MLC-93

Efficient Domain-Independent Experimentation

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

Abstract

Planning systems often make the assumption that omniscient world knowledge is available. Our approach makes the more realistic assumption that the initial knowledge about the actions is incomplete, and uses experimentation as a learning mechanism when the missing knowledge causes an execution failure. Previous work on learning by experimentation has not addressed the issue of how to choose good experiments, and much research on learning from failure has relied on background knowledge to build explanations that pinpoint directly the causes of failures. We want to investigate the potential of a system for efficient learning by experimentation without such background knowledge. This paper describes domain-independent heuristics that compare possible hypotheses and choose the ones most likely to cause the failure. These heuristics extract information solely from the domain operators initially available for planning (incapable of producing such explanations) and the planner's experiences in interacting with the environment. Our approach has been implemented in EXPO, a system that uses PRODIGY as a baseline planner and improves its domain knowledge in several domains. The empirical results presented show that EXPO's heuristics dramatically reduce the number of experiments needed to refine incomplete operators.

1 Introduction

Learning from the environment is a vital capability for an autonomous agent. The lack of knowledge affects the planner's capabilities, and learning requires both detecting a failure and determining a correction of the knowledge base. Experimentation is a powerful tool for gathering additional information from the environment that helps determine the appropriate correction. Previous work on learning from the environment has not addressed the issue of how to choose good hypotheses for efficient experimentation [Shen, 1989, Hume and Sammut, 1991]. Others have tried to reduce or eliminate the need for experimentation by relying on causal theories or other types of background knowledge to build explanations for the failures that determine what is to be learned [Rajamoney, 1988, Kedar *et al.*, 1991, Hammond, 1986]. Learning only seems feasible when detailed knowledge of the domain at hand is available. The central issue of the acquisition and refinement of additional and necessarily complex background knowledge is still far from resolved. While explanations are powerful, it is not realistic to assume that they are always available to planners in any task domain and with enough detail to explain any possible failure. Our work was inspired by observation of human behavior upon failed expectations in absence of adequate domain understanding. For example, if we are presented with a new pen and we fail to write with it, we would most probably try to vary the writing action and try to make it work. We might tilt the point a little. We might try writing on a different type of paper. We might try to press harder. These small variations are what we call experiments, perhaps of a more mundane nature than those performed in a laboratory but nevertheless greatly responsible for our autonomy and adaptability. Do we reason about friction between felt tips and types of paper? Do we have knowledge about how ink flows through a cartridge? Humans learn through these experiments in many domains. Do we all have theories about everything in the world that we interact with? More importantly, are these theories necessary for building systems able to learn autonomously from the world? The work in this paper suggests that they are not. We claim that efficient experimentation is possible without any theory that supports our actions and explains our failures.

This paper presents an efficient experimentation strategy that does not have access to a theory that produces explanations for failures. Our approach is to use domain-independent heuristics that extract infor-

```
(GRIND-INCOMPLETE
 (preconditions
   (and
     (is-a <machine> GRINDER)
     (is-a <tool> GRINDING-WHEEL)
     (is-a <part> PART)
     (holding-tool <machine> <tool>)
     (side-up-for-machining <dim> <side>)
     (holding <machine> <device> <part> <side>)))
 (effects (
     (add (surface-finish <part> <side> SMOOTH))
     (add (size-of <part> <dim> <value>)))))
```

Figure 1: An incomplete model of grinding. Notice that, upon an execution failure, a great deal of background information would have been needed to explain that the presence of cutting fluid is important for grinding.

mation solely from the domain operators initially available for planning and the planner's experiences in interacting with the environment. The paper also shows the performance of these heuristics in their implementation in EXPO, a learning by experimentation capability within the PRODIGY system [Carbonell *et al.*, 1990] that improves the planner's knowledge in several domains. We begin with a brief description of our previous work.

