April 1987 UILU-ENG-87-2223

COORDINATED SCIENCE LABORATORY
College of Engineering

BAGGER: An EBL System that Extends and Generalizes Explanations

Jude. W. Shavlik
Gerald F. DeJong

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) UILU-ENG-87-2223	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Coordinated Science Lab University of Illinois 6b. OFFICE SYMBOL (If applicable) N/A	7a. NAME OF MONITORING ORGANIZATION National Science Foundation
6c. ADDRESS (City, State, and ZIP Code) 1101 W. Springfield Avenue Urbana, IL 61801	7b. ADDRESS (City, State, and ZIP Code) 1800 G. Street N.W. Washington, D.C. 20550
8a. NAME OF FUNDING/SPONSORING ORGANIZATION National Science Foundation 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER NSF IST 85-11542
8c. ADDRESS (City, State, and ZIP Code) 1800 G. Street N.W. Washington, D.C. 20550	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)

BAGGER: An EBL System that Extends and Generalizes Explanations

12. PERSONAL AUTHOR(S) Shavlik, Jude and DeJong, Gerald

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical	FROM _____ TO _____	April 20, 1987	16

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Machine Learning, explanation-based learning, extending explanations, generalizing number, generalizing to N, learning situation calculus plans, learning blocks world plans

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This paper addresses the important issue in *explanation-based learning* of generalizing number. Most research in explanation-based learning involves relaxing constraints on the variables in an explanation, rather than generalizing the number of inference rules used. However, many concepts require generalizing the *structure* of the explanation. An explanation-based approach to the problem of generalizing to *N* is presented. The fully-implemented **BAGGER** system analyzes explanation structures and detects extendible repeated, inter-dependent applications of rules. When any are found, the explanation is extended so that an arbitrary number of repeated applications of the original rule are supported. The final structure is then generalized and a new rule produced. An important property of the extended rules is that their preconditions are expressed in terms of the initial state - they do not depend on the results of intermediate applications of the original rule. To illustrate the approach, portions of several situation calculus examples from the blocks world are analyzed. The approach presented leads to the acquisition of efficient plans that can be used to clear an object directly supporting an arbitrary number of other objects, build towers of arbitrary height, and unstack towers containing any number of blocks.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code) / 22c. OFFICE SYMBOL

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

Technical Report

UILU-ENG-87-2223

BAGGER: An EBL System that Extends and Generalizes Explanations *

Jude W. Shavlik†
Gerald F. DeJong

Artificial Intelligence Research Group
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
Urbana, IL 61801

April 1987

ABSTRACT

This paper also appears in the *Proceedings of the Sixth National Conference on Artificial Intelligence*, Seattle, WA, July 1987.

This paper addresses the important issue in "*explanation-based learning*" of generalizing number. Most research in explanation-based learning involves relaxing constraints on the variables in an explanation, rather than generalizing the number of inference rules used. However, many concepts require generalizing the *structure* of the explanation. An explanation-based approach to the problem of generalizing to N is presented. The fully-implemented **BAGGER** system analyzes explanation structures and detects extendible repeated, inter-dependent applications of rules. When any are found, the explanation is extended so that an arbitrary number of repeated applications of the original rule are supported. The final structure is then generalized and a new rule produced. An important property of the extended rules is that their preconditions are expressed in terms of the initial state - they do not depend on the results of intermediate applications of the original rule. To illustrate the approach, portions of several situation calculus examples from the blocks world are analyzed. The approach presented leads to the acquisition of efficient plans that can be used to clear an object directly supporting an arbitrary number of other objects, build towers of arbitrary height, and unstack towers containing any number of blocks.

---

* This research was partially supported by the National Science Foundation under grant NSF IST 85-11542.

† University of Illinois Cognitive Science/Artificial Intelligence Fellow.

