In D. Leake & E. Plaza (Eds.). (1997). *Case-Based Reasoning Research and Development: Second International Conference on Case-Based Reasoning* (pp 531-540). Berlin: Springer-Verlag.

Supporting Combined Human and Machine Planning: An Interface for Planning by Analogical Reasoning

Michael T. Cox and Manuela M. Veloso
Computer Science Department. Carnegie Mellon University
Pittsburgh, PA 15213-3891
{mcox;mmv}@cs.cmu.edu

Abstract. Realistic and complex planning situations require a mixed-initiative planning framework in which human and automated planners interact to mutually construct a desired plan. Ideally, this joint cooperation has the potential of achieving better plans than either the human or the machine can create alone. Human planners often take a case-based approach to planning, relying on their past experience and planning by retrieving and adapting past planning cases. Planning by analogical reasoning in which generative and case-based planning are combined, as in Prodigy/Analogy, provides a suitable framework to study this mixed-initiative integration. However, having a human user engaged in this planning loop creates a variety of new research questions. The challenges we found creating a mixed-initiative planning system fall into three categories: planning paradigms differ in human and machine planning; visualization of the plan and planning process is a complex, but necessary task; and human users range across a spectrum of experience, both with respect to the planning domain and the underlying planning technology. This paper presents our approach to these three problems when designing an interface to incorporate a human into the process of planning by analogical reasoning with Prodigy/Analogy. The interface allows the user to follow both generative and case-based planning, it supports visualization of both plan and the planning rationale, and it addresses the variance in the experience of the user by allowing the user to control the presentation of information.

1 Introduction

In mixed-initiative planning, automated and human planners need to interact in order to mutually construct a plan that satisfies a set of goals in a specific situation. Ideally, joint cooperation has the potential of achieving better plans than either the human or the machine can create alone [5, 9]. However, given the significant disparity between human and automated planning mechanisms, achievement of this potential is a difficult goal. The challenges of creating a successful mixed-initiative planning system fall into at least three categories: planning paradigms differ in human and machine planning; plan visualization is a complex, although necessary, task; and human users range across a spectrum of experience, both with respect to the planning domain and the planning technology. This paper describes the directions we are pursuing to address these problems when designing a mixed-initiative interface for the PRODIGY planning system.

One of the most significant problems facing the integration of human and automated planning is the cognitive metaphor shared between the participants. In general, AI planning assumes a model of actions in the world and generates new plans by searching the space of possible actions. Alternatively, the case-based metaphor of planning as a memory task [7] can be more accessible to the human user. Indeed, few humans plan as if they have never faced a problem like the current one before [10]. Instead, the solution is a matter of remembering concrete past experience to form a new plan similar to an old one previously performed [8, 11]. Yet, when gaps exist in experience, reasoning from first principles is equally as natural and necessary for deriving a successful plan. The Prodigy/Analogy planning system represents a hybrid metaphor that reflects planning in a more complete manner than is the case with traditional case-based planners. Within the same planning system, both generative and case-based algorithms have an equal role.

A related issue is the task of visualizing the plan itself. Although a plan can be conceptualized as a simple sequence of actions, the relationship of each action to the goals of the planner and the state of the world has complex structure. Plan steps often interact with each other depending on the conditions necessary for performing the action and the results of having carried out given actions. Instead of sequential action, a goal tree representation aids a planner when trying to discover or make concrete the structure of plans. Moreover, the *reasons* for why certain actions are chosen (that is, the planning justifications or rationale) are seldom represented, although beneficial for a user when trying to reason in complex domains either from a past case or from first principles. Therefore as presented in this paper, we have designed a mixed-initiative planning interface that supports the planning process through plan and rationale visualization. The inclusion of both facets of planning is aimed at supporting the metacognitive as well as the cognitive aspects of reasoning.

