LT Revisited: Explanation-Based Learning and the Logic of Principia Mathematica

PAUL O'RORKE (ORORKE@ICS.UCI.EDU)

Department of Information & Computer Science
University of California, Irvine 92717

(Technical Report Number 88-29)
November 14, 1988

ABSTRACT

This paper describes an explanation-based learning (EBL) system based on a version of Newell, Shaw and Simon's LOGIC-THEORIST (LT). Results of applying this system to propositional calculus problems from *Principia Mathematica* are compared with results of applying several other versions of the same performance element to these problems. The primary goal of this study is to characterize and analyze differences between not learning, rote learning (LT's original learning method), and EBL. Another aim is to provide base-line characterizations of the performance of a simple problem solver in the context of the *Principia* problems, in the hope that these problems can be used as a benchmark for testing improved learning methods, just as problems like chess and the eight puzzle have been used as benchmarks in research on search methods.

TABLE OF CONTENTS

1. Introduction

There is widespread agreement that explanations can form the basis for powerful learning strategies. A new subfield of machine learning called *explanation-based learning* (EBL for short) has begun to exploit this idea. To date, however, there have been few experiments on EBL involving more than a few examples.[1] Experiments involving the application of complete EBL systems to large sets of problems from challenging domains are needed in order to demonstrate claims about the advantages of EBL and to discover the limitations of proposed EBL methods.

This paper reports on an EBL experiment in a logical domain. The experiment involves complete machine learning systems built around one of the earliest, simplest AI problem-solvers: Newell, Shaw, and Simon's LOGIC-THEORIST (LT). Results of a comparison of the performances of three versions of LT (corresponding to *non-learning*, *rote learning*, and *explanation-based learning*) on a large number of problems are given.

2. The Domain: Principia Mathematica

The domain of this experiment is the propositional calculus of Alfred North Whitehead and Bertrand Russell's *Principia Mathematica* [26]. The calculus deals with a set of expressions or well-formed propositional formulae built upon a set of variables that are supposed to stand for arbitrary propositions such as "Agrippina killed Claudius." Complex propositions are built up by using connectives such as NOT ($\bar{}$) , AND ($\wedge$), OR ($\vee$), and IMPLIES ($\supset$).

Valid propositions are called theorems. It turns out that the theorems of the propositional calculus can be built up from an initial set of theorems called axioms, such as $(P \vee P) \supset P$.[2] Other theorems are derived from the axioms by applying rules of inference. *Detachment*, or modus ponens, is a rule of inference that allows one to infer B if one has A and $A \supset B$. Another rule allows substitution of any expression for a variable in any theorem. Other rules allow replacement of definitions for defined connectives (e.g., $\bar{P} \vee Q$ for $P \supset Q$).

A derivation of a theorem is called a proof. A proof of a desired theorem can be written as a sequence ending in that theorem where each step in the sequence is either an axiom or else follows from previous steps by a rule of inference. Alternatively, the proof can be depicted as a tree whose root is the desired theorem where each node in the tree is either an axiom or follows from its predecessors by an inference rule.

The particular theorems of interest in this chapter are listed in Appendix 2. They are from the second and third chapters of part one of *Principia* [26].

---

[1] Minton's work on MORRIS [10] and on PRODIGY [11, 12] are noteworthy exceptions.

[2] Complete lists of the axioms and of the problems from *Principia* used in the experiments are provided in an appendix.

One of the advantages of propositional logic is an advantage of mathematics in general, namely *high levels of abstraction and generality.* When one learns arithmetic one is learning something that will apply to an extremely wide range of tasks from the mundane (e.g., choosing best buys at the grocery store) to the esoteric (e.g., performing basic calculations in the astrophysics of black holes). Similarly, propositional logic is an abstract, general theory of deduction. One can apply logic in infinitely many definite, specific domains by specifying propositions that provide details about particular subjects. Any learning that takes place at the logical level can thus be applied in many domains.

Another advantage of logic is that the number of schemata required for complete coverage is small. This feature facilitates the execution of large scale experiments because it isn't necessary to hand code large amounts of knowledge in new schemata in order to introduce new examples.

The particular logical system of *Principia Mathematica* has the added advantage that a very simple and natural theorem prover exists, which can be used as the problem-solving performance element in learning experiments.

3. The Performance Element: LT (The Logic Theorist)

According to Donald Loveland's *Automated Theorem-Proving: A Quarter-Century Review* [9], Newell, Shaw, and Simon's *The Logic Theory Machine* [17] was the first publication reporting results of a computer program that proved theorems. The program (hereafter referred to as LT) proved some theorems in the propositional calculus of *Principia Mathematica* [26]. This section is a brief description of how a simple version of LT works, in the terminology of schema-based problem solving.

3.1. Schemata

In this context, a *schema* is merely a collection of related descriptions. Each LT schema has three descriptions (well-formed formulae of propositional calculus) related by a logical dependency. The dependency states that one description, (the *consequence*

Figure 1. The Detachment and Chaining Schemata

of the schema) is true if both of the others (the *antecedents*) are true. LT only uses two schemata: a *detachment* schema and a *chaining* schema (see Figure 1). The detachment schema captures *modus ponens* and is comprised of X, Y, and X⊃Y where Y depends on X⊃Y and X. The chaining schema captures the *transitivity of implication*, and is comprised of X⊃Y, Y⊃Z, and X⊃Z where X⊃Z is true if both X⊃Y, and Y⊃Z are true.

3.2. Problem Solving

LT's mission is to construct a proof of a given conjecture: building it out of detachment and chaining schemata, axioms, and previously proven theorems. LT's proofs are always linear trees similar to the one shown in Figure 2. The leaves (labelled $T_0 \ldots T_{n+1}$) are all taken from the set of known theorems. Each schema (labelled S_i) corresponds to a step in the proof and can be considered to reduce a problem (P_i) to a simpler one (P_{i+1}) as in Figure 3.

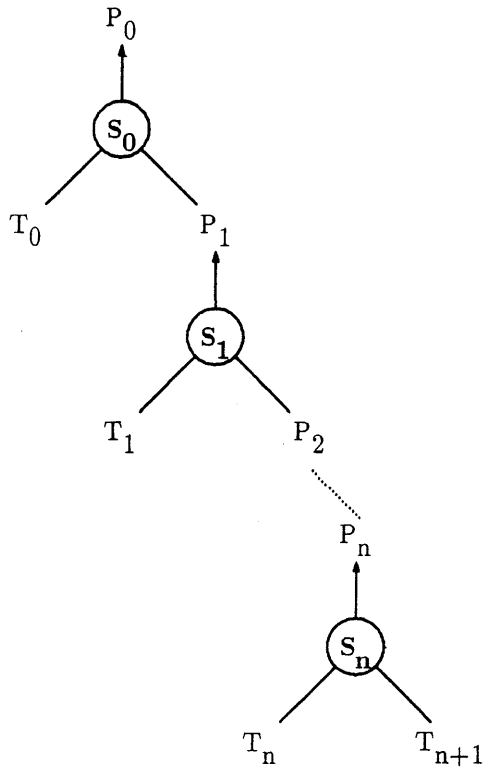

Figure 2. The Structure of LT's Proofs

New Problem P_{i+1} is $B_i\,\theta$,
where θ is the MGU of C_i with P_i and of A_i with T_i

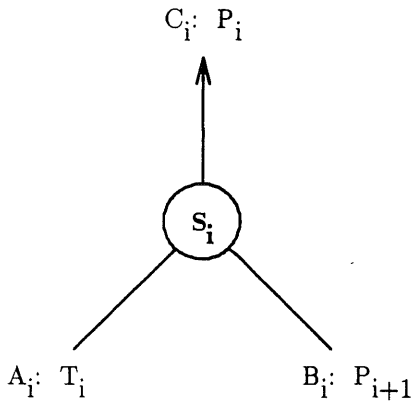

Figure 3. One Step in the Construction of a Proof

Figure 3 shows what happens, in general, during one step in the construction of a proof. Given a problem P_i, LT chooses a schema S_i and a known theorem T_i, such that the problem is an instance of the schema's conclusion C_i and the theorem T_i is compatible with one of the schema's antecedents A_i. The other antecedent B_i (viewed in the context of identifications between C_i and P_i, and between A_i and T_i) becomes the new problem P_{i+1}.

In other words, the problem P is matched against the conclusion C_i of a schema. If the match succeeds, the resulting substitution is applied to the conclusion and to both antecedents. The result of applying the substitution to one of the antecedents A_i is then matched against known theorems until a match succeeds. The resulting substitution is applied to the other antecedent, and it becomes a new subproblem.

It should be noted that the matcher used in this study is "smart" in the sense that it can decode defined symbols such as IMPLIES; but the matcher is "dumb" in the sense that it does not recognize other logical relationships (such as the fact that NOT NOT P is equivalent to P and so double negations could be stripped away). The logic itself deals with most logical relationships, not the matcher.

LT only extends a proof by means of a detachment schema in one way: given a problem, LT looks for a known implication whose conclusion subsumes the problem. If such a theorem is found, its antecedent becomes the new problem (see Figure 4). This amounts to applying a detachment *operator* corresponding to modus ponens, run backward.

New Problem P_{i+1} is X θ
where θ is the MGU of Y with P_i and of X⊃Y with T_i

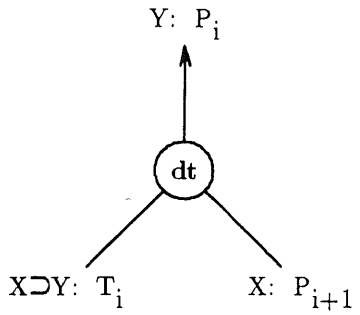

Figure 4. A Detachment Step

New Problem P_{i+1} is Y⊃Z θ,
where θ is the MGU of X⊃Y with T_i and of X⊃Z with P_i

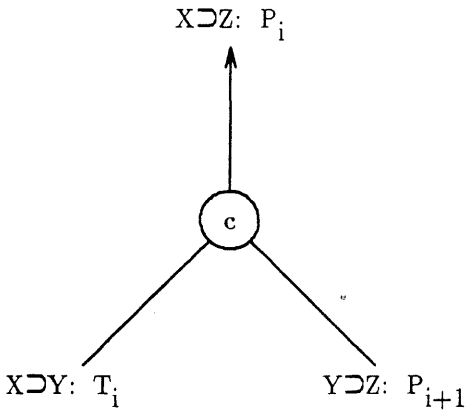

Figure 5. A Chaining Forward Step

The chaining schema can be used in two ways to extend the proof, so in effect LT has two chaining operators corresponding to transitivity of implication inference rules. Both operators attempt to prove an implication of the form X⊃Z. *Chaining forward*, (Figure 5), involves trying to show that an immediate consequence of X implies Z. In contrast, *chaining backward* (Figure 6) tries to show that X implies an immediate

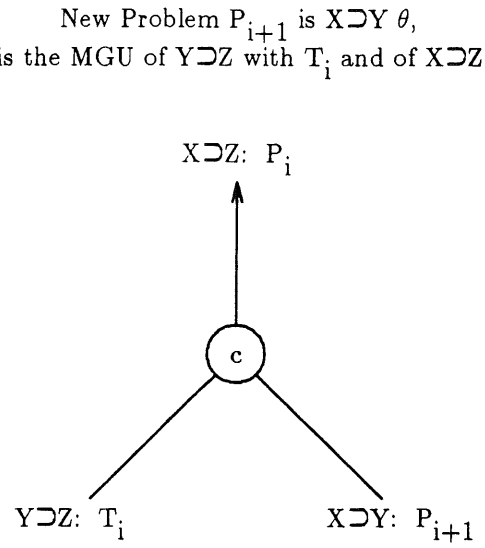

Figure 6. A Chaining Backward Step

antecedent of Z.

For a concrete example, consider the operation of LT on Principia-2.17: $(\overline{Q}\supset\overline{P})\supset(P\supset Q)$. This is the *only if* part of an important tautology called *contraposition* [4], which states that the contrapositive $\overline{Q}\supset\overline{P}$ holds if and only if the implication $P\supset Q$ holds.

LT proves Principia-2.17 by chaining forward (see Figure 7): it proves that the contrapositive $\overline{Q}\supset\overline{P}$ implies an intermediate result that eventually leads to the original implication $P\supset Q$. The first link of the chain is supplied by an instance of the axiom Principia-1.4: $(A\backslash/B)\supset(B\backslash/A)$. With A bound to $\overline{Q}$, and B bound to $\overline{P}$ this yields $(\overline{Q}\backslash/P)\supset(P\backslash/\overline{Q})$. By the definition of the implication connective, $P\backslash/\overline{Q}$ is the same as $P\supset\overline{Q}$; this serves as the intermediate step in the chain from $\overline{Q}\supset\overline{P}$ to $P\supset Q$. Chaining forward transforms the initial problem into the problem of proving $(P\supset\overline{Q})\supset(P\supset Q)$.

Next, the detachment operator is used. The detachment operation amounts to performing modus ponens backwards, so a theorem is needed whose conclusion subsumes the current subproblem. The conclusion of axiom Principia-1.6: $(A\supset B)\supset((C\backslash/A)\supset(C\backslash/B))$ meets this requirement. With C bound to $\overline{P}$, A bound to $\overline{Q}$, and B bound to Q, the axiom becomes $(\overline{Q}\supset Q)\supset((P\supset\overline{Q})\supset(P\supset Q))$. Detachment on Principia-1.6 transforms the problem of proving $(P\supset\overline{Q})\supset(P\supset Q)$ into the problem of proving $\overline{Q}\supset Q$. Assuming Principia-2.14 is known, $\overline{Q}\supset Q$ is the final subproblem because it is an instance of Principia-2.14.

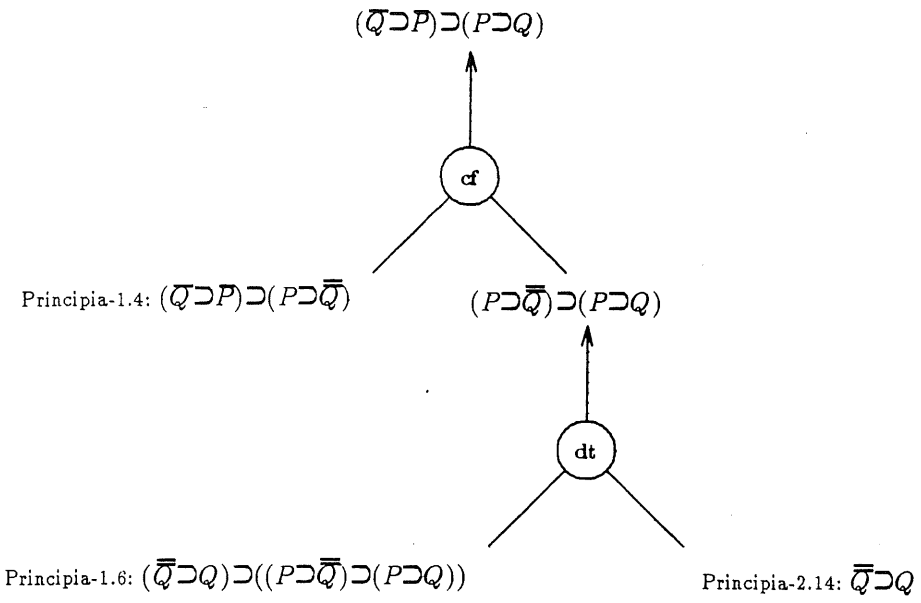

Figure 7. The Proof of Principia-2.17

The example just discussed may leave the impression that no search is involved in finding proofs. Of course this is not correct. Theorem provers are notorious for searches involving high branching factors and LT is no exception. LT performs breadth first search: when it uses its operators and known theorems to expand the unsolved subproblem in a partial proof it generates a number of new partial proofs each with one new subproblem. These are checked to see whether they are instances of known solutions and if not, they are added to the end of a queue of incomplete proofs to be worked on later.

