N90-29036

Plan Recognition for Space Telerobotics

Bradley A. Goodman[1]
BBN Systems and Technologies Corporation
10 Moulton Street
Cambridge, MA 02138

Diane J. Litman
AT&T Bell Laboratories
600 Mountain Avenue
Murray Hill, NJ 07974

Abstract

Current research on space telerobots has largely focussed on two problem areas: executing remotely controlled actions (the "tele" part of telerobotics) or planning to execute them (the "robot" part). This work (see [Rodriguez, ed. 87]) has largely ignored one of the key aspects of telerobots: the interaction between the machine and its operator. For this interaction to be felicitous, the machine must successfully understand what the operator is trying to accomplish with particular remote-controlled actions. Only with the understanding of the operator's *purpose* for performing these actions can the robot intelligently assist the operator, perhaps by warning of possible errors or taking over part of the task. We motivate the need for such an understanding in the telerobotics domain and describe an intelligent interface we are developing in the chemical process design domain that addresses the same issues.

1 Telerobotic Interfaces

Space telerobots are being developed to increase astronaut's productivity by allowing them to remotely perform cost-effective assembly, servicing and maintenance of equipment in outer space [Bejczy 87, Jenkins 87]. The telerobot extends the operator's abilities into the hostile environment of outer space while the operator stays in a non-hostile location. While the long-term goal is for autonomous telerobots that can perform their duties with little human intervention, current research is directed towards an active role of the human operator in the guidance of the telerobot. The operator monitors closely the activity of the robot through video and sensor information and issues instructions directly, through manipulator controls, and indirectly through prestored task plan data.

One problem with such a paradigm is the complexity of the required telerobotic interface. The operator must contend with limited resources and environmental issues such as imprecise video or sensor data, communication time delays, and the parallel coordination of numerous telerobot resources (e.g., the control of a mechanical arm and its numerous degrees of freedom [Bejczy 87]). An interface that can smoothly handle such issues in real time is beyond current technology. An operator instead must break the task down into reasonable self-contained pieces and separately execute actions to achieve each piece. For example, consider a telerobot having two arms with cameras mounted on each hand. Now suppose the operator is manipulating one arm to perform a task but would also like to move the second arm so that its camera can provide a view of the first arm from the side. An operator might find it quite awkward to manipulate both arms and cameras at the same time. If, however, the telerobot system could recognize the operator's goals and, seeing that the second arm was idle, infer that a view from it would be helpful to the operator, it could automatically move the second arm to track the first arm until the operator attempted to manually control it.

We propose that an intelligent telerobotics interface could address such problems and provide the operator with a more robust and habitable environment for controlling the robot. Development of such an interface requires adding another layer to the interface control mechanism that attempts to surmise the purpose of the operator's actions and, from those inferences, determine (1) more precise specifications for fine-tuning an operator's action and (2) helpful actions that the telerobotic system can independently perform to assist the operator and the telerobot in the performance of the task. Such an interface requires a mechanism, which we call a "plan recognizer," to infer the operator's intentions, and a response planner, to determine from the operator's intentions, the operator's actual actions, the telerobotic system's knowledge base, and sensor data appropriate ways of responding (e.g., adjusting an

[1]Partial support for this research was provided by the National Science Foundation under grant IRI-8701874.

operator's commands, by filling in more detailed information, or performing on its own initiative non-catastrophic actions). Figure 1 below illustrates the interface we envision. Notice that the plan recognizer has been added to the typical telerobot interface between the operator and the response planner where it can deduce an operator's goals and fine-tune an operator's commands. In this paper we concentrate only on plan recognition; others (e.g., [Dean and Brody 87], [Drummond, Currie, and Tate 87], [Durfee and Lesser 87], [Georgeff, Lansky, and Schoppers 87], and [Wilkins 87]) are considering response planning. Below we describe plan recognition and some of its recent instantiations and then we lay out a more robust formulation, constructive plan recognition, that we are attempting to employ. We describe as a case study how plan recognition has improved a chemical process design interface that we have built. Finally, we close with a brief discussion on how a similar interface could benefit telerobotics.

