Not the Last Word on EBL Algorithms*

Charles Elkan
Alberto Segre

TR 89-1010
May 1989

Department of Computer Science
Cornell University
Ithaca, NY 14853-7501

*This research is supported by the Office of Naval Research grant N0014-88-K-0123.

Not the Last Word on EBL Algorithms[1]

Charles Elkan
Alberto Segre

Department of Computer Science
Upson Hall
Cornell University
Ithaca, New York 14853-7501

May 1989

ABSTRACT: This paper describes a new domain-independent *explanation-based learning* (EBL) algorithm that is able to acquire useful new rules in situations where previous EBL algorithms would fail. The new algorithm is *complete* in the sense that every valid rule that can be extracted from an explanation can be extracted by this algorithm. The new algorithm is described inside a framework that provides insight into how the design of successful EBL systems takes into account operationality and imperfect domain theory issues.

[1]Support for this research was provided by the United States Office of Naval Research through grant N0014-88-K-0123.

1 Introduction

This paper presents a new domain-independent *explanation-based learning* (EBL) algorithm that we call EBL*. The EBL* algorithm is provably *complete* in that every valid rule that can be extracted from a given explanation is extractable by EBL*. This claim cannot be made about any other EBL algorithm.

Previous attempts to formulate EBL in a domain-independent fashion [4,15, 16] have failed to address the completeness issue, and they have failed to analyze convincingly the sources of the success of impressive EBL systems. This paper also presents a new perspective on EBL algorithms that provides insight into this question and others—EBL can be characterized as heuristic search through a space of transformed explanations; interesting EBL applications differ in the control heuristics they apply to guide this search. This point of view clarifies how operationality issues and imperfect domain theory issues affect the design of successful EBL systems.

We begin by describing our perspective on EBL algorithms, examining previous domain-independent algorithms such as EBG, EGGS, and IMEX inside this framework. We then present the new algorithm, EBL*, along with a proof of its completeness. Next we propose four domain-independent heuristics for controlling the search among transformed explanations that is necessary to acquire operational new rules. Finally we discuss the general control problem and its implications.

2 A perspective on EBL algorithms

In order to support the application of EBL techniques, a knowledge-representation formalism must meet certain conditions. Most obviously, the formalism must be declarative, in the sense that it must permit the construction of explanations. Consider for example the creatures designed by Brooks and his colleagues [2]. The knowledge a creature possesses about its environment is procedurally encoded in a network of finite state machines. The network specifies a direct mapping between external stimuli and appropriate responses. Applying EBL to amplify a creature's capabilities is impossible because it never constructs explanations.

Less obviously, knowledge must be represented in a way that supports the construction of explanations that are capable of being generalized. Knowledge should be expressed in terms of individuals and relationships between them, so that specific individuals in explanations can be replaced by generic individuals. Most declarative knowledge-representation formalisms do meet this criterion. Some EBL systems, for example LEX [14], use formulas of first-order logic to

express their knowledge. Other EBL systems, for example ARMS [18], construct explanations using knowledge structures such as schemata and frames that are not logical formulas. The knowledge contained in these structures can however be reformulated in first-order logic [9]. Moreover, like logical formulas, schemata and frames talk about individuals and relationships. So the generalization operations performed by an EBL algorithm on an explanation composed from logical formulas or from other knowledge structures are analogous.

There are declarative knowledge-representation formalisms with which EBL is difficult, although perhaps not impossible. Consider a system that stores its knowledge in a conventional way, but which uses a propositional reasoner to perform inference. The qualitative physics envisioner of Forbus [7] is just such a system. It answers questions about the behaviour of an artifact by compiling relevant facts about the artifact, expressed in a schema language, into assumptions which are passed to an ATMS. All inferencing happens inside the ATMS at a propositional level. Results are translated back into the schema language, but the derivations performed by the ATMS are not. Strictly propositional derivations cannot be generalized by current EBL techniques.