2 Learning by Experimentation

Suppose that a planner is given a process planning domain with the incomplete operator shown in Figure 1. This operator models the process of grinding a metallic surface. A grinder holds a part with some holding device, and, using a grinding wheel as a tool, it changes the size of the part along a selected dimension. This operator may seem correct, but it is incomplete. For example, it is missing a precondition that states that the grinder must have cutting fluid. Any plans that use this operator for grinding will not be useful for grinding parts in the real world. These plans will include steps that set up the tool and the part correctly in the machine, but will not provide the grinder with cutting fluid. Grinding is an abrasive operation that generates heat as a result of the friction between the tool and the part. Cutting fluids cool both the cutting edges of the tool and the part, aid in chip clearance, and improve the surface finish. If no cutting fluid if present to absorb the heat, then the grinding process will not produce the desired size (the grinder and the part will overheat instead.) The focus of our research is to design learning systems that would correct the operator's preconditions and effects.

EXPO learns new preconditions through the Operator Refinement Method [Carbonell and Gil, 1990], that we summarize here briefly. Suppose that the system has build a plan to grind a part to make its length smaller. Before grinding the part, the system checks that the preconditions are true in the external world. After grinding it, the postcondition of GRIND is checked in the external state. The size of the part has changed to be of size k, but the surface finish is not as it was expected. This may be because the known effect that specifies the new surface finish is wrong, or because the operator is missing a necessary precondition. This method addresses the failure by considering the latter possibility as the working hypothesis first (see [Gil, 1992] for a discussion on the first possibility): that some unknown precondition is not true in the state and thus the grinding action is not working as the given operator specifies. To find out what the missing precondition is, EXPO considers conditions that were true in an earlier successful application of the operator that are not true now. To do so, EXPO retrieves the description of the past state when the action execution succeeded before, which contains all the facts that the planner believed to be true of the world at that point in time. Among the things that were true in that state, which may be many, is the fact that the grinder had fluid when the operation worked in the past[1]. Experimentation is needed to determine which one of the differences is relevant for this particular failure. The experiments will point out that the presence of cutting fluid is relevant for grinding, and EXPO corrects the operator to reflect this fact.

A typical set of hypotheses obtained by EXPO in its process planning domain has 50 to 100 elements, but for simplicity consider the following subset:

```
(size-of <part> WIDTH 3)
(size-of <part> LENGTH 7)
(size-of <part> HEIGHT 2.5)
(material-of <part> BRASS)
(has-fluid <machine>)
(surface-finish part26 <side> SAWCUT)
(holding drill1 vise2 part26 <side>)
(material-of part26 STEEL)
(is-a drill1 DRILL)
(is-a drill-bit1 DRILL-BIT)
(material-of part37 COPPER)
(has-hole part37 <side>)
```

As described in [Gil, 1992], it is important to minimize the number of experiments and their requirements. For each experiment the planner has to build a plan to set the environment in a state that satisfies that many predicates. Apart from the planning effort involved, the execution of those plans raises non-trivial issues. Plan execution may use up valuable resources (including time), produce non-desirable changes in the environment that are hard to undo, and interfere with the main goals of the system's task. If any information is available to identify a smaller subset of these candidates as more relevant, the experimentation effort may

[1] For a discussion on the case when the missing condition is not present, see [Gil, 1992].

be greatly reduced. In particular, if we can devise a way of ranking the candidates from most relevant to least, then each candidate can be tested individually. EXPO follows this strategy by ranking the candidate hypotheses heuristically.

3 Domain-independent Heuristics for Efficient Experimentation

EXPO's efficient experimentation is based on heuristics that exploit knowledge about the planning task to evaluate which predicates in a set of differences are more likely to have caused the failure. This section briefly describes these heuristics, summarized in Table 1. Their implementation in EXPO follows. For more details see [Gil, 1991, Gil, 1992].

One heuristic is locality of actions. The idea behind it is the following. The fact that there is a steel part lying somewhere else in the machine shop is not likely to affect our grinding operation. Facts about the machine and tool used and the part being ground are more likely to be relevant to the failure. This heuristic prefers predicates which contain some object that appears in the operator's bindings.