Finally, users of mixed-initiative planning systems differ in experience along at least two dimensions. Users differ in experience with respect to their knowledge and skills of the planning domain (e.g., military deployment planning), and they also differ with respect to their knowledge and skills of the underlying planning technology in the mixed-initiative system. Therefore, a tension exists when presenting information to the user. In some cases a user may wish to override or have access to information concerning the operation of the automated planner, while such information and decisions may bewilder other users. From the beginning, the PRODIGY philosophy has been one of a *glass-box* approach, allowing the user to exert control over all aspects of decision making in the planner [15]. Within a mixed-initiative framework, however, a *black-box* philosophy that shields some of the technological and implementational details from the user (especially the naïve user) can be appropriate as well [4]. A major goal of the interface reported here is to develop a *selective-control mechanism* to support the range of expertise from novice to expert in both dimensions. That is, the user should be able to examine and control any desired decision, while the remainder is abstracted or hidden from the user. This paper reports on the implementation of the interface that begins to support such a philosophy.[1]

The mixed-initiative interface contains a number of features that solve or mitigate the above problems. The user is able to switch between generative and analogical planning manually, or the interleaving of both modes can be left to automation. The interface provides mechanisms to save in the case library planning cases created generatively or analogically, to retrieve old cases that match current demands (either automatically or manually) and to choose various case interleaving strategies for adaptation and replay. The evolving plan is graphically presented to the user as a goal-tree structure and justifications for automated choices is displayed upon demand. Finally, the user can maintain a custom level of information display and user-control depending upon experience and interests.

1. The interface is publicly available at http://www.cs.cmu.edu/afs/cs/project/prodigy/Web

Section 2 describes the PRODIGY control algorithms, both in generative and analogical mode. Section 3 then introduces the mixed-initiative interface of PRODIGY, describing how both the plan and the planning decisions are displayed to the user. Section 4 briefly describes the current efforts to make the interface more responsive to the user's level of expertise. Finally, Section 5 concludes with a short discussion.

2 PRODIGY: A Hybrid of Generative and Case-Based Planning

PRODIGY is an automated planner that combines generative state-space planning and case-based planning. In a generative planning mode (Prodigy4.0), the system uses search through a space of operator choices. When under case-based planning mode (Prodigy/Analogy), it retrieves from a case library past plans that are most similar to a given new problem. These plans are reused (replayed) to create a solution for the current goals.

2.1 The Generative Planning Algorithm

The Prodigy4.0 system [15] employs a state-space nonlinear planner and follows a means-ends analysis backward-chaining search procedure that reasons about both multiple goals and multiple alternative operators from its domain theory appropriate for achieving such goals. A domain theory is composed of a hierarchy of object classes and a suite of operators and inference rules that change the state of the objects. A planning problem is represented by an initial state (objects and propositions about the objects) and a set of goal expressions to achieve. Planning decisions consist of choosing a goal from a set of pending goals, choosing an operator to achieve a particular goal, choosing a binding for a given operator, and deciding whether to commit to a possible plan ordering and to get a new planning state or to continue subgoaling for unachieved goals. Different choices give rise to different ways of exploring the search space. These choices are guided by either control rules, by past problem-solving episodes (cases), or by domain-independent heuristics.

1. Initialize.
2. Terminate if the goal statement has been satisfied.
3. Determine which goals are pending, i.e. still need to be achieved.
4. Determine if there are any selected operators that have their preconditions satisfied in the current state, and hence could be applied to the state as the next step in the plan.
5. Choose to subgoal on a goal or to apply an operator: (backtrack point)
   * To subgoal, go to step 6 * To apply, go to step 7
6. Select one of the pending goals (no backtrack point), an instantiated operator that can achieve it (backtrack point); go to step 3.
7. Change the state according to an applicable operator (backtrack point); go to step 2.

Fig. 1. A top-level view of Prodigy4.0's planning algorithm.

As shown in Figure 1, Prodigy4.0 follows a sequence of decision choices, selecting a goal, an operator, and an instantiation for the operator to achieve the goal. Prodigy4.0 has an additional decision point, namely where it decides whether to "apply" an operator to the current state or continue "subgoaling" on a pending goal. "Subgoaling" can be best understood as regressing one goal, or backward chaining, using means-ends analysis. It includes the choices of a goal to plan for and an operator to achieve this goal. "Applying" an operator to the state means a commitment (not necessarily definite since backtracking is possible) in the ordering of the final plan. On the other hand, updating the state through this possible commitment allows Prodigy4.0 to use its state to more informed and efficient future decisions. Hence, the planning algorithm is a combination of state-space search corresponding to a simulation of plan execution of the plan (the head plan; [6]) and backward-chaining responsible for goal-directed reasoning (the tail plan).

