematically until the form of the model is specified. For now we can think of it as some global measure of how many steps it takes for the system to react to some inputs. In the robot example, it is some measure of how many steps it will take to arrange the objects properly in some desired configuration.

In this section we have developed a foundation of fundamental ideas for planning theory. When one begins to formulate a planning problem, one begins by modeling the problem domain; that is, the form of problem representation is chosen. Notice that the above properties are *both* domain and representation independent. Controllability and observability studies will quantify the feasibility of solving the problem at hand. Minimality will tell how well the problem domain was modeled. Stability of the problem domain is an important qualitative property that must be understood so that a planner of the proper form is designed. Stabilizability and detectability studies will say whether it is possible to attain stability in a feedback planning system. Rate of the planning system quantifies how quickly the problem can be solved. We expand on these ideas for specific planners and planning system structures in the following sections.

Open Loop AI Planning Systems

In this section we define open loop planning systems and outline some of their characteristics. They are named "open loop" because they use no feedback information from the problem domain. We begin by drawing analogies with the structure of open loop control systems.

Open Loop Control System–Planning System Structural Analogy

Here we develop a structural analogy between open loop conventional control systems and open loop planning systems, beginning with Fig. 2. In conventional control theory, the open loop control system has the structure shown at the bottom of Fig. 2. The outputs of the controller are connected to the inputs of the plant so that they can change the behavior of the plant. The input to the controller is the *reference input* $r(t)$, and it is what we desire the output of the plant to be. The controller is supposed to select the inputs of the plant $u(t)$ so that $y(t) \to r(t)$, or $y(t) - r(t)$ is appropriately small for all times greater than some value of t. Specifications on the performance of control systems speak of the quality of the response of $y(t)$. For example, we might want some type of transient response or we might want to reduce the effect of the disturbance on the output $y(t)$. However,

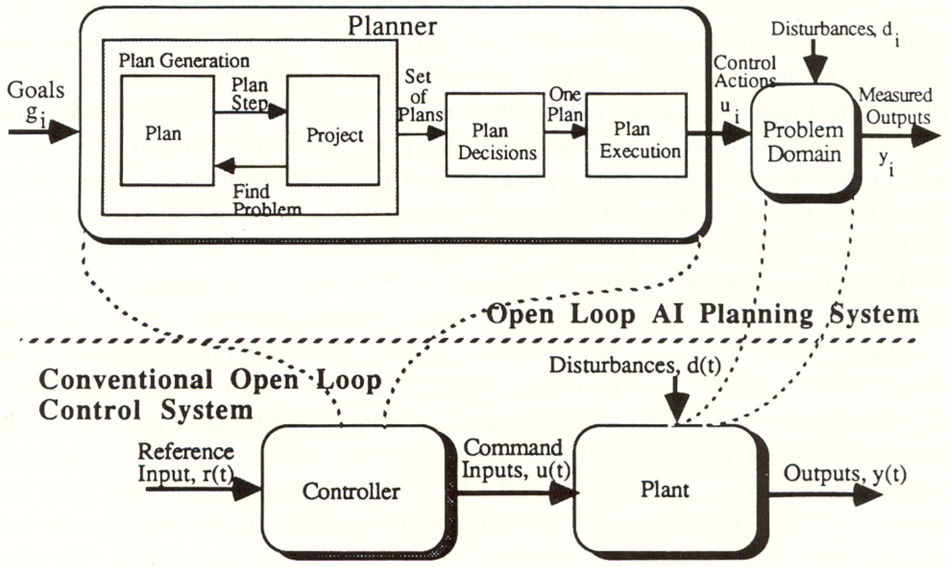

FIGURE 2. Open loop structural analogy.

an open loop control system cannot reduce the effect of disturbances in any way; notice that, by definition, the disturbances cannot be measured.

In the open loop planner, plan generation is the process of synthesizing a set of candidate plans to achieve the goal at step i, which we denote by g_i. The goals g_i may remain fixed, or change in time. In plan generation, the system projects (simulates, with a model of the problem domain) into the future, to determine if a developed plan will succeed. The system then uses heuristic plan decision rules based on resource utilization, probability of success, and so forth to choose which plan to execute. The plan executor translates the chosen plan into actions (inputs u_i) to be taken on the problem domain. Many AI planners implemented to date are open loop planners.

Characteristics of AI Open Loop Planning

We first consider the characteristics of the planner itself (not connected to the problem domain) by interpreting the results above. Then we outline the characteristics of open loop planning systems.