4. LT Plus Rote Learning

In *Human Problem Solving*, Newell and Simon describe experiments on LT augmented with simple forms of learning. They mention that perhaps the simplest learning method is to make LT's list of known theorems variable, modifying LT so as to add any new theorem it proves onto the list, so that "... in proving a sequence of theorems, LT would gradually become more capable" [16]. This form of learning is well suited for relatively uncontrolled situations such as *learning by discovery*, where conjectures are made that may turn out to be false, and problems are posed that may not be solvable. In more controlled learning situations such as *learning from a teacher*, sequences of problems are often carefully selected so that all of the problems are solvable. In fact, they are usually sequenced in order of increasing difficulty. The problems from *Principia* form just such a carefully constructed sequence. LT was

allowed (in both the original and the present studies) to use all prior theorems in its attempts on each new theorem, whether it had succeeded in proving them or not.

When solved problems are stored, each new problem is reduced to one that has been solved before. When all problems are stored, each new problem is reduced to one that has been seen before. In both cases, problems encountered are simply added to memory and one is left with the distinct impression that no "understanding" or "thinking" is taking place. For this reason, one can consider these strategies to be forms of *rote learning*.[3]

5. LT Plus Explanation-Based Learning

One problem with rote learning systems is that they tend to be very sensitive to the particular form of the examples they observe. Extraneous details of specific examples tend to be retained as well as essential facts. This can cause failure to recognize that a solution to one problem can also be used for another problem because the problems differ in trivial ways. One would prefer to forget about the extraneous details and to remember only the essentials of an example. The basic idea of *explanation-based learning* is that one can do this by constructing and using explanations. When one conducts an analysis and constructs an explanation of how and why a solution solves one particular problem, one is better prepared to see the general class of problems that the method can be successfully applied to.

LT can be augmented with EBL by focusing on explanations instead of focusing on problems. The explanations are LT's proofs, considered as structures built out of schemata. The EBL version of LT ignores the specific theorem that gave rise to a proof. Each proof is considered in its full generality in order to compute the most general theorem that can be concluded on the basis of the proof. The generalized conclusion of an LT proof can be defined recursively as follows. The generalized conclusion of an elementary proof (a proof that states that the desired conclusion is an instance of a known theorem) is simply the known theorem itself. The generalized conclusion of a complex proof is the result of viewing the conclusion of the topmost schema in the context of identifications (unifications) between one antecedent and a known theorem, and between the other antecedent and the generalized conclusion of the subproof (see Figure 8).

In particular, the generalized conclusion of a proof by detachment is the result of viewing the conclusion Y of the detachment schema in the context of the unification of one of its antecedents X⊃Y) with some known theorem (T_i) and of the other antecedent X with the generalized conclusion (G_{i-1}) of the subproof. (See Figure 9.) In forward chaining, X⊃Y is unified with a known theorem while Y⊃Z is unified with the generalized subconclusion to obtain a substitution that is applied to X⊃Z. In backward chaining, the roles of X⊃Y and Y⊃Z are reversed.

---

[3] The learning versions of LT add new theorems to the *end* of the list of known theorems. This will be seen to have important consequences in both rote and explanation-based learning.

Generalized Conclusion G_i is $C_i\,\theta$,
where θ is the MGU of A_i with T_i and of B_i with G_{i-1}

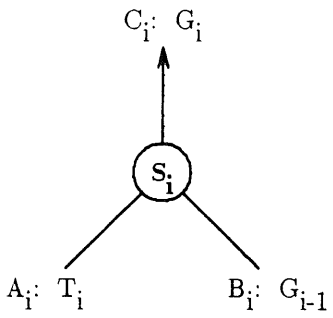

Figure 8. The Generalized Conclusion of One Step of a Proof

Generalized Conclusion G_i is $Y\,\theta$,
where θ is the MGU of $X \supset Y$ with T_i and of X with G_{i-1}

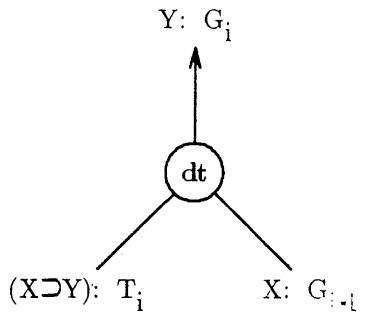

Figure 9. The Generalized Conclusion of a Detachment Proof

For a concrete example, reconsider the proof of Principia-2.17, $(\overline{Q} \supset P) \supset (P \supset Q)$. Recall that the proof involved two steps: chaining forward on Principia-1.4 reduced the original problem to a subproblem that was solved by detachment on Principia-1.6 and

Principia-2.14. Computing the generalized conclusion also involves two steps: first the generalized conclusion of the detachment subproof must be computed, then the generalized conclusion of the overall chaining forward proof can be determined.

The generalized conclusion of the detachment step is computed by identifying Principia-2.14 $\overline{D}\supset D$ with the antecedent $A\supset B$ of Principia-1.6, $(A\supset B)\supset((C\backslash A)\supset(C\backslash B))$. This identification binds A to $\overline{D}$, and B to D. The conclusion of Principia-1.6 *in this context* becomes the generalized conclusion of the detachment step, namely $(C\backslash\overline{D})\supset(C\backslash D)$. The generalized conclusion of the entire proof is computed by chaining forward on Principia-1.4 $(E\backslash F)\supset(F\backslash E)$ and the generalized conclusion of the subproof $(C\backslash\overline{D})\supset(C\backslash D)$.

The conclusion of Principia-1.4 plays the role of the middleman, and is identified with the antecedent of the generalized conclusion of the detachment. That is, $F\backslash E$ is unified with $C\backslash\overline{D}$: F is bound to C while E is bound to $\overline{D}$. The generalized conclusion of the chaining forward step kicks out the middleman and goes directly from $E\backslash F$, the antecedent of Principia-1.4, to $C\backslash D$, the conclusion of the generalized conclusion of the detachment step, *in light of this identification*. Substituting C for F and $\overline{D}$ for E in $(F\backslash E)\supset(C\backslash D)$, the generalized overall conclusion is $(\overline{D}\backslash C)\supset(C\backslash D)$. By the definition of implication, this can also be seen as $(\overline{D}\supset C)\supset(C\backslash D)$ as shown in Figure 10.

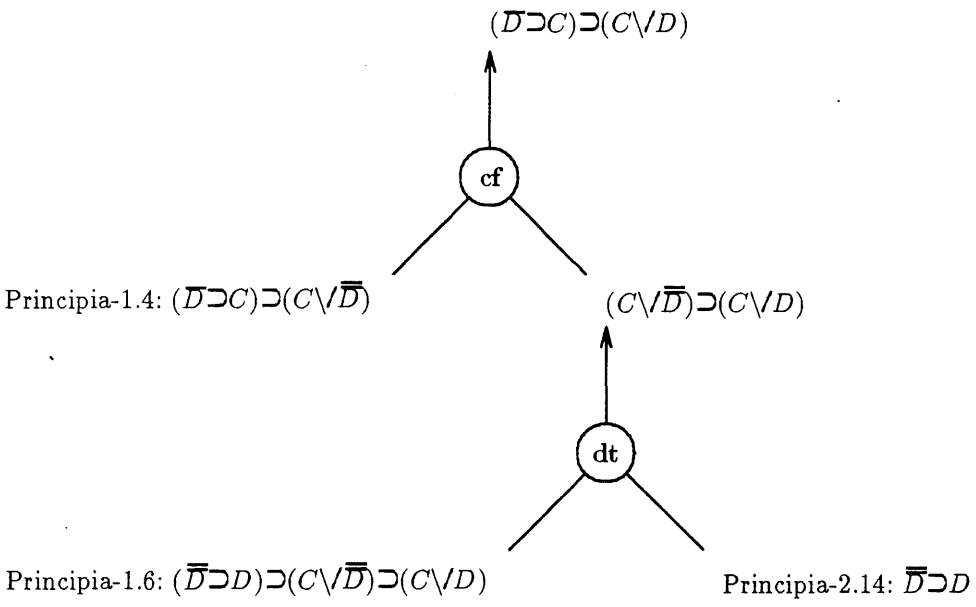

Figure 10. The Generalized Conclusion of the Proof of Principia-2.17

Comparing the generalized conclusion of the proof of Principia-2.17 to the particular problem:

Principia-2.17: $(\overline{Q} \supset P) \supset (P \supset Q)$
Generalized Conclusion: $(\overline{D} \supset C) \supset (C \backslash / D)$

one can see that the proof is strictly more general than the problem because the problem contains an extraneous negation. The NOT in P is not really necessary.

6. Methodology

The experiments reported here involve the application of three different versions of LT to propositional logic problems from *Principia Mathematica*. All three versions of LT start with the same axioms (the same initial set of known theorems). However, the first (*non-learning*) version of LT is not allowed to learn from its successes or failures. No new theorems are added to the list of known theorems. The non-learning LT attempts to solve each new problem by reducing it to one of the original axioms. The second version is allowed a form of *rote-learning*: problems are added to the list of known theorems whether they are solved or not. The rote-learning LT attempts to solve new problems by reducing them to problems it has seen before. The third version augments the basic LT by the simple form of explanation-based learning described earlier. Of course EBL is useless when the search for a proof fails, so rote-learning is resorted to in this case. However, when a novel theorem is proved, the generalized conclusion of the proof is added to the list of known theorems.[4]

The main questions asked on each *Principia* problem are:

- What is the nature of the search for a solution?
- What is the quality of the solution found?
- What is the quality of the learning?

6.1. Characterizing the Search for a Solution

For each problem in *Principia*, a record is made concerning whether each version of LT solves or fails to solve the problem in a limited search. LT does breadth first search maintaining a queue of untried problems. Before a problem is added to the queue, it is checked to see whether it is an instance of a known theorem. Problems are dequeued and attempted by applying LT's operators: first the detachment operator is applied, followed by chaining forward and then chaining backward. Each of these operators may produce new subproblems which may be added to the queue of untried problems. The search is restricted by limiting the number of problems attempted.

[4] It is important to note that the same basic problem solving machinery (LT) is being used in each version. LT is admittedly primitive by modern standards of theorem proving but our main interest is in differences in performance between no-learning, rote-learning and EBL systems, not in the particular performance element. All the learning and non-learning systems are handicapped by LT's lack of sophistication.

Whether the problem is solved or not, the total number of subproblems generated and the number of problems attempted in each search for a solution is recorded and used to compute the average branching factor (the number of sub-problems generated divided by the number of problems attempted) for each problem.

The numbers of subproblems generated and attempted can be thought of as "absolute" measures of the amount of search done by LT in solving a particular problem. The average branching factor can be thought of as a "relative" measure of the amount of search. Since it is a ratio, one must take care to realize that one search method can easily generate and attempt many more subproblems than another search method applied to the same problem, while still getting the same average branch. In reporting on results of experiments in this paper, we take care that when average branch is used as a measure of "amount of search", the absolute measures reflect the same relationships between the search methods being compared as does their ratio.

6.2. Evaluating the Results of Problem Solving and Learning

If a proof of a theorem is found, a measure of the size of the proof is recorded. In order to facilitate comparison of rote versus explanation-based learning, the theorems learned by the EBL version of LT are recorded. These degenerate to the input theorem in cases when no proof is found. When a non-trivial proof is found, the generalized conclusion is computed and the generality of the result is compared to the generality of the corresponding result of rote-learning.

7. Limited Search[5]

The main goal of all our experiments is to find and explain differences in problem solving performance between non-learning, rote-learning, and EBL versions of LT. It was known from the work of Newell, Shaw, and Simon that rote learning improves LT's performance dramatically over not learning. Before these experiments were conducted, one researcher (a well-known partisan of empirical approaches to learning) conjectured that generalization and performance would not improve in going from rote to explanation-based learning versions of LT on the *Principia* problems because these problems are so general to begin with. EBL advocates expected some improvement in performance, the only question for them was: *how much* would performance improve?

Our initial experiments involve severely limited search. To be exact, the number of subproblems LT is allowed to attempt in its effort to solve each Principia problem is limited to 15. Limited search experiments are interesting because people do not seem to search much; when they fail to find a solution fairly quickly in problem solving, they tend to give up and go on. In addition, advantages of one method over another tend to be magnified in limited search. For example, if one method requires more search than another to find a solution to a particular problem, it may not find a solution at all when the search is limited.

---

[5] As will be seen, this section describes the results of limited search experiments under some restrictions to be explained later (the IMPLIES restrictions).

7.1. Results on Search Performance

Figure 11 shows the search behavior of three versions of LT on the problems they fail to solve. Each point on the graph corresponds to an unsolved problem. The figure gives the "average branching factor", the number of problems generated in the search divided by the number of problems attempted. In the case of unsolved problems, the number of problems attempted is a constant, since this parameter is limited to 16 (15 subproblems and the original problem) and search has to be abandoned at that point when LT fails to find solutions. Thus, the average branching factor is proportional to the number of problems generated in this case.

The numbers of points on the curves indicate that LT solves far more problems with learning than without and that the learning versions are roughly comparable in

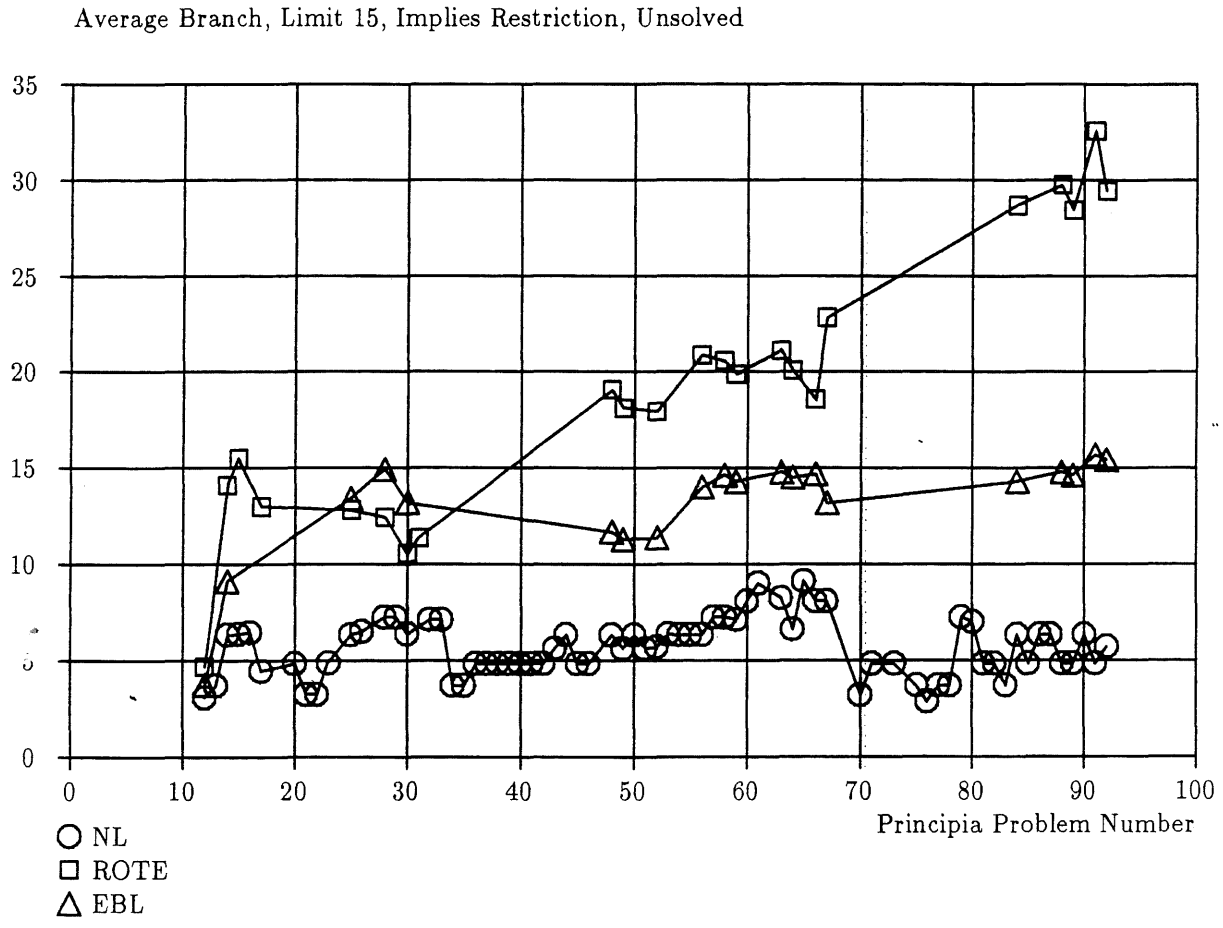

Figure 11. Limited Search Performance on Unsolved Problems

this respect. The fact that the RL curve is generally above the EBL curve and both are always above the NL curve indicates that, in most cases, RL generates more subproblems than EBL, which generates more subproblems than NL.

