Proceedings of the Eleventh International Conference on Machine Learning
July 10-13, 1994, Rutgers, NJ.

Learning by Experimentation: Incremental Refinement of Incomplete Planning Domains

Yolanda Gil
Information Sciences Institute, USC
4676 Admiralty Way
Marina del Rey, CA 90292
gil@isi.edu

Abstract

Building a knowledge base requires iterative refinement to correct imperfections that keep lurking after each new version of the system. This paper concentrates on the automatic refinement of incomplete domain models for planning systems, presenting both a methodology for addressing the problem and empirical results. Planning knowledge may be refined *automatically* through direct interaction with the environment. Missing conditions cause unreliable predictions of action outcomes. Missing effects cause unreliable predictions of facts about the state. We present a practical approach based on continuous and selective interaction with the environment that pinpoints the type of fault in the domain knowledge that causes any unexpected behavior of the environment, and resorts to experimentation when additional information is needed to correct the fault. Our approach has been implemented in EXPO, a system that uses PRODIGY as a baseline planner and improves its domain knowledge in several domains when initial domain knowledge is up to 50% incomplete. The empirical results presented show that EXPO dramatically improves its prediction accuracy and reduces the amount of unreliable action outcomes.

1 Introduction

Building a knowledge base is a process that requires iteration to correct errors that keep lurking after each new version of the system. Several types of imperfections can appear simultaneously in any type of domain theory, including incompleteness, incorrectness, and intractability [Mitchell *et al.*, 1986, Rajamoney and DeJong, 1987, Huffman *et al.*, 1992]. In an EBL system, for example, the rules of the theory are used to compose explanations and an imperfect theory may greatly impair the system's ability to build those explanations. In fact, EBL systems are very brittle with respect to errors in the domain theory, and a lot of the research in EBL concentrates on either correcting them or making the system more robust [Danyluk, 1991, Hall, 1988, Rajamoney, 1993, Rajamoney, 1988]. There is a well developed framework to classify these errors and understand how they affect the explanation process [Mitchell *et al.*, 1986, Rajamoney and DeJong, 1987].

In a planning system, the inaccuracies of the knowledge base may rend problems unsolvable or produce plans that yield unsuccessful executions. The different types of faults in a domain theory affect the planner's performance in different ways. Exploring this issue should provide a good framework for understanding and evaluating systems that learn planning domain knowledge. In this paper, we concentrate on the problematic of missing domain knowledge, which is technically known as incompleteness. Known operators may be missing preconditions and/or effects, or entire operators may be absent from the domain model. We describe the limitations of the capabilities of a planner in terms of the types of incompleteness of its domain knowledge. The imperfections of the domain knowledge have been closely related to planning and/or execution failures [Hammond, 1986, Huffman *et al.*, 1992], but we show that this is not necessarily the case: what they cause is prediction (or expectation) failures, including unexpected successful executions.

The second part of the paper presents a global perspective and empirical results of our work on autonomous refinement of incomplete planning domains [Carbonell and Gil, 1990, Gil, 1993, Gil, 1992]. Learning is selective and task-directed: it is triggered only when the missing knowledge is needed to achieve the task at hand. Our approach is based on continuous and selective interaction with the environment that leads to identifying the type of fault in the domain knowledge that causes any unexpected behavior of the environment, and resorts to experimentation when additional information is needed to correct the fault. The new knowledge learned by experimentation is incorporated into the domain and is immediately available to the planner. The planner in turn provides a performance element to measure any improvements in the knowledge base. This is a closed-loop integration of planning and learning by experimentation. Research in the area of acquiring action models is mostly subsymbolic [Mahadevan and Connell, 1992, Maes, 1991]. An important component of our approach is the ability to design experiments to gather additional information that is not available to the learner and yet is needed to acquire the missing knowledge. Experimentation is vital for effective learning and is a very powerful tool to refine scientific theories [Cheng, 1990, Rajamoney, 1993, Rajamoney, 1988], but other research on learning planning knowledge from the environment does not address this issue directly [Shen, 1993, Shen, 1989, Kedar *et al.*, 1991].

