Towards Mixed-Initiative Rationale-Supported Planning

Manuela M. Veloso

Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213-3891
veloso@cs.cmu.edu
http://www.cs.cmu.edu/~mmv

Abstract

This paper introduces our work on mixed-initiative, rationale-supported planning. The work centers on the principled reuse and modification of past plans by exploiting their justification structure. The goal is to record as much as possible of the rationale underlying each planning decision in a mixed-initiative framework where human and machine planners interact. This rationale is used to determine which past plans are relevant to a new situation, to focus user's modification and replanning on different relevant steps when external circumstances dictate, and to ensure consistency in multi-user distributed scenarios. We build upon our previous work in Prodigy/Analogy, which incorporates algorithms to capture and reuse the rationale of an automated planner during its plan generation. To support a mixed-initiative environment, we have developed user interactive capabilities in the Prodigy planning and learning system. We are also working towards the integration of the rationale-supported plan reuse in Prodigy/Analogy with the plan retrieval and modification tools of ForMAT. Finally, we have focused on the user's input into the process of plan reuse, in particular when conditional planning is needed.

Introduction

Our work within the ARPA planning initiative aims at developing a prototype system for mixed-initiative, rationale-supported planning. We investigate methods for capturing the machine-based and human-based rationale underlying the decision-making process during planning. We are developing techniques to store the derivation of the contingency plans, attaching the justifications for the successful and rejected choices during the mixed-initiative plan generation in addition to simply retaining the solution plan steps. This is being accomplished by extension of the Prodigy/Analogy derivational-trace extraction mechanism and combining it with conditional planning. In essence, the planner will provide on-demand justification for its decisions, and will permit reuse of past planning so long as the rationale remains valid in the instance situation.

Our work has pursued along three different fronts: (1) extending Prodigy/Analogy to incorporate user's rationale-capture and input; (2) providing visualization for plan generation, reuse, and modification; (3) combining analogical reasoning with conditional planning.

This paper is organized along these three directions. It should be seen as an entry point to the technical results we have achieved and approaches we are developing. Citations are given on the appropriate technical papers for follow up. In Section 2 we overview briefly Prodigy/Analogy, we discuss the user rationale-capture approach under development, and we identify the opportunities for user's input in the modification process during plan reuse. Section 3 presents the user interface we have developed in the Prodigy planning and learning system. We focus on presenting how the user can control the planning choices and on the visualization support for Prodigy/Analogy. Section 4 discusses the integration of analogical reasoning with Prodigy's conditional planning module. Finally, in Section 5 we draw conclusions and discuss our current working research directions.

Prodigy/Analogy: Combining Generative and Case-based Planning

Prodigy is an architecture for the study of the combination of planning and learning (Carbonell, Knoblock, & Minton 1990; Veloso *et al.* 1995). Different learning approaches are developed for learning planning domain models, and control knowledge for improving search efficiency and plan quality.

Prodigy/Analogy (Veloso 1994b; 1994a) achieves the integration of analogical reasoning into general problem solving as a method of learning at the strategy level to solve problems more effectively. Improvement in planning efficiency occurs by the generation, storage, retrieval, and replay of annotated derivational traces of problem solving episodes. Instead of investing substantial effort deriving general rules of behavior to apply to individual decisions, Prodigy/Analogy compiles complete problem solving cases that are used to guide future similar situations. Learned knowledge is flexibly applied to new problem solving situations even if only a partial match exists among problems.

Figure 1 illustrates the complete planning cycle of Prodigy/Analogy. Generative planning using an operator-based planner, Prodigy4.0 (Carbonell *et al.* 1992), is used if additional planning is needed when replaying the past planning cases.

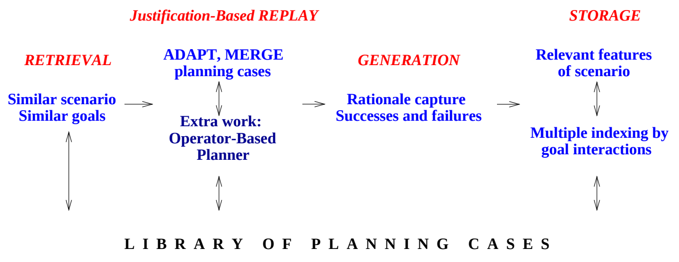

Justification-Based REPLAY
STORAGE
RETRIEVAL
ADAPT, MERGE planning cases
GENERATION
Relevant features of scenario
Similar scenario Similar goals
Extra work: Operator-Based Planner
Rationale capture Successes and failures
Multiple indexing by goal interactions
LIBRARY OF PLANNING CASES

Figure 1: Prodigy/Analogy: Retrieval, replay, generation, and storage of planning cases. An operator-based planner is used as a generative planner when additional planning is needed.

