Planning and Execution of Tasks in Cooperative Work Environments

Lawrence S. Leffkowitz and W. Bruce Croft

Computer and Information Science Department
University of Massachusetts
Amherst, MA 01003
CSNET Address: larry@cs.umass.edu
Tel: (413) 545-3147

Problem solving has long been recognized as an important component in many work environments. The potentially complex and often cooperative nature of even apparently "routine" tasks has led to attempts to use planning techniques to support work in real-world domains. This paper describes the work being done by the POLYMER project to construct a planning system to assist in the performance of multiagent, loosely-structured, underspecified tasks. Specifically, we present a representation for modeling tasks, agents and objects within such environments and describe the architecture and implementation of a planning system which uses these models to support cooperative work. We conclude with a description of how this planning system is being used to support further research in areas such as exception handling, negotiation and knowledge acquisition.

Principles: Planning, Knowledge Representation
Research
Office Automation
LISP and KEE
Two Person-years
Planning and Execution of Tasks in Cooperative Work Environments

Lawrence S. Lefkowitz W. Bruce Croft

Computer and Information Science Department University of Massachusetts Amherst, MA 01003

Abstract

Problem solving has long been recognized as an important component in many work environments. The potentially complex and often cooperative nature of even apparently "routine" tasks has led to attempts to use planning techniques to support work in real-world domains. This paper describes the work being done by the POLYMER project to construct a planning system to assist in the performance of multiagent, loosely-structured, underspecified tasks. Specifically, we present a representation for modeling tasks, agents and objects within such environments and describe the architecture and implementation of a planning system which uses these models to support cooperative work. We conclude with a description of how this planning system is being used to support further research in areas such as exception handling, negotiation and knowledge acquisition.

1 Introduction

Problem solving has long been recognized as an important component in many work environments [1,11,12]. The potentially complex nature of even apparently "routine" tasks has motivated the use of planning techniques to support work in real-world domains [5,17]. However, the work in many environments is cooperative in nature. In such settings, tasks often cannot be performed by an individual; the coordinated effort of a group of people is needed to accomplish a desired goal. The size and complexity of certain tasks and the limited abilities, knowledge, skills and resources of any individual often make a cooperative approach the only way to achieve results. Offices, design teams, management structures, and factories are all examples of cooperative work environments.

As the scope and complexity of the problems addressed by computer-based support systems grows, the need for planning and knowledge-based approaches becomes more apparent. While traditional tools have been adequate to support single-person, small scale tasks (e.g., mail systems and forms tools in offices, compilers and debuggers in software development, etc.), supporting cooperative work requires the coordination of multiple agents using a variety of tools. By using a model of the tasks, objects and agents in an application domain to generate multi-agent plans, the POLYMER planning system [5,7,10] can coordinate interdependent activities in complex, underspecified domains.

This research was supported in part by a contract with Ing. C. Olivetti and by the Air Force Systems Command, Rome Air Development Center, Griffiss Air Force Base, NY 13441-5700.

In this paper we present an overview of the POLYMER system, show its use in supporting cooperative work, and describe the current research centered around POLYMER. The following section describes the architecture and functionality of the POLYMER system. It presents POLYMER's model of an application in terms an example from the journal editing domain and shows how this model is used by the planner. Section 3 describes the planning process and shows how a plan is interactively generated and what happens when problems arise. Section 4 presents a brief overview of research projects which are extending the POLYMER planner to develop an integrated cooperative work environment. Finally, the current status of the POLYMER project is summarized in Section 5.
Over the past two years, the POLYMER system has been developed as a testbed for supporting cooperative work. Using descriptions of domain tasks and objects, POLYMER combines strategic and reactive planning and interacts with its users to generate a plan to accomplish a specified goal. POLYMER's domain description language evolved from one developed for the POISE intelligent interface system [9]. Using this formalism, POLYMER performs the type of hierarchical, non-linear planning described in NONLIN [19] and SIPE [20].