Structural similarity takes advantage of the fact a planning domain reflects the regularities of the actions needed to do a task. Consider the set of differences above as possible candidates for a new precondition of grinding. Many other operators change the size of a part. Many of them require the use of cutting fluid, which is in fact the relevant condition for this particular failure. Only some of them have conditions about the material of the part. And none of them has any conditions about the surface finish of a side of the part. This heuristic prefers predicates that are preconditions of operators that are similar to grinding according to some metric.

Generalization of experience takes advantage of the fact that the conditions needed for the action must have been present in all past successful executions of it. In fact, the precondition expression of an operator can be seen as a concept that represents the states in which the operator can be executed successfully, as in [Mitchell, 1978, Mitchell *et al.*, 1983, Langley, 1987, Langley *et al.*, In press]. Unlike these systems, EXPO takes the concept that reflects the LHS of the rule as a *heuristic* for learning, rather than as the sole basis for it. Thus, we can bias the generalization language without worrying about excluding the target concept. The generalization is used as a heuristic to guide the experiments, and it does not represent the precondition expression of the operator (although it is related). A summary of the planner's past experience of the action's behavior is useful to guide our search for the missing condition, because it highlights the conditions that were common to all the states when the action was executed before.

3.1 Implementation

To be able to generalize from experience, EXPO needs to keep track of the execution of actions. Each execution of an operator is either a success or a failure. A state in which a successful execution occurs corresponds to a positive instance of the concept, and a state in which a failure is obtained is a negative instance. EXPO keeps information about action executions in *situations*, which are composed of the operator whose action was executed, the result of the execution (success or failure), the list of bindings for the operator variables, and the list of predicates believed to be true immediately prior to the operator being executed (i.e., the state). The situations are used to maintain the current description of each operator's preconditions as a version space [Mitchell, 1978]. The algorithm is biased to produce conjunctive descriptions of the concept. This bias is appropriate for this application. The large majority of the precondition expressions in operators are conjunctions of predicates (or negations of predicates). This is because actions are easier to express if their effects under different conditions are described in separate operators. In this sense, even if the system aims to learn only conjunctive expressions of predicates it would be a great win. In fact, even though PRODIGY allows for a very expressive language in the preconditions, EXPO's generalization only contains the predicates in the preconditions that are part of the main conjunct. For example, if the precondition expression of an operator is `(and (A B C D (or E F)))`, E and F are never included in the generalization.

Version spaces implement the heuristic for selecting hypotheses based on its generalization of experience as follows. From the set of current candidate hypotheses, only the ones that appear in S (the ones that are common to all successful situations) and do not appear in G (since G contains the preconditions, they appear in the failure state) are selected.

The set of hypotheses selected by the generalization heuristic is then filtered by the locality heuristic. This heuristic selects only the hypotheses that contain constants and variables that appear in the bindings of the failure situation. This new subset of the hypotheses is then ranked by the heuristic of structural similarity as we explain now.

All the domain operators are organized by EXPO in a hierarchy using a simple clustering algorithm [Gil, 1992]. The root node contains all the operators in the hierarchy. For every node, the operators that are not in any of its children yet are examined to build a child node. The expression or expressions that appear in a larger number of operators[2] define the child node,

[2] in the preconditions, postconditions, or both. In our

heuristic	*description*
locality of actions	objects affected by the action are likely to be present in the operator's parameters
structural similarity	similar operators are likely to have similar preconditions
generalization of experience	necessary conditions have been present in all past successful executions of the action

Table 1: Domain-independent heuristics for suggesting better experiments.

and the operators that contain them are transferred to it. The algorithm works its way down in the tree until a node is reached that contains only one operator or all of its operators expressions are included in the node. When a new condition or effect for an operator is learned, the hierarchy is updated by recomputing the children of the node that contains the operator. Since this clustering algorithm is used as a heuristic, the emphasis is not so much in the accuracy of the result as long as it reflects to some extent the structural similarity behind the domain operators.