Generation of planning episodes

Reasoning by analogy in Prodigy/Analogy consists of the flexible reuse of derivational traces of previously solved problems to guide the search for solutions to similar new problems, avoiding a completely new search effort. Transformational analogy and most CBR systems reuse past solutions by modifying (*tweaking*) the retrieved final solution as a function of the differences found between the source and the target problems. Derivational analogy instead is a *reconstructive* method by which *lines of reasoning* are transferred and adapted to a new problem (Carbonell 1986) as opposed to only the final solutions.

Automatic generation of the derivational episodes that become the planning cases occurs by extending the base-level planner with the ability to examine its internal decision cycle, recording the justifications for each decision during its search process. Prodigy/Analogy has been re-implemented in Prodigy4.0, a state-space nonlinear planner (Carbonell *et al.* 1992; Fink & Veloso 1994).

Prodigy4.0's planning reasoning cycle involves several decision points, namely: the *goal* to select from the set of pending goals; the *operator* to choose to achieve a particular goal; the *bindings* to choose in order to instantiate the chosen operator; *apply* an operator whose preconditions are satisfied or continue *subgoaling* on a still unachieved goal. Prodigy/Analogy extends Prodigy4.0 with the capability of recording the context in which the decisions are made. Figure 2 shows the skeleton of the decision nodes. We created a language for the slot values to capture the reasons that support the choices (Veloso & Carbonell 1993a).

There are mainly three different kinds of justifications: links among choices capturing the subgoaling structure (slots `precond-of` and `relevant-to`), records of explored failed alternatives (the `sibling-` slots), and pointers to any applied guidance (the `why-` slots). A stored problem solving episode consists of the successful solution trace augmented with these annotations, i.e., the derivational trace.

Goal Node	Chosen Op Node	Applied Op Node
:step	:step	:step
:sibling-goals	:sibling-ops	:sibling-goals
:sibling-appl-ops	:why-this-op	:sibling-appl-ops
:why-subgoal	:relevant-to	:why-apply
:why-this-goal		:why-this-op
:precond-of		:chosen-at

Figure 2: Justification record structure. Nodes are instantiated at decision points during problem solving. Each learned episode is a sequence of such justified nodes.

Mixed-initiative rationale capture The generation of planning cases by Prodigy/Analogy captures the rationale of the machine-based planning. We have been addressing two different aspects of a mixed-initiative planning framework.

In a simpler approach, we assume that the user is familiar with the Prodigy planning cycle. We provide user interactive capabilities to allow a user to choose particular alternatives in the cases and plan merging strategies (see section below). Justifications provided by the user in general are selected from a set of available ones. If the user enters an unknown reason to the system, its functionality needs to be specified for validation at reuse time.

We are engaged in a technology integration experiment (TIE) with MITRE to address a more realistic user-based planning scenario. MITRE's ForMAT system provides a plan acquisition tool that allows users to generate and save force deployment plans (Mulvehill 1996). Stored plans are long lists of actions that are recorded without the underlying rationale structure. ForMAT incorporates mechanisms for the user to provide some organizational structure. It can also infer weighted functional information about different force modules of the plans.

The TIE will combine ForMAT for its user-based planner and for its case library with Prodigy/Analogy for its rationale-driven reuse capabilities. Rationale-supported reuse relies on the reasons for plan step selection, the actual goals to be achieved, and the alternative steps that were or should be considered. We are developing ways to extract dependency and ordering information from ForMAT plans. We recognized the need to augment the stored plans with the goals of the constituent force modules. Currently, we have a small case library of force deployment plans which includes a high-level definition of the user's rationale associated with the different force modules. This information is converted into Prodigy/Analogy goal-based planning case representation.

Replay of multiple guiding cases

When a new 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 (Veloso & Carbonell 1993b). In a nutshell, 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.

The replay functionality transforms the planner, from a module that costly generates possible operators to achieve the goals and searches through the space of alternatives generated, into a module that tests the validity of the choices proposed by past experience and follows equivalent search directions. The replay procedure provides the following benefits to the problem solving procedure:

- Proposal and validation of choices versus generation and search of alternatives.
- Reduction of the branching factor – past failed alternatives are pruned by validating the failures recorded in the past cases; if backtracking is needed PRODIGY/ANALOGY backtracks also in the guiding cases – through the links established during replay – and uses information on failure to make more informed backtracking decisions.
- Subgoaling links identify the subparts of the case to replay – the steps that are not part of the active goals are skipped.