Fundamental issues in the planner. It is useful to consider the planner to be a model of some human expert planner. The state of the planner is the situation describing the planner's problem-solving strategy at a particular instant. Planner controllability refers to the ability of the goal inputs to affect the state of the planner. Planner observability refers to the ability to determine the planner state using the goal inputs, planner outputs u_i, and the model of the planner. Minimality of the planner model reflects how well the planner was designed and modeled and thus, since an actual planner implementation depends on the model, how cheaply the planner can be implemented. Internal stability of the planner refers to its ability to stay in a problem-solving state of mind (ready to solve problems) when there are no goals to achieve (a null input). Input-output stability of the planner is attained if for all reasonable, admissible goals input the planner produces reasonable, acceptable outputs u_i. The planner is stabilizable if there exists a sequence of goals that will keep the planner properly focused on the problem. The planner is detectable if for all goal sequences that cause the planner to lose its focus of attention, the inputs, outputs, and model of the planner can be used to determine where the focus of attention is. The rate of the planner quantifies how quickly the planner can produce solutions with appropriate control actions.

Fundamental issues in the open loop system. All of the interpretations given above for the planner are valid here, the difference being that since we cascade the problem domain we are thinking of solving a particular problem. If the problem domain is uncontrollable, there may not exist a planner capable of solving the problem. If it is controllable, a planner does exist. This does not mean that if the problem domain is completely controllable, we can choose any planner and it will solve the problem. It just says that one exists. Situation assessment and execution monitoring cannot be done since there is no connection to the outputs of the problem domain. Consequently, there cannot be any replanning. If there are any disturbances in the problem domain, the planner canl become totally lost in its problem-solving process, because it has no ability to recover from plan failure; it is even unaware that there was a failure. We say that the planning system is sensitive to problem domain variations and open loop planners cannot reduce this sensitivity. This is closely related to the idea of sensitivity reduction in conventional control theory. Since, as explained under System Classification, there will *always* be some disturbances in the real world, open loop planners will necessarily fail at their task. However, they can work if the problem domain is well modeled and the disturbances are quite insignificant. This generally requires the use of a very complex, detailed model of the problem domain in the case of significant real-world problems. Notice that since the outputs are not sensed, if the problem domain is unstable (input-output or internally), then it is never stabilizable in open loop planning. Open loop stabilization requires absolutely exact knowledge of the problem domain. Since often this cannot be obtained, even insignificant disturbances can be catastrophic. The rate of the open loop system (planner and problem domain) can be increased over that of the problem domain, since the planner can choose shorter-path solutions. So analysis can be done to determine if certain specifications about the performance of the planning system can be achieved. World modeling and planner designing also cannot be done since the outputs are not sensed.

The length of projection in plan generation can be quite long, since if one is using open loop planning the disturbances must be assumed nonexistent or insignificant. The only reason for making the projection length shorter would be to begin plan execution. If the projection length is too short for the plan generator to specify a set of plans that will work, there will be uncertainty in the plan execution that may lead to ultimate plan failure. This is why current open loop planners build the complete plan and then execute it.

Open loop planners do have the advantage of simplicity. If the problem domain is stable and disturbances are insignificant, they should certainly be considered. They are cheaper to implement than the closed loop planners described in the next two sections, since one does not need to buy sensors to gather information about the states and outputs of the problem domain.

AI Feedback Planning Systems

AI feedback planning systems are analogous to conventional feedback control systems that do not use state estimation; they do not use situation assessment. Charniak and McDermott (1985, chapter 9) point out the inherent feedback in the planning process. They do not, however, make a clear distinction of the separation between the planner and problem domain and their interconnections. In this section the distinctions will be clarified. In the next section we will introduce AI feedback planning systems that use situation assessment.

The Closed Loop System Structural Analogy

The analogy between the feedback structures emerges from Fig. 3. The structure is the same as for the open loop system except that there is the feedback connection. This allows the planner to perform execution monitoring and replanning. The feedback planning system can recover from plan failures due to significant disturbances in the problem domain. The execution monitoring system uses the measured outputs, inputs, and domain model to determine if the current plan has failed. If a plan fails, it informs the plan generator that it must replan.

Fundamental Issues in AI Feedback Planning Systems

If the problem domain is not completely controllable, one can perhaps use more elaborate actuator systems that will properly affect the state of the domain, or perhaps rederive the model of the problem domain since controllability is a property of the mathematical model used. Therefore, controllability studies can be used for design guidelines for the problem domain, and likewise for observability. Situation assessment is not needed in a planner if the full state of the problem domain is measurable. This is analogous to full state feedback in conventional control theory. If observability studies show that some states of the