BAGGER:

An EBL System that Extends and Generalizes Explanations

I. INTRODUCTION

Most explanation-based generalization algorithms [Fikes72, Mitchell86, Mooney86, O'Rorke87, Rosenbloom86] do not alter the structure of their explanations. No additional objects nor inference rules can be incorporated into the explanation. Instead, these algorithms generalize by converting constants in the observed example to variables with constraints. Augmentation of the explanation, which can often be required to produce the appropriate generalization, is not performed. This paper addresses the important issue in explanation-based learning of extending explanations. This can involve generalizing such things as the number of entities involved in a concept or the number of times some action is performed. It is our thesis that explanation structures that suffice to understand a specific example do not always fully reflect the underlying general concept.

Many concepts require generalizing number. For example, concepts such as momentum and energy conservation apply to arbitrary numbers of physical objects, clearing the top of a desk can require an arbitrary number of object relocations, and setting a table involves an arbitrary number of guests. In addition, there is recent psychological evidence [Ahn87] that people can generalize number on the basis of one example.

A domain-independent, explanation-based approach to the problem of "generalizing to N" is presented in [Shavlik87b]. That paper presents a theory of generalizing number. It also motivates the need for augmenting explanations, discusses other approaches to generalizing the structure of explanations [Cheng86, Prieditis86, Shavlik85, Shavlik87a] and briefly discusses how this approach handles examples from several domains. This paper describes the details of a working system based on that theory. The system analyzes and generalizes structures of the form shown in the

left-hand side of figure 1.

Observation of the repeated application of a rule or operator indicates that generalizing the number of rules in the explanation may be appropriate. The desired form of structural recursion is manifested as repeated application of an inference rule in such a manner that a portion of each consequent is used to satisfy some of the antecedents of the next application. When such a sequence is detected, it is determined how an *arbitrary* number of instantiations of this rule can be concatenated together. This indefinite-length sequence of rules is conceptually merged into the explanation, replacing the specific-length collection of rules, and a standard explanation-based algorithm produces a new rule from the augmented explanation. An additional requirement is that the preconditions for the N rule applications be fully specified in terms of the state of the world when the new rule is applied. That is, the preconditions do not depend on the results of intermediate applications of the underlying rule.