For the purpose of comparing EBL algorithms, it is reasonable to adopt a very simple knowledge-representation formalism that meets the criteria just discussed. Our choice is a formalism involving only facts such as $hungry(jack)$ and rules such as $hungry(?x) \rightarrow angry(?x)$. Technically, facts and rules are first-order definite clauses.[2] This formalism is not expressive enough for many applications, and it is difficult to use with specialized inference mechanisms. Nevertheless it constitutes the core of MRS, a widely used automated reasoning shell [17], and it has been used in much previous EBL work [10,11,13].

2.1 Forming explanations from old rules

A special representation of explanations is useful for discussing EBL algorithms. If knowledge is represented as logical facts and rules, then explanations are simply proofs. Moreover, all proofs have the same tree structure, because a fact can only be proven by matching it with the consequent of a rule, and then recursively proving the antecedents of the rule.

The special representation makes explicit what rule instances are used in a proof. The nodes of the tree representing a proof are of two types: subgoals and consequents. Proof trees have edges of two types: rule edges and match edges. Rule edges link the consequent of each instance of a rule to its subgoals. Match edges link each subgoal to the consequent of the rule instance used to

[2] A good reference for the concepts of logic used in this paper is the textbook by Genesereth and Nilsson [8].

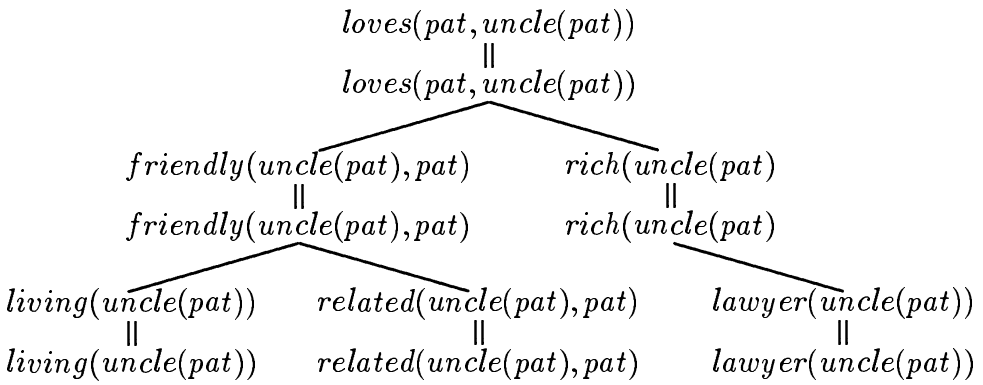


Figure 1: A proof of $loves(pat, uncle(pat))$

prove it. The root of any proof tree is a subgoal node, and all the leaves of trees representing completed proofs are consequents of rules having no antecedents.

Relative to a domain theory, a proof tree is *valid* if

(i) all subgoal-consequent pairs linked by match edges in the tree are identical, and

(ii) every rule instance in the tree is a legal instance of a rule in the domain theory.

If a proof tree is valid, then the goal appearing at its root is logically entailed by the set of leaves of the tree that are subgoals. The following example should make this clear.

Example 2.1: Consider the domain theory

$$friendly(?y, ?x) \wedge rich(?y) \rightarrow loves(?x, ?y)$$
$$lawyer(?x) \rightarrow rich(?x)$$
$$living(?y) \wedge related(?x, ?y) \rightarrow friendly(?x, ?y)$$
$$related(?x, uncle(?x))$$
$$living(uncle(pat))$$
$$lawyer(uncle(pat))$$

Figure 1 presents a proof obtained using this domain theory by an artificially intelligent kidnapper of the fact that Pat loves his uncle. Match edges are indicated by double lines and rule edges are indicated by single lines. In the proof of Figure 1, if a subgoal and a consequent are linked by a match edge, then they are identical. Also, every rule instance in the proof, say

$lawyer(uncle(pat)) \rightarrow rich(uncle(pat))$, is a legal substitution instance of a rule in the domain theory, in this case $lawyer(?x) \rightarrow rich(?x)$. Thus the proof is valid. ■

2.2 Learning new rules from explanations

If some of the leaves of a proof tree are unmatched subgoals, then the proof is said to be *partial*. Partial proofs can still be valid. A valid partial proof tree is a demonstration that the root goal of the tree is implied by its leaf subgoals. The rule formed by taking the root goal as consequent and the leaf subgoals of the tree as antecedents can be added to the domain theory without changing what is entailed by the theory. All EBL algorithms extract a new rule from an explanation in this way, summarizing it by chunking its premises and its conclusion together.