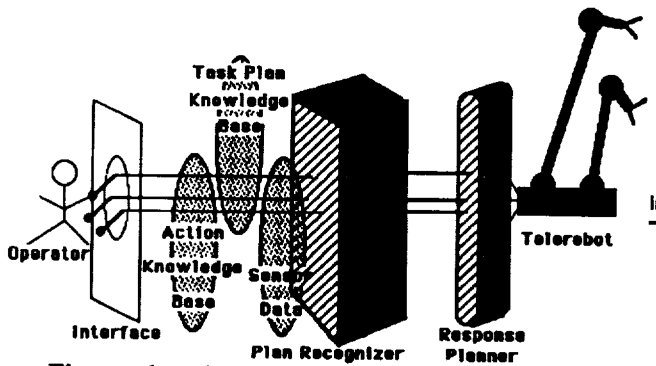


Figure 1: A proposed telerobot interface

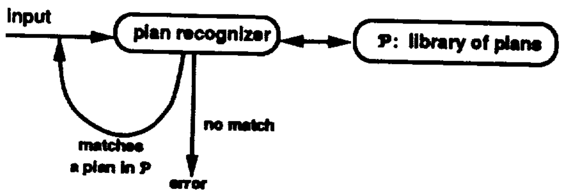


Figure 2: Standard plan recognition

2 Plan Recognition

2.1 Standard Plan Recognition

In the artificial intelligence literature, the issues mentioned in the previous section are addressed by work on plan recognition (e.g., [Allen 83], [Carberry 85], [Carver, Lesser, and McCue 84], [Genesereth 79], [Kautz 87], [Litman and Allen 87], [Pollack 86], [Schank and Reiger 74], [Schmidt, Sridharan, and Goodson 78], [Sidner 85]). Plan recognition is the process of inferring a user's intentions from his or her utterances or commands. In other words, since people don't normally maneuver in the world haphazardly and they usually have a purpose in mind, one wants to recognize the plan that underlies their actions A plan recognition component attempts to understand a sequence of observed input actions as accomplishing some high-level goal. Although these techniques have been largely used to understand the intent underlying linguistic actions (spoken or written utterances), they also lay a solid foundation for analyzing the purpose behind non-linguistic actions, such as teleoperated manipulations.

The first plan recognizers grew out of research on "scripts" [Schank and Reiger 74] for modelling stories or conversations. These early predecessors to plan recognizers were useful only if nothing out of the ordinary occurred. A richer formalism is necessary to infer the proper connections when an unusual or unexpected event occurs. Plan-based models have been employed to do just that. A plan-based system attempts to construct an explanation for the observations. Plan recognition, thus, becomes the process of searching the space of possible plans for an explanation. Since such a search is not tractable in any but the smallest domains, simplifying assumptions must be added to the plan recognition problem [Allen 87]. The typical assumption, called the "completeness assumption," has been to assume complete knowledge of all possible plans.

Several strategies have recently been employed to recognize and track a user's plan (e.g., Allen 83, Sidner 85, Carberry 85, Kautz 87). These plan recognition strategies typically use a plan library: a description of typical plans and actions that might occur in a particular domain. On the basis of the library and description of an action, plan recognition algorithms produce (possibly partial) descriptions of the corresponding plan. Allen's algorithm (Allen 83) uses a heuristic search to choose a preferred plan, while others (such as Sidner's (85) or Kautz's (87)) allow the recognition to occur incrementally. Sidner's and Kautz's plan recognition techniques operate in a "syntactic" framework. They perform a process similar to parsing by attempting to fit observed user actions into an expected user plan (as defined by a joint library of plans or goals). A successful parse of a user's observed actions, thus, entails finding an exactly matching prestored plan in the plan library. This strategy works given the assumption that the library is complete.

There are weaknesses to this approach. First, the observed user's actions may not match any of the plans in the plan library. Current algorithms fail in such a situation (as shown in Figure 2). Second, the parsing algorithms currently employed are slow making them less appropriate for real-time actions.

Our own research has been concerned with extending plan recognition to non-linguistic domains.[2] We have had, among others, two concerns relevant to space telerobots: (1) supporting real-time plan recognition, and (2) allowing for the recognition of novel plans.

2.2 Extended Plan Recognition for Space Telerobotics

The telerobot domain does not fit the current plan recognition paradigm described above. All possible plans of the operator or tasks the telerobot can achieve cannot be prestored ahead of time. Conditions in which tasks are undertaken can vary greatly and thus cannot be prepared for ahead of time; unexpected contingencies can occur. Hence user plans are often novel, created on the fly, or can have (unanticipated) errors in them. Given the hostile environment in which telerobots are intended to work, plan recognition must occur promptly in near real time before conditions change drastically. We discuss below a proposal for extending plan recognition so that it can function in a helpful manner in a telerobot domain. We describe a new view of plan recognition to allow it to respond to novel situations. While we do not discuss how to formulate plan recognition to occur in real time, we anticipate that a formulation based on chart parsing [Ross and Lewis 87, Vilain 88] could provide us with the necessary speed and flexibility.

2.2.1 Constructive Plan Recognition

Our proposal for constructive plan recognition is motivated by some early research in plan recognition. We describe that work briefly in the next section.

Motivating work

Plan recognition has primarily been investigated in the past at three different levels: formal, implementational, and psychological. Recently there have been numerous interesting theoretical results in the formal studies of plan recognition (Cohen and Levesque 85, Kautz 87), just as there have been interesting results about grammars independent of any particular grammar. We cannot, however, simply transfer the plan recognition algorithms developed from the formal studies and use them as implementations without first considering the domain to which they are to be applied. Instead, we propose looking further back to the early pioneering work in plan recognition (i.e., Schmidt, Sridharan, and Goodson 78, Genesereth 79, and Allen 83). Those works typically had particular applications in mind (e.g., Allen (83) wanted to provide helpful responses in a natural language understanding system). They also were not governed by the completeness assumption on the plan library. The analysis by Schmidt et al. is heavily motivated by psychological modelling and uses causality instead of (implicitly) prestored "legal" action sequences (as in the "syntactic" approaches to plan recognition) to tie together observed actions. After each observed action, a small number of alternative plan hypotheses are generated and used along with knowledge about the domain to generate expectations concerning what actions to expect next. Should the expectations not be satisfied, a revision process is employed to reformulate the current hypothesis. Allen developed a plan inference system that also chains together actions via decompositions and effects. His algorithm incorporates a set of plan recognition rules and a heuristic control strategy. The rules determine which inferences are possible. They define a valid chain of inferences from observed actions to plausible goals, while the control strategy considers the likelihood of these inferences in order to commit to a particular plan immediately. Allen's algorithm, as formulated, is intended to handle only single observations. Finally, Genesereth's plan recognition strategy draws heavily on a prestored model of the typical user to detect mistakes and misconceptions by the user. The procedure reconstructs the plan based on underlying beliefs in the user model. It suggests partial parsings of the user's actions and uses the user model to filter those suggestions. Thus, it too differs from the syntactic approaches by using knowledge about the user that is outside the plan library. As in these early approaches, we propose to *construct* valid plans from observed action sequences using first principles. We briefly describe our constructive view of plan recognition (CPR) in the section that follows.