Turning to solved problems, consider the search behavior of no-learning as opposed to rote. The NL system solves 22 problems (22/92 = 24%), including one problem not solved by RL (problem 31, Principia 2.41). RL solves 69 problems (75%), including 48 problems not solved by NL. When both systems solve a problem, RL never attempts more subproblems than NL, and RL almost always generates fewer (or the same number of) subgoals before a solution is found.

Figure 12 shows the search behavior of rote versus EBL on the solved problems. There are no isolated circles but three isolated boxes indicate that EBL solves everything that rote solves and more (72 problems in all, or 78%). (EBL picks up

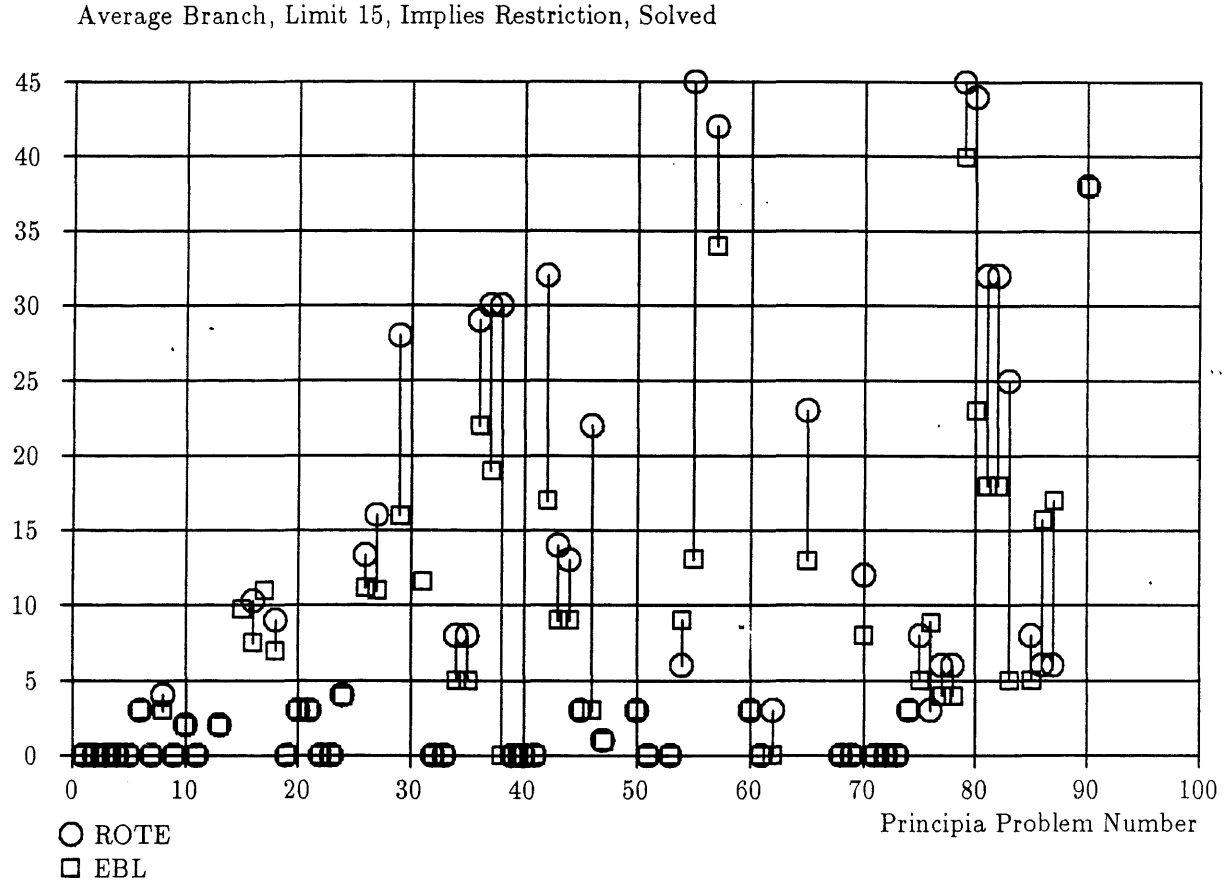

Figure 12. Rote vs EBL on Solved Problems in Limited Search

problem 31 and thus also solves everything NL solves as well.) In general, EBL does less search than rote, as measured in terms of problems attempted, subproblems generated, and in terms of average branching factors. However, there are a significant number of exceptions to this rule, e.g., problems 86 and 87.

7.2. Results on the Quality of the Solutions

Figure 13 shows the quality of the solutions delivered under NL vs RL, while Figure 14 compares the quality of the RL solutions vs those provided by EBL. Rote provides solutions that are always at least as good as NL, often better. However, the differences in quality between RL and EBL are mixed. They often get the same proof, but sometimes RL finds a shorter one, sometimes vice versa.

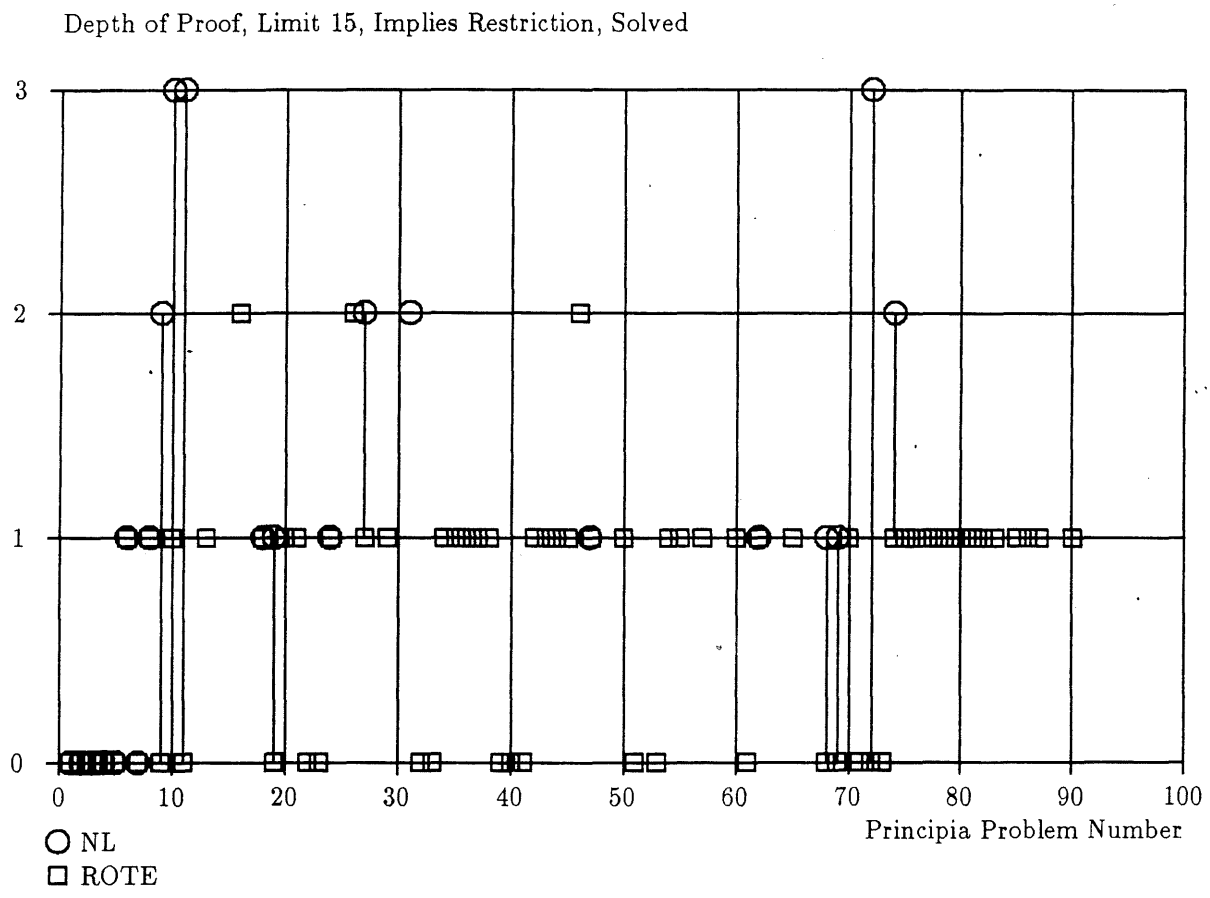

Figure 13. Quality of Solutions of NL vs Rote in Limited Search

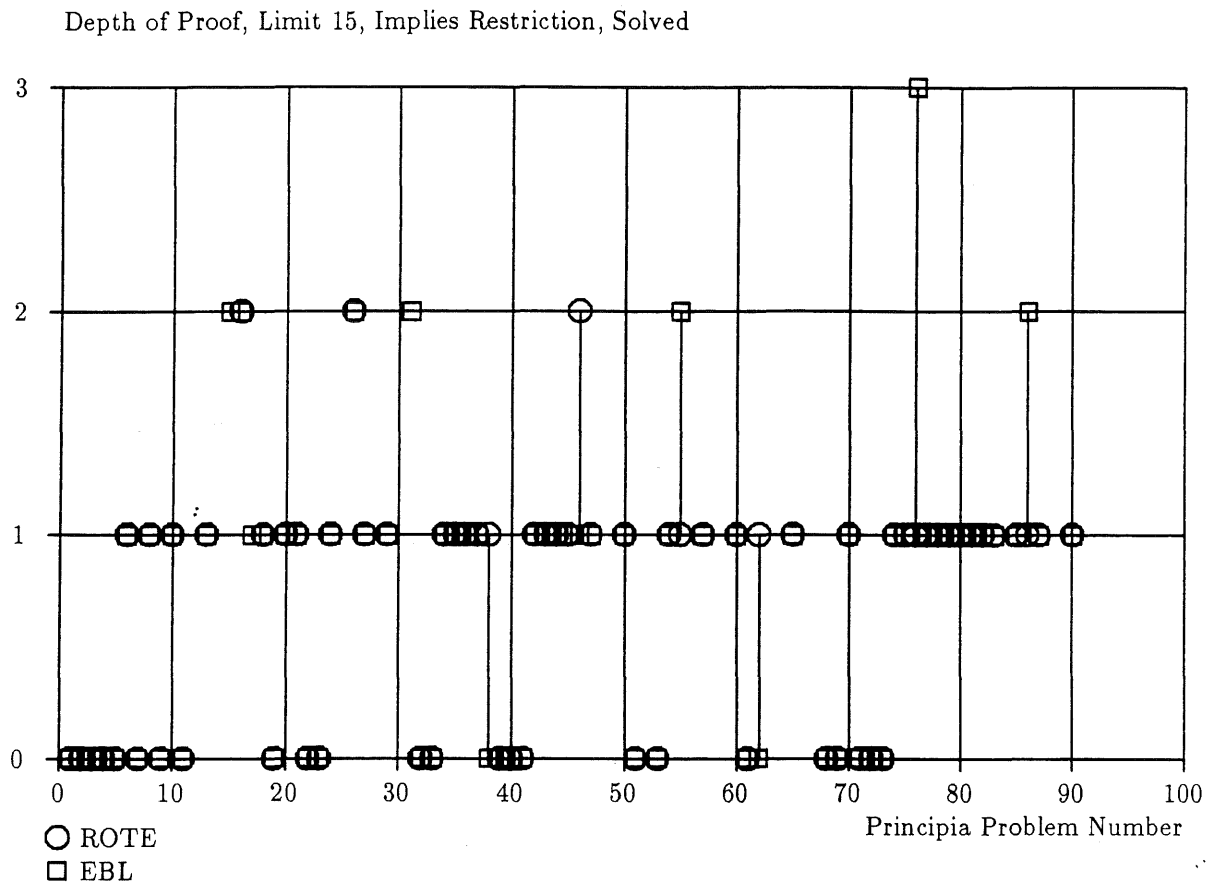

Figure 14. Quality of Solutions of Rote vs EBL in Limited Search

7.3. Discussion of Search Performance

In general, problems solved by the non-learning LT are also solved by the rote learning version and problems solved by the rote learning LT are also solved by the EBL version. Some exceptions to this rule are explained by new theorems leading the learning systems astray, so that they use up their limited search resources before finding a solution.

Looking at the number of subproblems generated and attempted by each version of LT, one sees increases in these "amounts of search" in going from non-learning to learning in some cases and drops in other cases. The drops indicate that some learned theorem is useful in solving a problem efficiently. The increases are due to the fact that learned theorems increase the number of possible next steps in proofs. The learning versions of LT generally have more ways of attempting to solve a problem. When one of the early attempts succeeds, the problem is solved immediately and less

search is done. Otherwise, more attempts are made before a solution is found or search is abandoned.

Relationships between the average branching factors are especially clear in the unsolved problems: they are relatively low for the non-learning system, much higher in both learning systems, but significantly lower in EBL than in rote learning. The branching factors appear to be relatively constant in NL, but increase with learning, more quickly in RL than in EBL.

It may seem odd that the no-learning performance is poor (it solves far fewer problems) as compared to the learning systems when it seems to have a less difficult search space (lower branching factor) to contend with. There are several reasons for this. First, learned theorems enable the learning versions of LT to "see more deeply into the search space." Search is limited and there are problems that cannot be solved without learning just because the required search exceeds the limit. The learning versions of LT may be able to effectively exceed the limit because search done in constructing the proof of learned solutions is not counted against searches that apply these earlier solutions to solve later problems.

Another reason for the drastic improvement in performance in learning as compared to non-learning has to do with LT's limited control strategy. LT is restricted to producing linear proofs: each operator (detachment and chaining) uses a known theorem in order to reduce a problem to a new subproblem. However, the learning systems add to the initial axioms theorems that follow from the initial axioms by one or more operations. This has the effect of allowing the learning versions of LT to break out of this restriction so that the search for a solution is taking place in a radically different search space, one which contains solutions that cannot be generated by the non-learning version of LT.

Figure 15 shows an example of a proof that is within the search space of the learning systems but denied to the NL LT. While NL does manage to construct an equivalent proof, using chaining forward as a mirror image of chaining backward, this is done at the cost of extra search.