The STRIPS [6], EBG [4], and EGGS algorithms, analyzed and compared by Mooney and Bennett [16], form the first generation of domain-independent EBL algorithms. They each transform an explanation in three stages:

(i) they prune away rules from the explanation, leaving unmatched subgoals;

(ii) they replace each remaining rule instance by the corresponding actual rule; and

(iii) they match up subgoals and consequents again.

The following example should make these three stages clear.

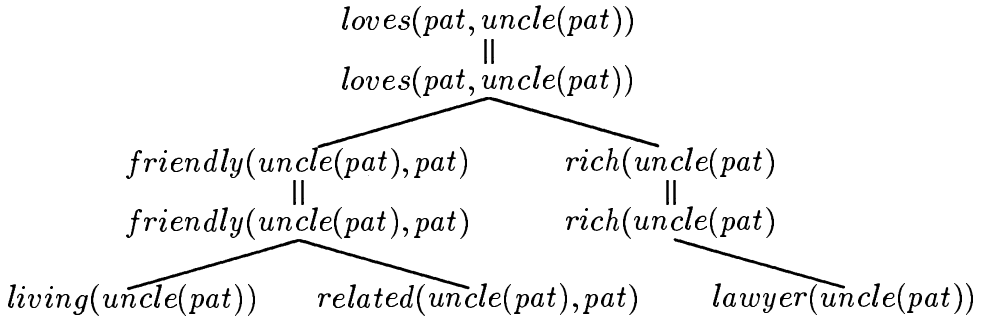
Example 2.2: Figure 2 shows the the result of applying each of the three stages of a traditional EBL algorithm to the proof tree of Figure 1. In stage (i) the rule instances $living(uncle(pat))$, $related(uncle(pat), pat)$, and $lawyer(uncle(pat))$ are pruned, leaving three unmatched subgoals. The rule that could be extracted after stage (i) would refer to Pat specifically; it would be valid but hardly worth learning. The proof tree after stage (ii) is shown next. This tree is not valid, because the constraints implicit in match edges are violated. The result of enforcing these constraints, stage (iii), is shown last. The rule that can be extracted from the final transformed explanation is

$$living(?v) \wedge related(?u, ?v) \wedge lawyer(?v) \rightarrow loves(?u, ?v).$$

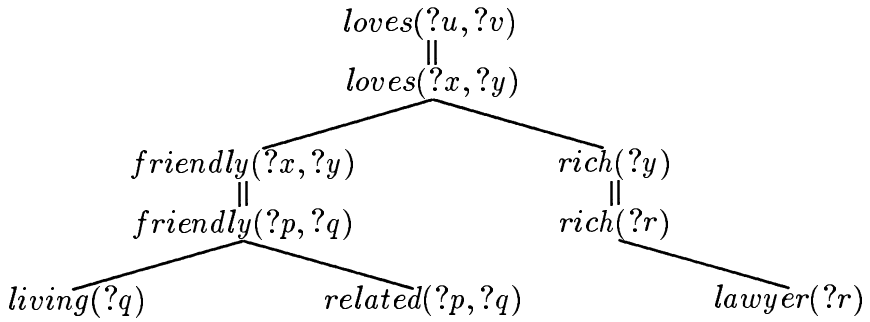
This rule is valid and worth learning. ■

Systems using the traditional EBL algorithms differ in how they determine what parts of an explanation to prune away. They also differ in the exact procedure they use for stage (iii), that is, to make generalized explanations valid

