**Appears in Working Notes of the AAAI Spring Symposium on Computational Models for Mixed-initiative Interaction**
**Stanford, CA, March 1997**

# Integrated Task Representation for Indirect Interaction Position Paper

**Robert P. Goldman** and **Stephanie Guerlain** and **Christopher Miller** and **David J. Musliner**
Honeywell Technology Center
3660 Technology Drive
Minneapolis, MN 55408
{goldman, guerlain, cmiller, musliner}@htc.honeywell.com

**Abstract**

We are developing mixed-initiative systems that provide users with flexible control over evolving processes. For such systems to be feasible, both users and automated agents must be able to operate at multiple levels of abstraction. We are designing mixed initiative systems organized around *shared* task models to achieve this multi-level task sharing. These task models will allow both human and automated agents to view the ongoing state, to control tasks and to project future requirements. Our task models will have semantics, allowing a suitable interpreter to directly execute tasks for users. We discuss this approach and our initial implementation of a system that assists users in proposal preparation.

## Introduction

We are an interdisciplinary research group at the Honeywell Technology Center (HTC), chartered to develop a general-purpose task representation scheme for mixed initiative systems. The knowledge encapsulated in this general-purpose task representation scheme will provide the foundation for applications in a number of areas, including crisis-management for industrial control systems, data mining and scheduling systems. The task representation scheme will support both (1) the inferences needed for the system to take action to achieve goals held jointly with users and (2) the design and population of user interfaces to the system.

Our working hypothesis is that mixed initiative systems must permit both users and automation to operate at *multiple levels of abstraction*. Systems in which the user can only delegate whole tasks are too inflexible for the applications that interest us. Systems in which the user can only act by direct manipulation overload their users.

We believe that the best way to achieve multiple-level intermingling of activities is to organize mixed initiative systems around *shared* task models. These task models should be understandable to both human and automated agents. They will capture the various tasks required by the system and user at multiple levels of abstraction, allowing both automation and user to view the ongoing state and future requirements of problem solving. We call systems designed in this way "indirect interaction" systems, because they permit users to act indirectly, delegating tasks to automation, but also to interact relatively directly, and to constrain the actions taken by the automation.

## Why Mixed Initiative Systems?

In this section we discuss the reasons our group is pursuing mixed initiative systems. One major motivating concern is the need to combine human and AI expertise in complex systems. Another, particularly relevant to Honeywell's controls business, is that automated systems often need to affect systems that can only be acted on or observed by people directly. Sociological and organizational reasons often dictate only partial automation even where full automation might be possible.

One reason for the adoption of mixed initiative systems is the limitations of knowledge acquisition in automating complex tasks. In many of the domains that concern us, there are aspects of human domain knowledge that we cannot hope to completely capture. One application area where we have confronted this issue is scheduling. Putting aside questions of computational practicality, scheduling is fairly a well-understood computational task: make a set of ordering decisions such that a set of actions will be carried out in a way that optimizes an objective function. We can point a search engine at the problem and wait for the answer. In actual manufacturing problems, however, we have rarely been able to formulate a tidy objective function. For example, we might wish, in general, to schedule production of a particular substance towards the end of a production period (because this task is most tolerant of residues left by earlier processes). However, if the plant manager gets an urgent order for this substance

from a very valuable customer, the ordering must be changed. There may be no time to formulate a suitable new objective function, and it would not be worthwhile to do so for every forseeable special situation.

In other cases, there are actions that cannot be performed by automation. For example, in many refineries there are valves that are not actuated by the control system and must be adjusted by field operatives with wrenches. We would like to build a mixed initiative system to support proposal preparation. The actual writing of the text cannot be automated of course, but many support tasks (e.g., construction of Gantt charts from milestone lists) can be at least partially automated.

Finally, there are cases which cannot be automated for reasons that are not technological. For example, researchers in medical informatics have stated that even were it possible to automate diagnosis, diagnostic systems would not be accepted, because diagnosis is one of the most interesting tasks physicians perform. We may not want to fully automate other tasks because we do not want to make humans dependent on automation. For example, often operators of point-of-sale devices are so dependent on mechanically-performed arithmetic that they are unable to "sanity check" these computations. We would not like workers in safety-critical professions, e.g. pilots, to allow their skills to erode in this way.