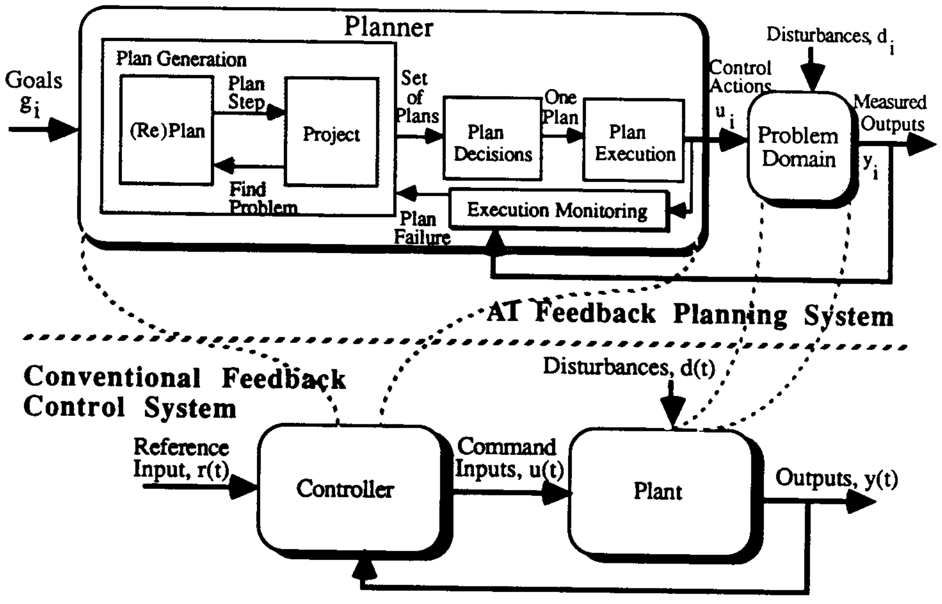

FIGURE 3. Closed loop structural analogy.

domain are unobservable, one can design and implement additional sensors that can provide the necessary information about the state. We see that there is a trade-off between expense of implementation of a planning system and planner complexity. It may be expensive to implement sensors to sense the whole state, but then situation assessment is not necessary, thus making the planner simpler.

If the problem domain is completely controllable and observable with respect to the chosen inputs and outputs, there exists a feedback planner that can stabilize the problem domain if it is unstable.

Goal Tracking/Following in AI Feedback Planning Systems

For the controller to force the plant output to *track* or *follow* the reference input, it compares the output to the current reference input and decides what to input into the plant. In comparing $r(t)$ and $y(t)$, the controller simply uses the difference $r(t) - y(t)$ to determine how well it is meeting its objective at any instant and takes appropriate actions. The difference $r(t) - y(t)$, called the *error,* is a control measure.

The planning system examines the difference between the current output situation and the goal to be achieved and takes subsequent actions. The error in the planning system is not as easy to form as in the conventional control case, because distance between symbols is more difficult to quantify. One could, however, say that a problem domain output is closer to the goal if the components that make up the output are closer to satisfying the goal. If the goal is a conjunction of several subgoals, it is closer to the output if the outputs make more of the subgoals true.

Suppose we fix the goal input to the feedback planning system to be the same for all time, that is, $g_i = g_0$ for all i. The feedback planning system is then considered to be a *regulatory planning system.* It achieves the goal state and regulates the inputs to the problem domain to ensure that the goals are met for all time even in the face of problem domain disturbances.

If the sequence of goals g_i, the exogenous inputs to the planner, change over time and the planner achieves the goals sequentially, the planning system is said to be a *goal-following* or *goal-tracking planning system.* Notice that if the goals change too quickly, the planner may not be able to keep up, and there will be some *tracking error.*

Design Issues in AI Feedback Planning Systems

When one designs a feedback planning system, there are certain properties that are desirable for the closed loop planning system. We refer to these properties collectively as the *closed loop specifications.* These could be stability, rate, performance measures, and so forth.

Normally, stability is always a closed loop specification. The planner is designed so that stability is present in the closed loop system. Take special note that even though feedback planning systems have the ability to stabilize any system that is stabilizable, they can also destabilize an otherwise stable problem domain. As an example, consider the case when, because of some delay in receiving feedback information, the planner applies the right plan but at the wrong time. One must be careful in the design so that this is avoided. Systems are often destabilized when one tries to increase the rate of the system. Therefore, if some rate is also a closed loop specification, it may be the case that only a certain rate is achievable, given that you want a safety margin to ensure that the system is stable.

A very important advantage of feedback planning systems over their open loop counterparts is their ability to reject problem domain disturbances (reach and maintain a goal even with disturbances) and to be insensitive to problem domain variations (reach a goal even though the model is inaccurate). In conventional control theory these objectives are designed for, using techniques that will produce optimal disturbance rejection and sensitivity reduction. Systems that meet these objectives are said to be *robust*. The theory of robust control addresses these questions.