(i) After pruning—



(ii) After generalizing—



(iii) After enforcing constraints—

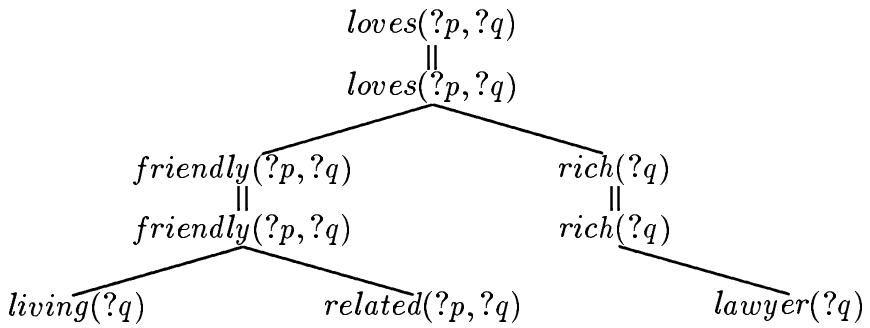


Figure 2: Applying EBL to the proof that Pat loves his uncle.

again. The EBG algorithm as originally described did not always perform stage (iii) correctly, and the EGGS algorithm in [16] also fails in some circumstances. In a longer version of this paper [5], we show how the problem of making a generalized explanation valid again reduces to the problem of computing most general unifying substitutions, even when explanation structures are not just proof trees, so it may be considered solved.

2.3 Learning useful rules by adjusting explanations

The point of transforming an explanation is to allow a rule to be extracted from it that is not only valid, but also worth learning, or useful. The quality of usefulness is traditionally called *operationality* [12]. A rule is operational if it can be applied in many different situations, and its subgoals can be proven efficiently when it is applicable.

The pruning stage of traditional EBL algorithms crucially affects the operationality of the rule that can be learned from an explanation. Consider two rules that might be extracted from an explanation, one before and one after pruning some rules from the explanation. The second rule is likely to be more often applicable: its subgoals can be proven in different ways later, whereas one particular way of proving those subgoals is compiled into the first rule. On the other hand, the first rule is likely to be more efficiently applicable: proofs of its subgoals are less deep than proofs of the subgoals of the second rule, unless the latter are proven in a different way.

The problem of ensuring that a rule acquired by EBL is operational is the problem of transforming an explanation so that its unmatched subgoals are both widely and efficiently provable. Suppose a subgoal is matched to a fact, that is, the consequent of a rule with no antecedents. Then it is reasonable to assume that another version of the subgoal could also be proven by retrieving a fact. This argument is an explicit operationality heuristic. It was in fact used tacitly to control pruning in the example above. EBL systems have used various methods to determine operationality. The most naive method is simply to flag some predicates as operational, once and for all. This approach is not always adequate [4], but it does have the merit of being explicit; more sophisticated operationality heuristics have typically been expressed procedurally, and never justified individually.

It may happen that an explanation to be pruned involves a rule whose antecedents are too specialized for a new rule involving them to be operational, but whose consequent is too high-level for it to be a suitable subgoal of a new rule. This will often be the case for a rule that is the result of earlier learning. The IMEX algorithm due to Braverman and Russell [1] can unravel a rule by

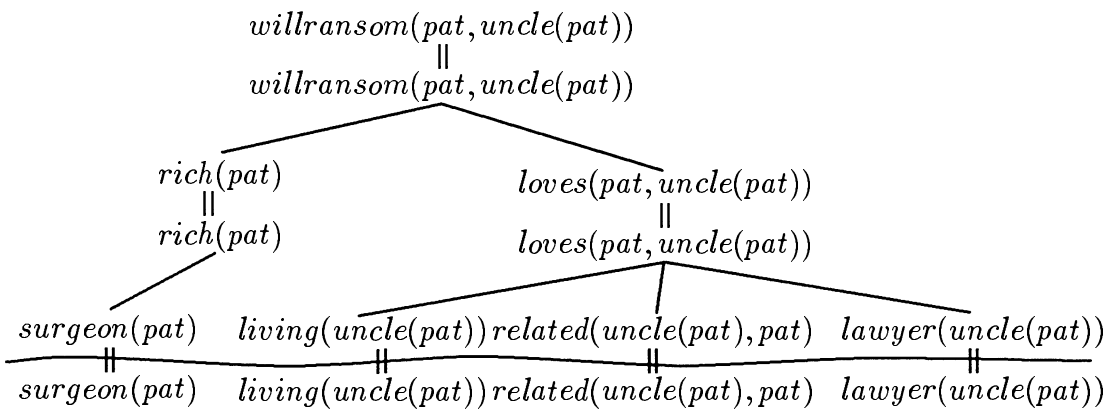


Figure 3: A proof that Pat will ransom his uncle.

