Proceedings of the Eight International Workshop on Machine Learni
Evanston, IL. June 1991.

A Domain-Independent Framework for Effective Experimentation in Planning

Yolanda Gil
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

This papers develops a general method for acquiring domain knowledge for planning by experimenting with their environment. When the expectations suggested by the domain knowledge and the observations differ, there is need and opportunity for learning. Since there are usually several possible ways to correct the domain, the system must experiment to gather additional information. This paper describes how to exploit the characteristics of planning domains in order to search the space of plausible hypotheses without the need for additional background knowledge to build causal explanations. Common features of planning domains are used in our system as heuristics to identify the most plausible hypotheses and avoid costly experiments. This work has been implemented in the PRODIGY planning architecture.

1 INTRODUCTION

Research on knowledge acquisition has lead to many methods for the acquisition of knowledge under different problem solving paradigms that include diagnosis and constraint satisfaction (Marcus, 1990). None of the current knowledge acquisition systems are designed for planning domains nor do they emphasize full automation. Planning systems offer the possibility of direct interaction with the environment, allowing autonomous learning. The best tool for the autonomous acquisition of a good domain model is the interaction with the external system being modeled (also called environment, or external world). Perceiving the result of the execution of actions in the environment allows the contrast of expectations given by the model and real results of the actions. When there is a difference between the expectations and the observations, there is opportunity for learning. The model must be corrected to avoid wrong expectations in the future. If several corrections to the model are possible, experimentation is needed to determine the adequate corrections.

Experiments must be designed appropriately to test hypotheses. This design process is complex, but it must be done efficiently. It must evaluate many parameters including the resources used in the experiment and the significance of the changes that the experiment produces in the external system. For these and other reasons, minimizing the amount of experiments is crucial. To do so, a system must be able to calibrate a set of hypotheses and determine which are the most plausible ones.

Much of the research on experimentation relies on some type of background knowledge that can be used to build causal explanations for each hypothesis (Rajamoney, 1988; Kedar *et al.*, 1991). They create an additional problem: the acquisition of that background knowledge. The framework for experimentation presented in this paper does not require domain-dependent background knowledge beyond the initial planning operators. It relies in the common characteristics that seem to be present in planning applications. Each application is a model designed for building plans to transform the state of the environment. The actions available in the model are related in different ways. For example, some actions reverse the changes that other actions do. Additionally, in planning domains actions can be isolated and identified along with their conditions and effects.

We have identified a set of heuristics to calibrate the relevance of hypotheses. These heuristics exploit characteristics of planning tasks, and include the locality of the actions, the structural regularity of domains, and the power of generalization of experience. We have augmented the PRODIGY planner (Minton *et al.*, 1989; Carbonell *et al.*, 1990) with experimentation capabilities that implement the ideas presented in this paper. The operators in PRODIGY represent actions. When an operator can be applied in the internal state, its preconditions are observed in the environment. If the observations agree with the internal state, the action that corresponds to the operator is executed. If the effects of the operator are observed to be true then the execution succeeded. Otherwise, our system considers it an execution failure and uses experimentation techniques to correct the model of the action represented by the operator. We assume deterministic actions whose effects can be immediately observed, noise-free sensors, and

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

Figure 1: An Incomplete Model of Grinding

that no other agents can affect the environment.

The paper is organized as follows. We begin by reviewing briefly our method for learning new preconditions of operators. Our system uses this method to find a set of hypotheses that contain a new precondition that must be satisfied in order to avoid future failures. The following Section describes several domain-independent heuristics for discriminating among this set of hypotheses and find the most plausible ones. Then, we review the related work and finish with the conclusions.

2 LEARNING BY EXPERIMENTATION IN PLANNING

Consider the operator in Figure 1, taken from a domain for machining metal parts (Gil, 1991). It 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 some dimension. This representation of grinding may seem correct, but it is missing many important facts. For example, grinding with a wheel of coarse grit produces a rough surface finish while a wheel of fine grit produces a smoother finish. Also, the grinder must have cutting fluid that it consumes in every operation, and grinding cannot be used for reducing the diameter of a part. After grinding, the part is no longer clean and it has metal burrs, and both the old surface finish and size are no longer valid.

We have used the *Operator Refinement Method* described in (Carbonell and Gil, 1990) to learn the missing preconditions and effects of GRIND. In essence, a new precondition can be learned when an action that succeeded before fails. The cause of the failure is typically that some unknown condition, required for correct performance, happened to be true earlier and is not true now. This condition can be any of the differences between the state in which a successful execution was possible and the state that lead to the failure. If there is only one difference, then it is added to the preconditions of the operator and a confirmation experiment is performed. If there are several predicates in the set of differences, experimentation is needed to find the relevant condition for the failure.