The approach has been implemented in a system called EXPO. EXPO's underlying planning architecture is the PRODIGY system [Minton *et al.*, 1989, Carbonell *et al.*, 1991] which provides a robust, expressive, and efficient planner. The examples included in this paper are based on a robot planning domain [Gil, 1992], but results are also shown for a complex process planning domain.

The paper is organized as follows. Section 2 presents a taxonomy of how incomplete domain knowledge can affect the performance of a planning system. Section 3 describes our approach to the automatic refinement of incomplete planning domains and its implementation in EXPO. Finally, the empirical results presented in Section 4 show that EXPO dramatically improves its prediction accuracy and reduces the amount of unreliable action outcomes.

2 Planning with Incomplete Models

This section groups the effects of incompleteness in planning domains in three categories: unreliable action outcomes, unreliable predicate beliefs, and unreliable coverage of the search space. The examples used are based on PRODIGY's robot planning domain [Gil, 1992].

2.1 Unreliable Action Outcomes

Suppose that a planner is given the following incomplete operator:

```
(OPEN'
 ;the condition (unlocked <door>) is missing
 (params (<door>))
 (preconds
   (and
     (is-door <door>)
     (next-to robot <door>)
     (dr-closed <door>)))
 (effects (
   (del (dr-closed <door>))
   (add (dr-open <door>)))))
```

OPEN' is incomplete: it is missing the condition `(unlocked <door>)`. If the planner uses OPEN' to open an unlocked door, the execution will be successful. If the planner uses OPEN' to open a door that happens to be locked, the action will have no effect. In this case, the planner made the wrong prediction of the outcome of the action execution: that the door would be open. So if the preconditions of an operator are incomplete, the planner's predictions of the operator's outcome are *unreliable*, because the desired effects of the operator may or may not be obtained. Notice that an execution failure is not necessarily obtained, since the missing conditions may happen to be satisfied. The success or failure of the action's execution is thus beyond the planner's control, and it depends solely on the chances that the unknown conditions happen to be true.

Missing conditions of context-dependent effects also cause unreliable action outcomes, since the planner cannot predict when the effect will take place.

2.2 Unreliable Predicate Beliefs

Consider the following incomplete operator:

```
(PUTDOWN-NEXT-TO'
 ;the effect (del (holding <ob>)) is missing
 (params (<ob>))
 (preconds
   (and (holding <ob>)
        (next-to robot <other-ob>)))
 (effects
   ((add (arm-empty))
    (add (next-to <ob> <another-ob>)))))
```

If the planner uses this action to put down an object, the action's execution will be reliable: the desired effects of the operator will be obtained. The planner will only notice the change in the status of holding at this point if it is monitoring the environment beyond the known effects. Although it may be possible in some applications [Kedar *et al.*, 1991, Shen, 1989], continuously monitoring the status of all the known facts is highly impractical in real domains, and furthermore it is not very cost-effective.

However, the planner may notice this change in the future. Suppose that it continues executing actions successfully. Now it wants to put the same object down again. Since it believes to be still holding the object it considers this operator to put the object down. It is when the system is checking the preconditions of the `PUT-DOWN` operator that now that it notices that the truth value of the predicate `(holding obj)` changed inadvertently. The incompleteness in the effects of the operator `PUT-DOWN-NEXT-TO` does not cause an unreliable action outcome. The action of putting down is reliable for the planner since it can predict the outcome of the action: the arm will be empty, and the object will be next to another one.

Notice that although the planner's prediction of the truth value of the predicate failed, in this case the planner does not obtain an execution failure. A missing effect is often mistakenly associated with an execution failure [Hammond, 1986, Huffman *et al.*, 1992], probably because of its negative implication: the planner needs to patch the plan and achieve the desired value of the predicate. In our example, holding needs to be reachieved. However, this is not necessarily the case. Incomplete effects do not always require plan repairs, as we illustrate in the following example.