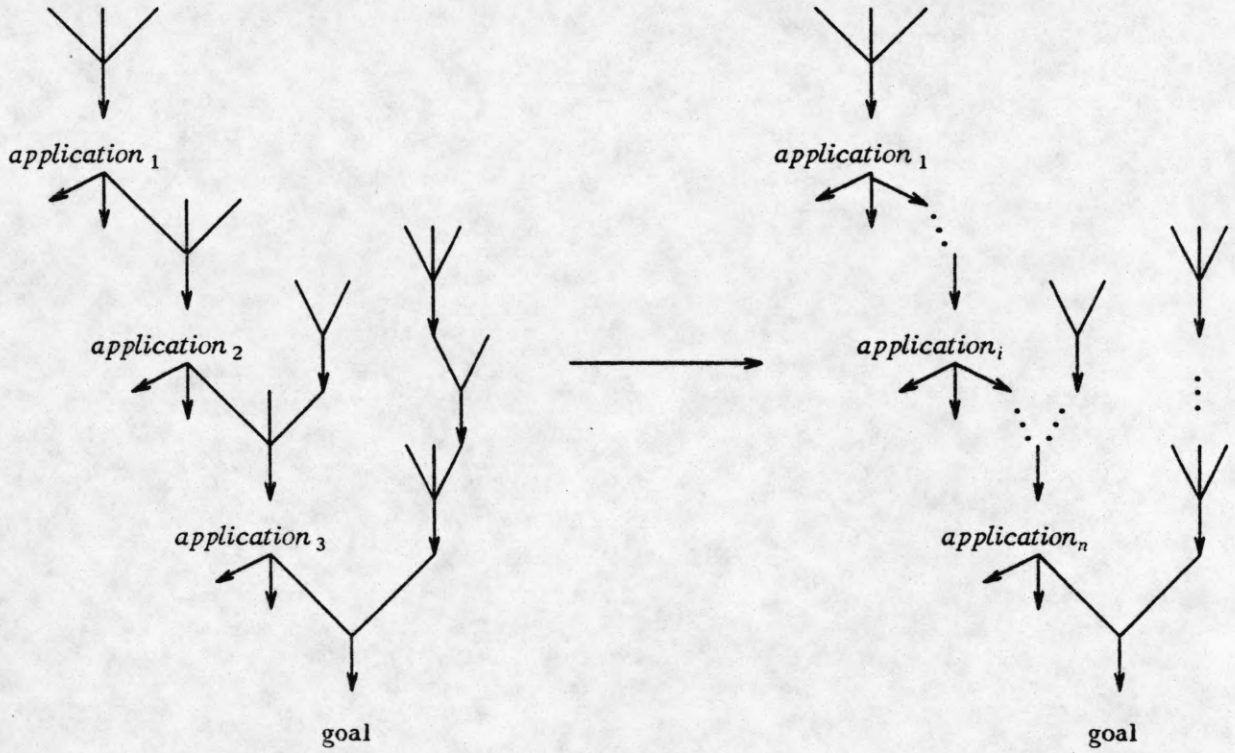

Figure 1. Augmenting the Explanation

II. THE BAGGER SYSTEM

The **BAGGER** system (Building Augmented Generalizations by Generating Extended Recurrences) analyzes predicate calculus proofs and attempts to construct concepts that involve generalizing to N. Most of the examples under study use the situation calculus to reason about actions.

One problem solution analyzed by **BAGGER** is shown in figure 2s. The goal is to clear block x. The system is provided low-level domain knowledge about blocks, including how to transfer a block from one location to another. Briefly, to move a block it must have nothing on it and there must be free space at which to place it. Additional inference rules (many of which are *frame axioms*) are used to reason about the effects of moving an object. The system produces a situation calculus proof validating the actions shown in figure 2s, in which two blocks must be moved to clear the desired block. By analyzing this example, the system acquires a general plan for clearing an arbitrary block contained in a tower of arbitrary height. The acquired plan applies, for example, to the problem of clearing block z in figure 2g. Note that there may be a different number of blocks on z than on x.

In another example, the system observes several blocks being stacked upon one another in order to satisfy the goal of having a block at a specified height. Extending the explanation of these actions produces a plan for stacking any number of blocks in order to reach any given height (provided enough blocks exist). Figure 3 illustrates this general plan.