The POLYMER system has been developed using the KEE system\(^1\) running on TI Explorers.\(^2\) POLYMER uses KEE's frame-based knowledge representation to encode domain activity, agent and object descriptions (see Section 2.1). KEE's assumption-based truth maintenance system (ATMS) is used to record all planning decisions and support dependency-directed backtracking (see Section 3.3). We have extended KEE's "world" system to permit the construction of a world hierarchy graph to represent the state of the application domain at each point in a POLYMER plan (see Section 3.2.2). Finally, KEE was selected to obtain an added degree of portability, especially in the design of POLYMER's interface.\(^3\)

The overall architecture of the POLYMER system is presented in Figure 1. The domain representation and planning methodology are presented below. POLYMER's exception handling capabilities and knowledge presentation and acquisition facilities are discussed briefly in Section 4.

2.1 Representing the application domain

POLYMER models an application domain through the description of activities, objects and agents within the domain. In this section we focus on the description of domain activities. A grammar for the POLYMER activities appears in Appendix A; a sample activity description from journal editing is shown in Figure 2.

The representation includes both the goal and preconditions of an activity as well as a decomposition of the activity into smaller steps. The steps are usually goals for which other activities will be selected during the planning process, but may also be specific activities or tool invocations (i.e., "actions"). Causal relations between steps may be specified and these relations are used to generate temporal ordering constraints as well as protection intervals.\(^4\) Control flow among the steps (e.g., looping, iteration, etc.) may also be specified directly.

\begin{figure}
\centering
\includegraphics[width=\textwidth]{figure1.png}
\caption{The POLYMER Planning System}
\end{figure}

ACTIVITY: REVIEW-PAPER

Goal: review(?submission, ?reviews)

Preconditions:

- member(?paper, papers)
- edits.journal(?editor, ?journal)
- submitted.to(?submission, ?journal)
- paper(?submission, ?paper)

Decomposition:

- goal reviewers-selected = and(reviewers(?submission, ?reviewers), sufficient-reviewers(?journal, ?reviewers))
- goal paper-distributed = has(?reviewer, copy-of(?paper))
- goal have-review = has(?editor, review(?reviewer))

Plan Rationale:

- enables reviewers-selected paper-distributed
- enables paper-distributed have-review

Control: repeat (paper-distributed to have-review) for ?reviewer in ?reviewers

Agents: (?editor, editors)

Consider an example from the domain of journal editing. When a paper is submitted for publication in a journal, the editor of that journal must select appropriate reviewers for the paper, invite them to review the paper, collect their reviews, make a decision based on the reviews, and inform the author of the decision. Figure 2 shows a high-level POLYMER activity description for reviewing a submitted paper. It contains subgoals which are named and specified in terms of states of domain objects. The subgoals are causally (and therefore temporally) linked since a reviewer must be selected before a copy of the paper can be sent to the reviewer and the reviewer must receive the paper before the editor can get a review back. The two latter steps are repeated for each reviewer selected in the first step.

\(^1\)KEE is a registered trademark of IntelliCorp, Inc.
\(^2\)Explorer is a trademark of Texas Instruments Inc.
\(^3\)A version of POLYMER has recently been ported to Sun (trademark of Sun Microsystems, Inc.) workstations by Olivetti.
\(^4\)Protection intervals maintain a certain state of the world during a portion of the plan on the assumption that a state generated at one point will be needed at some later point. See [20].
Using these domain descriptions, POLYMER interactively generates hierarchical, partially-ordered plans to accomplish a stated goal. The planning process, described below, is unique in its use of a combination of "script based" and "goal directed" descriptions of activities to overcome the rigidity of scripts while greatly reducing the cost of purely goal driven systems. It interleaves planning and plan execution in order to overcome the ambiguity inherent in complex, underspecified domains.

2.2 Preprocessing of domain descriptions

In order to make efficient use of the domain descriptions during the planning process, a certain amount of preprocessing is necessary. First, the external forms of the domain descriptions (shown above) are parsed and converted into Activity, Object, and Agent (KEE) units. Then, the activity descriptions are further processed to convert their information into a form more conveniently used during planning.