[2]In particular, mouse-oriented interfaces to a chemical plant design system and a naval information database.

Constructive Recognition

In [Goodman and Litman 89], we propose a new formulation of plan recognition called "constructive plan recognition" that can address some of the problems outlined above. We briefly describe that formulation below.

Many of the implementations of plan recognition algorithms that we mentioned earlier have never been embedded inside a complete working application. As a result, crucial issues of robustness, reliability and inherent limitations remain. In particular, most current algorithms make the incorrect assumption that valid and complete plan knowledge is specified and shared by all agents, and they do not consider the fact that users often make mistakes or get sidetracked. Our research attempts to rectify these deficiencies by building on the early more semantic-based work in plan recognition described in the last section.

We wish to remove the assumption that the system's knowledge of the user's plan is complete. Otherwise, action sequences that differ from those stored in the system's plan library, even if only minimally, cannot be recognized. Dropping the completeness assumption is especially necessary in contexts where the user is often creating new plans. If we remove this assumption, however, our plan recognition algorithm cannot be limited to matching into a predefined set of expected plans. It must be more *constructive* in nature and attempt to fill in potential knowledge gaps when presented with a novel plan. It must dynamically construct from the incomplete knowledge new ways of performing high level actions, if such actions can be seen as purposeful. One goal of our work is, thus, to identify the relationship *purposeful* so that our plan recognizer can efficiently conclude the proper inferences.

Many existing plan recognizers define a purposeful action sequence "syntactically" (e.g., Huff and Lesser 82, Carver et al. (84), Sidner 85, Kautz 87). They perform a process similar to parsing by attempting to fit observed user actions into an expected user plan as (implicitly) defined by a complete plan library, not unlike a language that defines all possible sentences. CPR attempts to extend this parsing analogy by developing a "cascaded" parsing algorithm where syntax and semantics work hand in hand (Woods 80). However, while a traditional cascaded parser uses semantics to verify or eliminate existing syntactic choices, we are using semantics to *construct* additions to an incomplete syntactic language.[3] That is, if CPR cannot parse an observed action sequence, it attempts to determine whether or not the sequence could be part of the plan language (i.e., is "purposeful") by applying semantic information to piece together actions. Purposeful actions must be able to be seen as being part of an action sequence on the way towards achieving some goal. In the case of novel sequences, however, that goal is not necessarily known. We propose to define a metric for "purposefulness." Suppose we observe two actions: action A followed by action B. "Purposefulness by entailment" occurs if the effects of A make the preconditions of B satisfied. If we can construct intervening actions between A and B to connect them, then we have "purposefulness by construction." Finally, we have "purposefulness by example" if we can contrast the partial sequence "A B" that was observed against a known "purposeful" action sequence (i.e., finding a partial match).

Similarly to the early work mentioned in the last section, CPR can use plan generation techniques to determine whether novel sequences of actions are potentially purposeful. For example, CPR incrementally examines (using local backward and forward chaining) the effects and preconditions of actions in an action sequence and the propagation of those effects to determine whether or not they fit together well. In other words, an action sequence is (potentially) purposeful if it could have been generated by a planner from first principles.[4] When effects of earlier actions neither violate nor achieve preconditions of later actions, CPR can say nothing definitive about the causal structure of the plan. Instead, the desired interactions are viewed as expectations that need to be satisfied before the plan specification is through in order for the action sequence to remain valid. Because CPR uses bottom up techniques to incrementally *verify* the coherence of action sequences, the search explosion involved in earlier bottom up approaches to plan recognition (where search spaces were generated to *propose* action sequences) can be controlled.

Besides plan recognition by plan generation techniques, we are also investigating the recognition of plans by plan adaptation. What we learn from failed plan parses from the incomplete library, along with techniques of case-based and adaptive reasoning (e.g., Alterman 86, Broverman and Croft 87, Hammond 87, Kolodner et al. 85) can be used to determine if novel actions and goals form reasonable plans. In other words, we attempt to use the incomplete plan library to dynamically construct new ways of performing high-level actions. Finally, we only detect an errorful plan in the case where our semantic plan recognition techniques fail. We can then employ relaxation techniques similar

[3]The plan parser, thus, fills in knowledge gaps in our plan library.

[4]In the early semantic-based recognition algorithms the chaining together of actions was either controlled by expectations once the goal was known, or if totally bottom-up, it exploded.

to those used in natural language (Carberry 85, Goodman 86) to provide diagnostic support to repair the plan. Figure 3 below provides an overview of our proposed constructive plan recognizer.

3 Intelligent Interfaces

We want to provide practical uses of artificial intelligence in human-machine interfaces. Our primary motivation for enhancing the plan recognition procedure is to provide the flexibility required for building such robust interfaces. In this section we discuss intelligent interfaces, citing an example of one that we built. That interface, CHECS (CHemical Engineering CAD System), uses a standard plan recognizer. We explain why a constructive plan recognizer is required to provide the full power necessary to bring about a more robust interface. We end the section with a brief discussion about using the same paradigm for telerobotic interfaces.