The reasons that these choices are made by the automated planner (or, if under user control, the user) are preserved in a plan derivation trace to improve planning efficiency in the future. When the user saves a planning episode, these decision justifications (i.e., the traces) are saved along with the solution in the case library. The case is indexed by the problem goals and a subset of the initial state responsible for the achievement of the goals.

2.2 The Derivational Analogy (CBR) Planning Algorithm

Under a case-based planning mode, Prodigy/Analogy [12] creates plans, interprets and stores planning episodes, and retrieves and reuses multiple past plans that are found similar to new problems. Stored plans are annotated with plan rationale so that, when the plans are retrieved in the future, new decisions can be guided and validated by the past rationale, hence avoiding inefficient search as can be the case in generative mode. The derivational-analogy strategy is to derive new solutions based on the *decision-making process* used in the past, rather than by adapting old *solutions* created in the past [3].

Figure 2 outlines the derivational analogy algorithm used by the system. When a new

1. Initialize.
2. Select a case to follow: follow the justifications and the selected strategy to merge the guidance from multiple past cases.
3. Get the relevant operators from the past cases.
4. Prune alternative failures from the current search path if the reasons for past failure hold.
5. Check syntactic operator-applicability by testing whether its left-hand side matches in the current state.
6. Check semantic applicability by determining whether the past reasons for their use still hold.
7. If choice not valid, choose a suitable action:
   - Suspend the guiding case if extra planning work needed to make choice valid:
     - Retrieve additional case
     - Or replan using domain theory.
   - Advance guiding case for past planning work that is not necessary.
   - Change the focus of attention by selecting another guiding case.

Fig. 2. A top-level view of the case-based planning algorithm.

problem is proposed, Prodigy/Analogy retrieves from the case library one or more problem solving episodes that may partially cover the new problem solving situation. The system uses a similarity metric that weighs goal-relevant features [14]. Essentially, it selects a set of past cases that solved subsets of the new goal statement. The initial state is partially matched in the features that were relevant to solving these goals in the past. Each retrieved case provides guidance to a set of interacting goals from the new goal statement. At replay time, a guiding case is always considered as a source of guidance, until all the goals it covers are achieved.

The general replay mechanism involves a complete interpretation of the justification structures annotated in the past cases in the context of the new problem to be solved. Equivalent choices are made when the transformed justifications hold. When that is not the situation, Prodigy/Analogy plans for the new goals using its domain operators adding new steps to the solution or skipping unnecessary steps from the past cases.

Prodigy/Analogy constructs a new solution from a *set* of guiding cases, as opposed to a single past case. Complex problems may be solved by resolving minor interactions among simpler past cases. However, following several cases poses an additional decision making step of choosing which case to pursue. Prodigy/Analogy includes several strategies to merge the guidance from the set of similar cases [13].

3 Visualization of the Plan and the Planning Process

One of the main goals underlying the design of the graphical user interface for PRODIGY is to provide a clear animation of the planning algorithm [2]. Given the planning algorithm outlined in the previous section, we discuss several features in the user interface that enable the visualization of the running of algorithm.

The user interface (UI) is implemented in Tcl/Tk, a scripting language which includes a set of Motif widgets (see [2] for additional implementational details). The user interface runs in a separate process from the planner, and the two communicate through Tcl/Tk's IPC mechanism. The planner currently runs in a Common Lisp process that listens for commands from two sources: the terminal running lisp and a socket connected to the user interface. This allows problems to be loaded or planning to be initiated from either the terminal or the interface. The interface makes use of a publicly available preprocessor for drawing directed graphs that represent parts of the domain, plan, and goal structure.

The original interface to PRODIGY is a simple command line interface from Lisp. The user loads a planning domain and a problem to solve, retrieves and loads past cases, and then calls the planner by invoking the run command. The output is composed of print statements at selected decision points in the search tree with indentation indicating search depth. The plan is printed as a list of instantiated operators, and then a brief statement reports success, running time, the number of search nodes expanded, and the number of solutions obtained.

