Mixed-Initiative Case Replay

Michael T. Cox

Department of Computer Science & Engineering
Wright State University
Dayton, OH 45435-0001
{mcox}@cs.wright.edu

Abstract

Mixed-initiative case replay introduces an active human into the case-based planning process. The goals of this novel technique are to utilize the strengths of machine-based case replay to improve human performance and to allow a human to override a machine's abstract representation of an actual domain. The synthesis of these two approaches in a planning domain enables flexible solutions to planning problems. This preliminary research introduces for the first time a human-centered technique into the well-developed planning method of derivational analogy.

Introduction

A mixed-initiative system integrates an intelligent system and an active user with the goal of producing results better than either the system or the user could alone. However many researchers concentrate on either the "mixed" side or the "initiative" side of the equation. Persons involved in mixed-initiative dialogue systems (e.g., Haller, McRoy, and Kobsa 1999) often worry most about issues such as the shift of control from one agent (either machine or human) to the other as is in the case of natural language dialogue. Persons involved in the mixed-initiative planning community (e.g., Burstein 1994) often stress the combination of human and machine abilities in a static relationship between the two during the planning process. Here we examine a new method within case-based reasoning that balances the focus between the two aspects of mixed-initiative computation. It adds to the emerging area of mixed-initiative case-based reasoning (Aha 2002).

Mixed-initiative case replay introduces a human into the case replay portion of case-based planning. By having a human in the case replay process we can gain certain advantages such as knowledge outside the transitive closure of the domain representation, improvements in search or plan results, and the ability to repair defective cases. In particular we consider two both novices and experts. Novices have little to no domain knowledge, and so require help in producing "good" plans. This assistance in case-based reasoning is found in sound prior cases. The interactive replay allows a novice user to see the reasoning used to solve an old case and to adapt it to the current problem. An expert user can generate plans that are more acceptable than that of the automated planner although the planner may be "correct." The expert user can also determine when and how to change a given goal, something that can be difficult to codify within a domain description.

Description of the PRODIGY Planner

PRODIGY (Carbonell *et al*. 1992; Veloso *et al*. 1995) is a domain independent non-linear state-space planning and learning architecture. The current core planner is Prodigy 4.0. In generative mode it uses a four-step decision process to solve planning problems. These control points are as follows.

- It selects a goal to solve from a list of candidate goals
- It selects an operator from a list of candidate operators that can achieve the selected goal
- It selects object bindings for open operator variables from a set of candidate bindings.
- If instantiated operators exist having no open preconditions and pending goals also exist, it chooses either to apply the instantiated operator (forward chain) or to continue subgoaling (backward chain).

This process repeats until a complete sequence of operators (a plan) is found or until the search space is exhausted. Using the Prodigy 4.0 User Interface 2.0 (Cox and Veloso 1997), a human can be involved in each step of the four-step process.

Prodigy/Analogy (Veloso 1994a; 1994b) is a case-based reasoner built on top of the Prodigy 4.0 core generative planner of the PRODIGY system. It uses past plans, stored as annotated cases, to solve a current problem using the technique of derivational replay (Carbonell 1986). It does this by retrieving a case that best matches the current problem from a set of all possible cases in the case repository. The retrieval criteria are based on the goal coverage that the case provides as well as the coverage of the new initial state by the footprint (Veloso 1994b) of the past states responsible for the past success. Once a case is retrieved, Prodigy/Analogy replays the past case by re-using the search tree created in the past solution. But derivational replay also checks the annotated justifications of the past plan (i.e., the plan derivation) to guarantee that the previous conditions responsible for success still hold.

Object bindings are rectified using a substitution set to match the current problem. Substitution takes the objects and goals defined in an old problem and maps them to objects and goals in the current problem. If a deviation from the case is necessary (perhaps because an operator or state is not valid in the current problem), Prodigy/Analogy can prune irrelevant steps and add new ones. If a case must be abandoned, the system will switch to a generative mode to finalize the planning process.