EXPO considers first the hypotheses that are selected by the three heuristics. Then, it considers the ones that the structural regularity heuristic rejected, then the ones rejected by the locality heuristic. Last, EXPO considers the rest of the hypotheses in the initial set.

Determining the missing precondition is done through iterative experimentation with the ranked list of candidate predicates. In EXPO, this process converges if the missing condition is an observable and non-inferred predicate that is within a conjunctive expression. If this is the case, the missing condition is included in the group of candidate hypotheses, and EXPO eventually encounters it and learns it through experimentation. If this is not the case, then the missing condition may be something else, e.g., a disjunction of some of those conditions, a quantified expression over some predicate, or an unobserved fact (see [Gil, 1992] for more details). EXPO does not learn these types of conditions.

Although the algorithms presented here can be made more sophisticated, we must keep in mind that they are used to implement heuristics and as such they are not required to be close to an optimal implementation of the idea behind them. In their simplicity, the results in the next section show that they are effective for this purpose.

4 Results

This section presents EXPO's performance with a robot planning domain in which 20% and 50% of the operator's preconditions were removed randomly. This means that from all the preconditions of all the operators a percentage was removed, so any operator can be missing any number of preconditions. We generated n problems randomly. All of the n problems were solvable within the time bound that PRODIGY was given. From the set of n solvable problems, we randomly chose m of them to be the training set. The rest constituted the test set. Notice that both sets are independent (they do not have any common instances). Initially, PRODIGY is given the incomplete domain and EXPO starts running the training problems. For each problem, EXPO obtains a plan from PRODIGY and tries to execute it in the external environment[3]. EXPO examines any expectation failures and applies the Operator Refinement Method together with the heuristics described in this paper for designing experiments. After the experiments determine the cause of the failure, EXPO corrects the operator and uses it for future plans.

Figures 2 and 3 present the number of experiments that are required to recover from the failures encountered by EXPO. The horizontal axis represents subsequent failures encountered by EXPO. The vertical axis shows the cumulative number of experiments needed until the missing condition is isolated. We show results with different combinations of the heuristics. We also plot the number of experiments needed when no heuristics are used, in which case EXPO tries in sequence the candidate predicates. The heuristics used are represented by a letter: g for generalization, s for structural similarity, and l for locality. A strategy that does less experiments in the absence of information is represented in the graph as *DC*. *DC* uses a divide-and-conquer strategy: it recursively splits the candidate set, using $log(n)$ experiments to isolate the correct hypothesis (n is the number of hypotheses).

Without any of the hypothesis-selection heuristics many experiments are needed, since the candidate hypotheses are tried one by one. Although DC does a smaller number of experiments than some of the heuristics used in isolation, we show below that there are other reasons why it is inefficient. The other curves show how effective each heuristic is individually and in combination with others. Each heuristic contributes in its own way to reducing the number of experiments.

experience with EXPO's domains, this does not make a difference in the effectiveness of the structural similarity heuristic.

[3]EXPO was not tested interacting with a physical environment, but with a software system that simulates one. The details of this simulation are described in [Gil, 1992].

