Learning Operator Transformations

*Bruce W. Porter**
Dennis F. Kibler

Information and Computer Science Department
University of California at Irvine
Irvine, Ca 92717**

Abstract

A relational model representation of the effect of operators is learned and used to improve the acquisition of heuristics for problem solving. A model for each operator in a problem solving domain is learned from example applications of the operator. The representation is shown to improve the rate of learning heuristics for solving symbolic integration problems.

I. Introduction

Machine learning research in problem solving domains has focussed on acquiring heuristics to guide the application of operators. Predominantly, researchers have assumed an operator representation (e.g. program code) which hides the operator semantics [5,6,7,10]. We call this operator representation *opaque* in that the transformation performed by the operator is not explicit. In contrast, *transparent* operator representations (e.g. STRIPS-like) enable the learning agent to reason with operator definitions. This research examines two issues:

- how to learn transparent operator representations from opaque representations.
- how to improve the process of acquiring problem solving heuristics by using transparent operator representations.

We demonstrate the approach with a PROLOG implementation, named PET, which learns to solve symbolic integration problems.

Section 3 formalizes the representation for operators used by PET and describes an algorithm for learning the representation. We call this representation of an operator a **relational model**. We discuss a two step algorithm for learning a relational model for an opaque operator OP from example applications of OP. First PET induces a general form, PRE, for states in which OP is usefully applied and a general form, POST, for states resulting from the application. Then PET selects relations from background knowledge [12] which link features of PRE with features of POST. Discovering a good relational model is formulated as a state space search.

Section 4 discusses how relational models improve the process of learning problem solving heuristics. The representation reveals features of heuristics which may be overly specific. Further, the representation suggests training instances which test these features, thereby guiding generalization.

For preliminaries, section 2 briefly reviews our past research on PET which serves as a "testbed" for experimenting with operator representations.

II. The PET System

This section presents an overview of the PET system [4,5]. Two central features of PET are **episodic learning** of useful problem solving macros and **perturbation** to automatically generate training instances.

Episodic learning is an incremental approach to learning heuristics which recommend problem solving operators and sequences. The LEX system [7,8] learns heuristic rules which recommend individual operators. The heuristics learned are an accurate compilation of past problem solving experience, but, taken together, may not enable efficient problem solving. The contextual information of an operator's position in problem solving sequences is not captured by LEX. MACROPS [3], on the other hand, learns operator sequences but does not acquire heuristics to select useful sequences to apply to particular problem states. Generally useful sequences are not identified and re-use of the macros during problem solving results in combinatorial explosion [2].

PET learns heuristics for operator sequences by incrementally learning new sub-goals. PET can only learn a heuristic for an operator if the purpose of the operator is understood. Initially, this restricts PET to learning heuristics for operators which achieve a goal state. Problem states covered by these heuristics are learned as sub-goals. Now PET learns heuristics for operators which achieve the subgoals. Operator sequences thus grow incrementally.

Perturbation is a technique for reducing teacher involvement during training by automatically generating near examples and near-misses. The role of the teacher in learning from examples is to generate and classify training instances. This role is diminished by shifting responsibility to the student. Given a positive instance POS for operator OP, PET generates and classifies further instances by:

- generation: make a minimal modification of POS by applying *perturbation operators* to POS. These operators select a feature F of POS and generate POS' by deleting F from POS or by replacing F by a sibling in a concept hierarchy tree.
- classification: POS' is a positive instance for operator OP *iff* apply(OP,POS') yields the same (sub)goal as apply(OP,POS).

---

* New address: Computer Science Department, The University of Texas at Austin.

** This research was supported by the Naval Ocean Systems Center under contract N00123-81-1165.

Viewed abstractly, episodic learning of problem solving involves learning *why* individual operators are useful and perturbation is useful in learning *when* operators should be applied. Sections 3 and 4 demonstrate the importance of learning an explicit representation of *what* individual operators do during problem solving.

III. Learning Relational Models

This section formalizes the relational model representation of operators and presents an algorithm for learning the representation from examples.

A relational model of an operator OP is an augmentation of a heuristic rule for OP. Following Amarel [1], a heuristic for OP is a production rule which explicitly represents OP's pre and post conditions. The form of the rule is:

$$PRE \overset{OP}{\rightarrow} POST$$

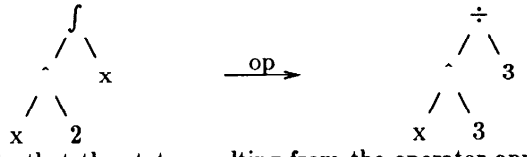
and has the interpretation:

> IF the current state, S, matches *PRE*, **and** the state resulting from apply(OP,S) matches *POST* **THEN** OP is recommended in S.