AI Feedback Planning Systems with Situation Assessment

Analogous to the conventional controller that uses state estimation, there are AI feedback planning systems that use situation assessment. In Wilensky (1983) the author's description of planning and understanding is quite similar in character to what is presented below. Understanding corresponds to situation assessment. An understanding system, according to Wilensky (p. 10), "is given the 'solution'" (the inputs and outputs of the problem domain) "and must reconstruct the goal and state of the world from it." In this section it is shown "that a good problem solver should incorporate some of the capabilities that were just attributed to understanding mechanisms" (Wilensky, 1983, p. 10). Wilensky also explains the ideas behind metaplanning, which are related to the next section on AI Adaptive planning.

AI Feedback Planning System with Situation Assessment-Control Structural Analogy

The structural analogy between the two feedback systems is shown in Fig. 4. If the problem domain is observable, then there exists a situation assessor that can determine the state of the problem domain from the domain inputs, outputs, and model. If this condition is not met, situation assessment cannot be successful at all times. Situation assessment is particularly useful in stabilizing the problem domain when it is detectable. The state estimator (also called an observer) in the conventional controller is analogous to the situation assessor. Both estimate the state and provide the state estimate for use in determining what actions ought to

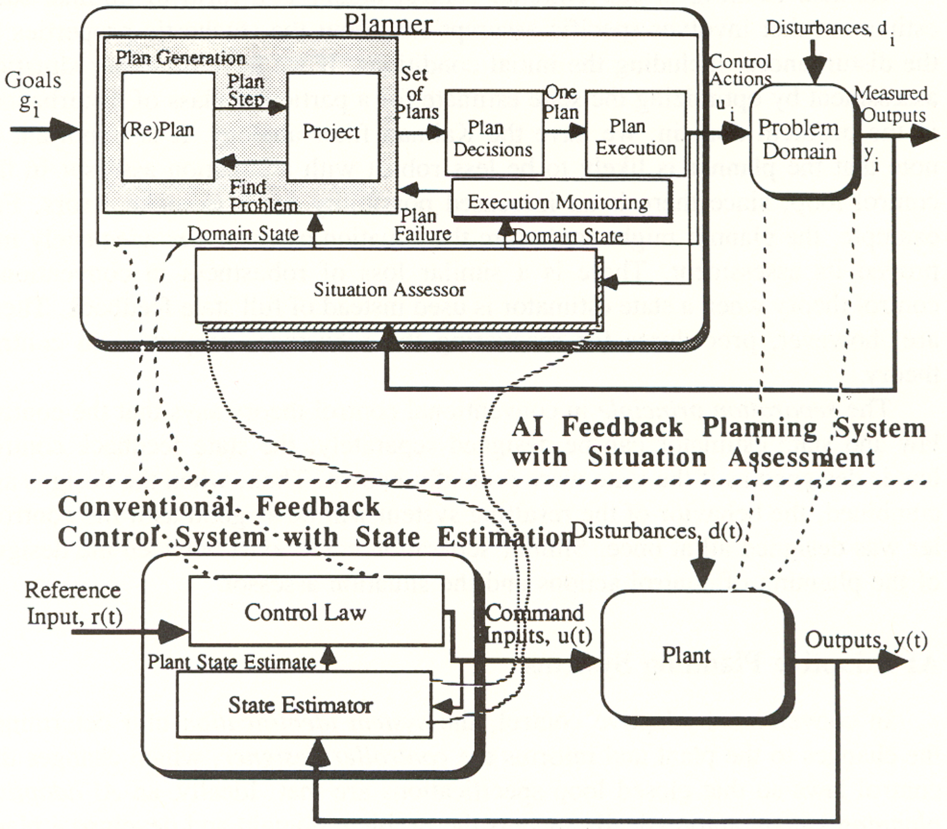

FIGURE 4. Closed loop structural analogy with situation assessment and state estimation.

be taken next (by the control law or rest of the planner). Notice that with the situation assessor it is possible for the execution monitor to perform better. With complete state information it will be able to detect plan failure more accurately or in cases where it was not possible without situation assessment.

Situation Assessment in AI Feedback Planning

The observability condition suggests that the situation assessor will need the problem domain inputs, outputs, and model to perform its task. When projection is done with a model of the domain, the planner knows where the state of the domain ought to be. The situation assessor uses this information and the domain inputs, outputs, and model to modify its own estimate of the state. This idea has already been mentioned in Hawker and Nagel (1987). There is the need for a measurement (of the error) between where the state ought to be and the current estimate of the state. This control measure is used to help determine where the state really is.

Kalman filtering in conventional control theory is a form of optimal state estimation that involves specific assumptions about the stochastic properties of the disturbances, including the initial conditions. If we have done our situation assessment by optimizing the state estimate for a particular class of disturbances in the problem domain, we have the Kalman filter analogy. It is important to note that the planner is likely to be less robust with a situation assessor in the control loop, since there is an increased possibility for it to make errors. For example, the planner might act before the situation assessor has adequately improved its assessment. There is a similar loss of robustness in conventional control theory when a state estimator is used instead of full state feedback. There are, however, procedures for recovering the robustness properties in control theory.