Here is a typical set obtained by our system. In this case, GRIND(grinder1, wheel1, vise1, part7, TOP) is successful but GRIND(grinder1, wheel1, vise1, part3, TOP) fails:[1]

```
(size-of part7 WIDTH 3)
(size-of part7 LENGTH 7)
(size-of part7 HEIGHT 2.5)
(material-of part7 BRASS)
(has-fluid grinder1)
(surface-finish part26 TOP SAWCUT)
(holding drill1 vise2 part26 TOP)
(material-of part26 STEEL)
(is-a drill1 DRILL)
(is-a drill-bit1 DRILL-BIT)
(material-of part37 COPPER)
(has-hole part37 TOP)
```

The problem can now be specified as follows:

Given: an operator $O_{incomplete-prec}$ and
a set of predicates $\{P_{candidates}\}_i$
Find: which of the predicates causes a successful
execution of $O_{missing-prec}$

The experiments consist on executing the action in states where different subsets of $\{P_{candidates}\}_i$ are true (we will come back to this in a moment). Each experiment is designed according with the following requirements:

- $E_{operator}$: $O_{incomplete-prec}$.
- $E_{current-state}$: State the system is currently in.
- $E_{exper-state}$: Any state that matches all the preconditions of the operator, plus an additional set of predicates $\{P\}_{current-candidates} \in \{P_{candidates}\}_i$.
- $E_{observe}$: Predicates to observe before and after the execution of the action that corresponds to $E_{operator}$.

In the absence of any criteria to discern which of the differences might be a more relevant cause for the failure, one strategy to follow is to do a binary search through through the set $\{P_{candidates}\}_i$. A binary search through a set of n predicates requires $log(n)$ experiments.

Each experiment requires building a plan to set the environment in a state as specified by $E_{exper-state}$. The execution of the plan may use up valuable resources, produce non-desirable changes the environment that are hard to undo, or to interfere with the main goals of the system's task. For all these reasons, it is important to minimize the number of experiments. If any information is available to determine a smaller subset of $\{P_{candidates}\}_i$ as relevant, the number of experiments may be greatly reduced. The next Section shows some domain-independent heuristics that guide this experimentation process doing precisely this.

[1]We omit the states in which the successful and the failed execution happened. The set of differences is formed by taking into account the bindings in each situation.

3 FINDING RELEVANT CONDITIONS FOR FAILURE

This Section presents different ways to exploit knowledge about the planning task to evaluate which predicates in a set of differences are more likely to have caused the failure. Our task is to identify a new precondition for an operator.

3.1 LOCALITY OF ACTIONS

A good heuristic to consider is the *locality of actions*. The preconditions and effects of actions are concentrated locally, affecting usually the objects under direct influence of the action. In our example we are grinding part7. The fact that this part is made of BRASS may be relevant for the failure obtained. However, it is probably not important that part37 is made of COPPER. This means that we can select the predicates in the set related to objects that the operator GRIND refers to directly.

This locality heuristic is implemented considering only the predicates in the state that contain some of the objects included in the bindings of the parameters of the operator. In our example, if we extract the predicates that include any of {grinder1, wheel1, vise1, part7, TOP} we obtain the following subset:

```
(size-of part7 WIDTH 3)
(size-of part7 LENGTH 7)
(size-of part7 HEIGHT 2.5)
(material-of part7 BRASS)
(has-fluid grinder1)
(surface-finish part26 TOP SAWCUT)
(has-hole part37 TOP)
(holding drill1 vise2 part37 TOP)
```

Notice that with this heuristic we elliminated from the list many predicates that were in fact irrelevant for grinding. For example, many facts about parts that are not the part being ground have disappeared.

This heuristic is not helpful when the set of variables that appear in an operator is incomplete. If the operator for grinding were missing any predicates that have to do with the tool being used, the system would never learn that the tool is important for the action. A possible way around this problem is to give some structured knowledge to the state. For example, to have information in the state about where everything is, and what things are close to each other. In this work, we avoid this kind of approach because it requires adding to the system knowledge that is not strictly required for planning.

Another problem is that this heuristic does not always propose relevant differences. Consider the subset of differences just obtained. Because grinding is being done to part7, all the facts about part7 could be relevant. But since the TOP is the side being ground, any facts that have to do with TOP are also considered relevant. This includes for example the fact that part37 has a hole on the TOP, which is not relevant to the application of the operator.

3.2 GENERALIZATION OF EXPERIENCE

A very helpful strategy is *generalization of experience*. Generalizing successful situations tells us what predicates appear in all success states. This summary of past experience helps us to locate relevant causes of failures.