One question worth asking is "Why not leave tasks like this completely free of automation?" In some cases, automation can fill gaps in human cognitive capabilities. Determining the presence of antibodies in blood samples is a task that people perform poorly, but that can be significantly aided by a mixed-initiative decision support system (Guerlain 1995). Scheduling often requires consideration of more possible orderings than can be evaluated by people. In other cases, forgoing automation is no longer even an option. For example, preparation of the complex documentation and accounting required to bid for a large contract is simply not feasible without tools like spreadsheets.

## Our Working Hypotheses

We have adopted three hypotheses to direct our design experiments.[1] First, mixed initiative systems should permit humans and automation to work at multiple levels of detail. Second, these efforts should be organized around a shared task model. Third, this task model will provide the backbone of the user interfaces of the systems we build.

[1] As yet, these are experiments only in the informal sense of the word.

Mixed initiative systems will be most flexible and most capable if both humans and automation are free to work at multiple levels of detail (see Figure 1). This trait sets our work apart from, for example, the kind of interfaces currently provided by Microsoft products. Current Microsoft products provide "Wizards" that allow users to delegate tasks to the automation. However, one cannot work *with* the wizard to achieve a goal: the delegation is effectively all or none. Complex AI systems typically cannot tolerate human intervention in details of their solutions. Further, human users, when delegating a (part of) a task to an automated actor, should be able to impose constraints on the automated actor's actions or propose a partial solution.

The joint efforts of humans and automation are to be organized around a *shared* task model that is understandable and manipulable by both users and automation. For example, the user may select a displayed subtask, thereby saying "Now let's talk about this" and from there can do things like say "What needs to be done to accomplish this?" "How is progress on this?" "I'm going to work on this now" or "I want you to work on this now," etc. If the user of the system acts through a shared task model, that will permit his/her actions and decisions to be incorporated into the network of constraints that the system uses to guide its own actions. That will prevent the confusion that currently occurs when humans intervene to change low-level aspects of automation actions. Conversely, the human-understandable task model will help the user comprehend actions taken autonomously by the system and relate them to jointly-held goals.

The shared task model is to be the keystone of the user interface of the mixed initiative systems we develop. Currently, a major expense in systems development is the development of user interfaces. Furthermore, the construction of direct manipulation interfaces requires us to develop direct manipulation metaphors, which are often strained by complex systems (Gentner & Nielson 1996). If we can develop a satisfactory cross-domain task modeling formalism, and a user interface approach that operates on this formalism, then we will be able to amortize the costs of user interface development for even complex software systems.

In our initial work, we are making three simplifying assumptions. First, we are focusing our efforts on systems with a single top-level goal. Our initial experimental system is an associate for proposal managers at HTC, to aid them in the process of preparing a single proposal. In this way we simplify the problem of inferring which goal the user is working on at any given time. Second, the systems we build will be