The pre and post state conditions are represented as parse trees of problem states. The following is an example production rule which recommends the operator

$$OP: \int x^n\,dx \rightarrow \frac{x^{n+1}}{n+1} + C$$

in state $\int x^2\,dx$ ("+C" is dropped for simplicity):



Note that the state resulting from the operator application, *POST*, is explicitly represented as the RHS of the rule.

Heuristic rules are generalized using "standard" generalization techniques. For example, the candidate elimination algorithm [7] is used by LEX to form generalizations of heuristic rules of the form $PRE \rightarrow OP$. Applying the algorithm to states resulting from OP's application yields a generalization of *POST*. For each operator OP in a problem solving domain, PET uses the dropping conditions and climbing hierarchy tree generalization operators to induce general forms both for states in which OP is recommended and for states resulting from recommended applications.*

Relational models are an augmentation of heuristic rules with background knowledge. The background knowledge consists of domain specific relations. In the domain of mathematics, PET uses the relations *equal*(X,Y), *suc*(N,M), *sum*(L,M,N), *product*(L,M,N), *power*(L,M,N), and *derivative*(X,Y).

A relational model is a tuple $\langle OP, PRE, POST, AUG\rangle$. The augmentation, *AUG*, is a set of relations $\{rel_1, \ldots, rel_n\}$ from background knowledge. Each relation $rel_i \in AUG$ has a relation name, or functor, and $m \geq 2$ arguments, $\{a_1, a_2, \ldots, a_m\}$. The purpose of the augmentation is to relate subexpressions of *PRE* with subexpressions of *POST*, thereby "linking" *PRE* to *POST*. To establish these links, each a_j is constrained to be a subexpression of either *PRE* or *POST*, such that not all a_j are from the same source. (Actually, this is a simplification. By allowing a_j to be a subexpression of an argument of another relation in *AUG*, composites of relational descriptors can be formed by "daisy-chaining" a link between *PRE* and *POST* through multiple descriptors. For example, the relation that *PRE* is the double derivative of *POST* is represented by *derivative*(*PRE*,X),*derivative*(X,*POST*). See [9] for a description of the algorithm which permits chaining and its ramifications.)

An evaluation function ξ estimates the "quality" of a relational model by measuring the coverage of *PRE* and *POST* by *AUG*. Intuitively, coverage is a measure of the number of nodes of *PRE* and *POST* which are in arguments of *AUG*. Formally,

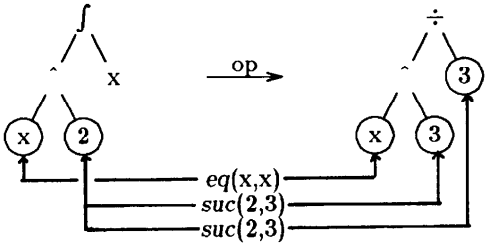
$$\xi(\langle OP, PRE, POST, AUG\rangle) = \mid S_1 \mid + \mid S_2 \mid$$

where $\mid S \mid$ is the cardinality of set S and

$$S_1 = \{nodes\ n\ in\ PRE: \exists rel(a_1, a_2 \ldots, a_m) \in AUG \wedge \exists i, 1 \leq i \leq m, such\ that\ descendantof(n, a_i)\}$$

S_2 is similarly defined for nodes in *POST*. Note that an individual node in *PRE* or *POST* can contribute to coverage at most once since S_1 and S_2 are sets not bags.

In addition to representing the transformation performed by an operator, a relational model constrains the interpretation of the heuristic. For example, the rule on page 3 is augmented with *eq* and *suc* relations, yielding the relational model:



The interpretation of the heuristic is:

> IF the current state, S, matches $\int x^2 dx$ and the state resulting from apply(OP,S) matches $\frac{x^3}{3}$ **such that** the relations in the augmentation hold, THEN OP is recommended in S.