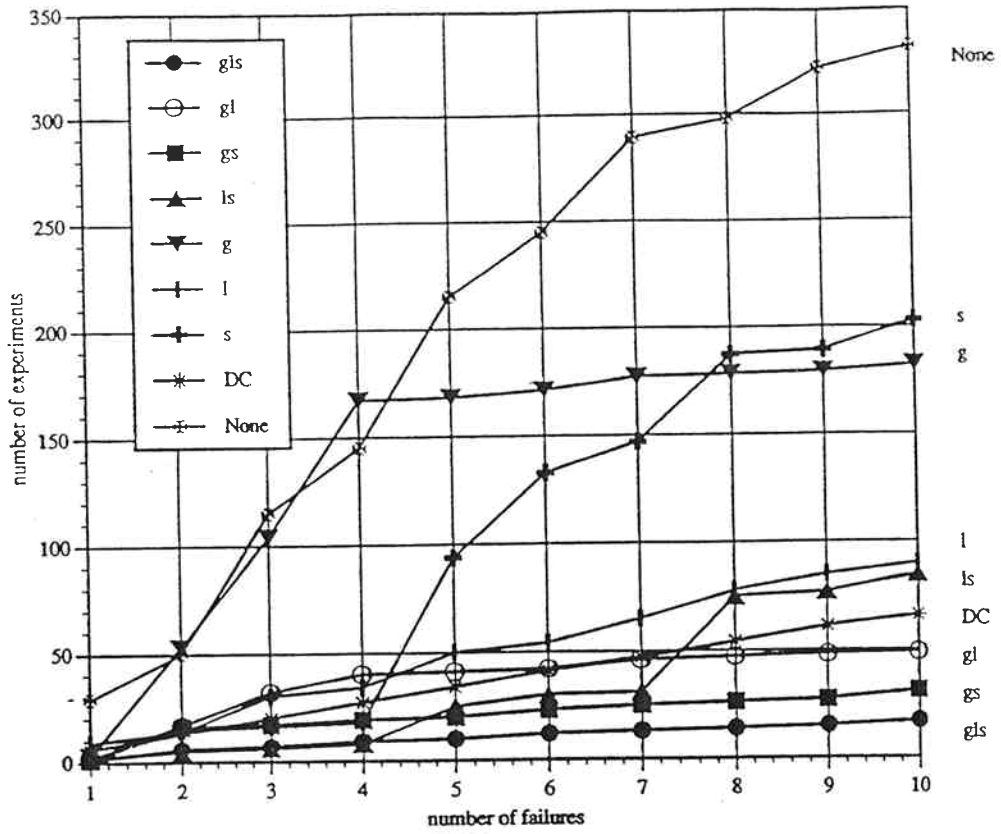

Figure 2: Given a domain missing 20% of its preconditions number of experiments that are necessary with all the combinations of the three hypotheses-selection heuristics: generalization of experience (g), locality (l), and structural similarity (s). The number of experiments needed is greatly reduced when the three of them are used. *DC* represents a divide-and-conquer strategy that does not use the heuristics.

For 20% incompleteness, the three heuristics combined yield the best results. For 50% incompleteness, gl is about as good as gls. This is because when the operators are very incomplete similar operators may be missing the same conditions, so s is not very helpful. The effectiveness of s improves as new knowledge is added to the domain.

As we mentioned above, even though *DC* needs few experiments, the planner must achieve $n-1$ additional goals for each failure (see [Gil, 1992] for more details). With 20% of the preconditions missing, the number of additional goals that are necessary to achieve for experimentation is as follows:

failure	*gls*	*g*	*l*	*s*	*none*	*DC*
5	10	168	50	94	215	341
10	17	172	90	110	332	652

Failure indicates the order of the failure in the sequence in which they are obtained. With 50% of the preconditions missing:

failure	*gls*	*g*	*l*	*s*	*none*	*DC*
5	40	172	27	118	205	377
10	71	276	102	177	460	691
17	89	370	201	325	728	1217

In summary, the combination of the three heuristics (generalization of experience, structural similarity, and locality) reduces dramatically the number of experiments required, and yields the best performance. A divide and conquer strategy over the set of candidates requires more experiments that also have much more complex setups.

5 Conclusion

The work presented in this paper shows that it is possible for a planner to recover from knowledge-level impasses autonomously without need of causal explanations. Our approach uses domain-independent heuristics for choosing good experiments that do not require any knowledge other than the operators defined for planning and the planner's experiences in interacting with the environment. EXPO's performance us-