The introduction of mixed-initiative case replay into Prodigy/Analogy required many changes. We modified Prodigy 4.0 to more fully support tracking of user interaction in plan generation and case storage. We also modified Prodigy/Analogy to include user interaction in the replay of cases. In particular when the user selects a goal binding or operator, an annotation is made in the search node that user interaction occurred. The annotation also includes the particular choice that the user selected. The implementation, called *Prodigy/AnalogyMI*, has also been modified to prompt the user only on human-made operator selections that occurred in the past case. We also modified it to enter a generative mode that continues user interaction if case-based reasoning fails.

Mixed-Initiative Case Replay

To illustrate mixed-initiative case replay, consider a user solving a simple problem in a kitchen domain. In this domain a variety of everyday actions can occur such as preparing and eating food, washing dishes, and storing/removing items from cabinets. Consider a problem in which an agent (Seamus) is hungry and wants to be made full. Two operators exist that can achieve this goal: EAT-BAD-SNACK or EAT-GOOD-SNACK. The domain associates multiple potential bindings with each operator variable, but it incudes constraints to filter the candidates. EAT-BAD-SNACK uses any available junk-food object, and eat-good-snack uses any available fruit object. EAT-GOOD-SNACK requires an extra operator to be used to satisfy its preconditions (slicing, washing, or peeling fruit). A sizeable search space can exist for problems with even a single goal when numerous binding choices are considered. Moreover, determining criteria for operator selections is not a trivial task, and in this particular case, may be impossible to determine with any degree of certainty before planning commences. It is impossible to know whether to use EAT-GOOD-SNACK or EAT-BAD-SNACK, because this is a choice that requires much knowledge about the problem. This knowledge is best added by the user implicitly, rather than explicitly building an overly complex domain to solve a relatively simple problem. It is therefore desirable to have a human involved in the planning process because the domain is subject to interpretation.

In generative mode with *no* user control, Prodigy 4.0 selects EAT-BAD-SNACK to meet the goal (see Figure 1[1]). This is a valid solution, but not necessarily desired or even correct when applied to a particular user. However, the Prodigy 4.0 User Interface Version 2.0 (Cox and Veloso 1997) allows users to interact with the automated planner, thereby allowing a user to determine choices at three of the four control points in the decision process described above.

In generative mode *with* user control, the user can select the alternative actions EAT-GOOD-SNACK(granny-smith) and WASH(granny-smith) with the granny-smith apple as an operator variable binding. This produces a satisfactory plan for the user at the current time. Perhaps the user wanted a healthy snack, and decided a quick snack would be to eat a washed apple. This plan is now stored as an old case in the plan library. In this instance, the user applied knowledge that is outside the transitive closure of the domain representation to the solution of the problem. This information yields a plan that more accurately reflects the idiosyncratic desire for healthy nutrition.