Figure 3 shows the graphical user interface for PRODIGY (in CBR mode). The main control and display window is at the top of the figure, while two cases for replay are shown below. As with the direct command-line interface in Lisp, the user can load a domain and problem with the "Load" button, control program parameters through the "Control Variables" pull-down menu, and execute the planner with the "Replay" button (called the "Run" button in generative mode) or incrementally step through the execution using the "Step" button. During planning, the user can "Break" the execution to pause and then "Restart" to continue or "Abort" to halt the execution after a break. During execution of either generative or case-based planning, the current state of the plan is maintained in two major graphical windows (Tk canvases). The UI generates a view of the tail plan in the Goal Tree Display on the left, and the head plan is printed as a list of committed plan-

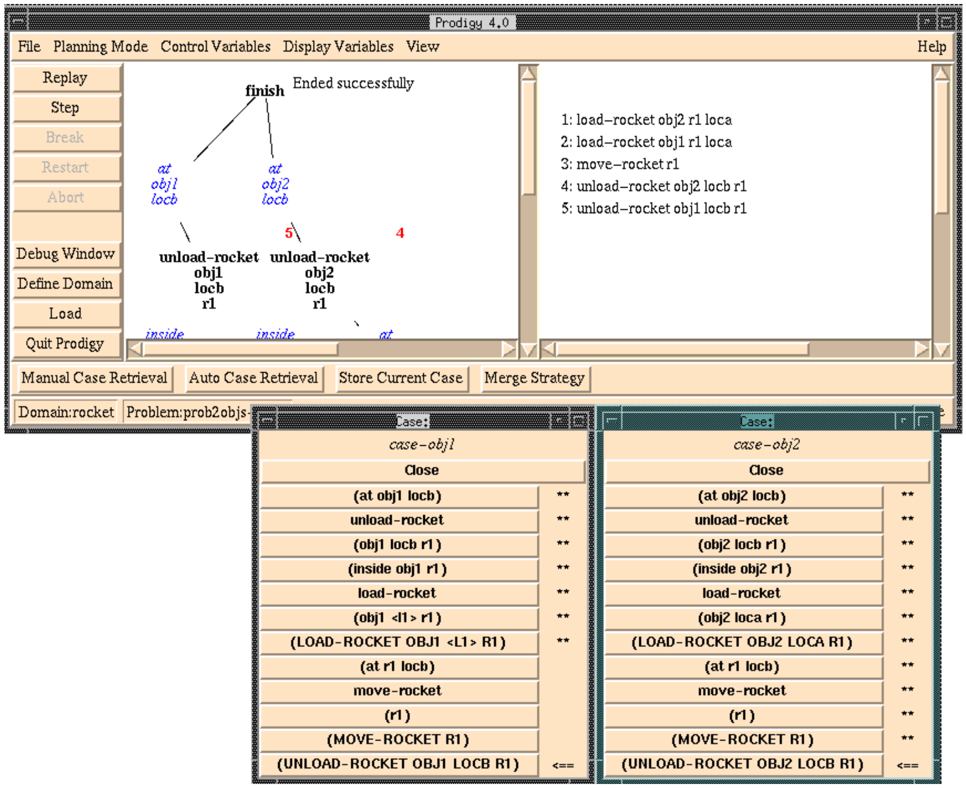

Fig. 3. Case replay, merge, and adaptation

ning steps in the Instantiated Operator Display on the right.

The segment of the goal tree displayed in Figure 3 shows that the user is finished with planning in the current problem in the one-way rocket domain if the two top goals are solved (i.e., objects 1 and 2 are at location `locb`). These goals can be solved if operator `unload-rocket` is executed twice while rocket `r1`containing both objects is at `locb`. The two previous cases below the main window show a merge of separate parts of the plan. Stars mark reused plan steps, the arrows mark current locations in the old plans, and blank spaces indicate skipped steps. The Instantiated Operator Display enumerates the steps of final merged plan. Although simplified due to space limitations here, the UI provides full access to PRODIGY to help the user visualize arbitrarily complex planning examples.