Given an unaugmented rule $R = \langle OP, PRE, POST\rangle$, a relational model of R is constructed by searching for the set of instantiated augmentation relations, *AUG*, which best covers R. This search is implemented in PET as a beam-search through the space of candidate augmentations. In this space, nodes are represented by the tuple $\langle AUG, Pool\rangle$ where *Pool* is the set of subexpressions of *PRE* and *POST* not covered by *AUG*. In particular, the initial state is $\langle nil, \{PRE \cup POST\}\rangle$. There is one operator in this search which is described by:

---

* This research does not present a novel generalization technique. Instead, representations which improve existing techniques are proposed.

Given a state $\langle AUG, Pool \rangle$,
  SELECT a relational descriptor, D, from the set of background concepts.
  INSTANTIATE D with members of $Pool$ or their sub-expressions.
  REMOVE selected Pool members from $Pool$, yielding $Pool'$.
  ADD instantiated descriptor to AUG, yielding AUG'.
Generate new state $\langle AUG', Pool' \rangle$.

The search terminates with AUG when continued search fails to improve coverage.

Built-in biases reduce the non-determinism of the search for an augmentation with maximal coverage and minimal complexity. In the selection of a relational descriptor, preference is given to more primitive relations, such as *equal* and *suc*, over more complex relations, such as *product*. Further, there are semantic constraints on the subexpressions selected to instantiate a relation. For example, the first parameter in the *derivative* relation must contain a variable of differentiation. Finally, note that the algorithm tries large subexpressions from PRE and $POST$ before small subexpressions, thereby maximizing the coverage of the augmentation. If two relational models have the same coverage, then the one with fewer relations is preferred.

This section introduced relational models and briefly described how they are constructed. It demonstrated that relational models can be built using the general techniques of state-space search. This search is constrained by built-in bias toward simple models with maximum coverage and by semantic constraints on relational descriptors. Section 4 describes an application of relational models.

IV. Using Relational Models

This section describes how PET uses relational models to improve learning of problem solving heuristics. Relational models explicitly represent the transformation performed by operators. This enables PET to reason with operator semantics to guide the generation of training instances.

As described in section 2, PET applies perturbation operators to a single teacher-supplied training instance to generate and classify multiple near-examples and near-misses.* Perturbation automates part of the teacher's role, but not the task of *selectively* generating training instances which are most useful in enabling concept convergence. Relational models present an alternative to naïvely generating all possible training instances.

PET selectively generates training instances which test features of a concept suspected to be spurious or overly specific. Spurious features are removed with the dropping conditions generalization operator. Rather than test the relevance of every feature, PET heuristically selects candidates. Given relational model $\langle OP, PRE, POST, AUG \rangle$ the heuristic states:

> Features of PRE which are not transformed by OP may be irrelevant to the rule recommending OP.

Those features of PRE which are not transformed are exactly those linked by the *eq* relation to features of $POST$. This heuristic identifies candidate irrelevant features which can be tested with perturbation.

* LEX uses a similar technique. See [8].

Relational models also guide the generation of training instances which test features suspected to be overly specific. Again, the selection of candidate perturbation operators is heuristically guided. The heuristic relies on two sources of information:

- relational models – which represent the transformation performed by an individual rule application.
- episodes – which represent the "chaining" of individual rules into a useful problem solving sequence.

Consider an episode E consisting of rule applications $r_1, r_2, \ldots, r_n$. Each rule r_i is represented with relational model $\langle OP_i, PRE_i, POST_i, AUG_i \rangle$. AUG_i represents the "intra-rule" links between PRE_i and $POST_i$. "Inter-rule" links are implicit in E. As reviewed in section 2, r_i is added to an episode if it enables r_{i+1}. This establishes an implicit link between $POST_i$ and PRE_{i+1}. Constraints imposed on r_i by r_j, $i < j$, are discovered by following inter-rule links through E and intra-rule links through rules. These constraints suggest perturbation operators for r_i.

The heuristic of locating overly specific features by propagating constraints through episodes is motivated by this observation:

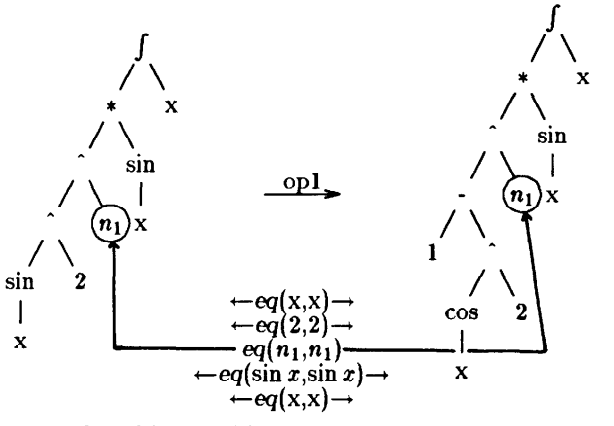
> Due to the incremental growth of episodes, for any pair of rules r_i and r_j, $i < j$ in E, the size of the training set for r_j exceeds the size of the training set for r_i because every training instance for r_i is also a training instance for r_j.