replacing it by a proof tree whose root is the consequent of the rule and whose leaves are the antecedents of the rule. The following example should make clear how the IMEX algorithm works.

Example 2.3: Suppose the rule learned in the previous example is used by the artificially intelligent kidnapper to deduce that Pat is willing to ransom his uncle. Figure 3 shows such a deduction. A traditional EBL algorithm might prune the proof as indicated by the wavy line in Figure 3, and extract the rule

$$surgeon(?p) \wedge living(?q) \wedge related(?p, ?q) \wedge lawyer(?q) \rightarrow willransom(?p, ?q).$$

Suppose now that subgoals involving the predicate *rich* are easy to prove. One would like the rule

$$rich(?p) \wedge living(?q) \wedge related(?p, ?q) \wedge rich(?q) \rightarrow willransom(?p, ?q)$$

to be learned. A traditional EBL algorithm cannot learn this rule from the given explanation because the $rich(uncle(pat))$ subgoal is hidden by the rule learned previously. The IMEX algorithm replaces the previously learned rule by the partial proof tree giving rise to it, and subsequently prunes the expanded proof tree as indicated in Figure 4. Thus it enables the desired rule to be learned. ■

3 The EBL* algorithm

Like the EBL algorithms described in Section 2, the new algorithm presented here extracts a rule from an explanation after transforming it. The difference

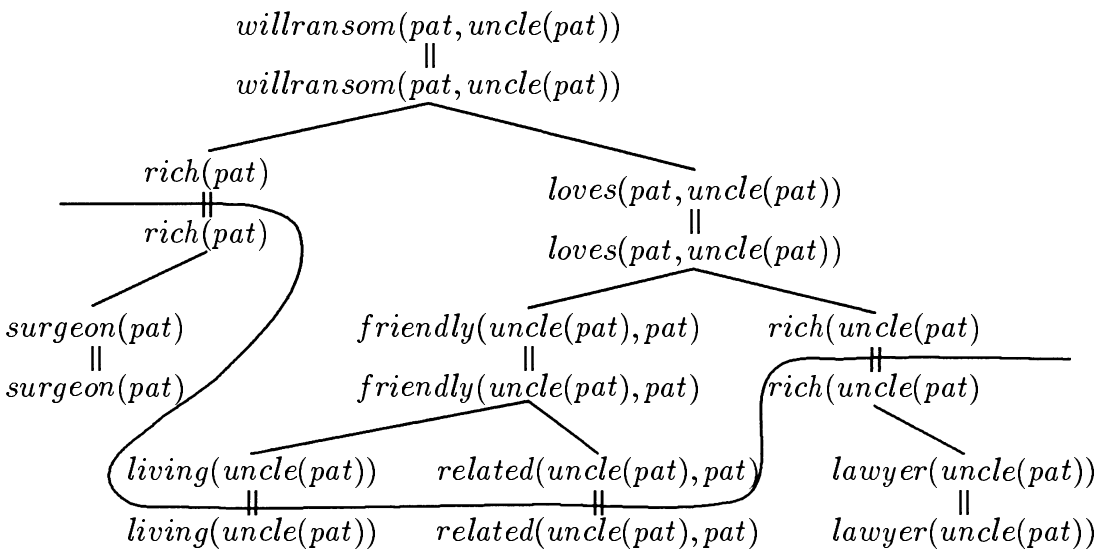


Figure 4: The proof of Figure 3, expanded by the IMEX algorithm.

is that the new algorithm does not operate in fixed stages. Instead it has a repertoire of basic proof transformation operations that is provably *complete*: any rule that can be extracted from a proof tree by transforming the tree in any validity-preserving manner can be extracted by a sequence of operations from the repertoire identified here. No other EBL algorithm is complete.