The *separation principle* in conventional control theory says that the control law and state estimator can be designed separately; the state feedback control law is designed as if the state were prefectly known. Then, when the designs are combined, the behavior of the resulting system will be as good as if the controller was designed all at once. Similar separation could exist between the designs of the planning of control actions and the situation assessor.

AI Adaptive Planning Systems

In conventional adaptive control, the *system identification* unit determines the changes to the plant and informs the *controller designer*, which changes the control laws so that closed loop specifications are met. Ideally, an *AI adaptive planning system* automatically models the problem domain and develops a planner that will solve the new problem in the domain. The structural analogy between the two is shown in Fig. 5.

A simpler adaptive planner begins with a model of the problem domain, and if there are domain structural changes, it updates the model of the domain. This is called *world modeling*. Using this updated model, the *planner designer* decides if it is necessary to make changes to the current planning strategy so that the problem represented by the new problem domain will be solvable. The AI adaptive planning structure is used to implement metaplanning. Metaplanning, discussed earlier, is examined in Wilensky (1983).

Notice that if the planner is not executing actions that excite the domain properly, the world modeler and thus the planner designer may not be able to perform their tasks. This is the problem of *sufficient and persistent excitation* in conventional control theory. Note that world modeling is not always needed; a planner designer can just change the strategy of the planner based on the occurrence of certain logical combinations of events. Notice also that the adaptive planner can perform fault detection and identification and ultimate accommodation for the failure.

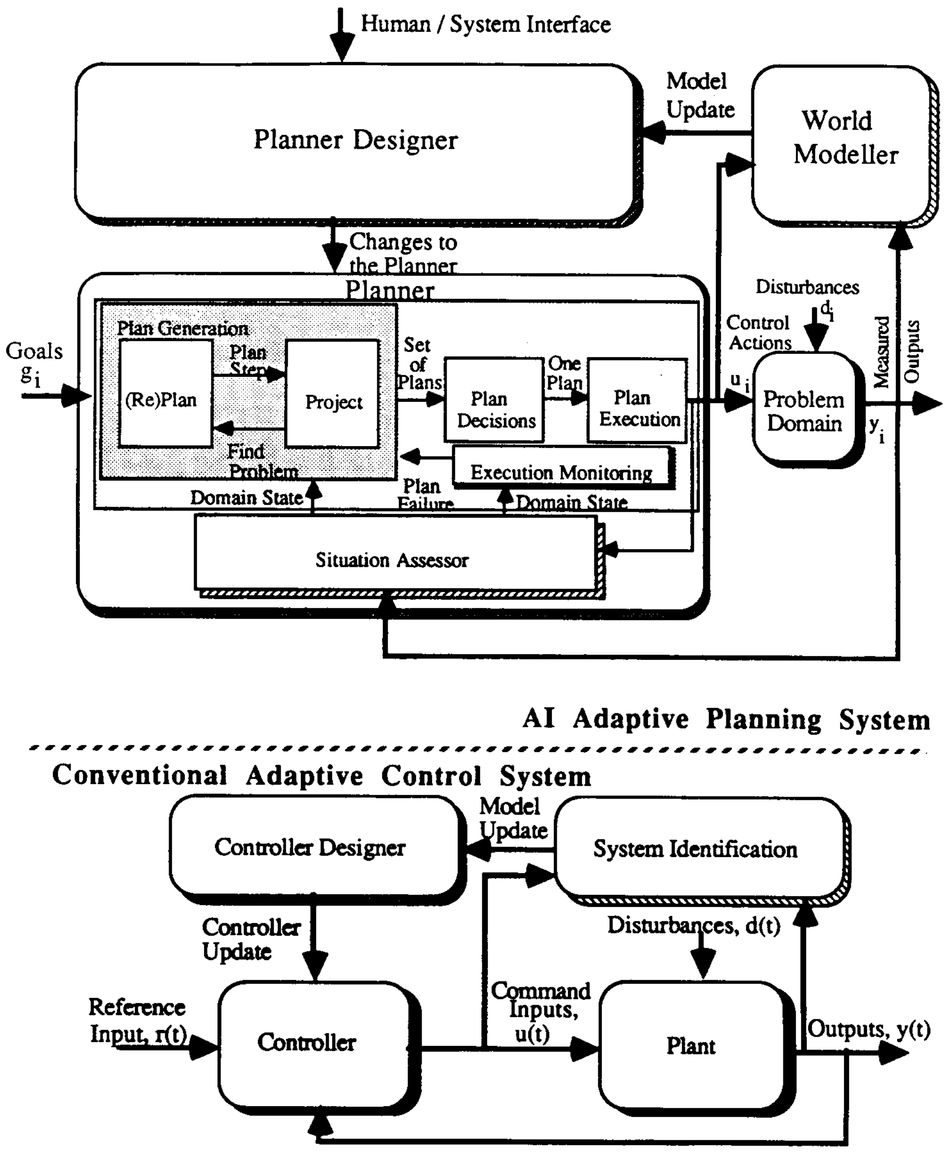

FIGURE 5. Adaptive structural analogy.