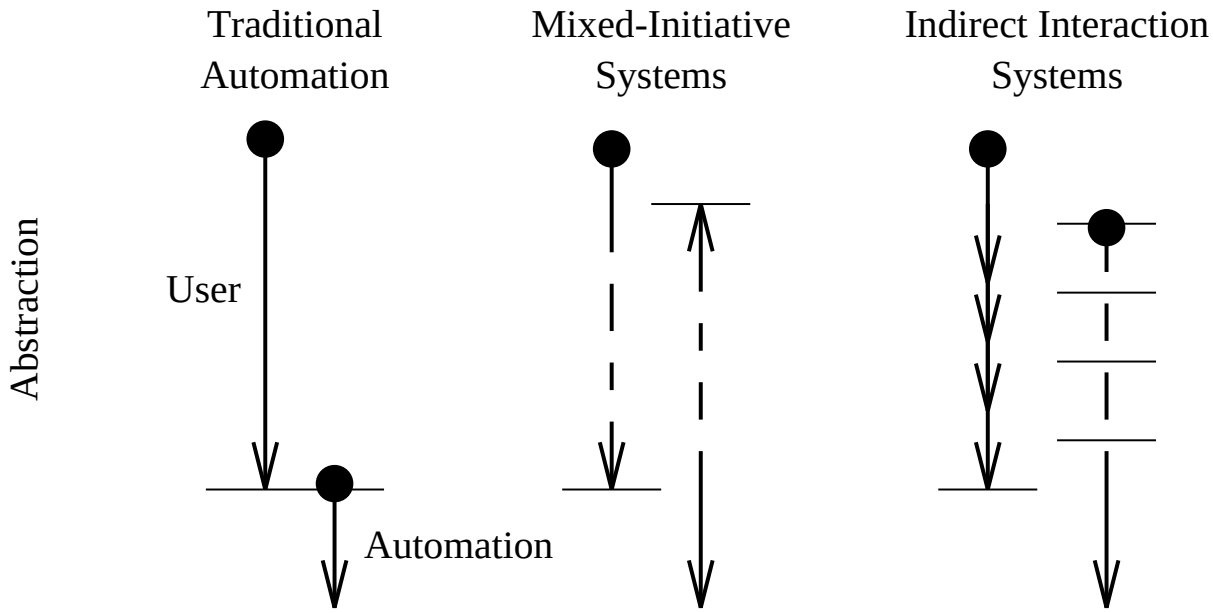


Figure 1: A spectrum of interaction.

environments within which users will work to achieve these goals. The user will always be working through a main system, rather than individually invoking multiple tools (the Unix model). This makes it easier for us to avoid situations where the user goes "behind the back" of the system. Finally, we are initially attempting only single-user systems, although many of the domains involve multiple human agents.

## Scenario

We are starting by developing a system for assisting in the preparation of contract proposals. This is a good test application for us, because it provides us with the opportunity to get user experiences and provides a useful corrective — if we fail to provide a benefit to these users over and above existing automation, they will not accept our system.

We are proceeding by iteratively developing demonstration scenarios with the task representation and inference algorithms necessary to support these scenarios. As a result, we are intertwining the development of our representation scheme with representation of sample task domains. In this section we present a sample interaction with our indirect interaction proposal associate.

On startup, the system will display the skeletal plan (see Figure 2).[2] Of the tasks in the initial, shallow, expansion, only the "Collect Proposal Information" task is marked as currently executable.

[2]This is only a diagram of the task model, not a sample of the user interface!

**Configuration** When the user selects "Collect Proposal Information," the display shifts to show the subplan for this task, illustrated in Figure 3. Once again, only one choice is executable, "Collect background information."

**Collecting Background Information** The assistant system will collect background information from the user. In particular:

- Formal and informal program titles.
- Customer (select from list or add new ones)
- Funding available to prepare the proposal
- Proposer's information packet (PIP) file location
- Deadline

When the user specifies the PIP file location, the Proposal Assistant recognizes an opportunity to be of assistance. The Proposal Assistant will invoke a local tool, the "dePIPer," that attempts to parse an ARPA PIP document and extract an initial outline for the proposal document.

Another opportunity will be spotted by the Proposal Assistant when the user specifies the deadline. This information will be propagated throughout the plan to generate bounds on completion times of the various goals.

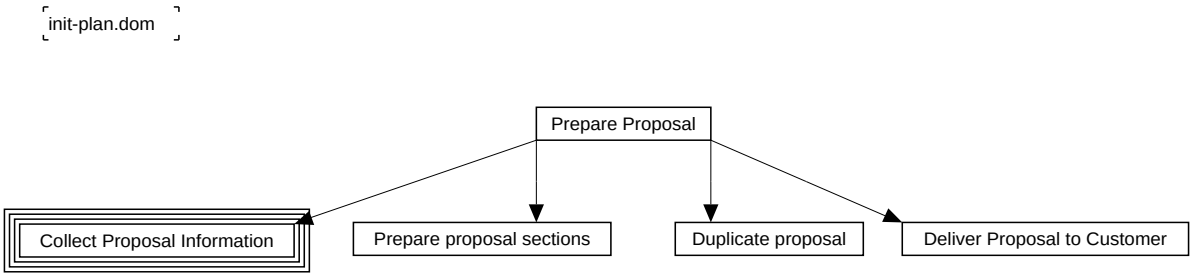


Figure 2: The initial plan. The "haloed" plan step is executable.