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. We explored several strategies to merge the guidance from the set of similar cases. We have experimented with exploratory and informed merging strategies. In the exploratory approach, choices are made arbitrarily when there is no other guidance available. This strategy allows an innovative exploration of the space of possible solutions leading to opportunities to learn from new goal interactions or operator choices.

We have applied the rationale-supported planning framework of Prodigy/Analogy in several domains including a realistic route planning domain using the real map of the Pittsburgh city (Haigh, Veloso, & Shewchuk 1996; Haigh & Veloso 1995).

Mixed-initiative plan reuse and modification There are several opportunities for user intervention in plan reuse. Before the actual plan reuse episode, the user may be involved in the retrieval process. In our TIE, ForMAT provides useful tools to allow users to query the library of planning cases until a sufficiently similar past scenario is found. Retrieval in Prodigy/Analogy is driven by goal and initial scenario similarities. We combine the functional querying in ForMAT with the goal information for efficient retrieval.

The replay procedure in Prodigy/Analogy includes role substitutions at the plan step level. It processes not only the substitutions returned by the retrieval procedure, but also the ones identified during the justification interpretation during the actual plan reuse. This avoids the user's effort to identify the implications of the partial match between the past and new planning scenarios.

During reuse, Prodigy/Analogy needs to modify and adapt the new plans. In the fully automated system, as described above, when extra planning is needed, the generative operator-based planner is invoked to plan for the additional goals encountered. In a realistic mixed-initiative framework, it is quite possible that the generative planner does not have the domain knowledge for extra planning needed. In this case, Prodigy/Analogy invokes ForMAT which accesses its case library or supports the user for plan building for the new goals. Within this system integration, Prodigy/Analogy will provide informed removal and insertion of new plan modules by following the rationale that links the different parts of the plans.

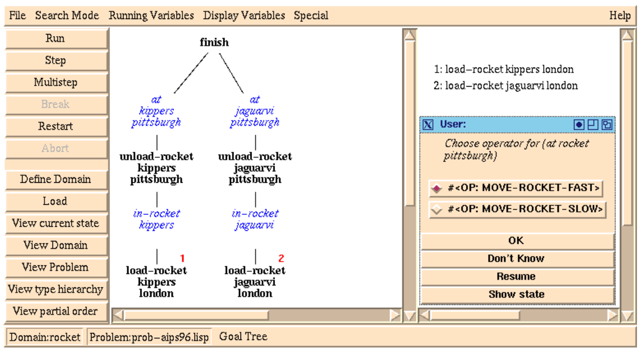
User Interface

Along the mixed-initiative paradigm, we have designed and developed a user interface for the Prodigy planning and learning system (Blythe, Veloso, & de Souza 1996). The interface supports the process of both building and running a planning domain. It was designed to be highly modular, requiring no changes to the underlying planner's code, and extensible, so that interfaces to modules added to the planner may could easily be integrated into the interface. When desired, a human user can step through and interrupt the planning process, as well as provide choices for the planning decisions. The user interface supports planning by reuse of past planning episodes and probabilistic planning.

Since Prodigy is an on-going research project, it was important to develop an interface that could be integrated easily with any variant of the system and with its different learning modules. This requirement led us to seek an architecture that is modular in two ways. First, the code for the planner should not need to be modified for the user interface to run. Second, the user interface itself should make very few assumptions about the planner's implementation. At the same time the interface should have a tight integration with the planner so that the planning process can be traced graphically and the interface can be used to interrupt and direct the planner while it is in operation. The architecture used for the PRODIGY user interface accomplishes these goals and the implementation details can be found in (Blythe, Veloso, & de Souza 1996). In this paper, we focus on describing briefly how the user can control the planning choices and on the visualization support for Prodigy/Analogy.

Control of planning choices

PRODIGY is a completely *open-controllable* architecture. This means that all the decision points are open to be controlled explicitly usually through control rules which can automatically dictate particular choices based on the planning scenario. Figure 3 illustrates a particular use of control rules that interrupt the planner to ask the user for guidance.