As we will see in the following section, POLYMER represents a plan as a partially ordered network of plan nodes called a plan network. To simplify the generation (and expansion) of a plan network, POLYMER creates a plan network to represent each activity during the preprocessing phase. Thus, all the information in an activity description is converted into a partially ordered set of plan nodes. A more formal description of POLYMER's Plan Network Maintenance System appears in [3].

To generate a plan network for an activity, POLYMER first creates a plan node for each step in the activity's decomposition. Goal, activity and action nodes are generated for each corresponding step type. In addition, structural nodes are generated to represent the start and finish of the entire activity. Next, a partial ordering is established among the nodes (using the node's "predecessor" and "successor" fields) based on information in the activity's control clause. The more complex control constraints (e.g., if, optional, star, plus and repeat) result in the insertion of additional structural nodes.

Next, the causal relations in the plan rational clause are used to generate additional ordering constraints and protection intervals for the plan network. Finally, the activity's preconditions are transformed into constraints on the network's start-node, the activity's effects are placed on the network's finish-node, and any additional constraints specified in the activity description are placed on appropriate nodes within the network.

An example of the plan network generated for the activity description in Figure 2 is shown in Figure 3. A grammar for the specification of POLYMER plan networks and plan nodes appears in Appendix B.

![Figure 3: The "Referee Paper" Plan Network](image)

In order to reduce the search for an activity to accomplish a goal during the planning process, one further preprocessing action is taken. For each goal node within an activity's plan network, POLYMER compares the value of the goal node to the goal of each known activity description. By finding and recording (during preprocessing) the set of all activities which could possibly satisfy each goal node, the planning process is made more efficient.

3 Interactive plan generation

To help a user achieve a desired goal, POLYMER attempts to generate a plan to accomplish the goal. To accomplish this, the goal and the required parameters must first be presented to the planner which then constructs the plan in a hierarchical manner. The planning proceeds in a top-down (in terms of plan abstraction), left-to-right (in terms of step ordering) fashion as far as possible without ambiguity. When the planning cannot proceed with certainty, it attempts to resolve the ambiguity by either 1) executing an action node (if any are "ready" as explained below) or 2) obtaining information from the user (such as which of several possible tasks it should use to accomplish an outstanding goal). In this section we describe how an initial goal is presented to the planner, how a plan network is interactively constructed to satisfy this goal, and what happens when difficulties arise during the generation and execution of the plan.

3.1 Statement of goal via an application interface

In a given application domain, there will typically be a fairly common set of goals that a particular user wishes to accomplish. For instance, in the journal editing domain, the editor of a journal will want to generate a request for papers, referee a submitted paper, modify a database of potential reviewers, etc. A reviewer will choose whether to review a particular paper and submit reviews to the editor. An author will write papers, submit them to journals, and rewrite the papers as necessary.

For a particular class of users in a particular domain, the user interface must allow the user to select the goal they want to accomplish and to specify the necessary parameters. Thus, a journal editor's interface allows the user to select a paper to review and to specify the paper. The current system contains both a menu-driven "developer's" interface (Figure 4) and an iconic "end-user's" interface for the office domain (Figure 5).

Because each goal may require an extended period of time to accomplish, a user will most likely want to interleave several tasks. Thus, the interface allows the user to suspend the current goal and select another (new or previously suspended) one.

Once a goal is selected, POLYMER generates a top-level plan network to represent that goal. It consists of a single goal node contained between a pair of "start" and "finish" structural nodes. The user's parameterized goal is used as the goal-value of the goal node; any initial state information is added to the top-level start-node. Planning begins by expanding this plan network as explained below.

3.2 Expanding the plan network

The POLYMER planner hierarchically constructs a plan by expanding a plan network. This expansion is performed by alter-
nately selecting a plan node to process and then processing that node. The way in which a node is processed is determined by the node type.