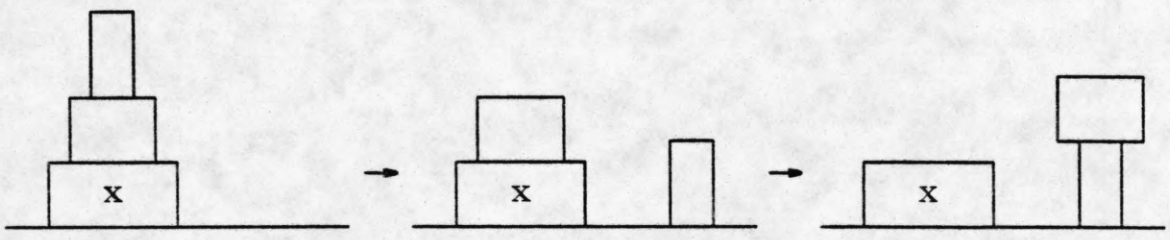

Figure 2s. Unstacking a Specific Tower

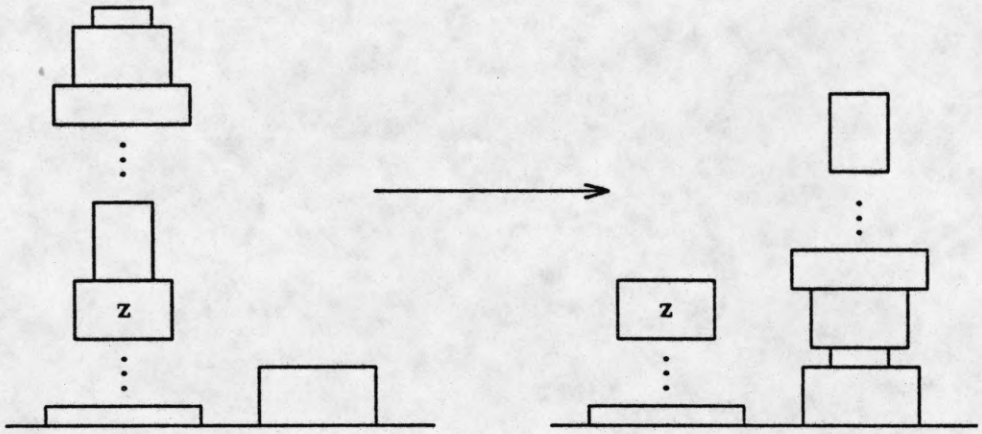

Figure 2g. A General Plan for Unstacking Towers

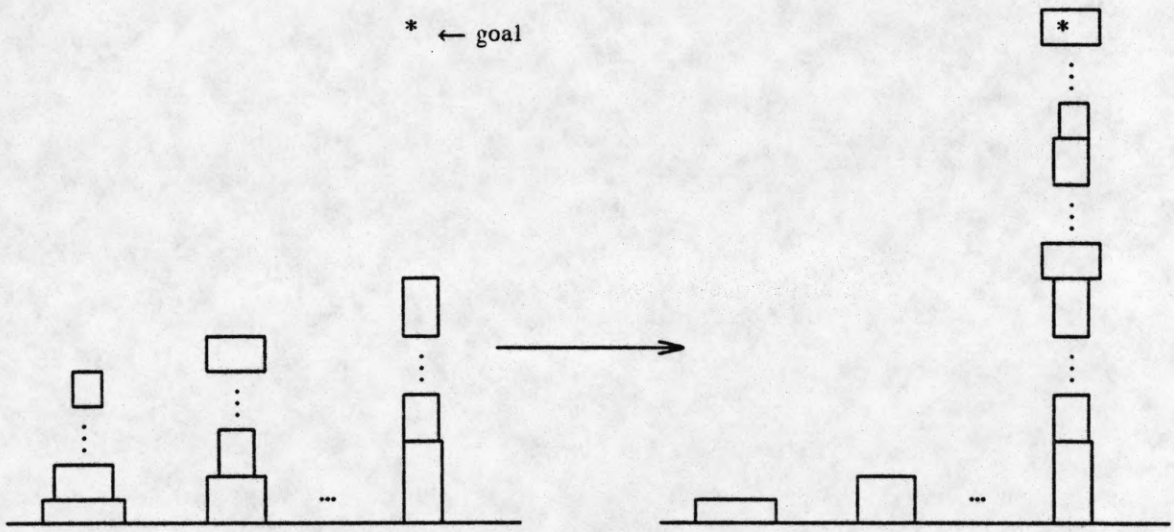

Figure 3. A General Plan for Building Towers

Unlike many other block-manipulation examples, in these examples it is *not* assumed that blocks can support only one other block. This means that moving a block does not necessarily clear its supporting block. Another concept learned by **BAGGER** is a general plan for clearing an object directly supporting any number of blocks. This plan is illustrated in figure 4.

The domain of digital circuit design has also been investigated. By observing the repeated application of DeMorgan's law to implement two cascaded *and* gates using *or* and *not* gates,

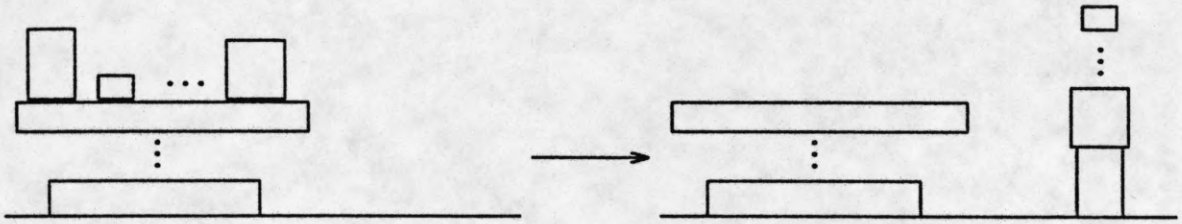

Figure 4. A General Plan for Clearing Objects

BAGGER produces a general version of DeMorgan's law which can be used to implement *N* cascaded *and* gates with *N* *or* and one *not* gate.