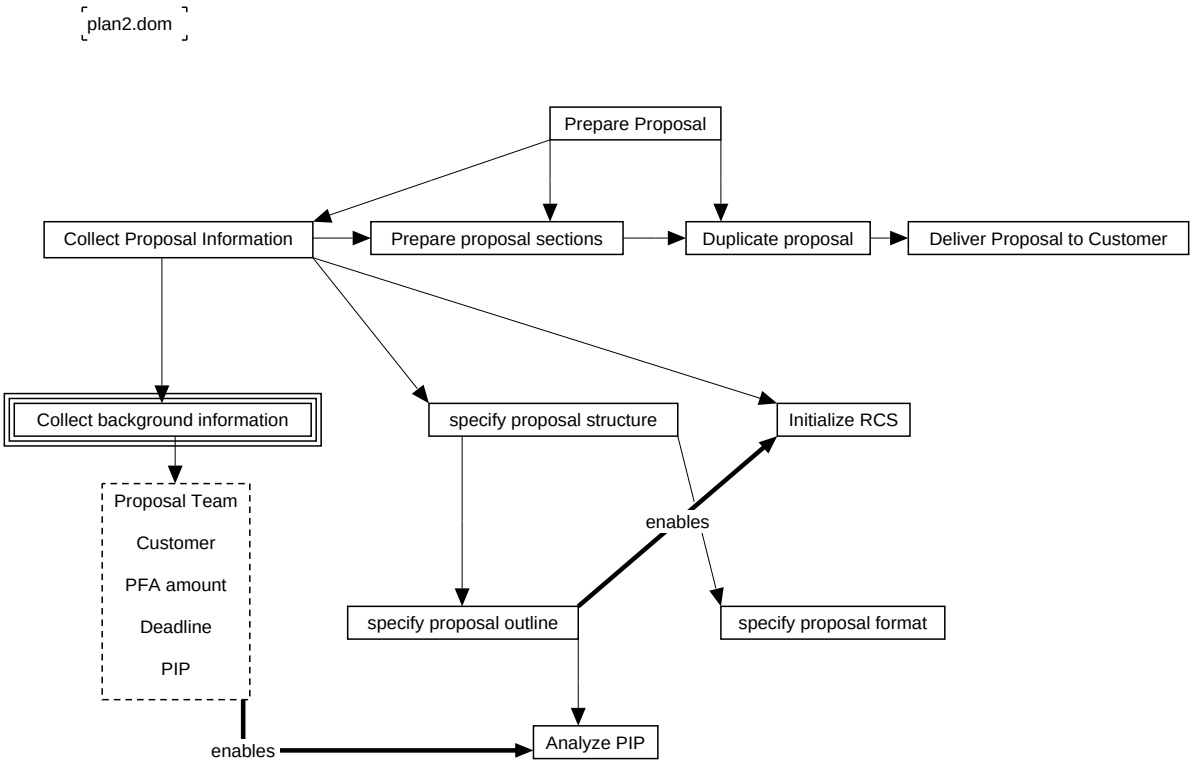


Figure 3: The plan after an initial move by the user.

**Specify Proposal Structure** When the user indicates that s/he is ready to specify the proposal outline, the Proposal Assistant brings up a display that contains the results of the attempt to parse the PIP. Note: we are assuming that there was an identifiable, machine-readable PIP in this scenario. If there were no such thing, then the system would simply display a blank outline or a set of defaults derived from prior proposals to similar customers).

When the sections have been defined, the Proposal Assistant notes another opportunity to act in support of the user. The Proposal Assistant determines what document preparation method will be used, by examining defaults and/or by interrogating users.[3] Noticing that the proposal team is working on engineering workstations, the Proposal Assistant creates a directory that will act as a repository for all of the proposal files. The Proposal Assistant also applies templates to generate boilerplate text for the various sections, and initializes the revision control system (RCS) for these files. The Proposal Assistant also initializes the main proposal file with information about the formatting and page layout.

This is a simple scenario illustrating an initial interaction with the Proposal Assistant system. We are currently implementing an initial version of the Proposal Assistant in Java.

## Task representation

### Desiderata

The desiderata for the task representation arise out of the needs of users of the indirect interaction system. The needs we have identified are:

- User can direct task plans. This includes
  - Choose alternative means to achieve goals: choose alternative expansions for a task;
  - Edit parameters to tasks.
- System provides user with feedback about plan quality.
- User must control system initiative.
- User must be able to explore scenarios.
- System must be able to assimilate actual with predicted data.
- User must be able to reverse system actions (characteristics of the domain permitting).

[3]For this discussion, we assume text editors and a text formatter (e.g., LaTeX). Similar functionality will be provided through a WYSIWYG system such as Word or Framemaker.

To provide these facilities, the system must be able to:

- track progress toward the top-level goal (**monitor**);
- look ahead and anticipate future events and threats to the goal's achievement (**project**);
- take initiative to execute sub-tasks on the user's behalf (**execute**).

In addition to the inferential desiderata, the task model must support comprehensible browsing and interaction.