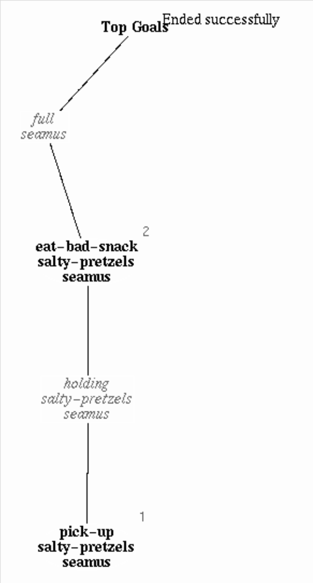
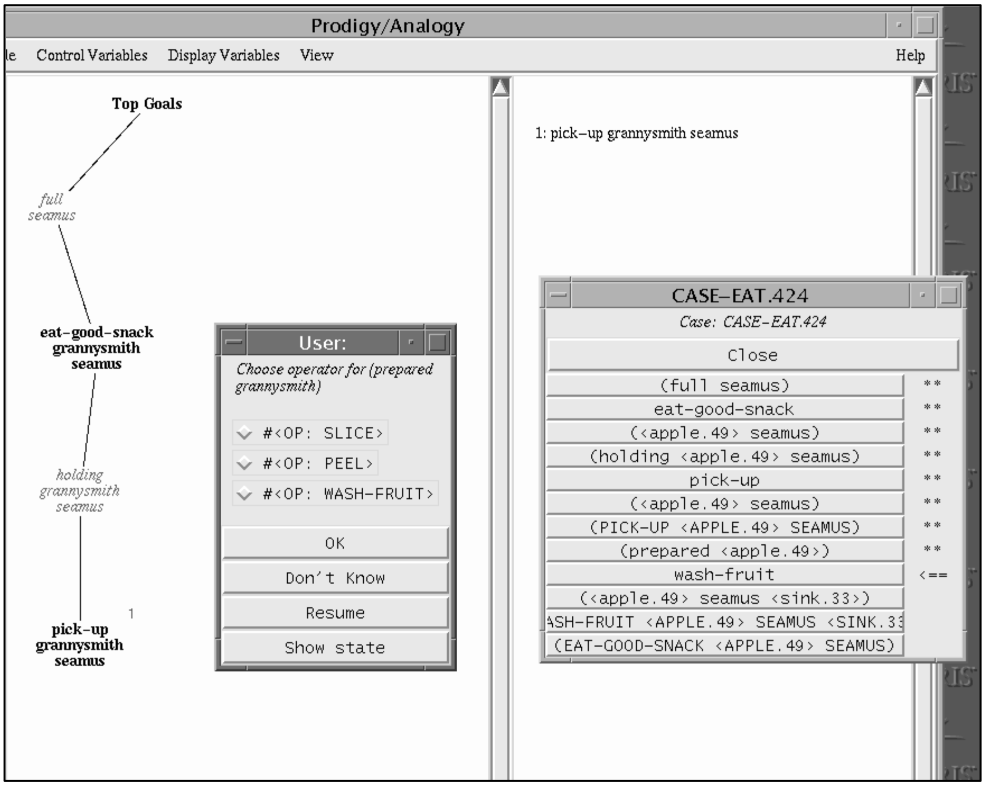


Figure 1. Plan created generatively, no user interaction

As noted above, the standard PRODIGY user-interface allows user interaction during generative mode, but does not during case replay. Normally Prodigy/Analogy would retrieve an old case such as the one created to eat the granny-smith, and if the old justification for choosing a step such as WASH still exists, then it will automatically repeat the step, even if other steps might be more attractive given a new situation or changing tastes. But instead of automatic replay, Prodigy/AnalogyMI allows user intervention throughout the derivational case-adaptation process. This differs from the standard case replay in that the user is

[1] Figures 1 through 3 are snapshots from PRODIGY's user interface. The graphs are goal-subgoal graphs. Blue nodes represent goal states, whereas black nodes represent operators. Red numbers indicate that the instantiated operator has been applied to the state, and the number itself corresponds to the plan step number.



1: pick-up grannysmith seamus

Choose operator for (prepared grannysmith)

#<OP: SLICE>
#<OP: PEEL>
#<OP: WASH-FRUIT>

Case: CASE-EAT.424

```
(full seamus)                          **
eat-good-snack                         **
(<apple.49> seamus)                    **
(holding <apple.49> seamus)            **
pick-up                                **
(<apple.49> seamus)                    **
(PICK-UP <APPLE.49> SEAMUS)            **
(prepared <apple.49>)                  **
wash-fruit                             <==
(<apple.49> seamus <sink.33>)
ASH-FRUIT <APPLE.49> SEAMUS <SINK.33
(EAT-GOOD-SNACK <APPLE.49> SEAMUS)
```