The next section describes how **BAGGER** constructs these general plans. Complete details on these examples, including the initial set of inference rules used, the situation calculus proofs, and the acquired inference rules, can be found in [Shavlik87c].

III. GENERALIZATION IN BAGGER

The system begins its analysis of a specific solution at the goal node. It then traces backward, looking for repeated rule applications. These repeated applications need not directly connect - there can be intervening rules. The general rule repeatedly applied is called a *focus rule*. Once a focus rule is found, **BAGGER** ascertains how an *arbitrary* number of instantiations of this rule and any intervening rules can be concatenated together (as illustrated in figure 1). This indefinite-length collection of rules is conceptually merged into the explanation, replacing the specific-length collection, and a new rule is produced from the augmented explanation.

Three classes of terms must be collected to construct the antecedents of a new rule. First, the antecedents of the initial rule application in the arbitrary length sequence of rule applications must be satisfied. To do this, the antecedents of the focus rule are used. Second, the preconditions imposed by chaining together an arbitrary number of rule applications must be collected. These are derived by analyzing instantiations of the focus rule in the sample proof. Those applications that provide enough information to be viewed as the arbitrary *ith* application produce this second class

of preconditions. Third, the preconditions from the rest of the explanation must be collected. This determines the constraints on the final applications of the focus rule.

The consequents of the new rule are produced by collecting the consequents of the last application in the chain of focus rule applications and any other terms in the goal expression. For example, the consequents for the rules illustrated by figures 2g and 4 state that in the final situation the object originally under the last object moved is clear. In the rule represented by figure 3, the consequents state that the last block moved is at the goal location.

In order to package a sequence of rule applications into a single *macro-rule*, the preconditions that must be satisfied at each of the N rule applications must be collected and combined. The preconditions for applying the resulting extended rule must be specifiable in terms of the initial state, and *not* in terms of intermediate states. This insures that, given that the necessary conditions are satisfied in the initial state, a plan represented in an extended rule will run to completion without further problem solving, regardless of the number of intervening states necessary. For example, there is no possibility that a plan will lead to moving $N-2$ blocks and then get stuck. If the preconditions for the *ith* rule application were expressed in terms of the result of the (i-1)*th* application, each of the N rule applications would have to be considered in turn to see if the preconditions of the next are satisfied. In the approach taken, extra work during generalization and a possible loss of generality are traded off for a rule whose preconditions are easier to check.

When a focus rule is concatenated an arbitrary number of times, variables need to be chosen for each rule application. A sequence of p-dimensional *vectors*, called the *rule instantiation sequence (RIS)*, is used to represent this information. The general form of the *RIS* is:

$$<v_{1,1}, \ldots, v_{1,p}>, <v_{2,1}, \ldots, v_{2,p}>, \ldots, <v_{n,1}, \ldots, v_{n,p}> \qquad (1)$$

In the unstacking example of figure 2s, $p = 3$: the current state, the object to be moved, and the object where the moved object will be placed.

Depending on the rule used, the choice of elements for this sequence may be constrained. For example, certain elements may have to possess various properties, specific relations may have to

hold among various elements, some elements may be constrained to be equal to or unequal to other elements, and some elements may be functions of other elements.

To determine the preconditions in terms of the initial state, each of the focus rule instantiations appearing in the specific proof is viewed as the *ith* application of the underlying rule. The antecedents of this rule are analyzed as to what must be true of the initial state in order that it is guaranteed the *ith* collection of antecedents are satisfied when needed. This involves analyzing the proof tree, considering how each antecedent is proved. An augmented version of a standard explanation-based algorithm [Mooney86] is used to determine which variables in this portion of the proof tree are constrained to be identical.[1] Once this is done, the variables are expressed as components of the p-dimensional vectors described above, and the system ascertains what must be true of this sequence of vectors so that each antecedent is satisfied when necessary. All antecedents of the chosen instantiation of the focus rule must be satisfied in one of the following ways for generalizing to N to be possible:

(1) The antecedent may be *situation-independent*. Terms of this type are unaffected by actions.