One important difference between the two structures is that in the AI adaptive planning system there is an input from a human or other supervisory system. A high-level goal that can be input at this level could be "go to manufacturing facility 4 and work there" (where it has never worked before). The adaptive planner would plan to achieve this goal by going to the facility, developing a world model, choosing a planning strategy, and forming subsequent plans and taking actions to solve problems so that low-level goals are met. The AI adaptive planner is more *autonomous* than the others. Autonomy is a characteristic of increasingly intelligent systems.

It is interesting to notice the similarities between the AI adaptive planner architecture and the *model of helping* developed in Egan (1986). We give one possible interpretation of the similarity here; many others are possible. Suppose we have a "helper" who is to aid in the solution of a problem by guiding the actions of a "subject." In Egan's theory of effective helping the "Present Scenario" corresponds to the problem representation of the problem that the subject is trying to solve and the planner, which represents the problem solving strategy of the subject. The "Preferred Scenario" consists of the accurate problem representation developed by the helper (world modeler) and the revised problem solving strategy designed by the helper (planner designer) for the subject. The "Action—Getting the New Scenario on Line" is the series of actions taken by the world modeler and planner designer to implement the methods used in the AI adaptive planner.

Issues Relevant to AI Planning Theory from Intelligent Autonomous Control

The AI planners described in this report are special cases of intelligent autonomous control structures. The area of intelligent autonomous control, studied from a control theorist's viewpoint, has been called the intersection or integration of AI, operations research, and control theory (Fu, 1971; DeJong, 1983). It has also been pointed out that intelligent controllers are at the top of the controller complexity scale and thus have the ability to solve increasingly difficult, poorly formulated, ill-structured problems (Saridis, 1979). *Intelligent autonomous controllers* coordinate the use of higher-level decision-making processes and conventional control techniques to control complex dynamical systems. An appropriate intelligent autonomous controller architecture and study of fundamental issues in intelligent autonomous control have recently been completed (Antsaklis and Passino, 1988). The field of intelligent control is in its infancy, yet some important ideas relevant to planning are valuable. We highlight these here.

Planning systems are hierarchical. There are three levels in the hierarchy. The lowest level is the *execution level.* At this level we find the system hardware and highly numeric-algorithmic techniques in use. In the robot example, the execution level contains the manipulator and the servomechanism that is used in the gripper. The next level up is the *coordination level.* At this level various execution-level controller actions are coordinated and supervised. It plans the actions of the low-level algorithms and hardware. Some intelligent decision making is used to perform the coordination and to interface to the highest level.

In the robot example, the coordination level would coordinate the movements of two manipulators so that they do not collide and plan their movements so that together they achieve some task. The highest level, the *organization* or *management level*, manages the systems actions and uses high-level decision-making processes and learning. It guides the actions of the coordination level and delegates duties to the various subsystems in the coordination and execution levels. In the robot example, the management level would, for instance, decide that a certain task needed two manipulators to achieve the goal efficiently. It is desirable to have hierarchical models to use with the hierarchical controller for compatibility reasons. Note that "hierarchical" as used here has different meaning from that in the section on AI planning systems.

As we go up the hierarchy, the *model abstractness* needed for the problem-solving processes increases, and as we go down, the needed *model granularity* increases. For example, differential equations are used to develop the gripper control laws, while higher-level decision-making processes might use a rule-based model of the robot's environment. Both symbolic and numeric processing and modeling are necessary.

The *time scale density* increases as we go from the management to the execution levels. This occurs because the management level has a macroscopic view of the actions that occur. It is not concerned with the details of force feedback of the manipulator but is only concerned that the object was properly moved. Consequently, the *decision rate*, or rate at which different parts of the controller take actions, decreases as we go from the execution to the management level.

CONCLUDING REMARKS

Although a foundation of fundamental concepts has been formed for AI planning systems by drawing an extensive analogy with control theoretic ideas, much work needs to be done to formalize mathematically the work presented here. At best, the results of this paper raise many questions and clarify some of the issues that may be important in quantitative studies of AI planning systems. Extensive research must be done on developing particular methods for modeling, analyzing, and designing AI planning systems. Because the results in this paper are independent of both domain and problem representation, they are applicable no matter what modeling and analysis methodology is chosen as long as the methodology provides for the study of the fundamental concepts developed here.

As it is quite fundamental to the quantitative study of AI planning systems, the *modeling* issue must be addressed first. Various questions must be answered: (1) What mathematical formalism should be used for the problem representation? (2) What is the expressive power of this formalism? That is, what class of problem domains can be modeled? (3) Does the formalism lend itself to analysis, design, and implementation? Which properties of the models will be important to study?