Incomplete effects may cause the elimination of unnecessary subplans that achieve a goal that is already satisfied in the world. Consider the following alternative description of the operator `PUTDOWN-NEXT-TO`:

```
(PUTDOWN-NEXT-TO''
 ;the effect (add (next-to <ob> <o>)) is missing
 (params (<ob>))
 (preconds
   (and (holding <ob>)
        (next-to robot <o>)))
 (effects
   ((add (arm-empty))
    (del (holding <ob>)))))
```

Now suppose that the goal is not to hold a key and to have it next to a certain box. The planner uses PUTDOWN-NEXT-TO" to achieve not holding the key, and then PUSH-OBJ to put the key next to the box. The planner is unaware that PUTDOWN-NEXT-TO" actually achieves both subgoals, and that PUSH-OBJ is thus an unnecessary subplan (provided that the subplan is not needed to achieve other goals). When the planner notices that the truth value of `next-to` was changed inadvertently, it can eliminate the unnecessary subplan. In this case, *the unreliable prediction did not have any negative implication* for the planner: it even saved some extra work in achieving the goals.

2.3 Unreliable Coverage of Search Space

The two previous sections describe how missing conditions and effects case undesirable behavior during plan execution. Incomplete domains may also cause unreliable coverage of the search space. Notice that affecting the search space can cause complications at problem solving time, not at execution time.

Consider the case of a missing operator. If there are no alternative operators to use during the search, then problems may have no solution (even though they would be solvable if the complete domain were available to the planner). For example, if OPEN is missing from the domain then no other operator would achieve the goal of opening a door, which would cause all the problems that include this subgoal to have no solution. The same type of behavior occurs if the missing effects of an operator were to be used for subgoaling. Consider for example that the domain included an operator OPEN that is missing the effect `(add (dr-open <door>))`. Any problem that needs to achieve the subgoal of opening a door would have no solution.

Notice that in the previous section the missing effects caused different complications. They did not preclude the operator from being part of a plan, since some other known effect of the operator allowed its use for subgoaling. So as long as some primary effect of each operator is known to the planner, the missing effects could be detected eventually as described in the previous section.

Another case of incompleteness occurs when a state is missing facts about the world. For example, consider a state containing a description of a door `Door45` that connects `Room4` and `Room5`. The state does not contain information about the door being either locked or unlocked. In this case, some operator's preconditions cannot be matched in the state. For example, OPEN has a precondition that the door must be unlocked, and the planner cannot consider using it for opening `Door45`. So when facts are missing from the state, the applicability of operators is restricted to the known facts and thus it may not be possible to explore parts of the search space until more information becomes available. [Gil, 1992] describes how to take advantage of this fact to learn whether the door is open by experimenting with `OPEN`.

2.4 Summary

Table 1 summarizes the taxonomy of limitations of a planner caused by incomplete domain knowledge. Missing conditions cause action execution failures. If the missing condition is identified, a plan is needed to achieve it before the action can be executed successfully. Missing side effects may cause either unnecessary subplans or additional planning, but they do not cause execution failures. Missing primary effects, operators, or data about the state may cause that some problems have no solution (even though they would be solvable if the complete domain were available to the planner).

3 Incremental Refinement of Planning Domains through Experimentation

When users define operators for a planning system, the resulting operators turn out to be operational for planning (i.e., the planner has some model of the actions that it can use to build plans) but are incomplete in that users often forget to include unusual preconditions or side effects. This section

what is missing	*what it may cause*	*when noticed*	*how noticed*
preconditions	action execution failure followed by plan repair	plan execution	unreliable action outcomes
conditions of context-dependent effects			
effects not needed for subgoaling	unnecessary subplans or plan repair	plan execution	unreliable predicate beliefs
effects needed for subgoaling	unreliable coverage of search space	problem solving	problems without solution
operators			
predicate beliefs			

Table 1: Limitations Caused by Incomplete Domain Knowledge in a Planner.

presents our approach to the problem of refining incomplete operators that are missing preconditions and effects. More details of our method for operator refinement can be found in [Gil, 1992, Gil, 1993, Carbonell and Gil, 1990]. We describe elsewhere [Gil, 1992] how our system addresses other limitations caused by incomplete domain knowledge in a planner, including how to acquire completely new operators that are missing from the domain and how to acquire new data about the state. [Gil, 1993, Gil, 1992] describe the experiment design process in more detail.