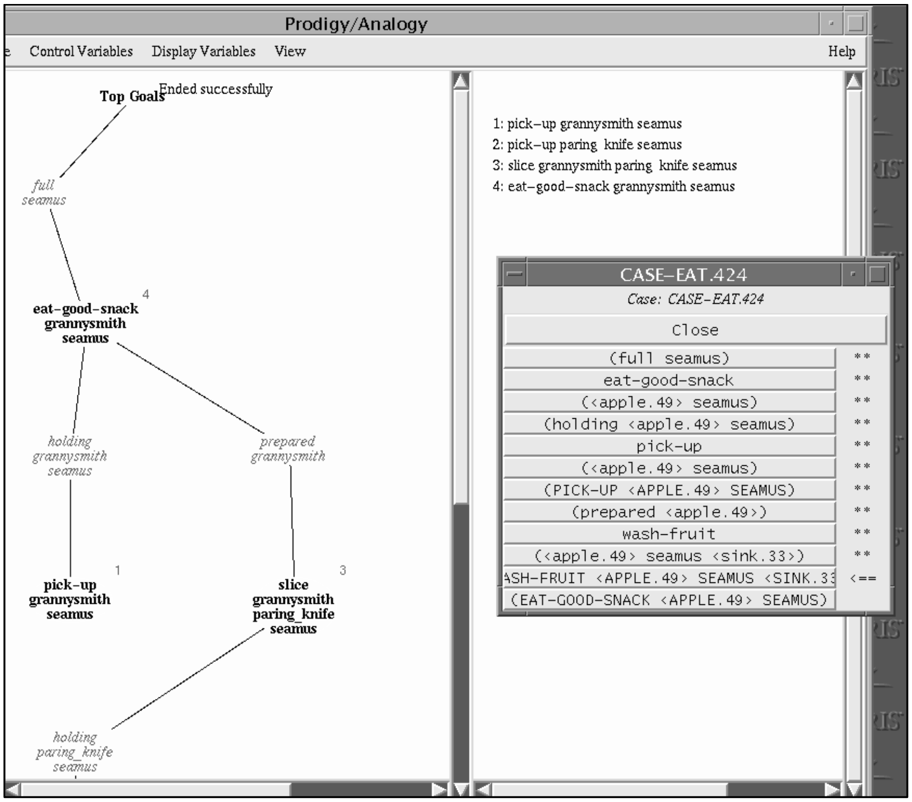
Figure 2. Mixed-initiative case replay: The user can choose to reuse case guidance or make another selection.

prompted at points within the old case where user interaction previously occurred and at new points where user interaction is currently possible that was not possible in the past.

In the old case, the user selected EAT-GOOD-SNACK and WASH as operators, but under mixed-initiative replay, the user is prompted at these points to allow reconsideration. During replay the user is prompted to select an operator to achieve the goal full(Seamus). The user decides to follow the case guidance and selects the operator EAT-GOOD-SNACK, so the case replay continues. Had the user selected another operator (e.g., EAT-BAD-SNACK), the case would no longer be valid because Prodigy/AnalogyMI is now using a different branch of the search tree than the one recorded in the case. Prodigy/AnalogyMI would then switch to generative mode (with user-interaction) to continue with plan generation.

But having chosen to replay the EAT-GOOD-SNACK action, the user now must choose an action to prepare the apple (see Figure 2). The user could choose to follow the guidance and wash the apple, or the user could deviate from the plan by slicing or peeling the fruit. Perhaps the user selects SLICE as the operator to fulfill the goal of preparing the apple. This deviation causes Prodigy/AnalogyMI to then enter generative mode, however, so no more choices are necessary to meet the goals generated by selecting the slice operator. The resulting plan can be seen in Figure 3 on the next page.

Humans can be very good at pruning search trees. This may not seem intuitive at first, but consider a simple problem from the logistics domain (Veloso 1994b). This domain involves moving packages from one post office in a city to another post office in a different city. When Prodigy 4.0 is presented with a problem involving one package, it examines 68 nodes. Increasing the number of packages to two, it now examines 152 nodes. A human solving the same task in generative mode examines 48 nodes, and then 72 nodes. This reduction in the size of the search tree occurs, because a human can spot obvious good selections for goal ordering (loading a truck with both packages, and then transporting them, for instance) and binding choices. These choices are easy for a human to make, but are often difficult to codify into control or inference rules in the domain and become more difficult the more detailed the domain becomes. Obviously the human will only do well in domains with which they are relatively familiar. They need to have an understanding of how to solve the problems before they can become better (faster, more flexible) at generating plans. Prodigy/AnalogyMI with mixed-initiative case replay solves the one package problem by examining only 48 nodes (the same as the human), and 97 nodes when solving the two package problem.