Second, systematic *analysis* methods must be developed so that planning system behavior can be studied quantitatively within the developed modeling framework. Before this is done, however, it will be important to determine what is important to analyze. Are there properties other than the ones developed here that need to be analyzed? It is also expected that planning methodologies that lend themselves to analysis will have to be developed. The question of what constitutes good planning system behavior must be answered. Finally, planning system *design* must be addressed. It is hoped that a systematic procedure for design is obtained—one that is similar in character to the control system design process.

REFERENCES

Adams, M., Deutsch, O., and Harrison, J. 1985. A hierarchical planner for intelligent systems. *Proc. SPIE Conf. on Applications of Artificial Intelligence,* April.

Allen, J. F., and Koomen, J. A. 1983. Planning using a temporal world model. *Proc. IJCAI* 8:741–747.

Antsaklis, P. J., and Passino, K. M. 1988. Towards intelligent autonomous control systems: Architecture and fundamental issues. *Journal of Intelligent and Robotic Systems,* Vol. 1, 1988.

Astrom, K., and Wittenmark, B. 1984. *Computer Controlled Systems.* Englewood Cliffs, N.J.: Prentice-Hall.

Barr, A., and Feigenbaum, E. A. 1981. *The Handbook of AI, Vol. 1.* Los Altos: Kaufmann.

Barr, A., and Feigenbaum, E. A. 1982. *The Handbook of AI, Vol. 2.* Los Altos: Kaufmann.

Broverman, C. A., and Croft, W. B. 1987. Reasoning about exceptions during plan execution monitoring. *Proc. IJCAI.*

Bruce, B., and Newman, D. 1978. Interacting plans. *Cogn. Sci.* 2:195–233.

Chapman, D. 1987. Planning for conjunctive goals. *Artif. Intell.* 32:333–377.

Charniak, E., and McDermott, D. 1985. *Introduction to Artificial Intelligence.* Reading, Mass.: Addison-Wesley.

Chen, C. T. 1984. *Linear System Theory and Design.* New York: Holt, Rinehart, & Winston.

Cohen, P. R., and Feigenbaum, E. A. 1982. *The Handbook of Artificial Intelligence, Vol. 3.* Los Altos: Kaufmann.

D'Azzo, J., and Houpis, C. 1981. *Linear Control System Analysis and Design.* New York: McGraw-Hill.

DeJong, K. 1983. Intelligent control: Integrating AI and control theory. *Proc. IEEE Trends and Applications, Gaithersburg, Md., May.*

Deutsch, O., et al. 1985. Heuristically-guided planning for mission control/decision support. *Proc. 1985 AIAA GNC Conf.*

Drummond, M., et al. 1987. Contingent plan structures for spacecraft. Proc. Workshop on Space Telerobotics, ed. G. Rodriquez, JPL Publ. 87-13, July.

Dudziak, M., et al. 1987. IVC: An intelligent vehicle controller with real-time strategic replanning. *Proc. IEEE International Symposium on Intelligent Control.*

Egan, G. 1986. *The Skilled Helper: A Systematic Approach to Effective Helping.* Monterey: Brooks/Cole.

Ernst, G., and Newell, A. 1969. *GPS: A Case Study in Generality and Problem Solving.* New York: Academic Press.

Fikes, R., and Nilsson, N. 1971. STRIPS: A new application of theorem proving to problem solving. *Artif. Intell.* 2:189–208.

Fikes, R., Hart, P., and Nilsson, N. 1972. Learning and executing generalized robot plans. *Artif. Intell.* 3:251–288.

Firschein, O., et al. 1986. *Artificial Intelligence for Space Station Automation.* Noyes.

Fu, K. S. 1971. Learning control systems and intelligent control systems: An intersection of artificial intelligence and automatic control. *IEEE Trans. Automatic Control,* February

Fu, K. S., Gonzalez, R. C., and Lee, C. S. G. 1987. *Robotics: Control, Sensing, Vision, and Intelligence.* New York: McGraw-Hill.

Georgeff, M. 1984. A theory of action for multiagent planning. *Proc. AAAI 84, Texas.*

Georgeff, M., and Lansky, A. 1987. Reactive reasoning and planning. *Proc. IJCAI.*

Gevarter, W. B. 1984. *Artificial Intelligence.* Noyes.

Giordana, A., and Saitta, L. 1985. Modelling production rules by means of predicate transition networks. *Inform. Sci.* 35:1-41.

Goodwin, E., and Sin, K. 1984. *Adaptive Filtering, Prediction, and Control.* Englewood Cliffs, N.J.: Prentice-Hall.