An additional source of the improvement in performance in learning is that learned solutions can increase the set of problems that can be solved. It is known that LT is an incomplete theorem prover, in other words there are theorems that it cannot prove in principle (even ignoring any limitations of the amount of search allowed). For example, *Principia* problem 2.13 cannot be solved by LT. Adding such problems to the list of known results covers for incompleteness in the theorem prover and leads to solutions of problems that otherwise could not be solved.

Focusing on search performance differences between the learning systems, we note that sometimes problems are solved by the EBL version alone, (for example, in this experiment, 2.16 and 2.18). Also, it is often the case that EBL finds proofs with less search than RL, measuring the amount of search in terms of problems attempted, subproblems generated, and in terms of average branch. One possible reason for improvement in performance in EBL as opposed to RL is the improved generality of the results of explanation-based learning. Another reason for this improvement is that

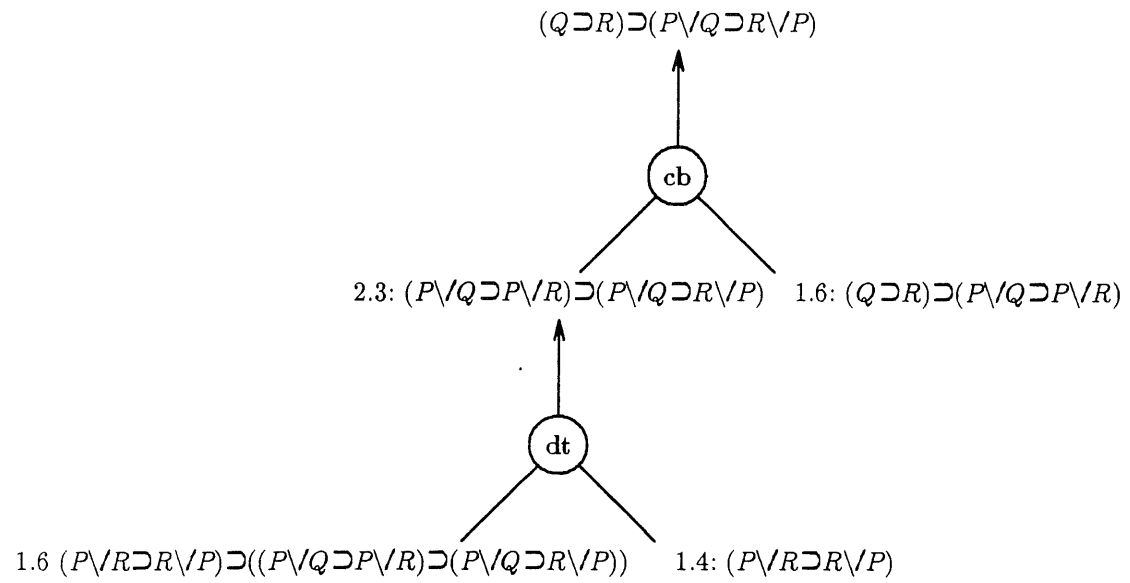

Figure 15. Rote-Learning and EBL Proof of Principia-2.36

the rote learning LT adds instances of known theorems to the list of known theorems. This violates a general principle of explanation-based learning that might be paraphrased: *only novel solutions to problems are worth remembering* [2]. Even without invoking EBL, however, there is no point in adding instances of known theorems to the list of known theorems because any instance of an instance X of Y is also an instance of Y. Or to put it another way, any "indirect" instance is also a "direct" instance. Adding instances hurts by increasing the branching factor of the search but provides no benefits, since the instances are added to the end of the list of known theorems, rather than the beginning.

Sometimes the search performance of EBL is inferior to RL. For example, if this experiment is run with a limit of 200 rather than 15, rote solves 2.37 but EBL misses it. Occasionally, one might expect a more general result learned by EBL to get in the way of finding the correct proof, but it turns out that this anomalous behaviour is actually due to the fact that LT is too restrictive in its use of problem solving operators. It requires subproblems to have IMPLIES as their top level connective before it will attempt to reduce them using forward or backward chaining. In addition, the known theorem used must also have IMPLIES as its top level connective. ([16], [25] page 24).[6] EBL often learns theorems that are more general than their RL counterparts; some of

these are not explicit implications. The conclusion to be drawn here is that it is necessary to have a reasonably "smart" matcher in order to take full advantage of the improved generality of explanation-based learning, otherwise one may wind up with degradation in performance rather than improvement! In fact, the overall number of problems solved can be lower in EBL with a "dumb matcher" than in rote learning because of this effect.

7.4. Discussion of the Quality of the Solutions

Turning to the quality of the solutions found by LT, the proofs discovered by the rote-learning LT are always at least as short as the proofs discovered by the non-learning LT. This is a consequence of the fact that the rote-learning LT needs only to reduce a problem to a previously seen problem, whereas the non-learning LT has to reduce it to one of the original axioms.

The rote and EBL proofs are of comparable quality, the same proofs being found in most cases. Sometimes (viz 2.49, 2.56, and 2.8) EBL produces shorter proofs. However, in other cases (namely 2.68, 3.24, and 3.41) the rote proofs are shorter.[7]

8. Limited Search Without the IMPLIES Restrictions

In this section, we examine some effects of lifting the IMPLIES restrictions. Lifting the IMPLIES restrictions has effects on LT regardless of whether or how it is learning. The question is how much of an effect? How will the relationships between non-learning and the competing learning strategies change?

8.1. Results on Search and Quality of Solutions

Figure 16 shows the search behavior of three versions of LT on unsolved problems. Lifting the IMPLIES restrictions has substantial impact. The branching factors are much higher, especially in the learning systems. EBL's average branch now seems to increase with learning at nearly the same rate as RL, whereas under the IMPLIES restrictions its branching factor curve seemed relatively flat. In addition, the relationships between the various systems are simpler: the RL curve is now always above the EBL curve and it in turn is always above the NL curve.

Turning to solved problems, Figure 17 shows the search behavior of rote versus EBL. There are no isolated circles but three isolated boxes indicate that EBL solves everything that rote solves and more. Furthermore, EBL does less search than rote on every problem now, as measured by problems generated, subproblems attempted, and by average branching factors. The exceptions observed in the initial experiments no longer occur once the IMPLIES restrictions are lifted.

---

[6] The resulting subproblem is, by definition, an explicit implication, but it may ground in (match with) a known theorem that is not an implication.

[7] We shall see in the next section that this is due to the IMPLIES restrictions. With a "dumb" matcher, EBL can actually hurt performance as much or more than it improves it, not just in terms of search costs, but in terms of the quality of the solutions found as well.

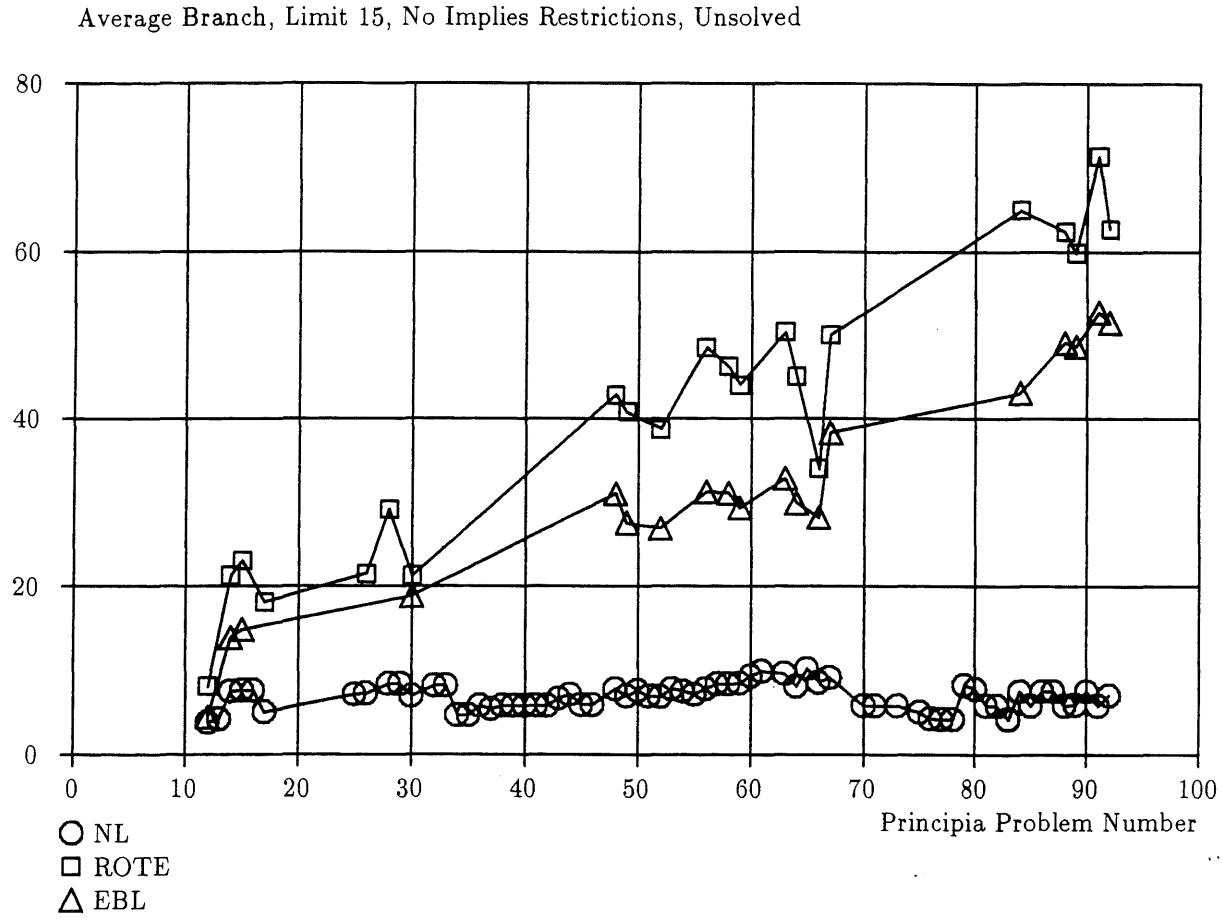

Figure 16. Search Behavior Without the IMPLIES Restrictions

Figure 18 shows the quality of the solutions provided by RL vs EBL. Note that RL no longer gets superior solutions, once the IMPLIES restrictions are lifted.

8.2. Discussion on the Effects of Lifting the IMPLIES Restrictions

Lifting the IMPLIES restrictions tends to increase branching factors, as reflected in the numbers of problems generated in limited attempts on the unsolved problems. However, this is offset by the fact that the new sub-problems generated sometimes lead to early proofs. This sometimes makes the difference, in limited search, between solving or not solving a problem. In other cases it means that a shorter proof is found.

Since problems must be reduced to the initial axioms when learning is disallowed, changes in behavior noted in the non-learning system are due solely to the fact that chaining is allowed to work on subproblems that are no longer required to be

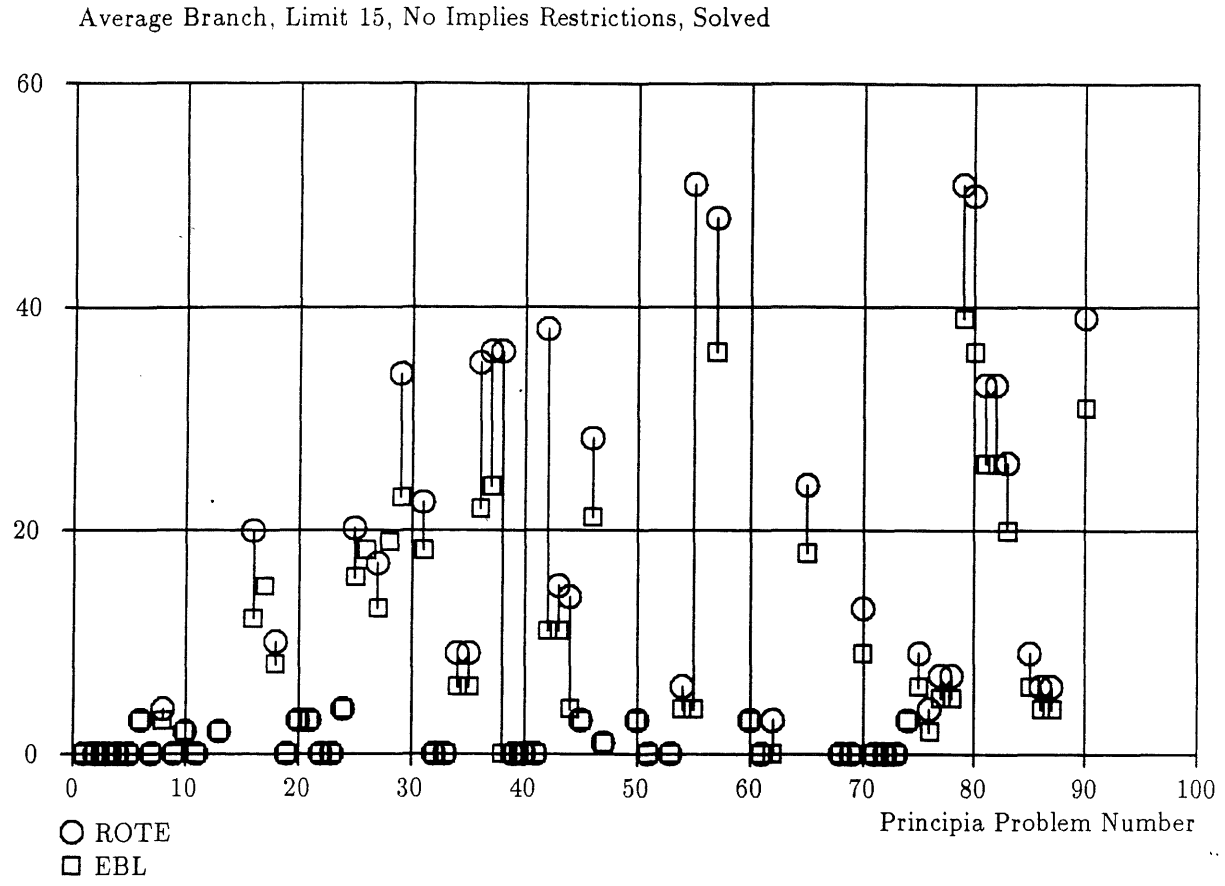

Figure 17. RL vs EBL Without the IMPLIES Restrictions

implications (the initial axioms are always implications). In rote learning, changes are due to both types of IMPLIES restriction, but the effect of the requirement that known theorems be implications is muted by the fact that almost all of the Principia problems are implications. Thus, once they are learned, they can be used even under the IMPLIES restrictions. In the EBL system, however, lifting the IMPLIES restrictions leads to much more pronounced changes in performance, because many of the theorems learned are not implications.

EBL does significantly more search than rote in some cases due to the fact that the IMPLIES restricted LT fails to put the results of explanation-based learning to full use. LT only uses the chaining schema to solve problems when they have the form of implications. In addition, it only uses known implications in chaining, in order to transform problems into new sub-problems. These restrictions effectively prevent the EBL LT from finding some legitimate proofs by preventing LT from using some of the

theorems learned by EBL.[8] Thus, in the remaining experiments discussed in this paper, the IMPLIES restrictions are lifted.

9. Extended Search Experiments

It is important to determine whether the relationships observed so far are dependent on the fact that search was restricted rather severely in the previous experiments. Increasing the search limits should enable each version of the problem solver to solve more problems. What else happens when the search limits are relaxed? In this section, we abandon the implies restrictions and loosen the limits on search.