**Monitoring** The system must be able to determine the status of various sub-tasks in the task model. Monitoring is a necessary prerequisate for the other inferential processes of projection and execution. Further steps in the exection of the task cannot be foreseen unless the system can identify what has already been done and the system cannot take initiative to act for the user if it cannot determine that a task remains to be done.

While we are interested in general purpose plan-recognition (Charniak & Goldman 1993; Kautz 1991; Carberry 1990; Schmidt, Sridharan, & Goodson 1978), we do not believe that plan-recognition algorithms yet handle sufficiently expressive representations or are sufficiently efficient and robust for our purposes. Accordingly, we are designing our systems to minimize the amount of plan recognition necessary. We hope to do this in two ways: first, by making our mixed initiative interface a "home base," from which the user will depart to carry out actions and to which the user will return (see discussion of simplifying assumptions in Section ), we hope to maximize the extent to which we can "just see," what tasks the user is performing. Second, when necessary we will fall back on asking the user directly.

Note that both of these tactics reinforce our desire for a task representation that is understandable to the user.

**Projection** In several of our domains of interest, resource and time management are of great interest to our prospective users. For example, in assembling a proposal, primary concerns are to establish and track a schedule and not to overspend the budget allocated to proposal preparation. Even in cases where resource management is of less interest (e.g., knowledge-assisted data mining), we need to project the effect of early actions in order to determine their effect on possible actions in the future.

**Execution** There are two primary means of execution for the systems we have in mind: first, automated execution of tasks by dispatch to other programs; second, explicit transfer of responsibility for a task to the user.

In order to be able to support execution, the system needs to make two primary kinds of inference: first, the system needs to be able to detect when a task the system is able to perform is ready for execution; second, the system needs to be able to detect when a task must be performed but the system itself is unable to do so. These inferences must take into account user-expressed preferences about system initiative, in order to afford the user control.

## Preliminary design

We are building our task modeling scheme on existing planning languages. In particular, we are going to use concepts from hierarchical planning. The notations used by "first principles" planning systems are not suitable for our purposes. First principles planning systems require modeling of domain mechanics in too great detail. E.g., it is not possible to require one action to follow another without an explicit producer-consumer relation between them. Further, these notations provide no way of capturing goal-subgoal and abstraction relationships.

Our task models are based on the Procedural Reasoning System (PRS) (Georgeff & Lansky 1986; Georgeff & Ingrand 1990; Ingrand, Georgeff, & Rao 1992). PRS provides a high-level language for describing agent behaviors and an interpreter to execute them. The PRS architecture, illustrated in Figure 4, performs somewhat like an enhanced production system.

Knowledge about how to accomplish given goals or react to certain situations is represented in PRS by declarative procedure specifications called *Knowledge Areas* (KAs). Each KA is triggered when its context is satisfied by current conditions. This encapsulation of procedures and context-dependent reactions allows both the modular structure of rule-based systems and the compound actions, subgoaling, and metalevel expressions of more powerful task representations (e.g. RAPS (Firby 1987)).

The *Intention Structure* (IS) contains all those tasks the system has chosen for execution, either immediately or at some later time, and is under full introspective control of KAs. Thus rather than simply executing one KA after another, at any given moment the IS may contain a number of intentions, some of which may be suspended or deferred, some of which may be waiting for conditions to hold prior to activation, and some of which may be metalevel intentions describing how to select from alternative future courses of action.

The following PRS characteristics appear especially useful as an enabling technology for mixed initiative tools and agents:

- The IS supports the identification of executable plans, necessary for the system to take initiative to achieve goals for the user.
- The interpreter will provide an initial framework for task execution.
- PRS provides more complex patterns of action than conventional planning representations, including looping.
- Its **procedural plan representation**, which is consistent with the form of many existing domain knowledge bases (e.g., standard operating procedures). Furthermore, the hierarchical, subgoaling nature of the procedural representation allows PRS to combine pieces of procedures in novel ways, which is important for flexible plan execution and goal refinement.
- Its ability to pursue parallel **goal-directed** tasks while at the same time being **responsive** to changing patterns of events in bounded time. The conventional planning assumption of a single-tasking agent is not sufficient for mixed initiative systems.
- Its ability to construct and act on **partial** (rather than complete) **procedures**.
- Its knowledge representation assumptions, which encourage **incremental refinement** of the plan (procedure) library, an enormous advantage for developing large-scale applications.