Selecting from the "Planning Mode" pull-down menu, the user can alternate control to and from a generative and case-based planning mode.[2] In response, a number of mode-specific buttons are arranged along the bottom of the window. The Prodigy/Analogy but-

2. Alternatively, the user can switch to a conditional-planning version that can use analogy along with a generative mechanism [1].

tons allow for retrieval (either automated or manual), storage, and merge selection. The user can visualize the merging procedure as it interleaves multiple cases, marks the steps that are used after successful validation, and skips the ones that are no longer necessary or are invalid. Guiding cases are displayed as shown in the two additional windows at the bottom of Figure 3.

Figure 4 shows the interface window for manual retrieval. The user is able to override the cases Prodigy/Analogy believes to be most similar to the new problem and to select one or more cases in the library for a particular domain. Cases are clicked for selection (or chosen with the "Select Case" button) and they are loaded in to the system via a "Load Selected Cases" button. The loading procedure establishes variable substitution according to the new goals and initial state and maps these values to the operator variables in the old cases.

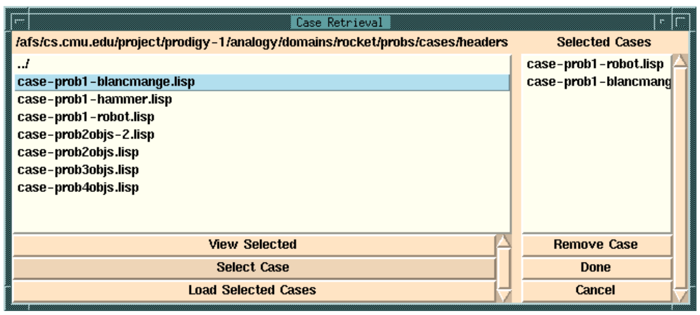

Fig. 4. Manual case retrieval

The UI attaches actions to the displayed graph nodes in the main window so that if the user clicks on them it will pass another message to the planner to display appropriate information. From the Goal Tree Display, the left mouse button causes the system to display the search tree node that relates justifications for the choice of that operator or goal (see Figure 5). Among other information, the sub-windows show that the choice of goal and operator bindings were made based on past experience linked to `case-obj2`. The middle (or right) mouse button causes the system to display the operator definition from the domain theory. From these opened windows the user can obtain the parent and children of a search tree node or information in the property lists of a given node.

The nodes also reveal other types of information about the rationale of the planner (whether human or automated). For example Figure 6 shows why some planning decisions fail.[3] Node number 25 fails dues to an attempt to re-establish a goal condition, while node 26 fails due to attempting to achieve a goal that is not possible (the rocket destination is not legal). Both of these failure justifications are saved to a case so that when following similar decisions in the future, the search path can be pruned *a priori*. In addition to these types of justifications, the cases represent decisions based on user selection, the application of control rules, failure due to state loops, and success due to goal achievement.

3. The failures shown are under generative mode and without control rules. Thus, variable bindings and goals are selected arbitrarily. Under case guidance this is less likely to occur given a good matching past case.