1: load–rocket kippers london
2: load–rocket jaguarvi london

Choose operator for (at rocket pittsburgh)

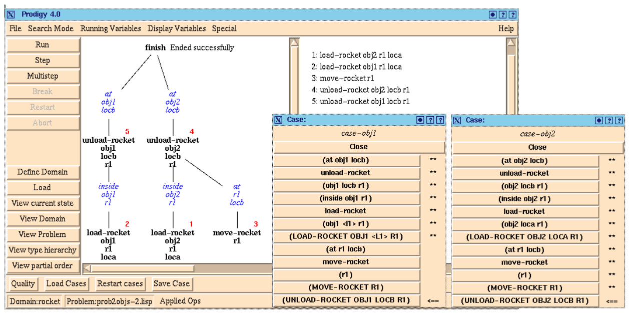
Figure 3: The user can control choices. In the right window, the user is shown the operator choices available to achieve the goal Prodigy is planning for.

The user is prompted with the choices available and can either select one of the alternatives, decide that doesn't know which one to select, or tell PRODIGY not to ask for any more guidance and resume its autonomous behavior.

The interface for Prodigy/Analogy

To reuse past planning experience to solve new complex planning situations, it may be necessary to reuse multiple past planning episodes. Planning becomes therefore a merging process of multiple plans, probably simpler and complementary. An automated merging procedure compares interactions in the past with the new scenario. In the new situation, the planner is capable of: reordering actions, adding new needed actions and deleting old actions as a function of possibly new or non-existing goals in the new scenario. We have developed a series of methods to merge different planning episodes which reason about different degrees of context information to guide their merging process. In particular, a user-driven merging procedure has been developed. We have created a simple language that allows to interact with the user. This is currently under further refinement to facilitate the user's task of combining and updating past plan rationale in new contexts.

Prodigy/Analogy's running mode can be selected from the interface. The user can solve problems and request that the planning episodes be stored. A new problem can then be solved by replaying and and merging possibly multiple past planning cases. Figure 4 shows a snapshot of the interface where one case is instantiated to guide two different goals.



1: load–rocket obj2 r1 loca
2: load–rocket obj1 r1 loca
3: move–rocket r1
4: unload–rocket obj2 locb r1
5: unload–rocket obj1 locb r1

case–obj1	
(at obj1 locb)	**
unload–rocket	**
(obj1 locb r1)	**
(inside obj1 r1)	**
load–rocket	**
(obj1 <l1> r1)	**
(LOAD-ROCKET OBJ1 <L1> R1)	**
(at r1 locb)	
move–rocket	
(r1)	
(MOVE-ROCKET R1)	
(UNLOAD-ROCKET OBJ1 LOCB R1)	<==

case–obj2	
(at obj2 locb)	**
unload–rocket	**
(obj2 locb r1)	**
(inside obj2 r1)	**
load–rocket	**
(obj2 loca r1)	**
(LOAD-ROCKET OBJ2 LOCA R1)	**
(at r1 locb)	**
move–rocket	**
(r1)	**
(MOVE-ROCKET R1)	**
(UNLOAD-ROCKET OBJ2 LOCB R1)	<==

Figure 4: A snapshot of the setup used by the analogical reasoning module. Guiding cases are displayed as shown in the two windows at the right. The steps reused are marked while the steps not needed in the new situation are skipped.

The user can visualize the merging procedure, as it interleaves the multiple cases, marks the steps that are used after successful validation, and skips the ones that are no longer necessary or are invalid.

Using Analogy in Conditional Planning

Given the uncertainty of planning in complex situations, we also developed a method to combine analogical replay with conditional probabilistic planning (Blythe 1994; Blythe & Veloso 1996).

Recently, several planners have been designed that can create conditionally branching plans to solve problems which involve uncertainty. These planners represent an important step in broadening the applicability of AI planning techniques, but they typically must search a larger space than non-branching planners, since they must produce valid plans for each branch considered. In the worst case this can produce an exponential increase in the complexity of planning. If conditional planners are to become usable in real-world domains, this complexity must be controlled by sharing planning effort among branches. We therefore argue that analogical plan reuse should play a fundamental role in this process. We have implemented a conditional probabilistic planner that uses analogical replay to derive the maximum benefit from previously solved branches of the plan. This approach provides valuable guidance for when and how to merge different branches of the plan and exploits the high similarity between the different branches, which have the same goal and typically a very similar state. Our experiments have been showing that analogical replay significantly reduces the complexity of conditional planning.