3.1 Current Intelligent Interface Technology

Before discussing the CHECS interface, we discuss briefly how others have preceded our efforts to increase the robustness of interfaces and the methodologies they employed. In particular, we mention a couple efforts. User modelling techniques have been employed in an interface to Unix[5] as part of the Unix Consultant (Wilensky et al. 84) to provide help to stereotypical users (from novice to expert users). User models are generally "static" so that the inferences that can be deduced are limited in comparisons to the ones plan recognition can provide. Chin (88) has extended the Unix Consultant to take into account some pragmatic information about a user's interaction with the system. It tracks the user's knowledge over the course of a session. This allows it, for example, to avoid repeating things the user already knows. Fischer and Rathke (88) describe an interface for spreadsheets based on object-oriented knowledge representation. Their enhanced interface is most useful for the developer of the system and less for the actual user. It utilizes constraint based programming to represent knowledge about the application domain. It has a layered architecture to make it easy to modify and extend the system. Using plan recognition instead of constraint propagation should make it possible to bring the full power of such a system to the user as well as the application builder since the system could infer the user's intentions and reflect them dynamically in the system.

3.2 CHECS: An Intelligent Interface for Computer Aided Design[6]

3.2.1 CHECS

We performed an empirical study that involved examining a representative plan recognizer based on standard technology and using it to add more power and flexibility to an intelligent interface. To concretely explore the connections between interfaces and plan recognition, we designed and implemented CHECS, a graphical interface to a process design system. Design appears to be an excellent forum for applying plan recognition. Recent research projects in AI and design have expanded design systems from mere bookkeepers to active participants in the design process (e.g., Brown et al. 86). As we will see, an interactive plan-based interface can add a further layer of sophistication to these approaches. We have chosen chemical process engineering as our design domain because the interface is illustrative of many graphical design tools, and the domain is particularly amenable to standard planning and plan recognition representations. For example, there is a small, well-defined set of unit operations out of which other operations are hierarchically and functionally composed. We also chose design to push on the robustness of plan recognition, that is, since most designs are usually new, the design process is more than simply finding an old design.

The architecture of CHECS (and of plan-based interfaces in general) is shown in Figure 4. Its user interface is a graphical interface, inspired by existing graphical interfaces to chemical design simulators. As shown in Figure 5, the interface allows the user to build up a flow diagram by introducing chemical equipment (e.g., heat exchanger), flow (e.g., pipes connecting equipment), and parameters (e.g., heat-exchanger.input = n-butane). For example, by clicking on one of the icons representing equipment types (in the "CHECS Object" window), the user can build up a

[5]UNIX is a registered trademark of AT&T Bell Laboratories.

[6]A more complete description of this work can be found in [Goodman and Litman 89]. We would also like to acknowledge Kimberly Claffy for help in programming CHECS.

design containing equipment instantiations (in the "Plant Design" window). Parameters of the particular instantiations are then specified via menu interactions (Figure 6).[7]

After each user input, CHECS calls a plan recognizer to infer the chemical reaction (or plan) underlying the design, propagates information through the design to the appropriate points, and checks the consistency of the design step. For example, a portion of the butane isomerization plan recognized from the current design is shown in the lower portion of Figure 5. Currently, CHECS uses a modified implementation of Kautz's covering model of plan recognition (Kautz 87). The algorithm is incremental since it cyclically receives inputs and infers conclusions. For each input, the conclusions of the plan recognizer are further constrained by the conclusions drawn from previous inputs. In CHECS we assume that a single plan explains all observations (i.e., that a user is working on only one design at a time).

A plan is inferred using pre-existing knowledge of actions. This knowledge is expressed in the form of interleaved frame hierarchies, defining abstraction and decomposition relationships between actions. For example, the fact that a *heating-action* "is-a" *change-temperature-action* is an abstraction, while the specification of the steps involved in the performance of a *heating-action* is a decomposition. The CHECS hierarchy contains frames representing primitive chemical actions (e.g., heat, cool), chemical plans composed of primitive actions and/or other plans (e.g., reaction with recycle, as in Figure 5), and a distinguished set of plans called *end* plans, which are not components of any other plans. There is also an interleaved object hierarchy, representing the equipment associated with each primitive action (e.g., a heat exchanger can be used to implement a heating or cooling action). Finally, there is a hierarchy of chemical substances, as well as a database of facts containing particular assertional information. In CHECS, the goal of plan recognition is the inference of an instantiated end plan given the specified configuration.

Finally, just as plan recognition has been used to enhance various systems, CHECS recognizes and then manipulates plans in order to provide new design interface capabilities. As will be discussed below, plan recognition is used to add conceptual design completion, error handling, advice, and plan-dependent system responses.

3.2.2 Plan Recognition Help for CAD Interfaces

Typical graphical front-ends to design simulators provide standard graphics capabilities such as the creation of icons to represent parts, the selection and placement of parts, and editing facilities. Everything is performed manually including the layout. More recent systems help the designer with layout, allow parameter values to flow through the design automatically, provide symbolic simulation, and determine design validity by contrasting the designer's selected values against other pieces of the design. With plan recognition, we were able to implement features in CHECS not found in current systems. These advances allow the system to reflect the context in the system's current state, such as through menus, system advice, error detection and recovery, and bookkeeping. What we achieved in CHECS can be seen as a step towards more intelligent *cooperative* design tools. It is not automatic design, but instead keeps the user in the loop with most but not all of the control.