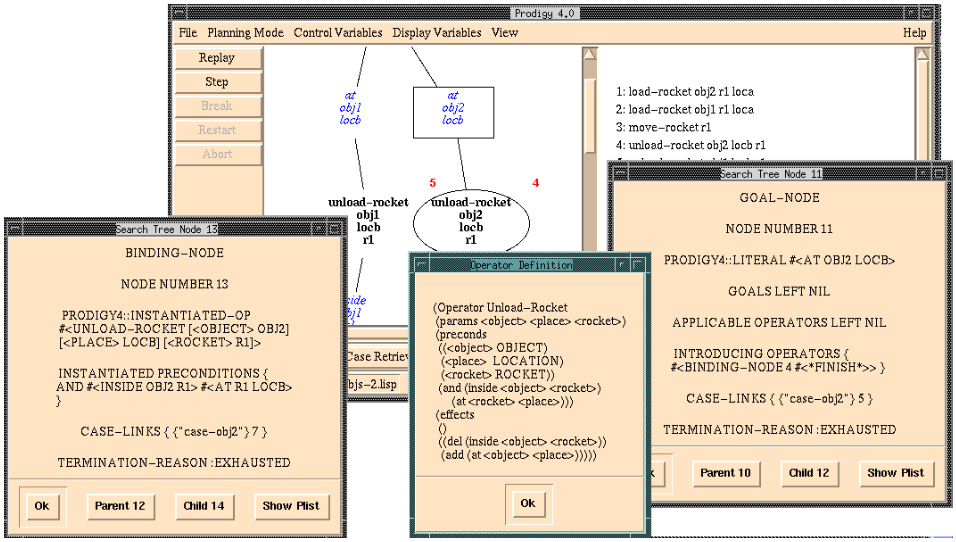

Fig. 5. Inspecting decision nodes from the goal tree

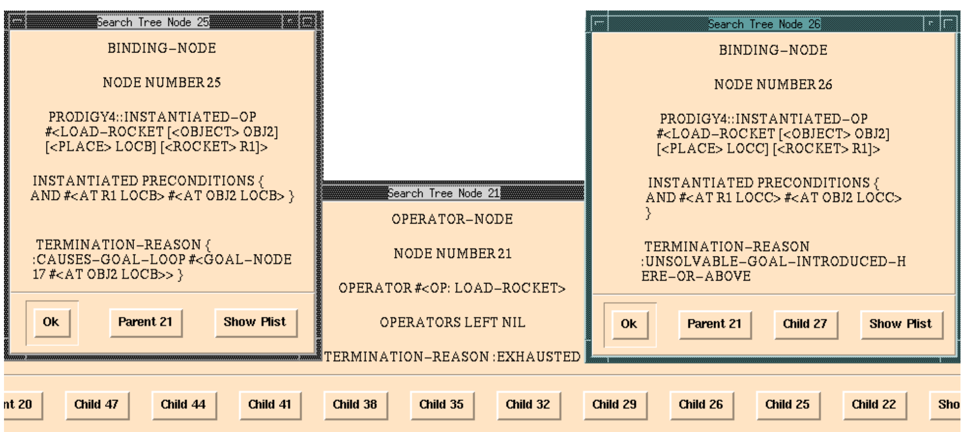

Fig. 6. Reasons for failed decisions.

4 Expertise in Planning Versus Expertise in Technology

At the current time, the interface to PRODIGY has been optimized for users who are familiar with the underlying planning system (e.g., the AI research community in planning). As such it provides a detailed level of access to the internal mechanisms, data representations, and plan structure. For example, the entire decision cycle described in Section 2 is open for inspection through the search-tree node windows. However, given that the development of the interface is still a work in progress, we have made substantial efforts to facilitate use by the more naïve user. A context sensitive help system is partially in place, the ease of execution of the planner has improved given menu selections to replace command-line switches, flags, and function calls, a substantial domain-specification subsystem exists, and the graphical display of the plan enhances plan visualization. Also, although not shown here (see [2] instead), the user can easily preempt automated planning decisions, making them manually instead. Finally, the system supports a concrete experiential basis for planning (i.e., planning is by cases, rather than strictly by first principles).

To further support the planner who is both a novice in the planning domain and knows little about planning formalisms, we have integrated Prodigy/Analogy with the ForMAT military-force deployment planner (see [16]). Using stored cases that match current planning problems, Prodigy/Analogy provides adaptation advice to the ForMAT user when modifying old deployment plans. In this mode, the user is remote across the internet and knows nothing about the manner in which Prodigy/Analogy works.

Finally, we are also working on a new *novice mode* that presents the output to the user in a more natural format. For example, instead of representing the predicates in prefix form in the goal tree display, infix is output so that is reads more like natural language. Likewise in this mode, we are substituting the technical data-structure terms in the justification windows with more intuitive language that describes their function.

5 Conclusion

This paper presents a case-based, mixed-initiative and extensible user interface built using Tcl/Tk that we developed for the PRODIGY architecture. Planning is a complex process and developing user planning interfaces is an important contribution for making implemented systems available to researchers, students, and practitioners. The user is able to switch between generative and analogical planning manually, or the interleaving of both modes can be left to automation. The interface provides mechanisms to save planning cases created generatively or analogically, to retrieve old cases that match current demands (either automatically or manually) and to choose various case interleaving strategies for adaptation and replay. The evolving plan is graphically presented to the user as a goal-tree structure, and justifications for automated choices is displayed upon demand.