Through its replay functionality, Prodigy/Analogy showed to be suitable to be combined with C-Prodigy, the conditional planner developed by Blythe. Table 1 introduces how conditional planning controlled by the probabilistic planner Weaver (Blythe 1994), is integrated with Prodigy/Analogy. It is interesting to notice the smoothness of this integration which is made possible by the common underlying framework of the Prodigy planning and learning system.

Procedure C-Prodigy-Analogy

1. Weaver and Analogy select which previously visited branch should be used to guide the new planning of conditional planning.
2. Based on the selected branching point, Analogy sets up accordingly the starting guiding point of the past planning episode.
3. C-Prodigy plans for the new branch guided by Analogy.
4. The integrated C-Prodigy/Analogy new planning episode proceeds in the same way as the usual Prodigy/Analogy: in this new context, previous decisions are followed if valid, unnecessary steps are not used, and new steps are added when needed. This process is equivalent in either of the two situations described above identified by Weaver. Whether C-Prodigy is trying to protect a potentially unsuccessful step, or replanning for a new branch, Analogy guides the new planning process based on the high similarity between the global current and past situations. This is particularly well suited for the analogical replay guidance and leads in general to minor interactions and a major sharing of the past planning effort.

Table 1: Top-level view of the integration of analogy in conditional probabilistic planning.

In general, Prodigy/Analogy allows for the replay of any number of past cases that are found to be jointly similar to a new planning situation. The planning cases are retrieved from a library of cases, similarities are evaluated, and appropriate substitutions are performed.

In this integration of conditional planning and analogy, the analogical replay within the context of different branches of the same problem can be better framed as an instance of internal analogy (Hickman, Shell, & Carbonell 1990). In fact, the accumulation of a library of cases is not required, and there is no need for an elaborated analysis of similarity between a new problem and a potentially large number of different cases. The branches of the problem need only to be cached in working memory and most of the objects do not need to be mapped into new objects, as the context remains the same. In our implementation, we set up Prodigy/Analogy with the ability to follow this same internal-analogy philosophy. However the full analogical reasoning paradigm leaves us with the freedom to reuse branches across different problems in the same domain. We could also find the need to merge different branches in a new situation. Our current implementation leaves these possibilities well open, as they are part of the functionality of the general Prodigy/Analogy.

Table 2 presents the analogical reasoning procedure combined with conditional planning. We follow a single case corresponding to the planning for the last branch visited according to the replanning order selected by Weaver.

```
procedure c-prodigy-analogical-replay
1. Let $C$ be the guiding case;
     and $C_i$ is the guiding step in the case.
2. The initial case step $C_0$ is set at the branching point.
3. Let $i = 0$.
4. Terminate if the goal state is reached.
5. Check which type of decision is $C_i$:
6. If $C_i$ adds a step $O_g$ to the head plan,
7.   If $O_g$ can be added to the current head plan
     and no tail planning is needed before,
8.     then Replay $C_i$; Link new step to $C_i$; goto 14.
9.     else B-planning plans for the new goals; goto 5.
10.If $C_i$ adds a step $O_g$ to the tail plan,
11.  If the step $O_g$ is valid,
12.    then Replay $C_i$; Link new step to $C_i$; goto 15.
13.    else Hold the case (if other planning needed), or
       Mark unusable all steps dependent of $C_i$; goto 14.
14.Advance the case to the next usable step $C_j$;
15.$i \leftarrow j$; goto 5.
```

Table 2: Sketch of the analogical replay procedure combined with conditional planning.

The adaptation in the replay procedure involves a validation of the steps proposed by the case. There may be a need to diverge from the proposed case step, because new goals exist in the current branch (step 9). There may be also the case that some steps in the old branch can be skipped, as they may be already true in the new branching situation (step 13). Steps 8 and 12 account for the sharing between different branches. These steps also account for the most part of the new planning, as the state is only slightly different and most of the goals are the same across branches. This selective use of replay controls the combinatorics of conditional planning.

Conclusion

We presented methods to extend Prodigy/Analogy to capture the user's rationale in a mixed-initiative framework. We have briefly described our technology integration experiment with MITRE's ForMAT case-based planning system. We have identified several opportunities where the retrieval and modification mechanisms of ForMAT can be merged with the rationale-supported reuse of Prodigy/Analogy. Our work has been based on ForMAT's force deployment plans and ACP analogs created at MITRE.

We presented a few of the user interactive capabilities that we developed in a user interface for the Prodigy system, including for Prodigy/Analogy. To address issues of efficient conditional planning to handle uncertainty, we have also integrated Prodigy/Analogy with C-Prodigy, Prodigy's conditional planner developed by Blythe.