Prodigy/Analogy
Control Variables
Display Variables
View
Help
Top Goals
Ended successfully
full seamus
4
eat–good–snack grannysmith seamus
holding grannysmith seamus
prepared grannysmith
1
pick–up grannysmith seamus
3
slice grannysmith paring_knife seamus
holding paring_knife seamus
CASE–EAT.424
Close

```
1: pick–up grannysmith seamus
2: pick–up paring knife seamus
3: slice grannysmith paring knife seamus
4: eat–good–snack grannysmith seamus
```

```
Case: CASE–EAT.424
(full seamus)                              **
eat-good-snack                             **
(<apple.49> seamus)                        **
(holding <apple.49> seamus)                **
pick-up                                    **
(<apple.49> seamus)                        **
(PICK-UP <APPLE.49> SEAMUS)                **
(prepared <apple.49>)                      **
wash-fruit                                 **
(<apple.49> seamus <sink.33>)              **
ASH-FRUIT <APPLE.49> SEAMUS <SINK.33      <==
(EAT-GOOD-SNACK <APPLE.49> SEAMUS)
```

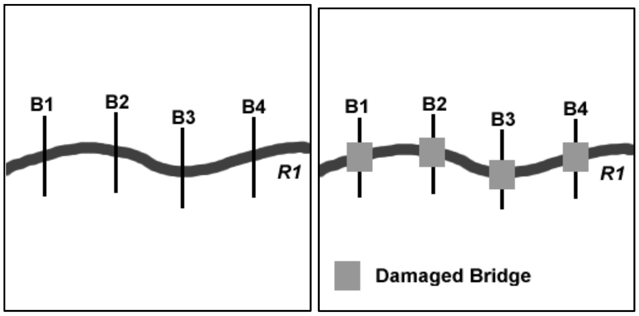
Figure 3. The final plan using mixed-initiative replay and generative planning

The Bridges Problem

More complex domains present different problems to the user and to Prodigy/Analogy. The Bridges Problem (Cox and Veloso 1998; Cox, Kerkez, Srinivas, Edwin, and Archer 2000) represents universally quantified planning tasks in a military domain. The goal of this class of problems is to maximize the "barriability" of rivers by destroying or damaging all bridges across them. Air units can destroy a single bridge and damage an arbitrary number of others. This domain includes many more operators, variables, and choices than does the simple kitchen domain.

Prodigy/Analogy's case retrieval process uses an interactively-footprint similarity matching algorithm (Veloso 1994a) to generate variable substitutions that map old cases to new problems. This allows both versions of Prodigy/Analogy to reason during case replay with the old plan using the resources available in the new problem. However, substitutions are not guaranteed to be optimal and can actually result in solutions that are impossible to replay automatically. For example an old case may include a solution that achieves an impassable river by destroying four bridges across it. A new problem may require that two rivers, one with three bridges and the other with two, be made impassable. The match algorithm might mistakenly map the extraneous fourth bridge in the old case to a bridge that crosses the second river in the new problem (i.e., the one that has two bridges). This results in a faulty case replay, because the scope of the universally quantified precondition for destroying a bridge restricts the operator variables to bridges across a single river. In our example the mismapped bridge crosses a second river instead. Thus Prodigy/AnalogyMI searches for an operator to move the bridge to the river with three bridges, but no such operator exists in the domain. Humans easily understand that suspension bridges cannot be moved between rivers and can reject the past case altogether. The user can then create a new solution from scratch.

Humans can improve upon case replay behavior in other ways. The system prompts the user for a decision based on old cases. When it uses cases that were generated by a novice user, the machine cannot suggest optimal solutions from naive experience. While these suggestions will hopefully lead to a solution, they will not necessarily result in the best solution. Novice users often choose to do the bare minimum to solve a task in the bridges problems. For example an initial state may consist of a river, four bridges that cross the river, and three aircraft, and the goal may be to make the river impassable (see Figure 4a). Because not enough resources exist to destroy all four bridges (each aircraft can only destroy one bridge), the user must change the goal. The user correctly performs this by selecting an operator that allows damage as well as destruction of bridges. However, because the user is a novice, she may damage all of the bridges instead of destroying three and damaging one (see figure 4b).



A. Problem state B. Novice solution

Figure 4. The bridges problem

While damaging all of the bridges does meet the requirements of restricting movement across a river, it is not an optimal solution. When this solution is played back in the form of a case, the user is prompted with the actions of the novice user. In this case, Prodigy/AnalogyMI suggests that the current user damage all of the bridges crossing the river. A more experienced user over-rides the suggestion and makes full use of the resources available in the problem.