(2) The antecedent may be supported by a consequent of an earlier application of the focus rule. Terms of this type place inter-vector constraints on the sequence of p-dimensional vectors.

(3) The antecedent may be supported by an "unwindable rule." When this happens, the antecedent is unwound to the initial state and all of the preconditions necessary to insure that the antecedent holds when needed are collected. This process is elaborated later. It, too, may place inter-vector constraints on the sequence of p-dimensional vectors.

(4) The antecedent is supported by other terms that are satisfied in one of these ways.

Notice that antecedents are considered satisfied when they can be expressed in terms of the initial state, and *not* when a leaf of the proof tree is reached. Conceivably, to satisfy these antecedents could require a large number of inference rules. If that is the case, it may be better to

[1] The rules used in the specific proof are replaced by their general versions and the algorithm determines which unifications must hold to maintain the veracity of the proof. That is, expressions must be unified wherever a rule consequent is used to satisfy an antecedent of another rule.

trace backwards through these rules until more *operational* terms are encountered. This *operationality/generality* trade-off [Mitchell86] is a major issue in explanation-based learning, and, except where it relates directly to generalizing to N, will not be discussed further here.

A second point to notice is that not all proof subtrees will terminate in one of the above ways. If this is the case, this application of the focus rule cannot be viewed as the *ith* application.[2]

The possibility that a specific solution does not provide enough information to generalize to N is an important point in explanation-based approaches to generalizing number. A concept involving an arbitrary number of substructures may involve an arbitrary number of substantially different problems. Any specific solution will only have addressed a finite number of these sub-problems. Due to fortuitous circumstances in the example some of the potential problems may not have arisen. To generalize to N, a system must recognize all the problems that exist in the *general* concept and, by analyzing the specific solution, surmount them. Inference rules of a certain form (described later) elegantly support this task in the **BAGGER** system. They allow the system to reason backwards through an arbitrary number of actions.

A specific solution will contain several instantiations of the general rule chosen as the focus rule. Each of these applications of the rule addresses the need of satisfying the rule's antecedents, possibly in different ways. For example, when clearing an object, the blocks moved can be placed in several qualitatively different types of locations. The moved block can be placed on a table (the domain model specifies that tables are always clear), it can be placed on a block moved in a previous step, or it can be placed on a block that was originally clear.

---

[2] One solution to this problem would be to have the system search through its collection of unwindable rules and incorporate a relevant one into the proof structure. To study the limits of this paper's approach to generalizing to N, we are requiring that *all* necessary information be present in the explanation; no problem-solving search is performed during generalization. Another solution would be to assume the problem solver could overcome this problem at rule application time. This second technique, however, would eliminate the property that a learned plan will always run to completion whenever its preconditions are satisfied in the initial state.

BAGGER analyzes all applications of the general focus rule that appear in the specific example. When several instantiations of the focus rule provide sufficient information for number generalization, **BAGGER** collects the preconditions for satisfying their antecedents in a disjunction of conjunctions (one conjunct for each acceptable instantiation). Common terms are factored out of the disjunction. Knowledge about the independence of the methods of satisfying the antecedents can be used to further simplify the disjunction of conjunctions.

The learned rule illustrated in figure 2g only allows clearing towers by unstacking each block (after the first) on the previously moved one. The first transfer of figure 2s provides no information that can be used to guarantee that the block to be moved in step i is clear at that step. The acquired rule would be more general if it contained provisions for placing moved objects in any of the types of locations mentioned above. When an example of unstacking a four-block tower is presented to the system, where one intermediate block is placed on the table, a disjunctive rule is learned. In this case, the learned rule provides a choice of places to locate moved blocks. The disjunctive rule represented by figure 3 involves a choice of where to get the next block for the tower being constructed. Either a block that is clear in the initial state is used, or a block that is cleared by earlier transfers is chosen.