For example, CHECS is able to constrain icons and parameter values by considering what is legal with respect to what the system infers that the designer is doing. Such constraints are visible in "dynamic menus" which reflect the current context by dimming out entries that are already filled by the designer, that are inferred and propagated by the plan recognizer, or that are illegal given the hypothesized design(s). Figure 6 illustrates the menu generated for specification of an input to a reactor vessel. The input menu is dimmed because an entry, *n*-pentane, has already been constrained as a result of inferences made by the plan recognizer from an earlier observation (e.g., by propagating the output from one part through a pipe to the input of the reactor). The menu could have been only partly dimmed if more than one entry was possible. Dynamic menus, thus, provide salience to their entries.

CHECS, once it infers the designer's goal, can also make inferences that provide shortcuts to the design process or even allow design completion. For example, once CHECS determines a unique plan in the library (or common elements in the remaining "possible" plans), it can complete all (or portions of) the user's design based upon the inferred plans(s). Consider Figure 7 in which the user has added first a feed (a container of a particular chemical substance) and then a still (a device used to separate a chemical from a mixture). From the plan library and from local constraints, CHECS can predict that a heat exchanger (a device used for heating or cooling, shown as dimmed in the figure) is needed on an input to the still. That is, stills require that their inputs are heated and, if no outputs of parts already on the screen are heated, the system could *predict* that the user will have to heat the input to the still.

[7]These figures show snapshots of the current system. CHECS is implemented in Common Lisp; the graphical interface runs on a Macintosh, while a textual version runs on other machines..

The heat exchanger is shown dimmed as a "suggestion" which the user can accept by clicking on it or reject by proceeding in a different manner.

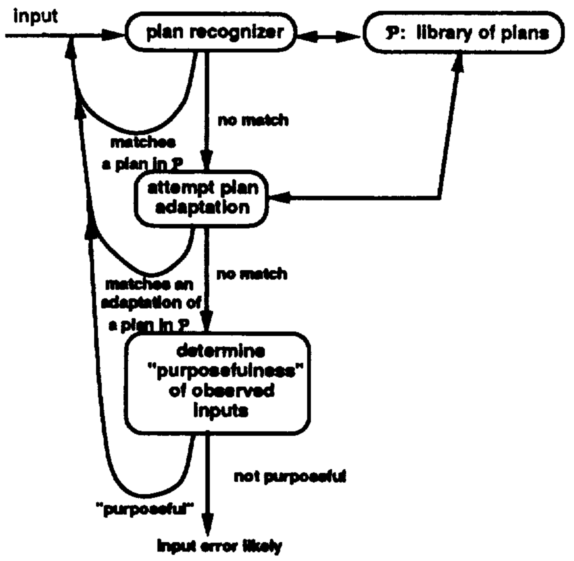


Figure 3: Constructive plan recognition

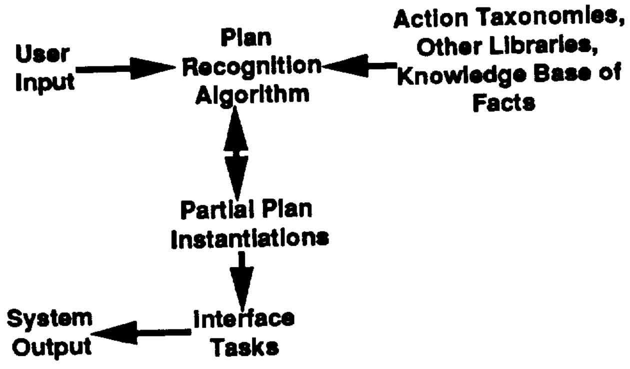

User Input
Plan Recognition Algorithm
Action Taxonomies, Other Libraries, Knowledge Base of Facts
Partial Plan Instantiations
Interface Tasks
System Output

Figure 4: A Generic Plan-Based Interface System

Because CHECS recognizes the plans that underlie designs, CHECS can infer and suggest designs that do not directly match the user's flow graph, but that instead have the same underlying functional (plan) structure. So, for example, even if the user's design of Figure 5 had a heat exchanger connected to the output of the reactor, the plan recognizer could still match the design conceptually against a design with a "heated reactor" (a reactor with a heating coil built in), because a chemical subplan such as "react-and-heat" can be achieved by either flow configuration.

CHECS also provides for error handling by checking design validity. Design validity entails that the user's design achieve a plan in the library. When it does not, the design is invalid with respect to the language of the design library. In its strongest instantiation, the system prevents the user from making mistakes by only allowing the user to perform actions that can lead towards one of the inferred designs.

Finally, CHECS provides advice and active bookkeeping. For example, CHECS can use conceptual matches to suggest functionally equivalent but more optimal designs (such as the use of a heated reactor for a reactor, pipe, and heat exchanger configuration). CHECS also does a better job at bookkeeping then previous CAD systems since it has a higher perspective of the design then available in object-oriented design systems. By inferring the design goal, CHECS can propagate information through the design and utilize it immediately. For example, even though we did not implement this feature in CHECS, inferring the possible matching designs in the design library can allow the system to do a better job (than the user) at the layout of the user's design.