3.2.1 Selecting a plan node  Because POLYMER constructs its plans in a top-down, left-to-right fashion, it selects the highest level (i.e., most abstract) node that is "ready" to be processed. In order to assure left-to-right processing, a node is "ready" to be processed if and only if 1) the node has not already been processed, 2) all of the node's necessary predecessors are complete and awaiting successors, and 3) all of the node's conditions are satisfied. A predecessor node is complete if its processing has been completed or if it does not need to be processed (i.e. a phantom goal node, as explained below). A node's conditions are evaluated in the node's "before-world" to assure that the node is applicable at this point in the plan.

To ensure top-down processing, each node is assigned an abstraction level denoting its "depth" from the original top-level goal. Thus, the nodes in the top-level network are assigned a level of 0; when a goal node is replaced by an activity network (as explained below), the level of the new nodes is 1 greater than the goal node they replace.

Once the "ready" nodes are sorted by abstraction level, they are ordered by node type. Structural nodes (except for loop-iteration nodes) are processed first, followed in order by goal nodes, activity nodes, action nodes and finally loop-iteration nodes. If more than one node of a given type are ready to be processed, the planner can either select one based on a (modifiable) set of heuristics (e.g., specificity of the node's conditions, a priori preferences among tasks, etc.) or ask the user to select which node to process.

3.2.2 Processing a plan node  Once POLYMER selects a plan node, the way in which the node is processed depends upon its type. For instance, it checks whether goal nodes are already satisfied by evaluating the goal node's value in the node's before-world. If the goal is already satisfied (either accomplished by some earlier steps or true in the initial world state), the node status is set to "phantom" and no further processing of the node takes place. If the goal is not satisfied, POLYMER determines if it can add additional ordering constraints to the existing plan network in order to place the node where its goal will be satisfied. If it is unable to achieve this, the planner must select an activity to achieve the goal.

As described in Section 2.2 above, each goal node template contains a list of all activities which may possibly satisfy the goal. These activities are now checked to see if their goals match that of the instantiated goal node and whether their preconditions are satisfied in the goal node's before-world. If more than one activity still qualifies as a means of accomplishing the goal, the planner can select one heuristically (using, for example, the closeness of the match between the activity's goal and that of the goal node, the specificity of the activity's preconditions, etc.) or by asking the user which activity should be performed.

Once an activity is selected for a goal, the plan network for that activity is instantiated and spliced into the current plan network in place of the goal node. Instantiating the network includes the instantiation of each of the nodes in the network, creation of KEE worlds corresponding to the new nodes, assertion of each node's effects in its after-world, and instantiating any needed protection intervals on these worlds. In addition to splicing the new nodes into the existing plan network, the new worlds are spliced into the existing world hierarchy.

Activity nodes are processed in a similar way, except that the selection of an appropriate activity is obviated. Action nodes require the invocation of tools (or interactions with other agents) and are handled by the execution monitor. The tools are invoked as specified in the "code" portion of the action node and the results are recorded in the after-world corresponding to the action node next.
Note that action nodes can be processed before the plan network is completely expanded. This permits the interleaving of planning and plan execution in order to prevent the planner from becoming swamped by the potentially explosive combinatorics of purely strategic planning in an underspecified and often ambiguous environment.

Most structural nodes simply serve as a means of demarcating activity boundaries and are the appropriate locations to place constraints and effects that belong at the beginning or end of an activity. Thus, an activity's preconditions and effects are placed on the activity's start and finish-nodes, respectively. The processing of these nodes only requires checking that their conditions are valid in order for them to be "complete". Structural nodes used to control looping, generated from certain control constructs, are more complex. For each loop construct, a loop-controller object is created and the loop-iteration and loop-termination nodes have conditions and effects which utilize the loop-controller. Thus, if the conditions of a loop-iteration node are satisfied, an additional instantiation of the nodes which comprise the body of the loop are created and inserted into the plan network. If a loop-termination node is satisfied, both the loop-iteration node and the loop-termination node are marked "complete" and the looping terminates.