Figure 5 contains a portion of the proof for the unstacking example. Portions of two consecutive transfers are shown. All variables are universally quantified. Arrows run from the antecedents of a rule to its consequents. Double-headed arrows represent terms that are equated in the specific explanation. The generalization algorithm used enforces the unification of these paired terms.

There are four antecedents of a transfer. To define a transfer, the block moved (x), the object on which it is placed (y), and a state (s) must be specified, and the constraints among these variables must be satisfied. One antecedent, the one requiring a block not be placed on top of itself, is type 1 - it is *situation-independent*. The next two antecedents are type 2. Two of the consequents of the i*th* transfer are used to satisfy these antecedents of the j*th* transfer. During

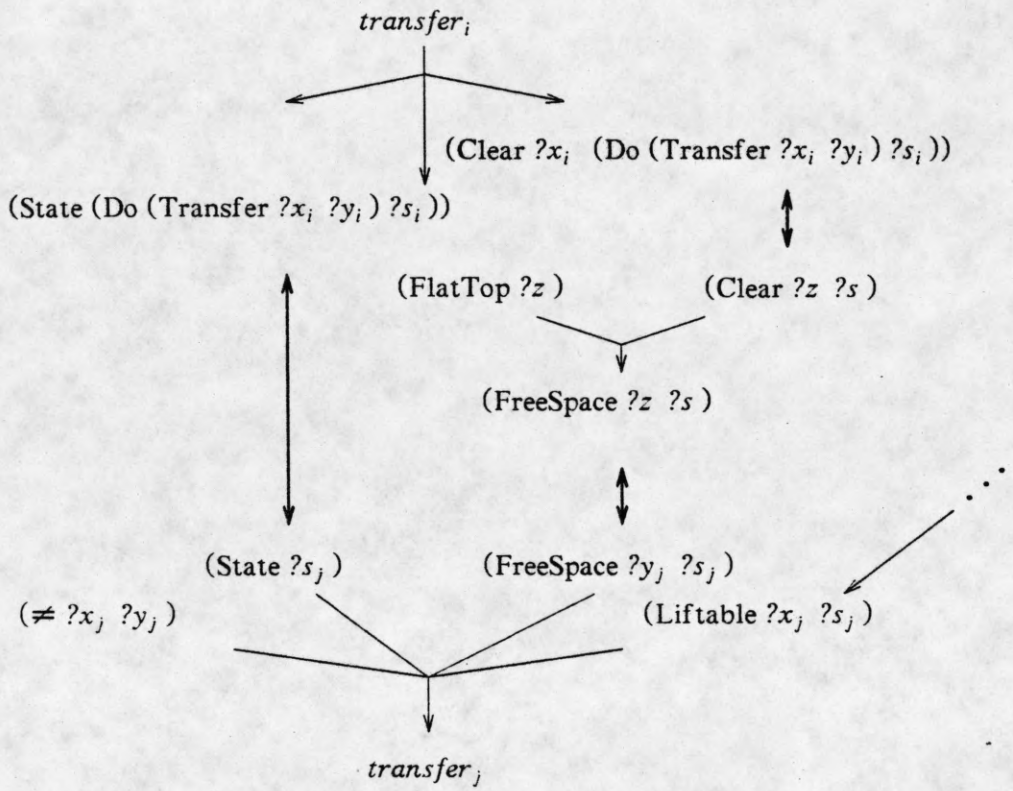

Figure 5. Satisfying Antecedents by Previous Consequents

transfer$_i$, in state s_i object x_i is moved on to object y_i. The consequents of this transfer are that a new state is produced, the object moved is clear in the new state, and x_i is on y_i in the resulting state. (The *On* term is not shown.)

The state that results from *transfer*$_i$ satisfies the second antecedent of *transfer*$_j$. Unifying these terms completely defines s_j in terms of the previous variables in the *RIS*.