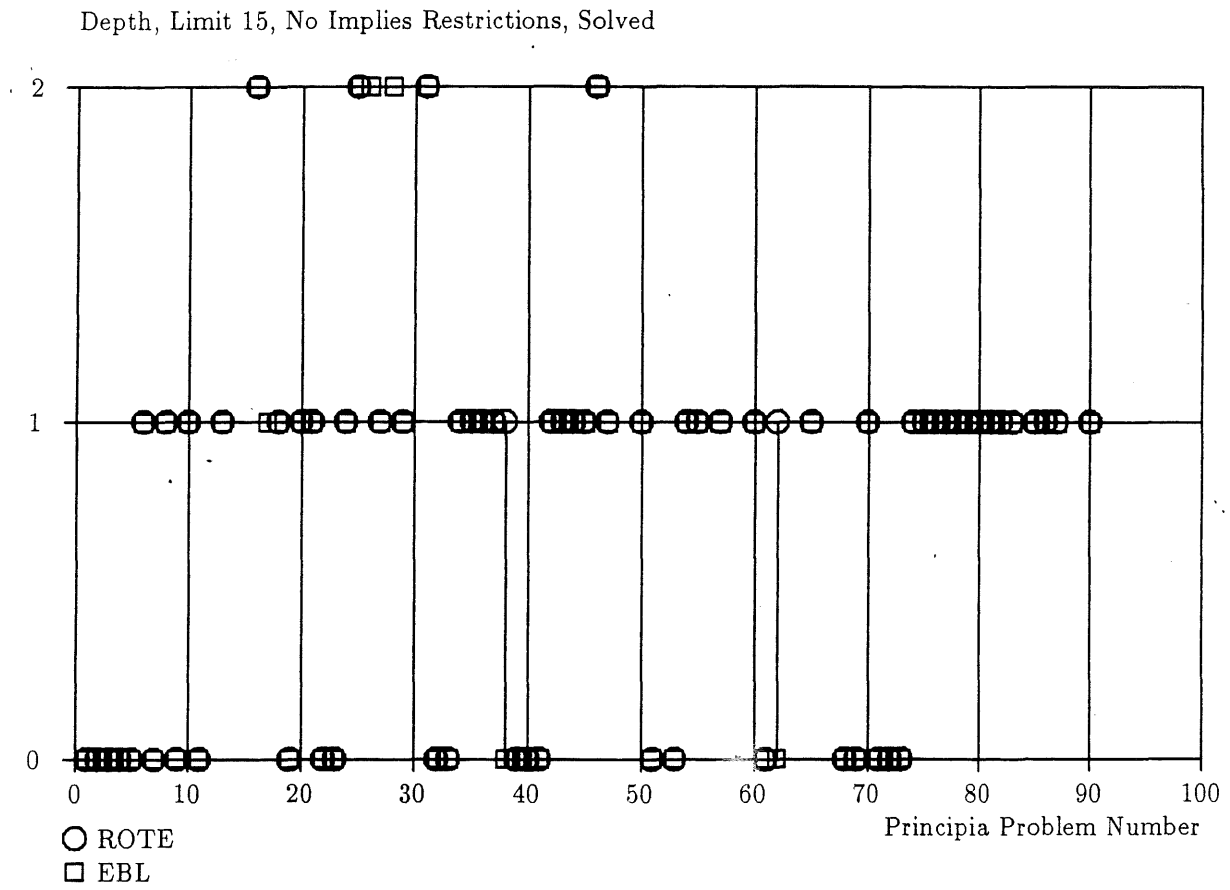

Figure 18. Quality of Solutions Without the IMPLIES Restrictions

[8] Some of the theorems learned by EBL are disjunctions more general than implications; they can be specialized to implications.

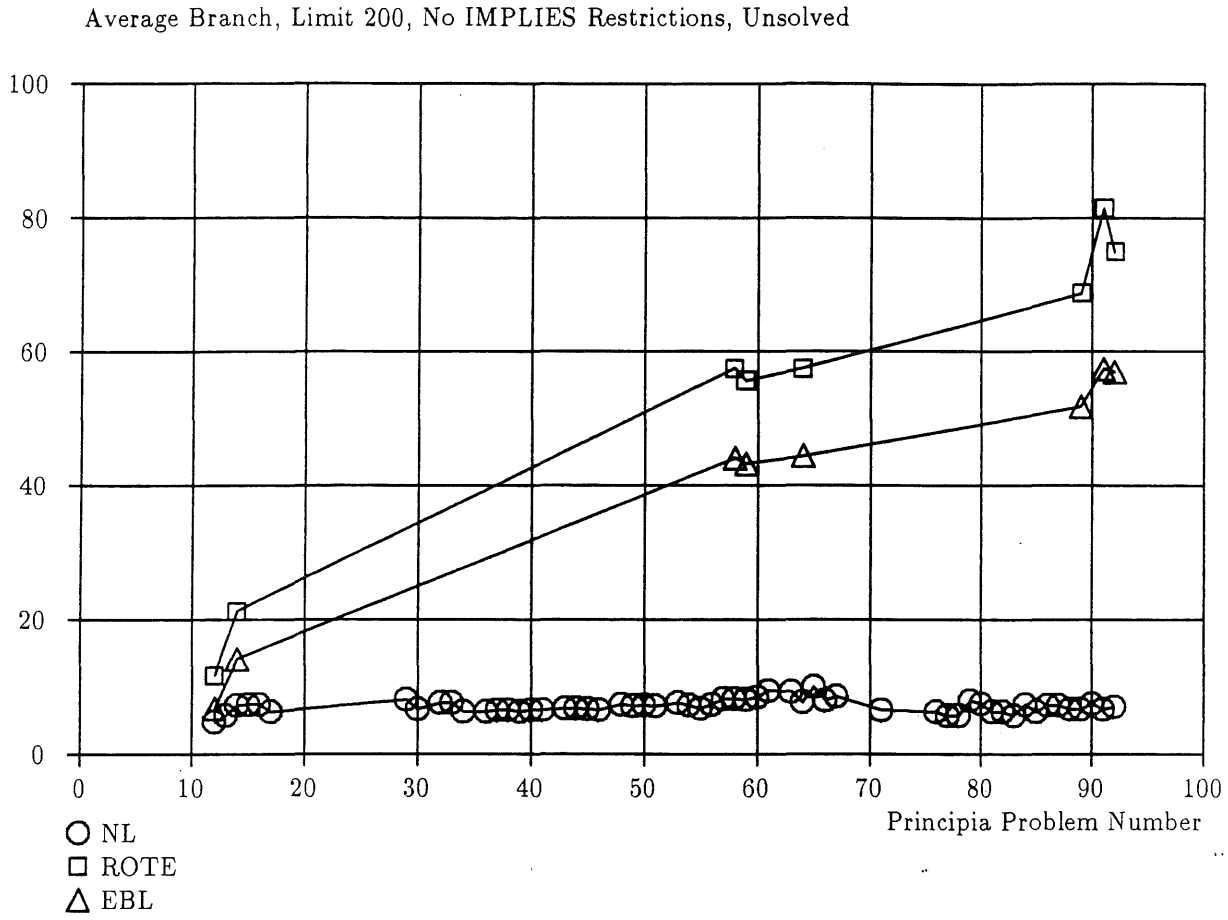

Figure 19. Search Behavior on Unsolved Problems in Extended Search

Instead of only attempting the first 15 subproblems, 200 subproblems may now be attempted.

9.1. Search Performance

Figure 19 shows the search behavior of NL, RL, and EBL on the problems they fail to solve. With extended search, the learning methods each fail to solve only 8 out of 92 problems. As in limited search, NL solves far fewer problems than the learning methods. The branching factor observed in NL seems to be roughly constant and much lower than the sharply increasing branching factors of the learning systems. Again, EBL's branching factors are uniformly below those of RL. With extended search, there is no difference in the number of problems solved by the learning systems, but RL does more search as indicated by subproblems generated and attempted, and

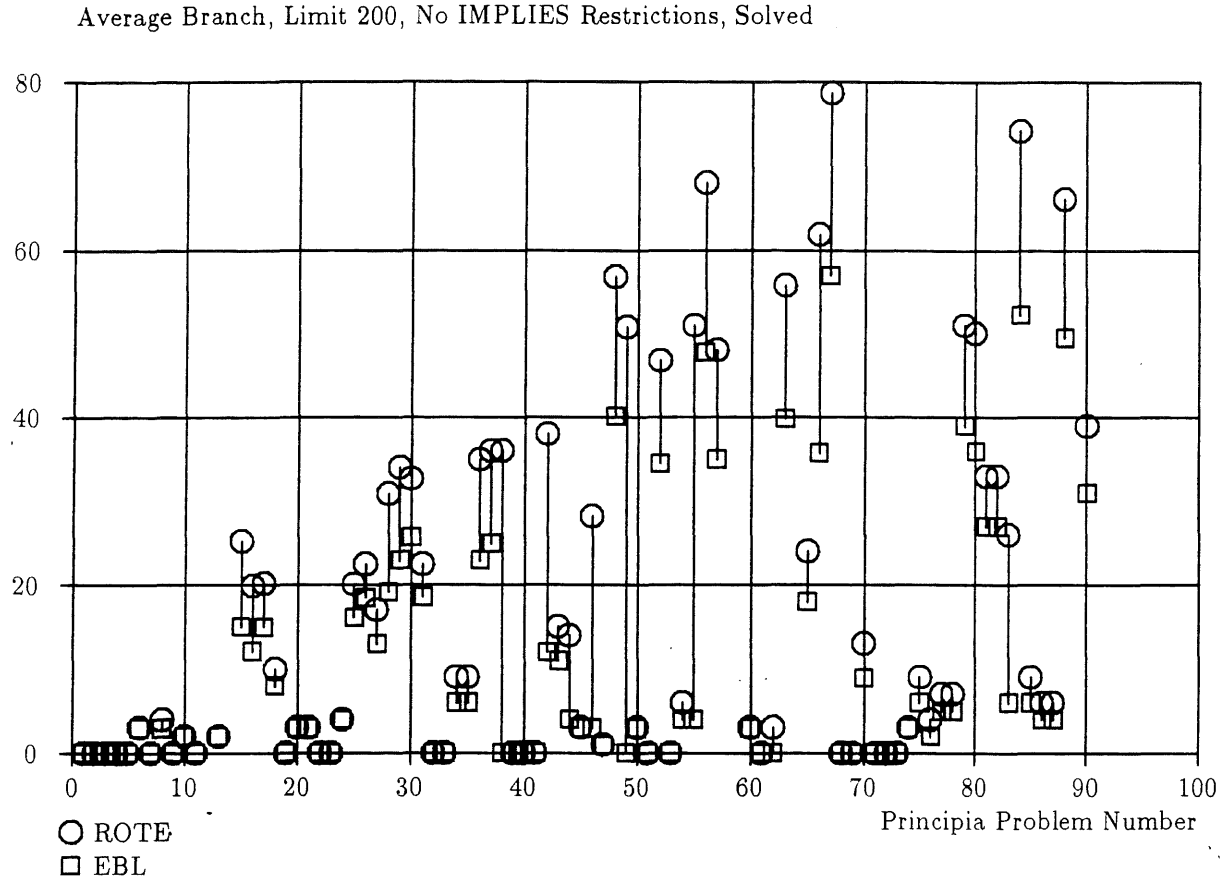

Figure 20. Rote vs EBL on Solved Problems in Extended Search

faces higher branching factors in all cases as shown in Figure 20.

9.2. Quality of Learning

A comparison of the quality of the results learned by RL vs those learned by EBL is shown in Table 1. Note that EBL is not attempted in many cases, specifically in case the original problem is an instance of a known theorem and in cases where no proof is found. The following is a discussion of the theorems learned by EBL versus rote-learning in the remaining cases.

A priori, one can see that when EBL works at all, it provides theorems that are at least as general as the original problems. *The original problems are always instances*

Table 1: Comparison of Theorems Learned by EBL vs Rote

Problem	EBL vs Rote	Theorem Learned by EBL	θ
2.06	>	$\overline{P\backslash/Q}\backslash/(\overline{Q}\backslash/R\backslash/(P\backslash/R))$	$\{P/\overline{p},Q/q,R/r\}$
2.08	=	$P\supset P$	
2.11	=	$P\backslash/\overline{P}$	
2.14	=	$\overline{\overline{P}}\supset P$	
2.16	>	$P\backslash/Q\supset\overline{\overline{Q}}\backslash/P$	$\{P/\overline{p},Q/q\}$
2.17	>	$\overline{\overline{P}}\backslash/Q\supset Q\backslash/P$	$\{P/p,Q/\overline{q}\}$
2.18	=	$(\overline{P}\supset P)\supset P$	
2.2	=	$P\supset P\backslash/Q$	
2.24	>	$P\backslash/(\overline{P}\backslash/Q)$	$\{P/\overline{p},Q/q\}$
2.25	=	$P\backslash/(P\backslash/Q\supset Q)$	
2.3	=	$P\backslash/(Q\backslash/R)\supset P\backslash/(R\backslash/Q)$	
2.31	=	$P\backslash/(Q\backslash/R)\supset(P\backslash/Q)\backslash/R$	
2.32	=	$(P\backslash/Q)\backslash/R\supset P\backslash/(Q\backslash/R)$	
2.36	=	$(P\supset Q)\supset(R\backslash/P\supset Q\backslash/R)$	
2.37	=	$(P\supset Q)\supset(P\backslash/R\supset R\backslash/Q)$	
2.38	=	$(P\supset Q)\supset(P\backslash/R\supset Q\backslash/R)$	
2.4	=	$P\backslash/(P\backslash/Q)\supset P\backslash/Q$	
2.41	=	$P\backslash/(Q\backslash/P)\supset Q\backslash/P$	
2.45	=	$\overline{P\backslash/Q}\supset\overline{P}$	
2.46	=	$\overline{P\backslash/Q}\supset\overline{Q}$	
2.47	>	$\overline{P\backslash/Q}\supset\overline{P}\backslash/R$	$\{P/p,Q/q,R/q\}$
2.48	>	$\overline{P\backslash/Q}\supset R\backslash/\overline{Q}$	$\{P/p,Q/q,R/p\}$
2.521	>	$\overline{\overline{P\backslash/Q}}\backslash/(\overline{Q}\backslash/R)$	$\{P/\overline{p},Q/q,R/p\}$
2.53	=	$P\backslash/Q\supset(\overline{P}\supset Q)$	
2.54	=	$(\overline{P}\supset Q)\supset P\backslash/Q$	
2.55	=	$\overline{P}\supset(P\backslash/Q\supset Q)$	
2.56	=	$\overline{P}\supset(Q\backslash/P\supset Q)$	
2.6	>	$P\backslash/(\overline{Q}\backslash/(R\backslash/Q))$	$\{P/\overline{\overline{p}},Q/q,R/\overline{\overline{p}\backslash/q}\}$
2.61	>	$P\backslash/Q\supset\overline{P\backslash/Q}\backslash/Q$	$\{P/\overline{p},Q/q\}$
2.621	=	$(P\supset Q)\supset(P\backslash/Q\supset Q)$	

Table 1: Comparison of Theorems Learned by EBL vs Rote (ctd.)