The goals of the interface have been to facilitate the interaction between human and machine during the planning process by integrating both generative and case-based planning in the same framework, by providing plan visualization and plan rationale information, and by working toward a more flexible architecture that allows the user to settle to a personal level of equilibrium between too little information and too much. Because the user should not be arbitrarily subjected to the full technological details of the underlying planning technology, our ultimate aim is to develop a user view that abstracts the details of the underlying planner according to need. Our belief is that the immediate focus should be upon what the user sees (the interface), what the user wants (the goals), and what the user does (the task). Additional details should remain transparent to the user unless requested.

Acknowledgments

This research is sponsored as part of the DARPA/RL Knowledge Based Planning and

Scheduling Initiative under grant number F30602-95-1-0018. The authors thank Jason Reisman for programming the Tk code that invokes the manual case-retrieval function.

References

1. Blythe, J. & Veloso, M. M. (in press). Analogical Replay for Efficient Conditional Planning. To appear in the Proceedings of the Fourteenth National Conference on Artificial Intelligence. Menlo Park, CA: AAAI Press.
2. Blythe, J., Veloso, M. M., & de Souza, L. (1997). *The Prodigy user interface* (Tech. Rep. No. CMU-CS-97-114). Carnegie Mellon University, Computer Science Dept.
3. Carbonell, J. G. (1986). Derivational analogy: A theory of reconstructive problem solving and expertise acquisition. In R. Michalski, J. Carbonell & T. Mitchell (Eds.), *Machine learning: An artificial intelligence approach, Vol. 2* (pp. 371-392). San Mateo, CA: Morgan Kaufmann.
4. Cox, M. T., & Veloso, M. M. (1997). *Controlling for unexpected goals when planning in a mixed-initiative setting*. Submitted.
5. Ferguson, G., Allen, J. F., & Miller, B. (1996). TRAINS-95: Towards a mixed-initiative planning assistant. In *Proceedings of the Third International Conference on AI Planning Systems (AIPS-96)*, Edinburgh, Scotland, May 29-31, 1996.
6. Fink, E., & Veloso, M. M. (1996). Formalizing the Prodigy planning algorithm. In M. Gahllab & A. Milani (Eds.), *New Directions in AI Planning* (pp. 261-271). Amsterdam: IOS Press
7. Hammond, K. J. (1989). *Case-based planning: Viewing planning as a memory task. Vol. 1.* of *Perspectives in artificial intelligence*. San Diego, CA: Academic Press.
8. Kolodner, J. L. (1993). *Case-based reasoning*. San Mateo, CA: Morgan Kaufmann.
9. Oates, T. & Cohen, P. R. (1994). Toward a plan steering agent: Experiments with schedule maintenance. In *Proceedings of the Second International Conference on Planning Systems (AIPS-94) (pp. 134-139)*. Menlo Park, CA: AAAI Press.
10. Riesbeck, C. K., & Schank, R. C. (1989). *Inside case-based reasoning* (pp. 1-24). Hillsdale, NJ: Lawrence Erlbaum Associates.
11. Seifert, C. M., Hammond, K. J., Johnson, H. M., Converse, T. M., McDougal, T. F., & Vanderstoep, S. W. (1994). Case-based learning: Predictive features in indexing. *Machine Learning, 16*, 37-56.
12. Veloso, M. (1994). *Planning and learning by analogical reasoning.* Springer-Verlag.
13. Veloso, M. (1997). *Merge strategies for multiple case plan replay.* This volume.
14. Veloso, M., & Carbonell, J. G. (1993). Towards scaling up machine learning: A case study with derivational analogy in PRODIGY (pp. 233-272). In S. Minton (Ed.), *Machine learning methods for planning.* Morgan Kaufmann.
15. Veloso, M., Carbonell, J. G., Perez, A., Borrajo, D., Fink, E., & Blythe, J. (1995). Integrating planning and learning: The PRODIGY architecture*. Journal of Theoretical and Experimental Artificial Intelligence, 7*(1), 81-120.
16. Veloso, M. M, Mulvehill, A., & Cox, M. T. (in press). Rationale-supported mixed-initiative case-based planning. To appear in *Proceedings of the Ninth Annual Conference on Innovative Applications of Artificial Intelligence*. Menlo, Park, CA: AAAI Press.