This heuristic is implemented generalising successful situations through the bindings of the operator. This will give us the set of predicates that have appeared in all of them. After removing from that set the predicates that correspond to the preconditions of the operator, we obtain the following set (variables are shown between angle brackets):

```
(material-of <part> BRASS)
(surface-finish <part> <side> SAWCUT)
(has-fluid <grinder>)
```

Notice that this set is much smaller than the one in the previous Section, where we only considered a single success situation. When the system encounters more successful situations, then the set of differences becomes smaller.

If the system has no previous experience with the application of the operator this generalization strategy is not helpful. Another reason for the failure of this strategy is when not much generalization can be extracted from successful applications.

A generalization of all the possible situations where grinding is successfully applied is exactly the correct precondition expression sought. The preconditions of an operator can be seen as a concept that expresses the class of states in which the operator is applicable. Thus, learning the precondition expression of an operator is a problem of concept learning. The initial precondition expression of an operator is the initial description of the concept. Each successful execution of an action is a positive example of the concept, and each failure a negative example. Experimentation is an additional source of examples, and it provides the learner with the ability to design instances and direct the learning.

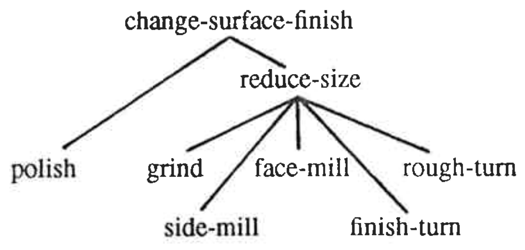
However, this concept learning is simpler due to common simplifying characteristics of planning tasks. There are no misclassified examples. The effects of actions can be observed immediately after execution. The observations are collected through noise-free sensors. Under these assumptions, our classification of execution success and failure never produces noisy data. As far as the language used for expressing the concepts, 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 in their effects under different conditions are described in separate operators. Disjunctions can be (and are) expressed explicitly in different operators. In this sense, limiting learning to conjunctive expressions is still useful.

3.3 THE STRUCTURE OF DOMAIN KNOWLEDGE

Operators for a single task are often closely related to one another. Some operators are inverses, i.e., they undo each other's effects. Some operators have similar effects, but are applied under different conditions. Both of these relations appear in the machining domain. There are operators for holding a part with a certain holding device, and there are operators to release the part from the device. There are operators for holding a tool in a machine, and operators for releasing tools from machines. The operators for drilling are all similar to one another. So are the operators for polishing surfaces. These relations of similarity and reversibility constitute the heuristic of *structural regularity* of the domain.

Structural similarity will help us identify what hypotheses are more plausible by looking at similar operators to the operator being considered. This is a very general idea, and it can be used for learning new preconditions, as described next.

One way to implement this heuristic is to organize the operators in a hierarchy, so that similar operators can be easily located. The hierarchy can be built through the preconditions and effects of operators. In our machining domain, part of the hierarchy that includes the grinding operation is as follows:



Consider the set of differences obtained in the precious Section. 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.[2] 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. The heuristic suggests that the differences should be considered in the following order:

1. (has-fluid <grinder>)
2. (material-of <part> BRASS)
3. (surface-finish <part> <side> SAWCUT)

This heuristic is not very helpful if there are no similar operators or if there are similar operators but they are also incomplete.

[2]Cutting fluids cool both the cutting edges of the tool and the part, aid in chip clearance, and improve the surface finish. Notice how much background information would be needed to explain that the presence of cutting fluid is important for grinding.

4 RELATED WORK

LEX (Mitchell *et al.*, 1983) is a system designed to learn by experimentation in a planning task. LEX uses version spaces to represent control rules. Its experiments are based in introspection, not interaction from the environment. This is why it can only learn control knowledge. The rules that it learns describe how to guide the search and are heuristics that recommend which operators to apply under what situations. Unlike our system, LEX never learns new domain knowledge. Many other systems (e.g., (Allen and Langley, 1990)) that modify the specifications of their operators cannot interact with the environment, just like LEX. In fact, they are learning better ways of applying known rules. Consequently, they are all learning a more effective way of expressing the knowledge that the system already has. In other words, they are symbol level learning systems while our system exhibits learning at the knowledge level (Dietterich, 1986).

LIVE (Shen, 1989) is a system that acquires domain knowledge from the environment. This system is designed for exploration and discovery. It can learn new operators and formulate new terms if the language is inadequate. Its experimentation capabilities are more limited than the ones presented in this paper. Our work is not so concerned with exploration, but it is more focused on what is required for the system in order to plan.

Another system that learns from the environment is Robo-Soar (Laird *et al.*, 1990). This system uses outside guidance to refine incomplete and incorrect domain knowledge. Soar is connected to a real environment via robotics systems. Our system learns without human guidance, although it is not connected to a real environment. (Maes and Brooks, 1990) describes an algorithm for learning in a behavior-based architecture implemented in a robotic system. The algorithm incrementally changes the preconditions of behaviors on the basis of positive and negative feedback from the environment. The architecture in our system is very different from this one, as well as the expressions to be learned.