Problem	EBL vs Rote	Theorem Learned by EBL	θ
2.64	=	$P\backslash/Q\supset(P\backslash/\overline{Q}\supset P)$	
2.67	>	$\overline{\overline{P\backslash/Q}\backslash/R}\backslash/(\overline{P}\backslash/R)$	$\{P/p,Q/q,R/q\}$
2.68	>	$\overline{\overline{P}\backslash/Q\backslash/R}\backslash/(P\backslash/R)$	$\{P/p,Q/q,R/q\}$
2.69	>	$\overline{\overline{P}\backslash/Q}\backslash/R\supset\overline{\overline{R}\backslash/P}\backslash/P$	$\{P/p,Q/q,R/q\}$
2.73	=	$(P\supset Q)\supset((P\backslash/Q)\backslash/R\supset Q\backslash/R)$	
2.76	=	$P\backslash/(Q\supset R)\supset(P\backslash/Q\supset P\backslash/R)$	
2.81	=	$(P\supset(Q\supset R))\supset(S\backslash/P\supset(S\backslash/Q\supset S\backslash/R))$	
2.83	>	$\overline{P\backslash/(Q\backslash/R)}\backslash/(\overline{P\backslash/(\overline{R}\backslash/S)}\backslash/(P\backslash/(Q\backslash/S)))$	$\{P/\overline{p},Q/\overline{q},R/r,S/s\}$
2.85	>	$\overline{P\backslash/Q}\backslash/(R\backslash/S)\supset R\backslash/(\overline{Q}\backslash/S)$	$\{P/p,Q/q,R/p,S/r\}$
2.86	>	$\overline{P\backslash/Q}\backslash/(\overline{Q}\backslash/R)\supset S\backslash/(\overline{Q}\backslash/R)$	$\{P/\overline{p},Q/q,R/r,S/\overline{p}\}$
3.12	>	$P\backslash/(Q\backslash/\overline{P\backslash/Q})$	$\{P/\overline{p},Q/\overline{q}\}$
3.21	>	$P\backslash/(Q\backslash/\overline{Q\backslash/P})$	$\{P/\overline{p},Q/\overline{q}\}$
3.22	>	$\overline{\overline{P\backslash/Q}\backslash/\overline{Q\backslash/P}}$	$\{P/\overline{p},Q/\overline{q}\}$
3.24	>	$\overline{\overline{P\backslash/\overline{P}}}$	$\{P/\overline{p}\}$
3.26	>	$\overline{\overline{P\backslash/Q}\backslash/P}$	$\{P/p,Q/\overline{q}\}$
3.27	>	$\overline{\overline{P\backslash/\overline{Q}}\backslash/Q}$	$\{P/\overline{p},Q/q\}$
3.3	>	$\overline{\overline{P\backslash/Q}\backslash/R}\supset P\backslash/(Q\backslash/R)$	$\{P/\overline{p},Q/\overline{q},R/r\}$
3.31	>	$P\backslash/(Q\backslash/R)\supset\overline{\overline{P\backslash/Q}\backslash/R}$	$\{P/\overline{p},Q/\overline{q},R/r\}$
3.33	>	$\overline{\overline{P\backslash/Q}\backslash/\overline{\overline{Q}\backslash/R}}\backslash/(P\backslash/R)$	$\{P/\overline{p},Q/q,R/r\}$
3.34	>	$\overline{\overline{P\backslash/Q}\backslash/\overline{R\backslash/P}}\backslash/(R\backslash/Q)$	$\{P/p,Q/q,R/\overline{r}\}$
3.35	>	$\overline{\overline{P\backslash/\overline{P\backslash/Q}}\backslash/Q}$	$\{P/\overline{p},Q/q\}$
3.37	>	$\overline{\overline{P\backslash/Q}\backslash/R}\supset\overline{\overline{P\backslash/\overline{R}}\backslash/Q}$	$\{P/\overline{p},Q/\overline{q},R/r\}$
3.4	>	$\overline{\overline{P\backslash/\overline{Q}}\backslash/(R\backslash/Q)}$	$\{P/\overline{p},Q/q,R/\overline{p}\}$
3.41	>	$\overline{P\backslash/Q}\backslash/(\overline{\overline{P}\backslash/R}\backslash/Q)$	$\{P/p,Q/q,R/\overline{r}\}$
3.42	>	$\overline{P\backslash/Q}\backslash/(\overline{R\backslash/\overline{P}}\backslash/Q)$	$\{P/p,Q/q,R/\overline{r}\}$
3.43	>	$\overline{\overline{P\backslash/Q}\backslash/\overline{P\backslash/R}}\backslash/(P\backslash/\overline{Q\backslash/R})$	$\{P/\overline{p},Q/q,R/r\}$
3.45	>	$P\backslash/Q\supset\overline{P\backslash/R}\backslash/\overline{Q\backslash/R}$	$\{P/\overline{p},Q/q,R/\overline{r}\}$

of the generalized conclusions of their proofs.[9]

From the experiment, one sees that while many theorems (25 of 57, or about 44%) learned via EBL are no more general than the given problems, in many cases (32 of 57, about 56%) the theorem added by EBL is strictly more general than the theorem added by rote-learning.

How are the results of EBL more general? In many cases (e.g., 2.06) EBL produced a variant of the original problem, minus an extraneous negation. The EBL version of LT acquires the theorem $\overline{P\backslash/Q}\backslash/(\overline{Q\supset R}\backslash/(P\backslash/R))$ from the solution of Principia-2.06. The problem, $(p\supset q)\supset((q\supset r)\supset(p\supset r))$, is an instance of this theorem with Q bound to q, R bound to r, and P bound to $\bar{p}$. (This is the meaning of the difference substitution in the theta column of Table 1.) The rote-learning LT demands a negation that is not required by the proof.

The EBL version of LT offers only modest improvements over rote-learning in such examples because it is well known that one can reverse the sign of a literal everywhere in a theorem to get a new theorem. One could easily modify the rote-learning LT to take advantage of the fact that whenever a literal appears only negatively in a *Principia* problem, one can safely delete the negative sign to obtain a logically equivalent but syntactically more general problem.

In a number of cases, however, there is no such simple fix that the rote-learning LT could use to obtain theorems as general as those acquired by the EBL version. For example, in a number of cases, rote-learning unnecessarily collapses two or more variables into one. Problem 36, Principia-2.47, is $\overline{p\backslash/q}\supset(p\supset q)$. The EBL version of LT acquires $\overline{P\backslash/Q}\supset(P\backslash/R)$ from the proof. The problem is obtained as an instance by substituting p for P and by binding both Q and R to q.

In other cases rote-learning results in more interesting overspecializations. In these examples, variables are not simply collapsed by rote-learning, instead they are required to be related logically in complicated ways when they really should be completely independent.

In Problem 42, one variable is made to be the negation of another when they should be independent and neither need be a negation. The problem (Principia-2.521) is $\overline{p\supset q}\supset(q\supset p)$, the generalized conclusion of the proof is $\overline{\overline{P\backslash/Q}\backslash/(\overline{Q}\backslash/R)}$. These match with bindings of P to $\bar{p}$, Q to q, and R to p.

In problem 47, three independent variables are specialized by effectively making one into a negation of an implication between the others and by requiring one to be a

---

[9] The reader should note that we are using a purely syntactic measure of generality (see [20]). We are considering our well-formed propositional formulae as terms, focusing on the syntax of these terms, and ignoring logical relationships between them such as the fact that P and NOT NOT P are logically equivalent. A one-way match between the theorems learned by EBL and the theorems learned by RL is used to determine whether they are variants. If they are variants, they are instances of each other and so equally general. Otherwise, the theorem learned by EBL is strictly more general than the corresponding theorem learned by RL and a substitution list shows how to specialize the EBL result to get the one learned by RL.

double negative. The problem (Principia-2.6) is $\overline{p}\supset(q\supset((p\supset q)\supset q))$ and the generalized conclusion of the proof is $P\vee(Q\supset(R\vee Q))$. These match with P bound to $\overline{\overline{p}}$, Q bound to q, and R bound to $\overline{p\supset q}$.

With LT's IMPLIES restrictions loosened, the improved generality afforded by EBL leads to performance that is superior to that of the rote-learning version of LT.[10] Sometimes the EBL system solves problems that could not be solved by the rote system due to limits on search. Sometimes both RL and EBL solve the problem but the EBL

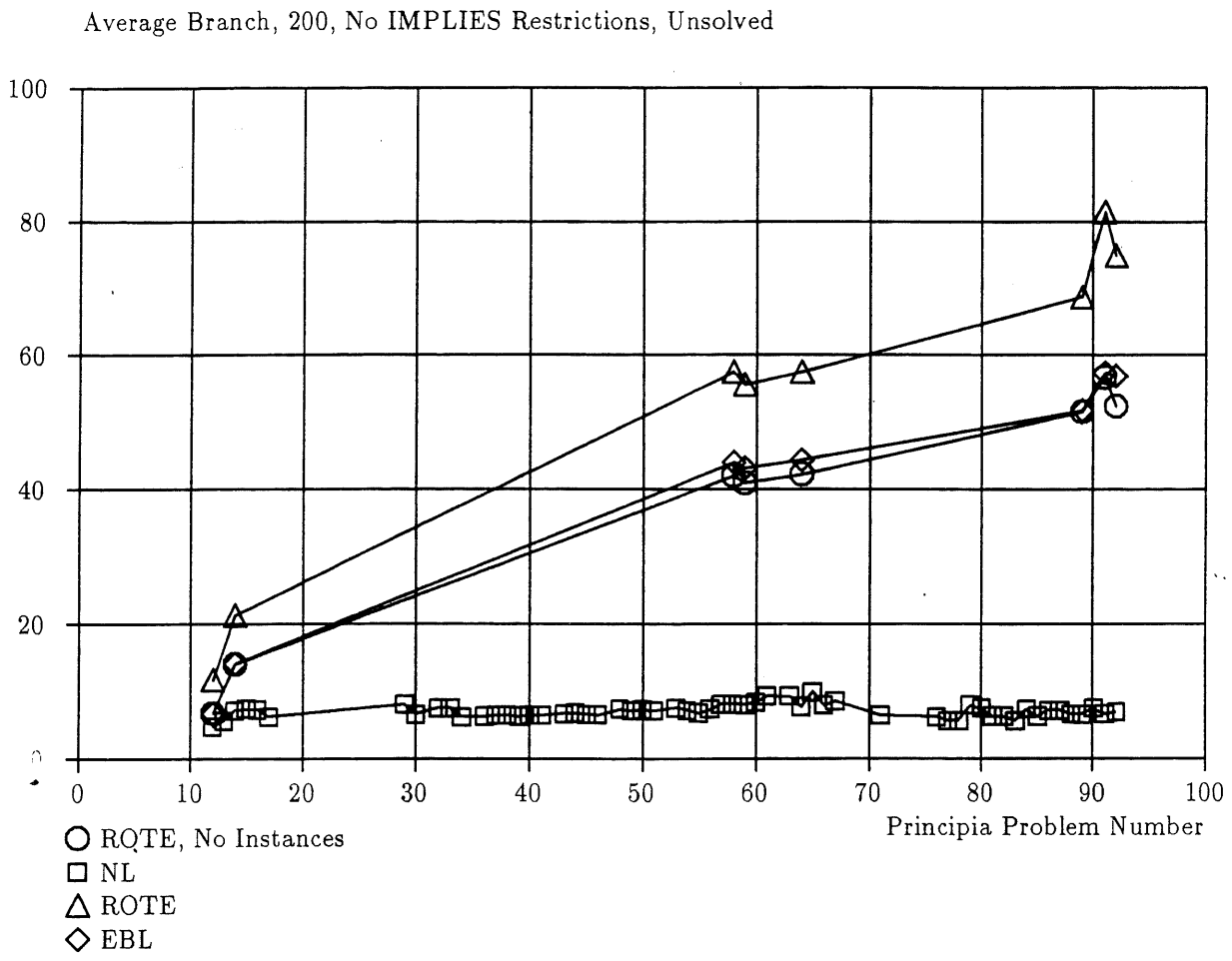

Figure 21. The Effects of Adding Instances During Rote Learning

[10] For detailed discussions of examples of how EBL improves performance, see the appendix on "Examples of Improvements in Performance Due to EBL."

solution is found earlier in the search. In some of these cases, the EBL solution is of higher quality in that it is simpler than the solution provided by RL. The quality of solutions found by the learning systems in extended search was not discussed in this section. Instead, this comparison will take place in the next section, in the context of a comparison between EBL and a version of rote learning that produces the same solutions as the present rote learning system using a more efficient search.

10. On the Effects of Adding Instances in Rote Learning

In the future work section of [18] it was stated that it would be interesting to determine the exact contribution of each source of the improved performance of the EBL version of LT. The improved generality of learned solutions helps, as does the fact that the EBL system does not add instances of known theorems. In this section, we report on an experiment that isolates these sources of improvement by allowing the

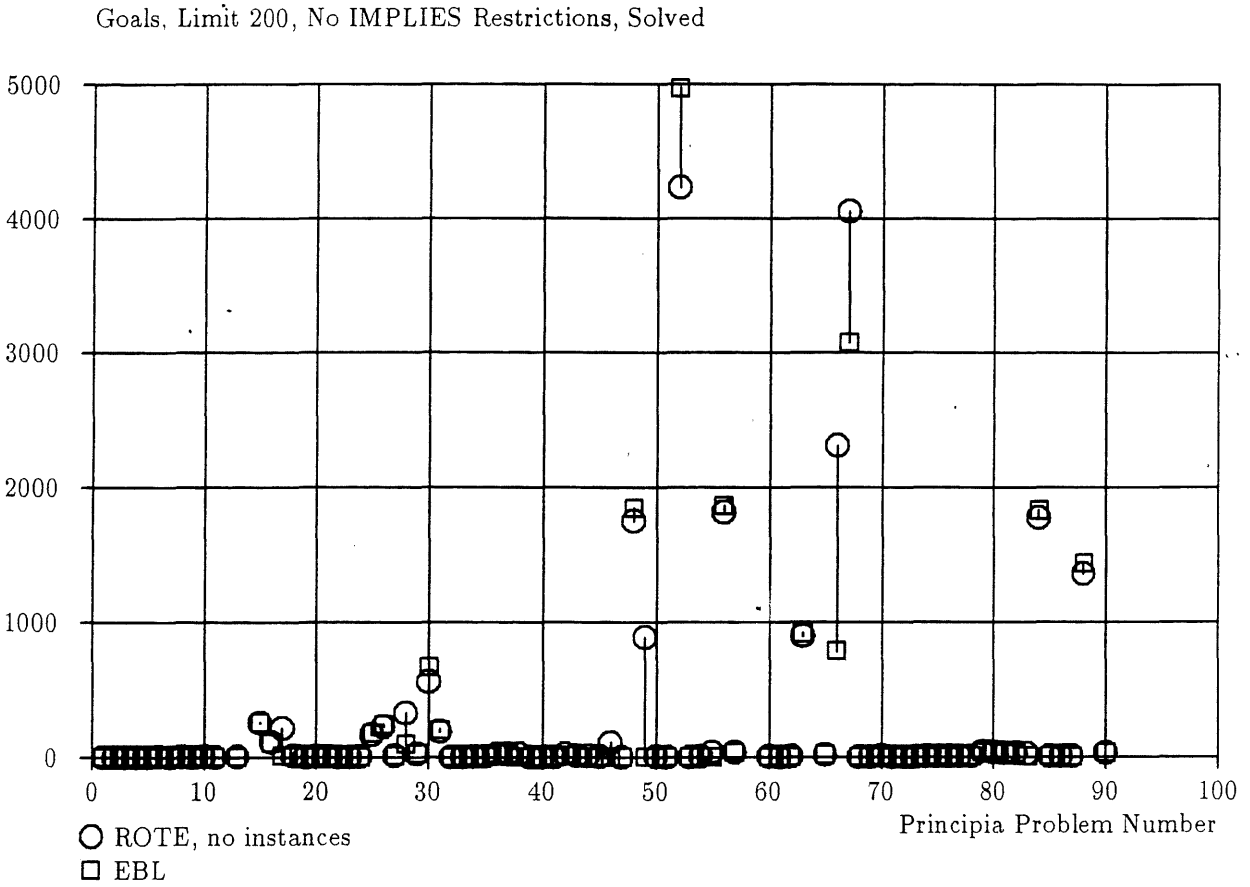

Figure 22. SubGoals Generated by Improved Rote vs EBL on Solved Problems

rote-learning LT to avoid adding problems that are instances of known theorems.