3.3 Detection and correction of plan problems

The expansion of the plan network and the execution of actions interleaved with plan generation can both cause problems to arise in the plan. These may be detected as 1) actions performed which were not expected as part of the plan, 2) goals selected by a user which were not currently expected, or 3) inconsistencies in the world model arising from violated protection intervals or failed constraints.

These problems can indicate an error on the part of the planner, "exceptional" behavior by the user, or simply a user error. The handling of exceptional behavior, i.e., intentional actions by the user that are not covered by the planner's domain model, is described in Section 4.1. In order to correct errors in the existing plan, POLYMER first considers the addition of node ordering constraints to resolve the problem. If this fails and the problem was caused by a violated goal, it considers reinstating the goal at a point after it was violated and before it is needed.

If POLYMER is unable to resolve the problem by manipulating the existing plan network, it is forced to backtrack and undo some of its previous planning decisions (e.g., the selection of an activity for a goal, a choice of alternative goals, etc.). Because POLYMER uses KEE's ATMS to justify and record each of its planning decisions, the planner needs to redo only those portions of the plan that led to the problematic results (i.e., dependency-directed backtracking).

4 Beyond the basic planner

While we believe that a planner such as POLYMER is an essential component to support work in cooperative environments, we also see the planner as the core of a set of sophisticated support tools. Several research projects are currently underway that build upon POLYMER's functionality and aim to extend its overall utility. These projects, described below, include systems to handle exceptional behavior, to present and acquire models of application domains, and to support conflict resolution.

4.1 Exception handling

Because POLYMER's domain model is inherently incomplete (as is any model of a "real-world" domain), there will be situations where the planner does not correctly anticipate a user's desired action(s). By combining the domain model with heuristic knowledge about how plans may deviate, the SPANDEX system [4] uses a process of plausible inference to generate explanations of how a user's exceptional behavior can be reconciled with an existing POLYMER plan. Using these explanations, SPANDEX constructs the necessary amendments to the domain model to incorporate this new behavior.

4.2 Knowledge presentation and acquisition

Though POLYMER's domain model may never be complete (or even necessarily correct), the ability to make modifications (e.g., additions, corrections) in a simple fashion is extremely important. In order to make such modifications by end users feasible, the domain model must be presented to and manipulated by the users in an "understandable" fashion. The DACRON project [15] has investigated how typical end-users perceive tasks and objects within their domains and has been able to map this more "natural" model onto the POLYMER formalism. An interactive, animated, iconic interface is being developed to permit the presentation of information to these users and allow them to modify existing information as well as to specify additional tasks, objects, etc.

4.3 Conflict resolution

Because cooperative work environments requires the interaction of multiple agents (as well as the planner), conflicts between agents will inevitably arise. Differing goals, limited knowledge about the domain, varying capabilities and incomplete models of other workers can lead one agent to perform in a way that another agent does not expect. The resolution of such conflicts is often difficult and usually occurs through negotiation between the affected agents[2]. The GENEVA project [8] has begun to explore ways in which POLYMER's models of the domain and of the current plan can be used to support the negotiation process. In particular, it aims to assist in 1) initiating negotiation (by identifying the needed agents and presenting them with an appropriate view of the conflict), 2) maintaining the state of current and past negotiation sessions, 3) suggesting and allowing the exploration of solutions, and 4) verifying that proposed solutions actually resolve the conflict.

5 Summary

The POLYMER planning system has been designed and implemented as the core of an environment to support cooperative work. The current prototype has been used to interactively generate plans in such diverse areas as journal editing, software development and house purchasing. A preliminary version of the system has been delivered to an Olivetti research laboratory where
it is being used to develop advanced applications in the area of office automation.

In addition to the development of further applications, POLYMER is serving as a testbed for several research projects. These projects are exploring the use of knowledge acquisition, exception handling, and computer-mediated conflict resolution as part of an effort to develop an integrated environment for the support of cooperative work.

REFERENCES