If we instead examine a novice user replaying the cases of a more experienced user, we find that Prodigy/AnalogyMI provides better advice. Assume that a domain expert has solved the same goal transformation problem (three aircraft, four bridges) discussed previously. When the novice user replays the case, Prodigy/Analogy suggests destroying three of the bridges and damaging the fourth (see Figure 5). This results in a much higher overall goal satisfaction (see Edwin and Cox 2002 for the formal goal satisfaction criterion).

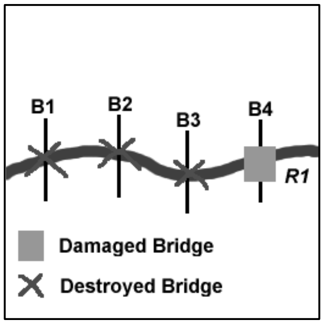


Figure 5. Using an expert's solution

Future Work

This paper has argued that derivational analogy in general and case-replay in particular can be improved by adding an active human user to automated case-based planning. Implemented examples demonstrate some of the strengths of this argument. However, empirical results would better support the efficacy of this approach.

Earlier research on mixed-initiative planning claims that presenting planning to humans as a search process is less effective than presenting it as a goal manipulation problem (Cox 2000). Experimental results from human studies lend support to this claim (Cox 2003; Zhang 2002). The studies compare users' performance solving bridges problems under a search metaphor and the Prodigy 4.0 User Interface 2.0 with the performance subjects using a goal manipulation metaphor and a PRODIGY-based mixed-initiative planner called GTrans (Cox 2000; Cox, Kerkez, Srinivas, Edwin, and Archer 2000; Zhang 2002; Zhang, Cox, and Immaneni 2002).

We have begun to examine the results from this study for the users under the search condition. We have their performance figures for the problems that they solved with PRODIGY under the user-control mode. That is, users themselves made the planning decisions at the control points enumerated in section two. But consider the potential performance difference if mixed-initiative case replay had augmented user choices. If the system had saved each solution as a case during the experiment, Prodigy/AnalogyMI could have retrieved problems they has already solved to provide guidance in subsequent test problems.

Manual examination of a small sample of the data show mixed results. In some instances mixed-initiative case replay would have produced better performance and in others it would have resulted in worse performance. The problem with this preliminary analysis is that we only estimate the experimental results by assuming that all test subjects would follow the case guidance without question. Yet this paper has shown that many examples exist where humans can improve over the suggested choices. A future experimental study with new human subjects can reveal a more complete and accurate understanding of these phenomena.

Conclusions

Both Prodigy 4.0 and Prodigy/Analogy generate plans that are correct, but they may not order operations in an efficient manner. Furthermore plans may be longer than necessary (e.g., moving a plane back and forth to move several packages instead of loading all packages at once). Humans can provide information that is outside the transitive closure of the domain representations in PRODIGY and thus reduce the amount of domain engineering. Moreover a smaller and more abstract domain theory leads to more efficient planning. This is also useful, because it is generally very difficult to describe operators in such a way that they mimic the subtle variations and requirements found in the real world. Domains can also contain situations that are interpreted differently by different users. An example we have used here is the individual preference between eating nutritious or tasty snacks.

The scientific goal to design fully autonomous, general planning systems has all but disappeared from the artificial intelligence community. Instead the field has continued in two broad research directions. The first is to constrain the computational definition of a planning problem and its representation so that new search methods can efficiently produce solutions. Many approaches using boolean satisfiability representations of plans follow this direction. One large problem, however, is that users of such systems are even more divided from the underlying technology and therefore less likely to be involved in its decision making.

The second research direction is toward more understandable and human centered systems. The concept of mixed-initiative computation emphasizes the synergy possible between humans and intelligent machines. This paper has introduced a new technique for case-based planning that examines active user participation as a potential method for escaping some computational limitations and for producing quality results using human experience with both the planning domain and with interaction with the system itself.

Acknowledgments

This research is funded in part through a grant from Science Applications International Corporation (SAIC) and a research challenge grant from Wright State University. The author thanks Seamus Sullivan for his hard work on implementing the changes to Prodigy/Analogy. I also thank Kevin Cleereman and the four anonymous reviewers for their helpful suggestions.