This suggests that features of PRE_j and $POST_j$ are more general than features of PRE_i and $POST_i$. PET selects perturbation operators which capitalize on this observation by back-propagating general features of PRE_j to potentially overly-specific features of PRE_i.

We illustrate this back-propagation with an example from Utgoff [11]. Assume that from prior training for operator

$$OP1 : \sin^2 x \rightarrow 1 - \cos^2 x$$

PET has acquired the following relational model:

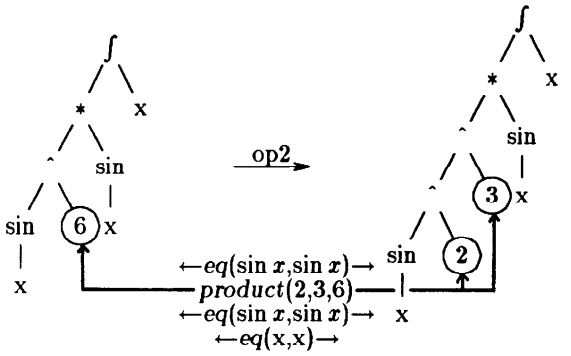


Note that this model has been generalized from ground instances such that PRE_{op1} matches states of the form $\int (\sin^2 x)^{nonzerointeger} \sin x \, dx$.

Now PET is given the training instance $\int \sin^6 x \sin x \, dx$ with the advice to apply the opaque operator:

$$OP2 : \sin^n x \rightarrow (\sin^2 x)^{\frac{n}{2}}$$

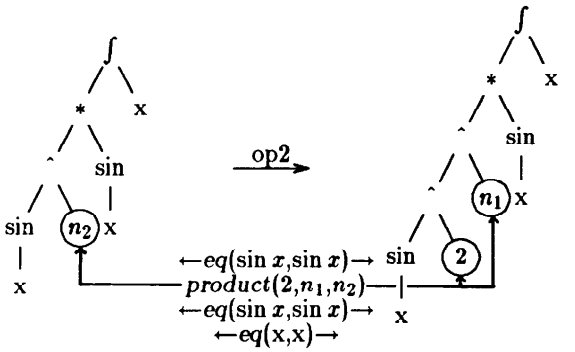
PET applies the operator, yielding $\int (\sin^2 x)^3 \sin x \, dx$. As reviewed in section 2, PET can only learn a rule for this training instance if it achieves a known (sub)goal (allowing the rule to be integrated into an existing episode). In this example, the training instance achieves the subgoal defined by PRE_{op1}. The following relational model for the training instance is built by the state-space algorithm in section 3:



Now that episodic learning has associated the relational models for $OP1$ and $OP2$, perturbation operators are applied to generalize the model for $OP2$. The relaxed constraint in PRE_{op1} is regressed through the episode with the potential of identifying a feature of PRE_{op2} which can be relaxed (generalized). The inter-rule link implicit in episodes connects the relational model of OP2 with the relational model of OP1. Matching $POST_{op2}$ with PRE_{op1} binds variable n_1 with 3. This suggests that the relational model for OP2 is overly-specific. Perturbation tests relaxing this constraint by generating a training instance with the feature slightly modified. This is done by traversing intra-rule links represented by the augmentation. Specifically, PET generates a useful training instance by the following steps:

1. Locate the relation $r \in AUG_{op2}$ with argument of 3 from $POST_{op2}$. In this case, $r = product(2,3,6)$.
2. Perturb r to generate a slight variant, r'. This is done in three steps: First, replace the argument with a neighboring sibling in a concept hierarchy tree. In this case, replace 3 with 4. Second, locate an argument p in r such that p is a sub-expression of PRE_{op2} and replace it by free variable x. In this case, $p = 6$. Third, evaluate the resulting partially instantiated descriptor to uniquely bind x to p'. In this example, $p' = 8$ and r' =product(2,4,8).
3. Generate PRE'_{op2}, a perturbation of PRE_{op2}, by replacing p by p'. Here, $PRE'_{op2} = \int \sin^8 x \sin x \, dx$
4. Classify PRE'_{op2} as an example or near-miss of a state in which $op2$ is useful. As reviewed in section 2, PRE'_{op2} is an example if apply(OP2,PRE'_{op2}) achieves the same subgoal as apply(OP2,PRE_{op2}). In this example, PRE'_{op2} is an example which achieves the subgoal of PRE_{op1}.