We need to augment PRS to handle

**temporal projection** For many applications the "schedule view" is crucial. We will support this by adding temporal information to the KAs and by providing meta-level temporal constraint propagation.

**feasibility inference** We need to be able to rapidly identify when choices made by the user or automation will threaten the successful completion of the plan. We will support this by special-purpose inference that detects when plan completion is impossible.

**user interface needs** We will augment the model with information about its presentation and about ways to gather appropriate information (e.g., parameter bindings) from the user.

## Summary

We are working to develop a general-purpose task representation scheme for mixed initiative systems. In this

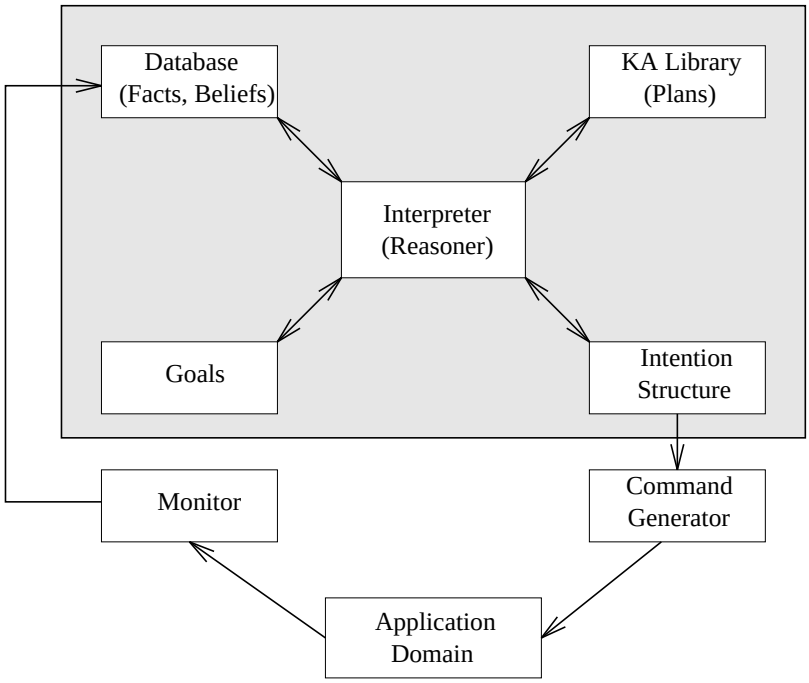


Figure 4: The PRS architecture (Georgeff & Ingrand 1990).

paper we have discussed the role of such a scheme in a class of mixed initiative systems we have called "indirect interaction" systems. These are systems in which both users and automation operate at multiple levels of abstraction. We have presented desiderata for the representation scheme and have discussed our initial steps in its development.

## References


Carberry, S. 1990. Incorporating default inferences into plan recognition. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, 471–478. Cambridge, MA: MIT Press.

Charniak, E., and Goldman, R. P. 1993. A Bayesian model of plan recognition. *Artificial Intelligence* 64(1):53–79.

Firby, R. J. 1987. An investigation in reactive planning in complex domains. In *Proceedings AAAI-87*, 196–201.

Gentner, D., and Nielson, J. 1996. The anti-mac interface. *Communications of the ACM* 39(8):70–82.

Georgeff, M. P., and Ingrand, F. F. 1990. Real-time reasoning: The monitoring and control of spacecraft systems. In *Proceedings of the Sixth Conference on Artificial Intelligence Application*, 198–204.

Georgeff, M., and Lansky, A. 1986. Procedural knowledge. *IEEE Special Issue on Knowledge Representation* 74:1383–1398.

Guerlain, S. 1995. *Critiquing as a Design Strategy for Engineering Successful Cooperative Problem Solving Systems*. Ph.D. Dissertation, The Ohio State University, Columbus, OH.

Ingrand, F.; Georgeff, M.; and Rao, A. 1992. An architecture for real-time reasoning and system control. *IEEE Expert* 7:6:34–44.

Kautz, H. A. 1991. A Formal theory of plan recognition and its implementation. In Allen, J. F.; Kautz, H. A.; Pelavin, R. N.; and Tenenberg, J. D., eds., *Reasoning About Plans*. Los Altos, CA: Morgan Kaufmann.

Schmidt, C.; Sridharan, N.; and Goodson, J. 1978. The plan recognition problem: an intersection of psychology and artificial intelligence. *Artificial Intelligence* 11:45–83.