3.2.3 Implications of CHECS

We built our first version of CHECS to study the weaknesses and strengths of an interface integrated with a plan recognition component. Our goal was to isolate shortcomings in a class of approaches and to propose ways around them. We discovered, as outlined in the previous section, that plan recognition can add numerous benefits to an interface in the areas of plan completion, error detection and recovery, advice generation, and context dependent responses. In turn, we also determined that current plan recognition systems are fundamentally limited in ways that limit the power of the associated interface. They have little to say about the recognition of novel plans; they typically search for known goals explained by known action sequences. However, one's planning knowledge is incomplete, can contain errors, and differs among different users. Hence, the recent implementations of plan recognizers make unreasonable assumptions that get in the way of a robust interface. They made some of the user enhancements awkward in practice. The plan recognizers caused the interfaces to exert too much control over the direction of the user in accomplishing his or her task. The more flexible we allow an interface to be (e.g., allowing the user to ReStart, BackUp, or Edit their commands), the more likely communication problems will occur. Current interfaces restrict such freewill at a cost of making the interface awkward to use. We contend that our proposed constructive plan recognizer can achieve a balance between the user and the system. Constructive plan recognition can also broaden the communication medium. As applications become increasingly complex, the interfaces between them and their users take on an ever increasing burden in providing effective communication. Such interfaces can expect mixed modalities in the communication channel. For example, a user can point with a mouse (or other pointing device), select commands from a menu, provide typed natural language commands, or give spoken commands.

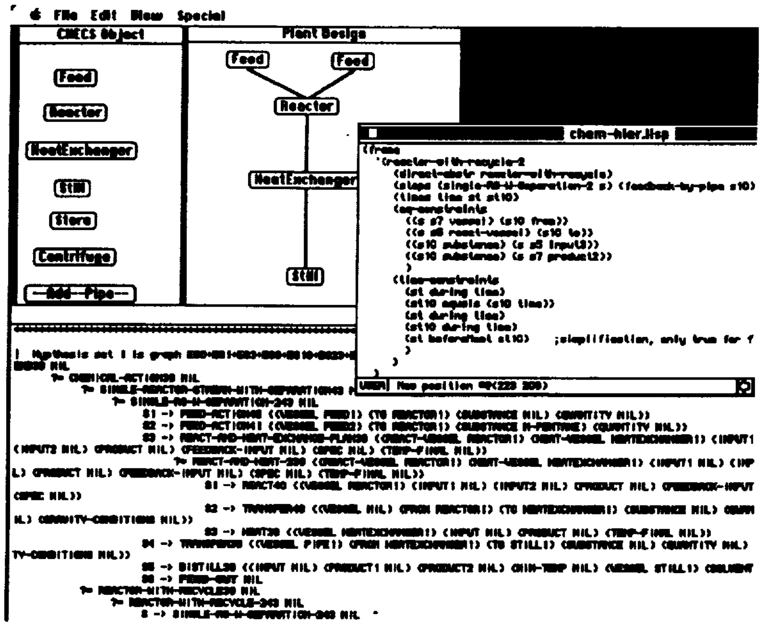
3.3 Intelligent Interfaces for Telerobotics

Our constructive plan recognition algorithm could serve as a backbone for an intelligent interface to a telerobot. Its strengths over standard plan recognizers have already been elucidated. In practice it would be capable of building on any prestored plans in the incomplete task plan library. CPR better fits the patched together nature of telerobotic actions. Since matching complete prestored plans to a particular situation won't always work in the unknowns of space, operators must adapt their plans or learn as they go. An interface that can effectively work for an operator in that environment, hence, must be capable of tracking varying user actions. It must detect and ignore side actions performed by the operator that aren't related to the overall goal while still responding to contingencies that arise.

A telerobot interface could be extended using CPR to add a more cooperative layer of sophistication and to bring the interface and telerobotics task closer together. CPR would monitor the operator's actions in an attempt to infer both high-level and low-level goals based on the latest action in the action sequence. When a higher-level goal is inferred, the interface can request the response planning component to complete the task. Otherwise, based on local actions, the interface can augment the operator's actions with more precise actions.

4 Conclusion

Plan recognition is an important part of any telerobot since it can help the robot determine an astronaut's intentions from his actions as well as working in conjunction with a planner to guide the robot in responding to contingencies. Previous plan recognition paradigms required the system builder to build large libraries of all possible plans and situations. Constructive plan recognition can provide the robust form of plan recognition required for telerobotics. CPR is more realistic than previous plan recognition by recognizing that tasks are often approached in a novel way. CPR is meant to survive without a complete and predefined plan library relieving some of the burden from the user.