Finally, PET generalizes the original training instance with examples generated by perturbation. The following relational model is the minimal generalization of this (2 member) training set:



Note that the $product(2,n_1,n_2)$ augmentation descriptor corresponds to the concept of *even-integer*(n_2). PET uses this relational model to guide the incremental refinement of the rule with subsequent training (see [9]).

In addition to this example, PET learns relational models for eighteen other operators. The longest episode is a sequence of seven operations. We are currently examining metrics for measuring performance of learning algorithms. Representational adequacy is of major importance. For heuristic accuracy, description languages for rules should represent relations observed among features during training. Relational models address this concern.

V. Summary

This paper examines the effect of operator representation on the acquisition of heuristics for problem solving. Opaque operator representations, which conceal the transformation performed by the operator, are frequently used. Transparent operator representations reveal the transformation, allowing reasoning about operator effects. However, it is unreasonable to assume transparency in "real-world" learning domains.

This paper presents an approach to learning transparent representations from examples of opaque operator applications. The transparent representation is called a **relational model**. Domain-specific background knowledge, represented as a set of relations, augments rules which model the transformation of each operator. The learning algorithm is described as a state-space search for an augmentation which is simple yet predictive. Once learned, a relational model for an operator OP is also a heuristic which identifies states in which OP is recommended.

Lastly, the paper examines an advantage of the relational model representation over "traditional" opaque representations. The representation reveals features of heuristics which are candidates for generalization. A method for automatically generating training instances which test these candidates is presented.

The research ideas are implemented in a system which learns to solve symbolic integration problems. Please refer to [9] for a more complete description of this research including an algorithm for generalizing over a set of rules represented as relational models.

References

[1] Amarel, S. "On Representations of Problems of Reasoning About Actions," in *Machine Intelligence 3*, D. Michie (Ed.), 131-171, 1968, Edinburgh Univ. Press.

[2] Carbonell, J. "Learning by Analogy: Formulating and Generalizing Plans from Past Experience," in *Machine Learning*, Michalski, Carbonell, Mitchell (Eds.), 137-162, 1983, Tioga Press.

[3] Fikes, R., Hart, P. and Nilsson, N. "Learning and Executing Generalized Robot Plans," *Artificial Intelligence, 3*, 251-288, 1972, North-Holland Publishing Co.

[4] Kibler, D. and Porter, B. "Perturbation: A Means for Guiding Generalization," *Proceedings of International Joint Conference on Artificial Intelligence*, 415-418, 1983.

[5] Kibler, D. and Porter, B. "Episodic Learning," *Proceedings of National Conference on Artificial Intelligence*, 191-196, 1983.

[6] Langley, P. "Learning Effective Search Heuristics," *Proceedings of International Joint Conference on Artificial Intelligence*, 419-421, 1983.

[7] Mitchell, T. *Version Spaces: An Approach to Concept Learning*, PhD Dissertation, Stanford University Computer Science Dept, December 1978, CS-78-711.

[8] Mitchell, T., Utgoff, P., and Banerji, R. "Learning by Experimentation: Acquiring and Refining Problem Solving Heuristics," *Machine Learning*, Michalski, Carbonell, Mitchell (Eds.), 163-190, Tioga Press, 1983.

[9] Porter, B. *Learning Problem Solving*, PhD Dissertation, University of California at Irvine, Information and Computer Science Dept, (forthcoming).

[10] Silver, B. "Learning Equation Solving Methods from Worked Examples," *International Machine Learning Workshop*, 99-104, June 22-24,1983, Monticello, Illinois.

[11] Utgoff, P. "Adjusting Bias in Concept Learning," *International Machine Learning Workshop*, 105-109, June 22-24,1983, Monticello, Illinois.

[12] Vere, S.A. "Induction of Relational Productions in the Presence of Background Information," *Proceedings of International Joint Conference on Artificial Intelligence*, 349-355, 1977.

[13] Waldinger, R. "Achieving Several Goals Simultaneously," *Machine Intelligence 8*, 1977 Elcock, E.W. and Michie D. (eds.), New York: Halstead and Wiley.