Failures are very useful for learning. Most of the systems that learn from failures do so by building causal explanations using some type of background knowledge. Kedar *et al.* [1991] present an approach to building explanations for operator failures using a set of domain constraints. Rajamoney [1988] describes a system that learns preconditions of processes. The system uses qualitative reasoning and explanation-based learning for explaining the failures. Salzberg [1985] describes some heuristics for designing hypotheses when a failure occurs. The heuristics are applied to nondeterministic systems (results of horse races), and causal knowledge is used to calibrate the hypotheses.

Scientific discovery is naturally related to the subject of experimentation. Many programs have been built for formulating theories by experimentation (Langley *et al.*, 1987; Kulkarni, 1988). Experimentation to understand scientific phenomena requires a great deal of domain specific knowledge. The system described in this paper presents a more pragmatic approach to experimentation, and relies on knowledge-independent heuristics.

5 CONCLUSIONS

We have described several heuristics for proposing relevant conditions for failure based exclusively on the knowledge given to a planner. No additional background knowledge is needed to recover from the failures. The heuristics include locality of actions, structural regularity, and generalization of experience. They are very different in nature and useful in different cases, but they can be combined to guide the experimentation process. We have found them helpful for many different planning domains. Work is under way to measure the number of experiments performed by the system using these heuristics and compare them with an optimal experimentation strategy.

Our system can currently learn preconditions of operators when they are conjunctive expressions of predicates that are directly observable in the state. Much work remains to be done, mainly about generalization through an object type hierarchy, learning negations of predicates, and the detection of disjunctions and other functional constructors. In particular, learning negations of predicates presents interesting problems because of the closed-world assumption made in planning representations.

Experimentation is a very powerful tool for the autonomous acquisition of knowledge for planning systems. The impact of this work should be felt in robotic and other autonomous planning tasks.

Acknowledgements

The research reported in this paper has benefited enormously from discussions with Jaime Carbonell, Tom Mitchell, Herbert Simon, and Jeff Schlimmer. Kevin Knight and Eduardo Perez provided very useful comments on an earlier draft. The author would like to thank past and present members of the PRODIGY project for their helpful criticism of this work.

This research was sponsored by the Office of Naval Research under Contract N00014-86-K-0678. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government.

References

[Allen and Langley, 1990] J. A. Allen and P. Langley. Integrating memory and search in planning. In *Proceedings of the 1990 DARPA Symposium on Planning*, San Diego, CA, 1990.

[Carbonell and Gil, 1990] J. G. Carbonell and Y. Gil. Learning by experimentation: The operator refinement method. In *Machine Learning, An Artificial Intelligence Approach, Volume III*. Morgan Kaufmann, Irvine, CA, 1990.

[Carbonell *et al.*, 1990] J. G. Carbonell, Y. Gil, R. 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.

[Dietterich, 1986] T. G. Dietterich. Learning at the knowledge level. *Machine Learning*, 1(3), 1986.

[Gil, 1991] Y. Gil. A specification of process planning for PRODIGY. Technical report, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, 1991. Technical Report, forthcoming.

[Kedar *et al.*, 1991] S. Kedar, J. Bresina, and L. Dent. The blind leading the blind: Mutual refinement of approximate theories. In *Submitted to the Proceedings of the Eighth International Workshop on Machine Learning*, Chicago, IL, 1991.

[Kulkarni, 1988] D. S. Kulkarni. *The Process of Scientific Research: The Strategy of Experimentation*. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1988.

[Laird *et al.*, 1990] J. E. Laird, M. Hucka, E. S. Yager, and C. M. Tuck. Correcting and extending domain knowledge using outside guidance. In *Proceedings of the Seventh International Conference on Machine Learning*, Austin, TX, 1990.

[Langley *et al.*, 1987] P. Langley, H. A. Simon, G. L. Bradshaw, and J. M. Zytkow. *Scientific Discovery: Computational Explorations of the Creative Processes*. MIT Press, 1987.

[Maes and Brooks, 1990] P. Maes and R. A. Brooks. Learning to coordinate behaviors. In *Proceedings of the Eight National Conference on Artificial Intelligence*, Boston, MA, 1990.

[Marcus, 1990] S. Marcus, editor. *Knowledge Acquisition: Selected Research and Commentary*. Kluwer Academic Publishers, 1990.

[Minton *et al.*, 1989] S. Minton, J. G. Carbonell, C. A. Knoblock, D. R. Kuokka, O. Etzioni, and Y. Gil. Explanation-based learning: A problem solving perspective. *Artificial Intelligence*, 40(1-3):63–118, 1989.

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

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

[Salzberg, 1985] S. Salzberg. Heuristics for inductive learning. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, 1985.

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