3.1 Detection of an Imperfection

A planner's ability to interact with its environment allows the detection of knowledge faults. EXPO monitors the external world *selectively* and *continuously*. Before the execution of an operator, EXPO expects the operator's known preconditions to be satisfied, so it checks them in the external world. If they are indeed satisfied, then EXPO executes the corresponding action. The operator's known effects are now expected to have occurred in the world, so EXPO checks them in the internal world. Any time that the observations disagree with the expectations, EXPO signals an imperfection and learning triggered.

3.2 Operator Refinement

EXPO uses the Operator Refinement Method [Carbonell and Gil, 1990] to learn new preconditions and effects of operators. We briefly describe now the implementation of this method in EXPO.

Acquiring New Preconditions

When an operator O executed in state S has an unpredicted outcome, EXPO considers the working hypothesis that the preconditions of O are incomplete and triggers learning to find out the missing condition C. C must have been true (by coincidence) every time that O was executed before. EXPO keeps track of every state in which each operator is executed. It looks up S_O, a generalization of all the states in which O was successfully executed in the past.[1] All the predicates in S_O are considered potential preconditions of O. (Notice that the currently known preconditions of O must be in S_O). EXPO then engages an experimentation process to discern which of those predicates is the missing condition.

Because of the bias needed in the generalization of S_O, the missing condition may not appear in S_O. If this is the case, none of the experiments would be successful. EXPO then would retrieve any successful past application of O, S_{suc}, and builds a new set of candidate preconditions with the differences between S and S_{suc}. If experimentation is not successful in this stage, the current implementation of EXPO prompts the user for help. Ideally, it would look for additional candidates (for example, predicates that are not included in the state S because they were never observed), and even consider the alternative working hypothesis that O has conditional effects (instead of missing a precondition).

Previous work on refinement of left-hand sides (LHS) of rules has used the concept learning paradigm in considering each LHS as a generalization of states where the rule is applicable [Mitchell, 1978, Mitchell *et al.*, 1983, Langley, 1987]. However, EXPO uses this paradigm as a heuristic that guides the search for a new condition, and not as a basis for finding it. EXPO uses other heuristics to make the experimentation process more efficient. This is described in detail in [Gil, 1993, Gil, 1992].

Acquiring New Effects

When a predicate P is found to have an unpredicted value, EXPO considers the working hypothesis that some operator that was applied since the last time P was observed had the unknown effect of changing P. EXPO retrieves all operators executed since then, and considers them candidates for having incomplete effects. Experiments with each operator monitoring P closely yield the incomplete operator.

[1]The generalization of states is done through the operator's bindings and uses a version space framework.

what is noticed	*working hypothesis*	*candidates*	*state before experiment*	*operator in experiment*	*observations in experiment*	
					before	*after*
unreliable outcome of O	O is missing some condition	Predicates $\{P_i\}$ that were true in previous executions of O	Preconditions of O and some P_i are satisfied	O	—	effects of O
unreliable belief of P	P is an unknown effect of some previously executed operator	Operators $\{O_i\}$ executed since last time P was observed	Preconditions of some O_i are satisfied	O_i	P	P

Table 2: Operator Refinement in EXPO.

	robot planning	*process planning*
number of rules	14	120
average number of preconditions	4	8
average number of effects	4	6
number of predicates	11	93
number of object types	4	33

Table 3: The robot planning and the process planning domains.

3.3 Summary

Table 2 summarizes operator refinement in EXPO. EXPO triggers learning when something unpredicted happens, and focuses on experiments that find the missing information that yields the correct prediction. Experimentation is *task-directed*: always engaged within a particular context that sets specific aims and purpose for what is to be learned.

4 Empirical Results

This section contains results that show the effectiveness of EXPO, i.e., that it can indeed be used to acquire new knowledge that is useful to the problem solver.

The results presented in this section show EXPO learning in two different domains: a robot planning domain and a complex process planning domain. The robot planning domain is an extension of the one used by STRIPS that has been used in other learning research in PRODIGY (see [Carbonell *et al.*, 1991] for references). The process planning domain contains a large body of knowledge about the operations necessary to machine and finish metal parts, and was chosen because of its large size. The domain is described in detail in [Gil, 1991] The domains are compared along some dimensions in Table 3. [Gil, 1992] describes them in detail.