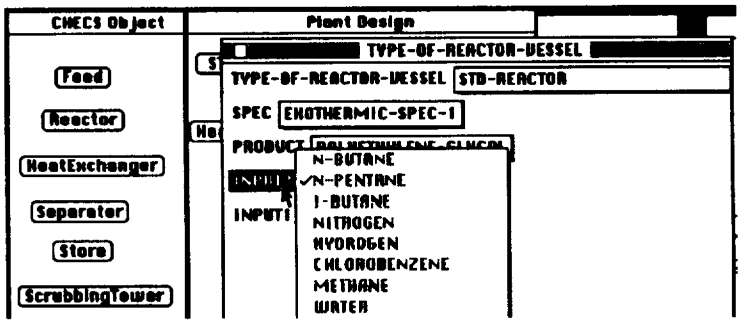


```
((frame
  '(reactor-with-recycle-2
    (direct-abstr reactor-with-recycle)
    (steps (single-RG-W-Separation-2 s) (feedback-by-pipe s10)
    (times time st st10)
    (eq-constraints
      ((s s7 vessel) (s10 from))
      ((s s8 react-vessel) (s10 to))
      ((s10 substance) (s s5 input2))
      ((s10 substance) (s s7 product2))
      )
    (time-constraints
      (st during time)
      (st10 equals (s10 time))
      (st during time)
      (st10 during time)
      (st beforeMeet st10)    ;simplification, only true for f
      )
    )
  )
USER  New position @(223 309)
```

```
| Hypothesis set 1 is graph [illegible]
[illegible] NIL
    ?= CHEMICAL-ACTION[illegible] NIL
        ?= SINGLE-REACTOR-STREAM-WITH-SEPARATION[illegible]
            ?= SINGLE-RG-W-SEPARATION-243 NIL
                S1 -> FEED-ACTION40 ((VESSEL FEED1) (TO REACTOR1) (SUBSTANCE NIL) (QUANTITY NIL))
                S2 -> FEED-ACTION41 ((VESSEL FEED2) (TO REACTOR1) (SUBSTANCE N-PENTANE) (QUANTITY NIL))
                S3 -> REACT-AND-HEAT-EXCHANGE-PLAN30 ((REACT-VESSEL REACTOR1) (HEAT-VESSEL HEATEXCHANGER1) (INPUT1
(INPUT2 NIL) (PRODUCT NIL) (FEEDBACK-INPUT NIL) (SPEC NIL) (TEMP-FINAL NIL))
                        ?= REACT-AND-HEAT-280 ((REACT-VESSEL REACTOR1) (HEAT-VESSEL HEATEXCHANGER1) (INPUT1 NIL) (INP
L) (PRODUCT NIL) (FEEDBACK-INPUT NIL) (SPEC NIL) (TEMP-FINAL NIL))
                            S1 -> REACT40 ((VESSEL REACTOR1) (INPUT1 NIL) (INPUT2 NIL) (PRODUCT NIL) (FEEDBACK-INPUT
(SPEC NIL))
                            S2 -> TRANSFER40 ((VESSEL NIL) (FROM REACTOR1) (TO HEATEXCHANGER1) (SUBSTANCE NIL) (QUAN
L) (QUANTITY-CONDITIONS NIL))
                            S3 -> HEAT30 ((VESSEL HEATEXCHANGER1) (INPUT NIL) (PRODUCT NIL) (TEMP-FINAL NIL))
                S4 -> TRANSFER[illegible] ((VESSEL PIPE1) (FROM HEATEXCHANGER1) (TO STILL1) (SUBSTANCE NIL) (QUANTITY NIL)
TY-CONDITIONS NIL))
                S5 -> DISTILL30 ((INPUT NIL) (PRODUCT1 NIL) (PRODUCT2 NIL) (MIN-TEMP NIL) (VESSEL STILL1) (COOLANT
                S6 -> FEED-OUT NIL
        ?= REACTOR-WITH-RECYCLE[illegible] NIL
            ?= REACTOR-WITH-RECYCLE-243 NIL
                S -> SINGLE-RG-W-SEPARATION-243 NIL
```

Figure 5: Designing a Butane Isomerization Process

ORIGINAL PAGE IS
OF POOR QUALITY



TYPE-OF-REACTOR-VESSEL STD-REACTOR
SPEC EXOTHERMIC-SPEC-1
PRODUCT
INPUT1

Figure 6: CHECS Dynamic Menu

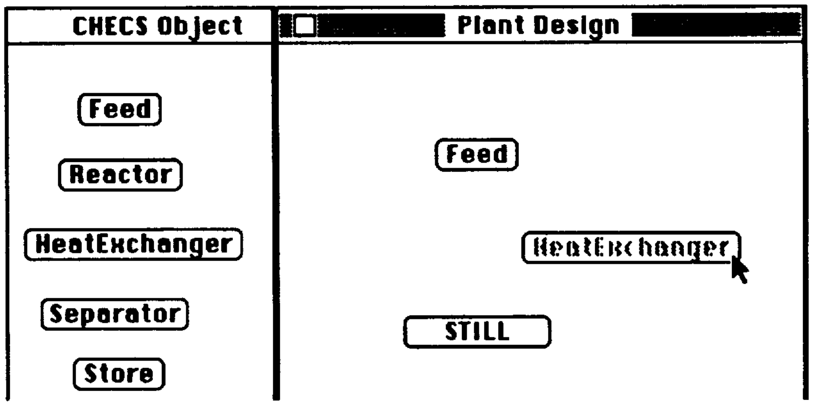


Figure 7: Local Prediction

5 References

Allen, James F. Recognizing Intentions from Natural Language Utterances. In *Computational Models of Discourse,* M. Brady and B. Berwick (eds.), MIT Press, 83.

Allen, James F. *Natural Language Understanding.* The Benjamin/Cummings Publishing Company, Inc., Menlo Park, Ca., 87.

Alterman, Richard. An Adaptive Planner. In *Proceedings of AAAI-86,* pages 65-69. Philadelphia, Pa., August, 86.

Bejczy, A. K., Man-Machine Interface Issues in Space Telerobotics: A JPL Research and Development Program. In *Proceedings of the Workshop on Space Telerobotics,* G. Rodriguez (ed.), NASA and JPL, July 87.

Broverman, Carol A. and W. Bruce Croft. Reasoning About Exceptions During Plan Execution Monitoring. In *Proceedings of AAAI-87,* pages 190-5. Seattle, Wa., July, 87.