References

Aha, D. ed. 2002. *Proceedings of the Workshop on Mixed-initiative Case-Based Reasoning at the Sixth European Conference on Case-Based Reasoning*.

Burstein, M. ed. 1994. *Proceedings of the ARPA / Rome Laboratory Knowledge-Based Planning and Scheduling Initiative*. San Francisco: Morgan Kaufmann.

Carbonell, J. G. 1986. Derivational Analogy: A theory of reconstructive problem solving and expertise acquisition. In R. Michalski, J. Carbonell and T. Mitchell eds., *Machine learning: An artificial intelligence approach,* Vol. 2, 371-392. San Mateo, CA: Morgan Kaufmann Publishers.

Carbonell, J. G.; Blythe, J.; Etzioni, O.; Gil, Y.; Joseph, R.; Kahn, D.; Knoblock, C.; Minton, S.; Perez, A.; Reilly, S.; Veloso, M. M.; and Wang, X. 1992. *PRODIGY4.0: The Manual and Tutorial*, Technical Report, Computer Science Department, Carnegie Mellon University.

Cox, M. T. 2003. Planning as Mixed-initiative Goal manipulation. In G. Tecuci, D. Aha, M. Boicu, M. T. Cox, G. Ferguson, and A. Tate eds., *Proceedings of the Workshop on Mixed-Initiative Intelligent Systems at the 18th International Joint Conference on Artificial Intelligence*, 36-41. Menlo Park, CA: International Joint Conference on Artificial Intelligence, Inc.

Cox, M. T. 2000. A Conflict of Metaphors: Modeling the planning process. In *Proceedings of the 2000 Summer Computer Simulation Conference*, 666-671. San Diego: The Society for Computer Simulation International.

Cox, M. T.; Kerkez, B.; Srinivas, C.; Edwin, G.; and Archer, W. 2000. Toward agent-based mixed-initiative interfaces. In H. R. Arabnia ed., *Proceedings of the 2000 International Conference on Artificial Intelligence, Vol. 1,* 309-315. CSREA Press.

Cox, M. T.; and Veloso, M. M. 1998. Goal Transformations in Continuous Planning. In M. desJardins ed., *Proceedings of the 1998 AAAI Fall Symposium on Distributed Continual Planning*, 23-30. Menlo Park, CA: AAAI Press / The MIT Press.

Cox, M. T.; and Veloso, M. M. 1997. Supporting combined human and machine planning: An interface for planning by analogical reasoning. In D. B. Leake and E. Plaza eds., *Case-Based Reasoning Research and Development: Second International Conference on Case-Based Reasoning*, 531-540. Berlin: Springer-Verlag.

Edwin, G., and Cox, M. T. 2001. Resource Coordination in Single Agent and Multiagent Systems. In *Proceedings of the 13th IEEE International Conference on Tools with Artificial Intelligence*, 18-24. Los Alamitos, CA: IEEE Computer Society.

Haller, S.; McRoy, S.; and Kobsa, A. eds. 1999. *Computational models of mixed-initiative interaction*. Boston: Kluwer.

Veloso, M. M., 1994a. *Planning and Learning by Analogical Reasoning.* Berlin: Spring-Verlag.

Veloso, M. M. 1994b. Prodigy/Analogy: Analogical Reasoning in General Problem Solving. In *Topics in Case-Based Reasoning*, S. Wess, K.-D. Althoff, and M. Richter eds., Berlin: Springer Verlag, 33-50.

Veloso, M. M.; Carbonell, J.; Perez, A.; Borrajo, D.; Fink, E.; and Blythe, J. 1995. Integrating planning and learning: The PRODIGY Architecture. *Journal of Theoretical and Experimental Artificial Intelligence 7.*

Zhang, C. 2002. *Cognitive models for mixed-initiative planning*. Masters dissertation, Wright State University, Computer Science and Engineering Department, Dayton, OH. (Available at URL `www.cs.wright.edu/~mcox/GTrans/zhang2002.pdf`)

Zhang, C.; Cox, M. T.; and Immaneni, T. 2002. *GTrans version 2.1 User manual and reference,* Technical Report, WSU-CS-02-02. Dayton, OH: Wright State University, Department of Computer Science and Engineering.