We want to control the degree of incompleteness of a domain in the tests. We have available a complete domain D which has all the operators with all their corresponding conditions and effects. With this complete domain, we can artificially produce domains D' that have certain percentage of incompleteness (i.e., 20% of the preconditions are missing) by randomly removing preconditions or effects from D. We will use D'_{prec20} to denote a domain that is incomplete and is missing 20% of the conditions. D'_{post20} is a domain missing 20% of the postconditions. Notice that EXPO never has access to D, only to some incomplete domain D'.

EXPO learns new conditions and effects of incomplete operators. What is a good measure of the amount of new knowledge acquired by EXPO in each case? Missing preconditions may cause action execution failures. To show that EXPO is effectively learning new preconditions, we run the test set several times during training. We compared the cumulative number of wrong predictions during training with the number of problems in the test set that could be executed successfully to completion. Missing effects may cause wrong predictions of literals. We compared the cumulative number of incorrect literals found during training with the number of incorrect literals in the final state of the problems in the test set. Each wrong prediction encountered during training, is an opportunity for learning. At certain points during learning, we run the test set. Learning is turned off at test time, so when a wrong prediction is found the internal state is corrected to reflect the observations but no learning occurs.

Training set and test set were generated randomly, and they were independent in all cases.

4.1 Results

Figure 1 shows the number of action execution failures that EXPO detects during training with D'_{prec20} and D'_{prec50} in the robot planning domain. The figure also shows how many solutions for problems in the test set were successfully executed with D'_{prec20} and D'_{prec50}. The number of plans that PRODIGY is able to execute correctly increases with learning.

The maximum number of unexpected action outcomes, indicated by the upper limit of the y-axis, corresponds to learning all the missing preconditions. For D'_{prec20}, notice that although EXPO does not acquire all the

(a) Cumulative number of unexpected action outcomes during training

(b) Number of plans successfully executed in the test set

Figure 1: Effectiveness of EXPO in the robot planning domain with 20% and 50% of the preconditions missing (D'_{prec20} and D'_{prec50}). (a) Cumulative number of unexpected action outcomes as the size of the training set increases. (b) The number of plans successfully executed in the test set increases as EXPO learns.

missing domain knowledge, it has learned the knowledge necessary to execute successfully the solutions to all the problems in the test set. In fact, after training with 40 problems EXPO can solve all the problems in the test set. Even though EXPO learns new conditions with further training they do not cause any improvement in the performance. For D'_{prec50}, very few solutions to the test problems are executed successfully in one case. This is because the situations encountered during training do not cover the situations encountered in the test problems in that the knowledge needed to solve the test problems is not needed to solve the training problems. (In fact, after training with the test set one more new condition is learned which turns out to be common in the test set and thus the solutions to all the test problems can be successfully executed).

In the process planning domain, the tests were run in domains with 10% and 30% incompleteness using two training sets and two test sets. Figure 2 presents results for D'_{prec10} and D'_{prec30} when EXPO acquires new preconditions. Even though this is a more complex domain, the curves show results very similar to the results obtained for the robot planning domain.

We also ran tests with domains where postconditions of operators were missing. Figure 3 shows the results for D'_{post20} and D'_{post50} respectively in the robot planning domain. As more incorrect literals are found in the state, EXPO acquires new effects of operators. Thus, the number of incorrectly predicted literals when running the test set is reduced continuously.

4.2 Discussion

The new preconditions and postconditions learned through EXPO improve PRODIGY's performance by reducing the amount of wrong predictions during plan execution. The effectiveness of learning is not solely a characteristic of the learning algorithm: it is heavily dependent on the situations presented to EXPO during training. This is expected of any learning system: if the training problems cover situations that are comparable to the ones in the test problems, then learning is more effective.

Another effect of the nature of the training problems is that EXPO rarely acquires all the knowledge that is missing from the domain. However, PRODIGY's performance is always improved, and in many cases all the test problems can be executed successfully after learning even though the improved domain may not be complete. EXPO's knowledge is becoming increasingly more complete, because learning is directed to find the missing knowledge needed to solve the task at hand.

(a) Cumulative number of unexpected action outcomes during training

(b) Number of plans successfully executed in the test set

Figure 2: Effectiveness of EXPO in the process planning domain with 10% and 30% of the preconditions missing (D'_{prec10} and D'_{prec30}).