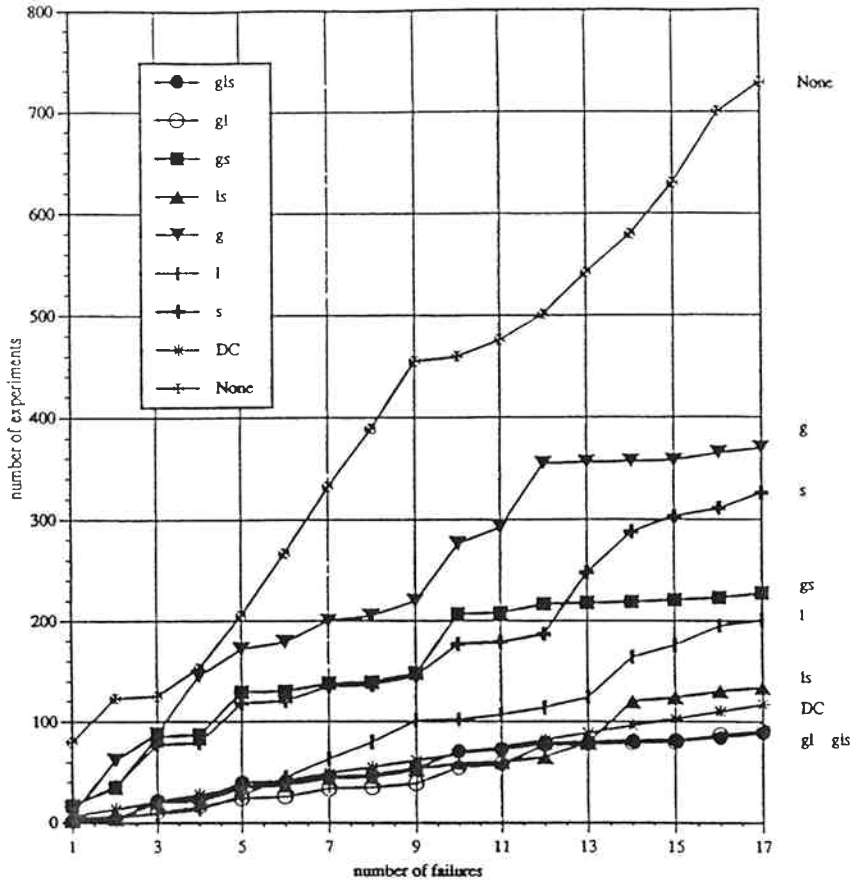

Figure 3: Given a domain missing 50% of its preconditions number of experiments that are necessary with all the combinations of the three hypotheses-selection heuristics: generalization of experience (g), locality (l), and structural similarity (s). The number of experiments needed is greatly reduced when the three of them are used. *DC* represents a divide-and-conquer strategy that does not use the heuristics.

ing all the heuristics combined shows that our method for experimentation is efficient. Additional domain-independent heuristics would improve this approach. The structural similarity heuristic can be extended to exploit other regularities in the domain, including inverse relations between operators. The knowledge intensive explanation-based methods would still outperform our system, but the problem of acquiring that additional knowledge remains. It would be interesting to combine the strengths of both approaches, relying on background domain knowledge when it exists and falling back on our heuristics otherwise.

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. The clarity of the paper and the work itself has also improved through comments from many people, including Kevin Knight, Eduardo Perez, and other members of the PRODIGY group. Thanks also to Ramesh Patil and Tom Russ for their comments on the paper.

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

[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.*, 1990] J. G. Carbonell, Y. Gil, R. L. Joseph, C. A. Knoblock, S. Minton, and M. M. Veloso. Designing an integrated architecture: The PRODIGY view. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society*, Boston, MA, 1990.

[Gil, 1991] Yolanda Gil. A domain-independent framework for effective experimentation in planning. In *Proceedings of the Eight International Workshop on Machine Leaning*, Evanston, IL, 1991. Morgan Kaufmann.

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

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

[Hume and Sammut, 1991] David Hume and Claude Sammut. Using inverse resolution to learn relations from experiments. In *Proceedings of the Eighth Machine Learning Workshop*, Evanston, IL, 1991.

[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.

[Langley *et al.*, In press] Pat Langley, Ken Thompson, Wayne Iba, John H. Gennari, and John A. Allen. An integrated cognitive architecture for autonomous agents. In Walter Van De Velde, editor, *Representation and Learning in Autonomous Agents.* North Holland, Amsterdam, The Netherlands, In press.

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

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

[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.

[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.