3.1 The repertoire of explanation transformations

The repertoire consists of operations of four types:

- replace a term by another, either specializing or generalizing the original term;
- delete the rule whose consequent is matched to a subgoal, leaving the subgoal unmatched;
- add a rule, matching its consequent to an unmatched subgoal;
- delete a subgoal and the subtree under it, preserving the argument bindings imposed by the subtree.

Traditional EBL algorithms use the first two types of operation listed above, while the IMEX algorithm also uses the third. The extra power of the EBL* algorithm, and its completeness, come from the fourth operation. The following

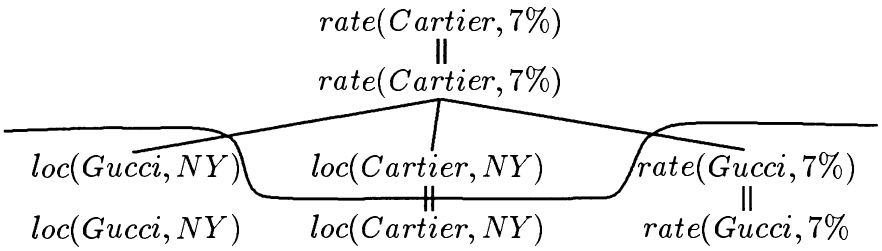


Figure 5: A proof that the sales tax rate is 7% at Cartier.

example shows how the fourth operation works, and at the same time demonstrates its usefulness.

Example 3.1: Suppose an artificially intelligent accountant knows that if two stores are located in the same state, then the sales tax percentage at the two is the same:

$$loc(?x, ?u) \wedge loc(?y, ?u) \wedge rate(?x, ?v) \rightarrow rate(?y, ?v).$$

Given the common location of Gucci and Cartier and the sales tax rate at Gucci, one can find the rate at Cartier, as the proof tree in Figure 5 shows. Neither a traditional EBL algorithm nor the IMEX algorithm can transform this proof into a version from which an interesting rule can be extracted. However the new rule

$$loc(x, NY) \rightarrow rate(x, 7\%)$$

can be learned after deleting two subgoals and their subproofs while preserving the bindings generated for the arguments of the subgoals by the deleted subproofs. This new rule is valid and operational. It says that the sales tax rate at any store in New York is 7%. ■

A rule of the form "stores in the same state pay sales tax at the same rate" is a *determination*: a higher-order regularity that by itself is useless in reasoning, but which together with some premises leads to useful conclusions. Determinations underly many instances of analogical reasoning, which are therefore deductively sound [3]. In the presence of determinations, the EBL* algorithm is capable of similarity-based learning: from one store the algorithm infers the sales tax rate at all similarly-located stores. The EBL* algorithm performs deductively sound similarity-based learning from a single prototype because a determination is a rule stating that one feature of the prototype is functionally dependent on another.

3.2 Completeness of the EBL* algorithm

A transformation operation is *validity-preserving* if it always transforms valid explanations into valid explanations. The second, third, and fourth EBL* operations are validity-preserving but all of their suboperations fail to be validity-preserving. They can thus be called *minimal* transformations. The first EBL* operation applied to an arbitrary term is not validity-preserving. In the longer version of this paper [5] we give a more precise definition of all the EBL* operations, including an operation that specializes or generalizes a term in a minimal validity-preserving way.

Essentially, the EBL* repertoire is complete because it consists of a set of minimal transformations that can be sequenced to reproduce the effect of any larger-scale transformation. For example, the effect of the IMEX operation of replacing a rule by the whole proof tree from which the rule was learned can be achieved by applying the second EBL* operation repeatedly. The EBL* operations can also be sequenced to reproduce the way the BAGGER system [20] transforms explanations in order to generalize number.