Even though an action may have many more conditions and effects than those currently known, only the ones that are relevant to the current situation are acquired. EXPO shows that learning can improve a system's performance and bring it to a point where it can function reasonably well with whatever knowledge is available, be it a perfect model of the world or not.

Finally, EXPO is a *proactive* learning system. When a fault in the current knowledge is detected, the information available to the learner may well be insufficient for overcoming the fault. An important component of EXPO's learning is the ability to design experiments to gather any additional information needed to acquire the missing knowledge. Work on learning theory has shown that the active participation of the learner in selecting the situations that it is exposed to is an important consideration for the design of effective learning systems [Angluin, 1987, Rivest and Sloan, 1988, Ling, 1991, Kulkarni *et al.*, 1993].

5 Conclusion

Learning from the environment is a necessary capability of autonomous intelligent agents that must solve tasks in the real world. Planning systems that model a physical system can be given the ability to interact with it, and thus they can directly examine the actual behavior of the physical system that the domain is supposed to model. This presents an opportunity for autonomous refinement of the imperfections of a domain theory. Our approach combines selective and continuous monitoring of the environment to detect knowledge faults with directed manipulation through experiments that lead to the missing knowledge. The results presented in this paper show the effectiveness of this approach to improve a planner's prediction accuracy and to reduce the amount of unreliable action outcomes in several domains through the acquisition of new preconditions and effects of operators.

This work is applicable to a wide range of planning task, but there are some limitations. The state of the world must be describable with discrete-valued features, and reliable observations must be available on demand. Actions must be axiomatizable as deterministic operators in terms of those features.

Our work assumes an initially incomplete knowledge base. Future work is needed to address other types of imperfections, including incorrectness of planning domain knowledge.

Acknowledgments

This work was done while the author was at Carnegie Mellon University. I would like to thank Jaime Carbonell, Tom Mitchell, Herb Simon, and Nils Nilsson for their suggestions and support throughout my thesis work. Thanks also to all the members of the PRODIGY group for many helpful suggestions.

This research was supported by the Avionics Laboratory, Wright Research and Development Center, Aeronautical Systems Division (AFSC), U.S. Air Force, Wright-Patterson AFB, Ohio 45433-6543 under Contract F33615-90-C-1465, ARPA Order No. 7597. The view and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of DARPA or the U.S. government.

References

[Angluin, 1987] Dana Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1987.

[Carbonell and Gil, 1990] Jaime G. Carbonell and Yolanda Gil. Learning by experimentation: The operator refinement method. In Y. Kodratoff and R. S. Michalski, editors, *Machine Learning, An Artificial Intelligence Approach, Volume III.* Morgan Kaufmann, San Mateo, CA, 1990.

[Carbonell *et al.*, 1991] Jaime G. Carbonell, Craig A. Knoblock, and Steven Minton. PRODIGY: An integrated architecture for planning and learning. In Kurt VanLehn, editor, *Architectures for Intelligence.* Lawrence Erlbaum Associates, Hillsdale, NJ, 1991.

[Cheng, 1990] Peter C-H. Cheng. *Modelling Scientific Discovery.* PhD thesis, The Open University, Milton Keynes, England, 1990.

[Danyluk, 1991] Andrea D. Danyluk. *Extraction and Use of Contextual Attributes of Theory Completion: An Integration of Explanation-Based and Similarity-Based Learning.* PhD thesis, Columbia University, New York, NY, 1991.

[Gil, 1991] Yolanda Gil. A specification of manufacturing processes for planning. Technical Report CMU-CS-91-179, School of Computer Science, Carnegie Mellon University, 1991.

[Gil, 1992] Yolanda Gil. *Acquiring Domain Knowledge for Planning by Experimentation.* PhD thesis, Carnegie Mellon University, School of Computer Science, 1992.

[Gil, 1993] Yolanda Gil. Efficient domain-independent experimentation. In *Proceedings of the Tenth International Conference on Machine Leaning*, Amherst, MA, 1993. Morgan Kaufmann.