Figure 21 shows the search behavior of RL (without adding instances) on unsolved problems superimposed on that of the systems previously studied. Figure 22 illustrates the search performance of the improved RL system vs EBL on solved problems. The differences in numbers of problems attempted are very similar to differences shown between the number of subproblems generated.

The results on search behavior indicate that adding instances accounts for much of the decrease in branching factors that occurs in going from the original rote to EBL. While EBL does uniformly less search than the original RL system on both solved and unsolved problems, EBL tends to do slightly more search than the improved RL method. This is always the case on the unsolved problems, and often the case in solved problems as well. However, exceptions in the solved problems occur when EBL pays off

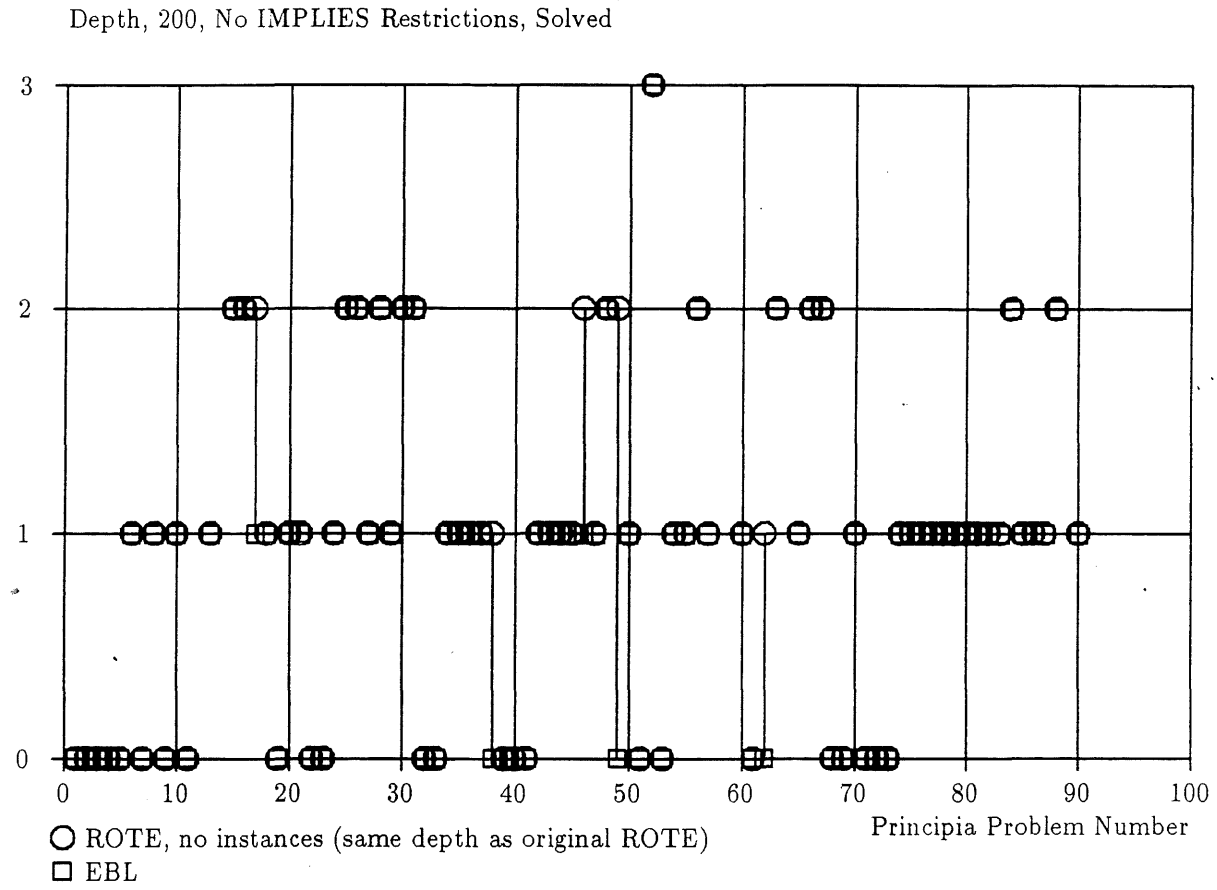

Figure 23. Quality of Solutions Provided by (Improved) Rote vs EBL

by enabling the problem solver to hit upon a solution earlier in the search. As a result, EBL generates 2791 fewer subproblems than the improved RL method and attempts 84 fewer problems overall on the problems solved by both methods.

RL should never produce shorter proofs than EBL since the theorems learned by EBL are always at least as general as those learned by RL. Figure 23 shows that some of the proofs discovered by EBL are shorter than the proofs provided by the improved RL system. The bottom line seems to be that, for a small increase in search on most problems, one can buy large decreases in search and improved solutions on some problems by using EBL rather than the improved rote learning procedure.

11. Conclusion

This section summarizes the contributions of this study toward an empirical understanding of explanation-based learning (EBL). It might be nice to make grand general claims of the form "learning method X is universally superior to all other forms of learning" at this point. No such claim can follow from experiments, however, because one can only test a given learning method against a limited set of competitors, in the context of a limited set of performance elements, on limited sets of test data.

It would make no sense to claim that the performance element used in these experiments is a "representative" performance element. It would make even less sense to claim that the problems used in this study are "typical" problems. Performance elements and problem domains vary dramatically. Thus the conclusions of this study must be carefully qualified. Peculiarities of the performance element that seem to interact with the learning methods and the problem domain under study are identified. The test problems used here are also characterized, especially when aspects of the problems influence the effectiveness of learning.

One virtue of this study is that it explores the performance of a complete EBL system against competing learning methods on a significant number of examples. The examples used in the study are uncontrived in the sense that the *Principia* problem set was designed for entirely different purposes three quarters of a century ago — long before electronic computers existed and with no consideration of machine learning experiments in mind. During the study a number of interactions between performance elements, learning elements, and the problem set were observed. Some of these interactions were quite unexpected.

The experiments reported here show that both EBL and rote-learning are much better than no learning at all on the Principia problems. The experiments focus on the difference between EBL and rote learning in an abstract, purely logical setting, using very general problems, where neither learning method is allowed an advantage in "turning constants to variables."

Before these experiments were done, it was hypothesized that generalization and performance would not improve in going from rote to EBL on purely logical problems because neither learning system is allowed to make inductive leaps from concrete propositions like "It is raining" or "Today is Monday." It seems harder for EBL to "win big" on the Principia problems because any improvements of EBL over rote-

learning are forced to occur at a very high level of abstraction. Indeed, the experiments can be interpreted as supporting this hypothesis, because rote-learning is "roughly comparable" to EBL on the Principia problems in the sense that the difference between the learning systems is small when compared with the large differences between non-learning and learning.

Ignoring non-learning, however, the experimental data focusing on the differences between the learning systems clearly shows that EBL is significantly more effective than rote-learning even in highly abstract settings. The experiments and analysis show that even in this context EBL learns strictly more general results quite frequently and that — *provided the matcher is "smart" enough* — EBL's superior generalization contributes to superior problem solving performance.

The caveat that the matcher should be "smart" is critical. The experiments reported here include all problems from chapter 3 of Principia, thus going beyond the 52 problems from chapter 2 used in Newell, Shaw, and Simon's work and our early work reported in [18]. We were surprised to find that, in a reversal of our earlier results, EBL's overall performance was actually worse than rote in some experiments involving the new problems. The explanation turned out to be that, while the matcher supplied to our version of LT was capable of decoding $P \supset Q$ using its definition in terms of $P \backslash / Q$, this was short-circuited by LT's "IMPLIES restrictions". These restrictions forced problems and known theorems to have IMPLIES as the main connective, rendering unusable some of the generalizations of problems learned by EBL. Thus it was found that LT's IMPLIES restrictions can cause EBL to be *less effective* than rote-learning. This is an instance of a more general observation, that it is important not to restrict matching unnecessarily in EBL.

When one considers the goals of the designers of a logical system such as the one found in *Principia Mathematica*, it seems surprising that the EBL LT (with the IMPLIES restrictions removed) performs significantly better than rote-learning on the *Principia* problems. Surely, the authors of *Principia* must have intended each problem to be as general as possible. Imagine Whitehead and Russell creating their sequence of theorems. Assume they have proposed a certain problem as the next theorem in the sequence. If they found a proof for this problem that actually proved a more general theorem than the problem they began with, they should have crossed out the proposed theorem, replacing it with the more general one. With this in mind, one can view the superior generalizations produced by the EBL version of LT as suggesting improvements on the logical system of *Principia Mathematica*.

12. Relation to Previous Work

This is part of the growing body of work on Explanation-Based Learning. The papers by DeJong and Mooney [3, 14] and Mitchell, et al [13] provide overviews and pointers into the EBL literature. The present paper is basically an experimental study of EBL versus no learning and rote learning in a particular "domain", a logic where rote learning was expected to do well because of the generality of the problems. Other experimental studies of EBL involving relatively large numbers of examples have been

done recently in a number of other task domains, including planning [11, 12]. Mooney's Ph.D. thesis [15] reports on experiments with a general EBL system solving examples from a number of task domains such as planning, recognition, etc. Shavlik's Ph.D. thesis [24] contains the results of a number of experiments on planning and physical reasoning tasks and Segre's Ph.D. thesis [22, 23] contains experimental work on applications of EBL in robotics.

The present study is also related to work on macro-operators, and can be viewed as a step in the direction of viewing theorem proving as a process that can benefit from the acquisition of macro-operators (as suggested, for example, by Korf in [7]). However, it should be noted that the way explanation-based generalization contributes to future problem solving in the present study is a bit unusual. In contrast to EBG macro-learning systems based on PROLOG [6, 19] and MRS [5] only generalized conclusions are learned in the EBL LT. The leaves of the "generalized" proof tree are thrown away; they are not needed because they are (always true) theorems.

It might be interesting to try an alternative approach: an EBL version of LT could learn general macro operators which would play roles similar to detachment and chaining. These operators could be constructed by ignoring not just the specific problem proved by a composition of schemata, but also by ignoring the specific known theorems used to ground the leaves of the proof tree.

Another way to view the work presented here is as evidence bearing on the relative value of methods for generalizing examples by turning constants into variables versus EBL methods of generalizing examples by specializing existing general knowledge. Thus, this work may be viewed as providing experimental evidence that conversions such as the one described in [21] should lead to improved performance.

13. Future Work

More experiments should be done involving complete explanation-based learning systems and large numbers of examples. Several possible extensions to the experiments reported here suggest themselves. For example, we have recently completed new experiments involving reversals and random permutations of the problem sets, in an effort to determine how changes in the order of the *Principia* problems affect the learning methods. In another experiment, we have disallowed learning of unsolved problems, in order to address concerns that this strategy, while it may be appropriate for highly structured learning situations, is not appropriate in general. The results of these experiments will be presented in a follow-on paper.

It would be interesting to see how much of the improvement in going from non-learning to rote learning and EBL is due to the limited control structure of LT requiring that proofs be linear. How much of the improvement is due to macro learning? It might also be interesting to augment the simple EBL version of LT with *subgoal learning*. This could yield results on how much improvement occurs in *between trial learning*. The original LT would have to be modified to produce non-linear proof trees in order to get results on the effectiveness of *within trial learning*.

The results on the *Principia* problems presented in the present paper are part of an effort to provide a baseline characterization of the performance of learning methods such as rote and basic EBL on these problems. It is hoped that this will make it possible to use the *Principia* problems as a benchmark for testing improved learning methods, just as problems like the eight puzzle have been considered the *Drosophila* or fruit fly of research on search in AI [8].

If one looks at the theorems learned by EBL in the present study, it is obvious that they could stand some improvement. Many of these theorems are still less general than they could be (e.g., some contain double negations). It would be interesting to see how performance changes in this domain when basic EBL is augmented by a system designed to transform learned expressions into their most general form.

The present domain also seems like a good place for studies of proposed improvements in storing and accessing learned results. It was noted here that it is a mistake to add instances to the end of a list of known theorems in rote learning. It would be interesting to see if they can make a positive contribution if they are added to the beginning of the list rather than the end. Unlike the EBL LT discussed in this paper, EBL systems typically try the most recently learned solutions first. This strategy may make especially good sense when facing highly structured sequences of problems where new problems often build upon their immediate predecessors as in *Principia*. Indeed, it would be interesting to try out more sophisticated storage and retrieval mechanisms in this highly organized domain. Rather than looking down linear lists as was done in this study, discrimination nets [1]) could be used. Perhaps more powerful methods for organizing learned knowledge can be used so as to benefit EBL.

Evidence that carefully controlled "forgetting" will play an important role in the management of memories containing learned results is beginning to accumulate. One such method is based on tracking the usefulness of learned results in terms of measurements of search utility. It would be interesting to see whether Minton's results on search utility in planning [11] can be replicated in the purely logical domain of *Principia*.

Acknowledgements

Some of the results presented here were originally published in the Proceedings of the Fourth International Conference on Machine Learning [18]. The research presented here grew out of a discussion with Pat Langley at the Cognitive Science Conference held at the University of California, Irvine in the summer of 1985. Early work was carried out by the author when he was a member of Gerald DeJong's Explanation-Based Learning Research Group at the University of Illinois at Champaign-Urbana. Special thanks to Gerald DeJong and Scott Bennett at Illinois, to Ray Mooney (now at the University of Texas at Austin), and also to Pat Langley, Michael Pazzani, Tony Wieser, and Heping He at Irvine. Wieser provided a great deal of research assistance to the author, including reimplementation of the author's LISP systems in PROLOG, as well as having discovered that the IMPLIES restriction forced the EBL LT's performance to deteriorate in the extended experiments first reported here. Heping He

provided programming support and assisted in the analysis and presentation of the data. This research was supported in part by the National Science Foundation under grant NSF IST 83-17889, by a Cognitive Science/AI Fellowship from the University of Illinois, and by a McDonnell-Douglas University External Relations award to UCI.

Appendix 1: Examples of Improvements in Performance Due to EBL

The superior generality of EBL contributes to superior problem solving performance in two main ways. Sometimes it enables the problem solver to solve problems that could not be solved before. Alternatively, when both rote-learning and EBL systems solve a problem, the EBL solution is sometimes found more quickly and is sometimes simpler than that provided by rote-learning.

For an example of an EBL system solving more problems as a result of improved generality note that the EBL version of LT found a proof for Problem 17 (Principia-2.18) while the rote-learning version failed to find a proof in small search (with the subproblems attempted limited to 15). The proof found by EBL involves the theorem learned from Problem 16 (Principia-2.17). It was obtained by chaining forward on the learned theorem and axiom Principia-1.2 (see Figure 24). The generalized conclusion of this proof of Principia-2.18 is $(\overline{A} \supset A) \supset A$. The proof is not constructed by rote-learning because of the extraneous NOT in Principia-2.17.

Chaining forward on the result of rote-learning on Principia-2.17 yields the less general conclusion $(\overline{\overline{A}} \supset \overline{A}) \supset \overline{A}$, (Figure 25). Principia-2.18 $(\overline{P} \supset P) \supset P$ is not an instance of this conclusion.