The following theorem states that the EBL* algorithm is as powerful in principle as any EBLalgorithm, where an EBL algorithm is taken to be any algorithm that transforms an explanation in a validity-preserving way, and then extracts a rule from the new explanation.

Theorem: (Completeness) If a rule can be validly extracted from an explanation by any EBL algorithm, then it can be extracted by the EBL* algorithm.
Proof: Given any valid explanation α and valid transformed version α', some sequence of operations from the EBL* repertoire produces α' from α. Explicitly, apply the second operation repeatedly to α, eventually leaving just one unmatched subgoal, the root of α. Then use the first operation to make this subgoal maximally general, and use the third operation to construct a proof with the same tree structure as α'. Finally, use the fourth operation to remove unwanted subgoals and use the first operation to specialize the new proof as necessary. ■

The proof just given is disappointing: all it says is that the EBL* operations can take a explanation into the empty explanation and then build up another arbitrary explanation from scratch. A similar phenomenon occurs in planning: if one is working in a world where there is a ground state attainable from every other state, and the operators for moving from state to state are reversible, then planning is in principle easy: one can go from any state to any other through the ground state.

The difficulty lies in deciding *when* to apply operations from the proposed repertoire to an explanation. The theoretical merit of the EBL* algorithm de-

pends only on the completeness of its repertoire, whereas its practical usefulness depends entirely on knowing when particular transformations of an explanation contribute to learning an operational rule from it. This control issue is discussed next.

3.3 Controlling the EBL* algorithm

We propose the use of explicitly stated search-control heuristics to guide the explanation transformation process. The purpose of these heuristics is to restrict exploration and encourage the extraction of operational rules. We have identified four general heuristics.

- A subgoal not involving any variable mentioned in the root goal of the proof tree should be deleted along with its subproof, preserving the bindings induced. Intuitively, such a subgoal provides background information that can be compiled into the rule to be learned.
- A subgoal that is provable for only one set of bindings should be deleted along with its subproof. There is no need ever to prove the subgoal again, so it can also be compiled into the rule to be learned.
- A rule with a unary consequent and a single unary subgoal, for example $surgeon(?x) \rightarrow rich(?x)$, should be deleted. Typically, a rule of this form expresses a taxonomic *isa* relationship, and chains of *isa* deductions should not be compiled.
- If the same subgoal appears repeatedly in an explanation, then the different appearances of the subgoal should be kept as one antecedent in the learned rule. Later proofs using the learned rule will then not prove the same subgoal more than once.

These four heuristics are similar in sophistication to those of procedurally controlled traditional EBL systems. They are sufficient to automate the examples of this paper, and others, but they have not been tuned yet for a large application.

As the proof of the theorem of Section 3.2 suggests, an EBL algorithm might explore the space of all alternative explanations before extracting a rule from a given explanation. Traditional EBL algorithms restrict this search by relying on preestablished transformation stages and fixed criteria for what is operational. The EBL* algorithm succeeds in transforming explanations quickly and generating operational rules, using the four heuristics above. The implementation traverses an explanation in depth-first fashion, testing the preconditions of each heuristic at each node, and applying the suggested transformation immediately if they are satisfied.

4 Conclusion

Considered from the perspective of this paper, EBL is quite weak: only rules logically entailed by current knowledge can be learned. Yet paradoxically some impressive EBL applications exist, systems that have surprising success at useful tasks. Paradoxically also, the EBL algorithms proposed to date have been incomplete and incapable of extracting new knowledge from explanations involving similarities.

Both these paradoxes can be understood by looking at the distinction between explanation-transformation operations and heuristics for guiding their application. Impressive EBL systems owe their power to the heuristics built into them. Because these heuristics have been conflated with transformations in previous EBL algorithms, their incompleteness has not been apparent.

The control heuristics of an EBL system must ensure that explanations are transformed quickly, and in such a way that operational new rules are learned. Operationality has many dimensions, of which only the two most universal are taken into account by the heuristics presented above. Indeed, because the framework adopted in this paper does not admit uncertainty [19], some important dimensions of operationality cannot be discussed in it. The complexity of the operationality issue typically makes the control heuristics of effective EBL systems domain-dependent.