Graglia, P., and Meystel, A. 1987. Planning minimum time trajectory in the traversability space of a robot. *Proc. IEEE Int. Symp. on Intelligent Control.*

Hawker, J., and Nagel, R. 1987. World models in intelligent control systems. *Proc. IEEE Int. Symp. on Intelligent Control.*

Hayes, C. 1987. Using goal interactions to guide planning. *Proc. IJCAI.*

Hayes-Roth, B., and Hayes-Roth, F. 1979. A cognitive model of planning. *Cogn. Sci.* 3:275-310.

Hopcroft, J., and Ullman, J. 1979. *Introduction to Automata Theory, Languages and Computation.* Reading, Mass.: Addison-Wesley.

Hodgson, J. 1987. Structures for intelligent systems. *Proc. IEEE Int. Symp on Intelligent Control.*

Kalman, R., Falb, P., and Arbib, M. 1969. *Topics in Mathematical System Theory.* New York: McGraw-Hill.

Kempf, K. G. 1987. Planning and scheduling: Is there a difference? *Proc. Workshop on Space Telerobotics,* ed. G. Rodriquez, JPL Publ. 87-13, July.

Korf, R. E. 1987. Planning as search: A quantitative approach. *Artif. Intell.* 33:65-88.

McDermott, D. 1978. Planning and acting. *Cogn. Sci.* 2:71-109.

McDermott, D. 1982. A temporal logic for reasoning about processes and plans. *Cogn. Sci.* 6:101-155.

McDermott, D. 1985. Reasoning about plans. In *Formal Theories of the Commonsense World,* eds. Hobbs, J. R., and Moore R. C. Ablex.

Miller, R., and Michel, A. 1982. *Ordinary Differential Equations.* New York: Academic Press.

Nilsson, N. J. 1980. *Principles of Artificial Intelligence.* Palo Alto: Tioga.

Passino, K. M., and Antsaklis, P. J. 1988. Planning via heuristic search in a Petri net framework. *Proc. Third IEEE Int. Symp. on Intelligent Control,* August.

Passino, K. M., and Antsaklis, P. J. 1988. Artificial intelligence planning problems in a Petri net framework. *Proc. Ameircan Control Conf.,* pp. 626-631, June.

Pearl, J. 1984. *Heuristics: Intelligent Search Strategies for Computer Problem Solving.* Reading, Mass.: Addison-Wesley.

Rosenschein, S. 1981. Plan synthesis: A logical perspective. *Proc. Int. Joint Conf. on Artificial Intelligence,* pp. 331-337.

Sacerdoti, E. D. 1973. Planning in a Hierarchy of Abstraction Spaces. SRI Tech Note 78, Stanford University.

Sacerdoti, E. D. 1975. The Nonlinear Nature of Plans. SRI Tech Note 101, Stanford University.

Saridis, G. N. 1979. Toward the realization of intelligent controls. *Proc. IEEE* 67(8):August.

Shapiro, S. (ed). 1987. *Encyclopedia of Artificial Intelligence.* New York: Wiley.

Stefik, M. 1981. Planning with constraints. *Artif. Intell.* 16:111-140.

Stefik, M. 1981. Planning and meta-planning. *Artif. Intell.* 16:141-170.

Stephanopoulos, G., et al. 1987. An Intelligent System for Planning Plant-Wide Process Control Strategies. MIT Dept. of Chemical Engineering Report No. LISPE-87-016, March.

Tanimoto, S. L. 1987. *The Elements of Artificial Intelligence.* Rockville, Md.: Computer Science Press.

Tate, A. 1977. Generating project networks. *Proc. IJCAI,* 5:888-893.

Tate, A. 1985. A review of knowledge-based planning techniques, in *Expert Systems 85,* ed. M. Merry. London: Cambridge University Press.

Vere, S. 1983. Planning in time: Windows and durations for activities and goals. *IEEE Trans. Pattern Analy. Machine Intell.* PAMI-5(3):May.

Waldinger, R. 1975. Achieving Several Goals Simultaneously. SRI Tech Note 107, Stanford University.

Warren, D. 1974. WARPLAN: A system for generating plans. Memo No. 76, Univ. of Edinburgh, Edinburgh, June.

Wilensky, R. 1981. Meta-planning: Representing and using knowledge about planning in problem solving and natural language understanding. *Cogn. Sci.* 5:197–233.

Wilensky, R. 1983. *Planning and Understanding*. Reading, Mass.: Addison-Welsey.

Wilkins, D. 1983. Representation in a domain-independent planner. *Proc. IJCAI.*

Wilkins, D. 1984. Domain-independent planning: Representation and plan generation. *Artif. Intell.* 22:269–301.

Winslett, M. 1987. Validating generalized plans in the presence of incomplete information. *Proc. IJCAI.*

Received July 25, 1988

Request reprints from K. M. Passino.