The paper summarized our directions of research within the ARPI planning initiative. The paper reports on our on-going work towards efficient mixed-initiative rationale-supported planning.

Acknowledgements

This research is sponsored in part by the Advanced Research Projects Agency (ARPA) and the Rome Laboratories (RL) as part of the ARPA/RL Knowledge Based Planning and Scheduling Initiative the under grant number F30602-95-1-0018. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Rome Laboratories or the U.S. Government.

Thanks to Jim Blythe, Luiz de Sousa, and Karen Haigh for their help in the development of the user interface. The work on using analogy in conditional planning has been done jointly with Jim Blythe. Thanks also to Jaime Carbonell for his suggestions and discussions on this work. Finally, special thanks to Alice Mulvehill and Steven Christey from MITRE, and Michael Cox for our TIE with ForMAT.

References

Blythe, J., and Veloso, M. 1996. Using analogy in conditional planners. Technical Report forthcoming, Computer Science Department, Carnegie Mellon University.

Blythe, J.; Veloso, M.; and de Souza, L. E. 1996. The Prodigy user interface. Technical Report forthcoming, Computer Science Department, Carnegie Mellon University.

Blythe, J. 1994. Planning with external events. In de Mantaras, R. L., and Poole, D., eds., *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, 94–101. Seattle, WA: Morgan Kaufmann.

Carbonell, J. G.; Blythe, J.; Etzioni, O.; Gil, Y.; Joseph, R.; Kahn, D.; Knoblock, C.; Minton, S.; Pérez, A.; Reilly, S.; Veloso, M.; and Wang, X. 1992. PRODIGY4.0: The manual and tutorial. Technical Report CMU-CS-92-150, Department of Computer Science, Carnegie Mellon University.

Carbonell, J. G.; Knoblock, C. A.; and Minton, S. 1990. Prodigy: An integrated architecture for planning and learning. In VanLehn, K., ed., *Architectures for Intelligence.* Hillsdale, NJ: Erlbaum. Also Technical Report CMU-CS-89-189.

Carbonell, J. G. 1986. Derivational analogy: A theory of reconstructive problem solving and expertise acquisition. In Michalski, R. S.; Carbonell, J. G.; and Mitchell, T. M., eds., *Machine Learning, An Artificial Intelligence Approach, Volume II*, 371–392. Morgan Kaufman.

Fink, E., and Veloso, M. 1994. PRODIGY planning algorithm. Technical Report CMU-CS-94-123, School of Computer Science, Carnegie Mellon University.

Haigh, K. Z., and Veloso, M. 1995. Route planning by analogy. In *Case-Based Reasoning Research and Development, Proceedings of ICCBR-95*, 169–180. Springer-Verlag.

Haigh, K. Z.; Veloso, M. M.; and Shewchuk, J. 1996. Exploring geometry in analogical route planning. *Submitted to Journal of Experimental and Theoretical Artificial Intelligence.*

Hickman, A. K.; Shell, P.; and Carbonell, J. G. 1990. Internal analogy: Reducing search during problem solving. In Copetas, C., ed., *The Computer Science Research Review 1990.* The School of Computer Science, Carnegie Mellon University.

Mulvehill, A. M. 1996. Building, remembering, and revising force deployment plans. In *In this same volume.*

Veloso, M. M., and Carbonell, J. G. 1993a. Derivational analogy in PRODIGY: Automating case acquisition, storage, and utilization. *Machine Learning* 10:249–278.

Veloso, M. M., and Carbonell, J. G. 1993b. Towards scaling up machine learning: A case study with derivational analogy in PRODIGY. In Minton, S., ed., *Machine Learning Methods for Planning.* Morgan Kaufmann. 233–272.

Veloso, M.; Carbonell, J.; Pérez, M. A.; Borrajo, D.; Fink, E.; and Blythe, J. 1995. Integrating planning and learning: The PRODIGY architecture. *Journal of Experimental and Theoretical Artificial Intelligence* 7(1):81–120.

Veloso, M. M. 1994a. Flexible strategy learning: Analogical replay of problem solving episodes. In *Proceedings of Twelfth National Conference on Artificial Intelligence*, 595–600. Seattle, WA: AAAI Press.

Veloso, M. M. 1994b. *Planning and Learning by Analogical Reasoning.* Springer Verlag. (Monograph of Ph.D. thesis, Carnegie Mellon University, 1992).