Moreover, successful EBL systems assemble explanations in complex ways from special-purpose knowledge structures, and the validity of explanations constructed from these structures often depends on tacit constraints embodied in the part of a system that finds explanations. The explanation-finding and learning components of such a system are necessarily interdependent because they must cooperate to ensure that explanation transformations are valid. This is the reason why the most interesting and successful EBL systems (for example ARMS [18]) are tightly coupled.

Finally we note that focusing on the issue of control makes the elusive problem of imperfect theories relatively concrete and manageable. With an imperfect domain theory, not all explanations that may be constructed are actually valid. For example, explanations involving long chains of reasoning may be shakier than shallower ones. EBL is more difficult with an imperfect domain theory than with a perfect theory, but not intrinsically different. With an imperfect theory, control heuristics must restrict the class of acceptable transformed explanations to those in which one has confidence, as well as ensure operationality.

References

[1] Michael S. Braverman and Stuart J. Russell. IMEX: Overcoming intractability in explanation based learning. In *Proceedings of the National Conference on Artificial Intelligence*, pages 575–579, August 1988.

[2] Rodney A. Brooks, Jonathan H. Connell, and Peter Ning. Herbert: A second generation mobile robot. MIT AI Memo 1016, January 1988.

[3] Todd R. Davies and Stuart J. Russell. A logical approach to reasoning by analogy. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, 1987.

[4] Gerald F. DeJong and Raymond J. Mooney. Explanation-based learning: An alternative view. *Machine Learning*, 1(2):145–176, April 1986.

[5] Charles Elkan and Alberto Segre. The compleat EBLer. Forthcoming, 1989.

[6] Richard E. Fikes, Peter E. Hart, and Nils J. Nilsson. Learning and executing generalized robot plans. *Artificial Intelligence*, 3:251–288, 1972.

[7] Kenneth D. Forbus. Interpreting measurements of physical systems. In *Proceedings of the National Conference on Artificial Intelligence*, pages 113–117, 1986.

[8] Michael R. Genesereth and Nils J. Nilsson. *Logical Foundations of Artificial Intelligence.* Morgan Kaufmann Publishers, Inc., Los Altos, California, 1987.

[9] Patrick J. Hayes. The logic of frames. In Nils J. Nilsson and Bonnie L. Webber, editors, *Readings in Artificial Intelligence*, pages 451–458. Morgan Kaufmann Publishers, Inc., Los Altos, California, 1981. First published 1979.

[10] Haym Hirsh. Explanation-based generalization in a logic programming environment. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 221–227, 1987.

[11] Smadar T. Kedar-Cabelli and L. Thorne McCarty. Explanation-based generalization as resolution theorem proving. In *Proceedings of the Fourth International Machine Learning Workshop*, June 1987.

[12] Richard M. Keller. Defining operationality for explanation-based learning. In *Proceedings of the National Conference on Artificial Intelligence*, pages 482–487, 1987.

[13] Steven Minton. EBL and weakest preconditions. In *AAAI Workshop on Explanation-Based Learning*, pages 210–214, 1988.

[14] Tom M. Mitchell. Learning and problem solving. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 1139–1151, 1983.

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

[16] Raymond J. Mooney and Scott W. Bennett. A domain-independent explanation-based generalizer. In *Proceedings of the National Conference on Artificial Intelligence*, pages 551–555, 1986.

[17] Stuart Russell. The compleat guide to MRS. Technical Report KSL-85-12, Computer Science Department, Stanford University, 1985.

[18] Alberto M. Segre. *Machine Learning of Robot Assembly Plans.* Kluwer Academic Publishers, Norwell, Massachusetts, 1988.

[19] Alberto M. Segre. Operationality and real-world plans. In *AAAI Workshop on Explanation-Based Learning*, pages 158–163, 1988.

[20] Jude W. Shavlik. Generalizing the structure of explanations in explanation-based learning. Technical Report UILU-ENG-87-2276, University of Illinois at Urbana-Champaign, December 1987.