Brown, David C. and B. Chandrasekaran. Knowledge and Control for a Mechanical Design Expert System. *Computer* (7):92-100, July, 86.

Carberry, Mary Sandra. *Pragmatic Modeling in Information System Interfaces.* PhD thesis, University of Delaware, 85.

Carver, N.F., Lesser, V.R., and McCue, D.L. Focusing in Plan Recognition. In *Proceedings of the National Conference on Artificial Intelligence,* The American Association for Artificial Intelligence, Austin, TX, 84.

Chin, David, Exploiting User Expertise in Answer Expression. In *Proceedings of AAAI88,* pages 756-760, St. Paul, Mn., 88.

Cohen, Philip R. and Hector J. Levesque, Speech Acts and Rationality, in *Proceedings of the 23rd ACL Conference,* pages 49-59, Chicago, July, 85.

Dean, T. and M. Brody, Prediction and Causal Reasoning in Planning. In *Proceedings of the Workshop on Space Telerobotics,* G. Rodriguez (ed.), NASA and JPL, July 87.

Drummond, M., K. Currie and A. Tate, Contingent Plan Structures for Spacecraft. In *Proceedings of the Workshop on Space Telerobotics*, G. Rodriguez (ed.), NASA and JPL, July 87.

Durfee, E. H. and V. R. Lesser, Incremental Planning to Control a Blackboard-Based Problem Solver. In *Proceedings of the Workshop on Space Telerobotics*, G. Rodriguez (ed.), NASA and JPL, July 87.

Fischer, Gerhard and Christian Rathke, Knowledge-Based Spreadsheets. In *Proceedings of AAAI-88*, pages 802-807, St. Paul, Mn. 88.

Genesereth, Michael R. The Role of Plans in Automated Consultation. In *Proceedings of IJCAI-79*, pages 311-3, Tokyo, Japan, August, 79.

Georgeff, M. P., A. L. Lansky and M. J. Schoppers, Reasoning and Planning in Dynamic Domains: An Experiment With a Mobile Robot. In *Proceedings of the Workshop on Space Telerobotics*, G. Rodriguez (ed.), NASA and JPL, July 87.

Goodman, Bradley A. Reference Identification and Reference Identification Failures. *Computational Linguistics* 12(4):273-305, October-December, 86.

Goodman, Bradley A. and Diane J. Litman, Plan Recognition for Intelligent Interfaces. Submitted for review to IJCAI-89, 89.

Hammond, Kristian J. Explaining and Repairing Plans that Fail. In *Proceedings of IJCAI-87*, pages 109-114, Milan, Italy, August, 87.

Huff, Karen and Victor Lesser. *Knowledge-Based Command Understanding: An Example for the Software Development Environment.* Report TR 82-6, Computer and Information Sciences, University of Massachusetts at Amherst, Amherst, Ma., 82.

Jenkins, L., Space Telerobotic Systems: Applications and Concepts. In *Proceedings of the Workshop on Space Telerobotics*, G. Rodriguez (ed.), NASA and JPL, July 87.

Kautz, Henry A. *A Formal Theory of Plan Recognition.* PhD thesis, University of Rochester, 87. Also, TR215, University of Rochester, Dept. of Computer Science, Rochester, N.Y.

Kolodner, Janet L., Robert L. Simpson, Jr., and Katia Sycara-Cyranski. A Process Model of Case-Based Reasoning in Problem Solving. In *Proceedings of IJCAI-85*, pages 284-290, Los Angeles, August, 85.

Litman, Diane J. and James F. Allen. A Plan Recognition Model for Subdialogues in Conversations. *Cognitive Science* 11:163-200, 87.

Pollack, Martha E. Inferring Domain Plans in Question-Answering. PhD thesis, University of Pennsylvania, 86. Also, Report MS-CS-86-40 of the Department of Computer and Information Science, University of Pennsylvania

Rodriguez, G. (ed.), *Proceedings of the Workshop on Space Telerobots*, NASA and JPL, July 87.

Ross, Peter and John Lewis, *Plan Recognition and Chart Parsing*, DAI Research Paper No. 309, University of Edinburgh, Edinburgh, Scotland, 87.

Sacerdoti, Earl D. Planning in a Hierarchy of Abstraction Spaces. *Artificial Intelligence* 5:115-135, 74.

Schank, R. C. and C. J. Reiger, Inferences and the computer understanding of natural language, *Artificial Intelligence* 5, pages 333-412, 74.

Schmidt, C. F., Sridharan, N. S., and Goodson, J. L. The Plan Recognition Problem: An Intersection of Psychology and Artificial Intelligence. *Artificial Intelligence* 11:45-83, 78.

Sidner, Candace L. Plan parsing for intended response recognition in discourse. *Computational Intelligence* 1(1):1-10, 85.

Wilensky, Robert, Yigal Arens, and David Chin. Talking to UNIX in English: An Overview of UC. *Communications of the ACM*: 27(6): 574-593, 84.

Vilain, Marc, Chart Parsing for Plan Recognition Proposal, Personal communication, BBN, 88.

Wilkens, D. E., Recovering from Execution Errors in SIPE. In *Proceedings of the Workshop on Space Telerobotics*, G. Rodriguez (ed.), NASA and JPL, July 87.

Woods, William A. Cascaded ATN Grammars. Amer. J. Computational Linguistics 6(1):1-15, Jan.-Mar., 80.