[Hall, 1988] Robert J. Hall. Learning by failure to explain: Using partial explanation to learn in incomplete or intractable domains. *Machine Learning*, 3(1):45–78, 1988.

[Hammond, 1986] Chris J. Hammond. *Case-based Planning: An Integrated Theory of Planning, Learning, and Memory.* PhD thesis, Yale University, New Haven, CN, 1986.

[Huffman *et al.*, 1992] Scott B. Huffman, Douglas J. Pearson, and John E. Laird. Correcting imperfect domain theories: A knowledge-level analysis. In *Machine Learning: Induction, Analogy and Discovery.* Kluman Academic Press, Boston, MA, 1992.

[Kedar *et al.*, 1991] Smadar T. Kedar, John L. Bresina, and C. Lisa Dent. The blind leading the blind: Mutual refinement of approximate theories. In *Proceedings of the Eight Machine Learning Workshop*, Evanston, IL, 1991.

[Kulkarni *et al.*, 1993] S. R. Kulkarni, S. K. Mitter, and J. N. Tsitsiklis. Active learning using arbitrary binary valued queries. *Machine Learning*, 11(1), 1993.

[Langley, 1987] Pat Langley. A general theory of discrimination learning. In *Production System Models of Learning and Development.* MIT Press, Cambridge, MA, 1987.

[Ling, 1991] Xiaofeng Ling. Inductive learning from good examples. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, Sydney, Australia, 1991.

[Maes, 1991] Pattie Maes. Adaptive action selection. In *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*, Chicago, IL, 1991.

[Mahadevan and Connell, 1992] Sridhar Mahadevan and Jonathan Connell. Automatic programming of behavior-based robots using reinforcement learning. *Artificial Intelligence*, 55(2-3):311–365, 1992.

[Minton *et al.*, 1989] Steve Minton, Jaime G. Carbonell, Craig A. Knoblock, Dan R. Kuokka, Oren Etzioni, and Yolanda Gil. Explanation-based learning: A problem solving perspective. *Artificial Intelligence*, 40(1-3):63–118, 1989.

[Mitchell *et al.*, 1983] Tom Mitchell, Paul Utgoff, and Ranan Banerji. Learning by experimentation: Acquiring and refining problem-solving heuristics. In *Machine Learning, An Artificial Intelligence Approach, Volume I.* Tioga Press, Palo Alto, CA, 1983.

[Mitchell *et al.*, 1986] Tom M. Mitchell, Richard M. Keller, and Smadar T. Kedar-Cabelli. Explanation-based learning: A unifying view. *Machine Learning*, 1(1):47–80, 1986.

[Mitchell, 1978] Tom M. Mitchell. *Version Spaces: An Approach to Concept Learning.* PhD thesis, Stanford University, Stanford, CA, 1978.

[Rajamoney and DeJong, 1987]
Shankar A. Rajamoney and Gerald F. DeJong. The classification, detection, and handling of imperfect theory problems. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Milano, Italy, 1987.

[Rajamoney, 1988]
Shankar A. Rajamoney. *Explanation-Based Theory Revision: An Approach to the Problems of Incomplete and Incorrect Theories.* PhD thesis, University of Illinois at Urbana-Champaign, Urbana, IL, 1988.

[Rajamoney, 1993] Shankar A. Rajamoney. The design of discrimination experiments. *Machine Learning*, 12(1/2/3), 1993.

[Rivest and Sloan, 1988] Ron L. Rivest and Robert Sloan. Learning complicated concepts reliably and usefully. In *Proceedings of the Workshop on Computational Learning Theory*, Pittsburgh, PA, 1988.

[Shen, 1989] Wei-Min Shen. *Learning from the Environment Based on Percepts and Actions.* PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1989.

[Shen, 1993] Wei-Min Shen. Discovery as autonomous learning from the environment. *Machine Learning*, 12(1/2/3), 1993.

(a) Cumulative number of incorrect literals found during training

(b) Incorrect literals in the final state of test problems

Figure 3: Acquisition of new effects in the robot planning domain with 20% and 50% of the effects missing (D'_{post20} and D'_{post50}). (a) Cumulative number of incorrect literals found in the internal state as the size of the training set increases. (b) The number of incorrect literals of the final state in the test set decreases as EXPO learns.