Concrete examples of EBL constructing simpler solutions as a result of improved generality also occurred in the experiments (see Figure 26). While both learning versions of LT solve Problem 38 (Principia-2.49), the EBL version recognizes it as an instance of the class of problems solved by a previous solution while the rote-learning version has to regenerate that solution. The problem is $\overline{P \vee Q} \supset (\overline{P} \vee \overline{Q})$, an instance of the generalized conclusion of the proof of Problem 36 (Principia-2.47), namely $\overline{A \vee B} \supset (\overline{A} \vee D)$. This proof is not constructed during rote-learning because Principia-2.47 identifies B and D but Q and $\overline{Q}$ are incompatible. It turns out that in order to prove Principia-2.49, the rote-learning LT winds up having to prove the theorem that the explanation-

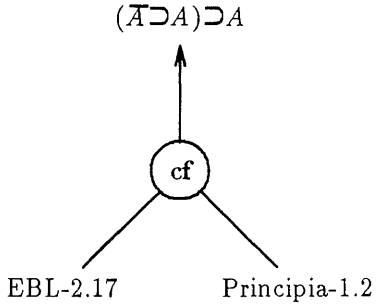

Figure 24. The Proof of Principia-2.18

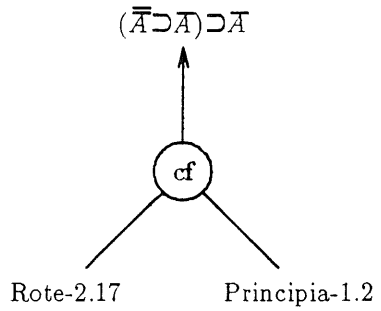

Figure 25. Inferiority of Rote Learning on Principia-2.18

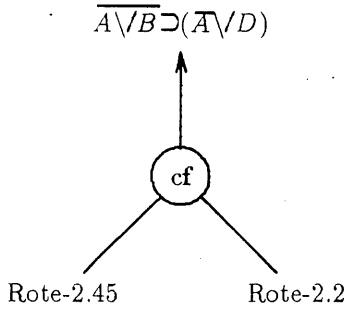

Figure 26. Rote-Learning Proof of Principia-2.47 and 2.49

based learning LT extracted from Principia-2.47. That is, it regenerates the same proof that it used before on Principia-2.47 because it failed to learn all it could from this proof. This is a very clear case of rote-learning losing because it simply stores problems instead of generalizing and computing the class of problems that a novel solution solves.

Appendix 2: Axioms and Problems from Principia

This appendix contains the propositional logic axioms and problems used in experiments with non-learning, rote-learning and explanation-based learning versions of LT. The axioms corresponding to numbers 1.2 through 1.6 in Whitehead and Russell's *Principia Mathematica* are shown in Table 2. Note that Principia-1.1 is not included because it corresponds to a rule of inference, an operator in LT.

Table 2: Axioms from Principia

Designation in Principia	Axiom
1.2	$p \vee p \supset p$
1.3	$p \supset q \vee p$
1.4	$p \vee q \supset q \vee p$
1.5	$p \vee (q \vee r) \supset q \vee (p \vee r)$
1.6	$(p \supset q) \supset (r \vee p \supset r \vee q)$

The problems are shown in Table 3. They are the first 92 theorems from chapters two and three of part one of *Principia*. The axioms and problems are listed in an abbreviated format using operators for (in order of increasing priority) *logical equivalence*, *implication*, *disjunction*, *conjunction*, and *negation*. Machine readable versions of these (and other) *Principia* axioms and problems may be obtained by writing the author (preferably via electronic mail).

Table 3: Problems from Principia

Problem	Theorem	Problem	Theorem
2.01	$(p \supset \bar{p}) \supset \bar{p}$	2.6	$\bar{p} \supset (q \supset ((p \supset q) \supset q))$
2.02	$q \supset (p \supset q)$	2.61	$(p \supset q) \supset ((\bar{p} \supset q) \supset q)$
2.03	$(p \supset \bar{q}) \supset (q \supset \bar{p})$	2.62	$p \vee q \supset ((p \supset q) \supset q)$
2.04	$(p \supset (q \supset r)) \supset (q \supset (p \supset r))$	2.621	$(p \supset q) \supset (p \vee q \supset q)$
2.05	$(q \supset r) \supset ((p \supset q) \supset (p \supset r))$	2.63	$p \vee q \supset (\bar{p} \vee q \supset q)$
2.06	$(p \supset q) \supset ((q \supset r) \supset (p \supset r))$	2.64	$p \vee q \supset (p \vee \bar{q} \supset p)$
2.07	$p \supset p \vee p$	2.65	$(p \supset q) \supset ((p \supset \bar{q}) \supset \bar{p})$
2.08	$p \supset p$	2.67	$(p \vee q \supset q) \supset (p \supset q)$
2.1	$\bar{p} \vee p$	2.68	$((p \supset q) \supset q) \supset p \vee q$
2.11	$p \vee \bar{p}$	2.69	$((p \supset q) \supset q) \supset ((q \supset p) \supset p)$
2.12	$p \supset \bar{\bar{p}}$	2.73	$(p \supset q) \supset ((p \vee q) \vee r) \supset q \vee r)$
2.13	$p \vee \bar{\bar{p}}$	2.74	$(q \supset p) \supset ((p \vee q) \vee r) \supset p \vee r)$
2.14	$\bar{\bar{p}} \supset p$	2.75	$p \vee q \supset (p \vee (q \supset r) \supset p \vee r)$
2.15	$(\bar{p} \supset q) \supset (\bar{q} \supset p)$	2.76	$p \vee (q \supset r) \supset (p \vee q \supset p \vee r)$
2.16	$(p \supset q) \supset (\bar{q} \supset \bar{p})$	2.77	$(p \supset (q \supset r)) \supset ((p \supset q) \supset (p \supset r))$
2.17	$(\bar{q} \supset \bar{p}) \supset (p \supset q)$	2.8	$q \vee r \supset (\bar{r} \vee s \supset q \vee s)$
2.18	$(\bar{p} \supset p) \supset p$	2.81	$(q \supset (r \supset s)) \supset (p \vee q \supset (p \vee r \supset p \vee s))$
2.2	$p \supset p \vee q$	2.82	$((p \vee q) \vee r) \supset (((p \vee \bar{r}) \vee s) \supset ((p \vee q) \vee s))$
2.21	$\bar{p} \supset (p \supset q)$	2.83	$(p \supset (q \supset r)) \supset ((p \supset (r \supset s)) \supset (p \supset (q \supset s)))$
2.24	$p \supset (\bar{p} \supset q)$	2.85	$(p \vee q \supset p \vee r) \supset p \vee (q \supset r)$
2.25	$p \vee (p \vee q \supset q)$	2.86	$((p \supset q) \supset (q \supset r)) \supset (p \supset (q \supset r))$
2.26	$\bar{p} \vee ((p \supset q) \supset q)$	3.1	$p \wedge q \supset \bar{p} \vee \bar{q}$
2.27	$p \supset ((p \supset q) \supset q)$	3.11	$\bar{p} \vee \bar{q} \supset p \wedge q$
2.3	$p \vee (q \vee r) \supset p \vee (r \vee q)$	3.12	$\bar{p} \vee (\bar{q} \vee (p \wedge q))$
2.31	$p \vee (q \vee r) \supset (p \vee q) \vee r$	3.13	$p \wedge q \supset \bar{p} \vee \bar{q}$
2.32	$(p \vee q) \vee r \supset p \vee (q \vee r)$	3.14	$\bar{p} \vee \bar{q} \supset p \wedge q$
2.36	$(q \supset r) \supset (p \vee q \supset r \vee p)$	3.2	$p \supset (q \supset p \wedge q)$
2.37	$(q \supset r) \supset (q \vee p \supset p \vee r)$	3.21	$q \supset (p \supset p \wedge q)$
2.38	$(q \supset r) \supset (q \vee p \supset r \vee p)$	3.22	$p \wedge q \supset q \wedge p$
2.4	$p \vee (p \vee q) \supset p \vee q$	3.24	$p \wedge \bar{p}$
2.41	$q \vee (p \vee q) \supset p \vee q$	3.26	$p \wedge q \supset p$
2.42	$\bar{p} \vee (p \supset q) \supset (p \supset q)$	3.27	$p \wedge q \supset q$
2.43	$(p \supset (p \supset q)) \supset (p \supset q)$	3.3	$(p \wedge q \supset r) \supset (p \supset (q \supset r))$
2.45	$p \vee q \supset \bar{p}$	3.31	$(p \supset (q \supset r)) \supset (p \wedge q \supset r)$
2.46	$p \vee q \supset \bar{q}$	3.33	$(p \supset q) \wedge (q \supset r) \supset (p \supset r)$
2.47	$p \vee q \supset \bar{p} \vee q$	3.34	$(q \supset r) \wedge (p \supset q) \supset (p \supset r)$
2.48	$p \vee q \supset p \vee \bar{q}$	3.35	$p \wedge (p \supset q) \supset q$
2.49	$p \vee q \supset \bar{p} \vee \bar{q}$	3.37	$(p \wedge q \supset r) \supset (p \wedge \bar{r} \supset \bar{q})$
2.5	$p \supset \bar{q} \supset (\bar{p} \supset q)$	3.4	$p \wedge q \supset (p \supset q)$
2.51	$p \supset \bar{q} \supset (p \supset \bar{q})$	3.41	$(p \supset r) \supset (p \wedge q \supset r)$
2.52	$p \supset \bar{q} \supset (\bar{p} \supset \bar{q})$	3.42	$(q \supset r) \supset (p \wedge q \supset r)$
2.521	$p \supset \bar{q} \supset (q \supset p)$	3.43	$(p \supset q) \wedge (p \supset r) \supset (p \supset q \wedge r)$
2.53	$p \vee q \supset (\bar{p} \supset q)$	3.44	$(q \supset p) \wedge (r \supset p) \supset (q \vee r \supset p)$
2.54	$(\bar{p} \supset q) \supset p \vee q$	3.45	$(p \supset q) \supset (p \wedge r \supset q \wedge r)$
2.55	$\bar{p} \supset (p \vee q \supset q)$	3.47	$(p \supset r) \wedge (q \supset s) \supset (p \wedge q \supset r \wedge s)$
2.56	$\bar{q} \supset (p \vee q \supset p)$	3.48	$(p \supset r) \wedge (q \supset s) \supset (p \vee q \supset r \vee s)$

REFERENCES

1. E. Charniak and D. McDermott, *Introduction to Artificial Intelligence Programming*, Lawrence Erlbaum Associates, Hillsdale, N.J., 1986.
2. G. F. DeJong, "An Approach to Learning from Observation," in *Machine Learning: An Artificial Intelligence Approach, Vol.* 2, Ryszard S. Michalski, Jaime G. Carbonell, and Tom M. Mitchell (ed.), Morgan Kaufmann, Los Altos, California, 1986, 571-590.
3. G. F. DeJong and R. Mooney, "Explanation-Based Learning: An Alternative View," *Machine Learning* 1, 2 (1986), 145-176.
4. H. B. Enderton, *A Mathematical Introduction to Logic*, Academic Press, Inc., New York, 1972.
5. H. Hirsh, "Explanation-Based Generalization in a Logic-Programming Environment," *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Milan, Italy, August 1987, 221-227.
6. S. T. Kedar-Cabelli and L. T. McCarty, "Explanation-Based Generalization as Resolution Theorem Proving," *Proceedings of the Fourth International Workshop on Machine Learning*, Los Altos, CA, June 1987, 383-389.
7. R. E. Korf, *Learning to Solve Problems by Searching for Macro-Operators*, Morgan Kaufmann Publishers, Inc., Palo Alto, CA, 1985.
8. R. E. Korf, "Search: A Survey of Recent Results," in *Exploring Artificial Intelligence: Survey Talks from the National Conferences on Artificial Intelligence*, Howard E. Shrobe (ed.), Morgan Kaufmann, San Mateo, CA, 1988, 197-237.
9. D. W. Loveland, "Automated Theorem Proving: A Quarter-Century Review," in *Proceedings of the Special Session on Automatic Theorem Proving at the American Math. Soc. Annual Meeting*, Denver, Colorado, January 1983.
10. S. Minton, "Selectively Generalizing Plans for Problem-Solving," *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, August 1985, 596-599.
11. S. Minton, "Quantitative Results Concerning the Utility of Explanation-Based Learning," *Proceedings of the Seventh National Conference on Artificial Intelligence*, Saint Paul, Minnesota, August 1988, 564-569.
12. S. Minton, "Learning Effective Search Control Knowledge: An Explanation-Based Approach," CMU-CS-88-133, CMU Computer Science Department, Pittsburgh, PA, March 1988.
13. T. M. Mitchell, R. M. Keller and S. T. Kedar-Cabelli, "Explanation-Based Generalization — A Unifying View," *Machine Learning* 1, 1 (1986), 47-80, Kluwer Academic Publishers.

14. R. Mooney and S. Bennett, "A Domain Independent Explanation-Based Generalizer," *Proceedings of the National Conference on Artificial Intelligence*, Philadelphia, PA, August 1986.

15. R. J. Mooney, "A General Explanation-Based Learning Mechanism and its Application to Narrative Understanding," UILU-Engineering-87-2269, Coordinated Science Laboratory University of Illinois at Urbana-Champaign, Urbana IL, December 1987.

16. A. Newell and H. A. Simon, "The Logic Theorist: An Example," in *Human Problem Solving*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1972, 105-140.

17. A. Newell, J. C. Shaw and H. A. Simon, "Empirical Explorations with the Logic Theory Machine: A Case Study in Heuristics," in *Computers and Thought*, Edward Feigenbaum and Julian Feldman (ed.), Robert E. Krieger Publishing Company, Inc., Malabar, Florida, 1981, 109-133. (This article was also published in the *Proceedings of the Joint Computer Conference* (pp 218-230) in 1957. The original edition of *Computers and Thought* was published in 1963 by McGraw-Hill, Inc.)

18. P. O'Rorke, "LT Revisited: Experimental Results of Applying Explanation-Based Learning to the Logic of Principia Mathematica," *Proceedings of the Fourth International Machine Learning Workshop*, Irvine, CA, June 1987, 148-159.

19. A. E. Prieditis and J. Mostow, "PROLEARN: Towards a Prolog Interpreter that Learns," *Proceedings of the National Conference on Artificial Intelligence*, Seattle, WA, August 1987, 494-498.

20. J. C. Reynolds, "Transformational Systems and the Algebraic Structure of Atomic Formulas," in *Machine Intelligence* 5, Bernard Meltzer and Donald Michie (ed:), Edinburgh University Press/ American Elsevier, New York, 1970, 135-151.

21. P. S. Rosenbloom and J. E. Laird, "Mapping Explanation-Based Generalization onto SOAR," *Proceedings of the National Conference on Artificial Intelligence*, Philadelphia, PA, August 1986.

22. A. M. Segre, "Explanation-Based Learning of Generalized Robot Assembly Tasks," UILU-Engineering-87-2208, Coordinated Science Laboratory University of Illinois at Urbana-Champaign, Urbana IL, January 1987.

23. A. M. Segre, *Machine Learning of Robot Assembly Plans*, Kluwer Academic Publishers, Dordrecht, the Netherlands, 1988.

24. J. W. Shavlik, "Generalizing the Structure of Explanations in Explanation-Based Learning," Ph.D. thesis, Coordinated Science Laboratory University of Illinois at Urbana-Champaign, Urbana IL, January 1988.

25. E. Stefferud, "The Logic Theory Machine: A Model Heuristic Program," Memorandum RM-3731-CC, The Rand Corporation, Santa Monica, CA, June, 1963.

26. A. N. Whitehead and B. Russell, *Principia Mathematica*, Cambridge University Press, London, 1913.