Another antecedent requires that, in state s_j, there be space on object y_j to put block x_j. This antecedent is type 4. Another inference rule specifies that a clear object with a flat top has free space. The clearness of x_i after *transfer*$_i$ is used. Unifying this collection of terms leads, in addition to the redundant definition of s_j, to the equating of y_j with z and x_i. This means that the previously moved block always provides a clear spot to place the current block. No provisions need be made to insure the existence of a clear location to place intermediate blocks.

The fourth antecedent, that x_j be liftable, is also type 4. A rule (not shown) states that an object is liftable if it is a clear block. Block x_j is determined to be clear because the only object it originally supports is moved in $transfer_i$. Tracing backwards from the liftable term leads to several situation-independent terms and the term $(Supports\ ?x_j\ (?x_i)\ ?s_i)$. Fortunately, although this term contains a situation variable, it is satisfied by an "unwindable rule," and is type 3.

Equation 2 presents the form required for a rule to be unwindable. The consequent must match one of the antecedents of the rule. Hence, the rule can be applied recursively. This feature is used to "unwind" the term from the ith state to the initial state.[3] The variables in the rule are divided into three groups. First, there are the x variables. These appear unchanged in both the consequent's term **P** and the antecedent's term **P**. Second, there are the y variables which differ in the two **P**'s. Finally, there is the state variable (s). There can be additional requirements of the x and y variables (via predicate **Q**), however, these requirements cannot depend on a state variable. Only the definition of the next state can depend on the current state, as it is assumed the sequence of repeated rule applications completely determines the sequence of states.

Applying equation 2 recursively i times produces equation 3. This rule determines the requirements on the initial state so that the desired term can be guaranteed in state i. Except for the definition of the next state, none of the antecedents depend on the intermediate states. Notice

---

$$\begin{aligned}
&\mathbf{P}\,(x_{i,1}, \ldots, x_{i,\mu}, y_{i-1,1}, \ldots, y_{i-1,\nu}, s_{i-1}) \\
&\qquad \text{and} \\
&\mathbf{Q}\,(x_{i,1}, \ldots, x_{i,\mu}, y_{i-1,1}, \ldots, y_{i-1,\nu}, y_{i,1}, \ldots, y_{i,\nu}) \\
&\qquad \text{and} \\
&s_i = Do\ (x_{i,1}, \ldots, x_{i,\mu}, y_{i-1,1}, \ldots, y_{i-1,\nu}, s_{i-1}) \\
&\qquad \longrightarrow \\
&\mathbf{P}\,(x_{i,1}, \ldots, x_{i,\mu}, y_{i,1}, \ldots, y_{i,\nu}, s_i)
\end{aligned} \qquad (2)$$

---

[3] Actually, recursive rules are not always unwound to the initial state. If two (or more) of rules of this form are in a pathway, the first is unwound from state i to state t and the second is unwound from state t' to the initial state. For example, a block can be supporting another block during some number of transfers, can be cleared, can remain clear during another sequence of transfers, and finally be added to a tower.

that a collection of y variables must be specified. Any of these variables not already contained in the *RIS* are added to it.

Frame axioms often satisfy the form of equation 2. Figure 6 shows one way to satisfy the need to have a clear object when placing the i*th* block in a tower. On the left-hand side of figure 6 is a portion of the proof of a tower-building example. Block x_i is clear in state s_i because it is clear in state s_{i-1} and the block moved in $transfer_{i-1}$ is not placed upon x_i. Unwinding this rule leads to the result that block x_i will be clear in state s_i if it is clear in state s_1 and x_i is never used as the new support block in any of the intervening transfers.

$$
\begin{aligned}
&P(x_{i,1}, \ldots, x_{i,\mu}, y_{1,1}, \ldots, y_{1,\nu}, s_1) \\
&\qquad \text{and} \\
&\forall\, k \in 2, \ldots, i \\
&\qquad Q(x_{i,1}, \ldots, x_{i,\mu}, y_{k-1,1}, \ldots, y_{k-1,\nu}, y_{k,1}, \ldots, y_{k,\nu}) \\
&\qquad\qquad \text{and} \\
&\qquad s_k = Do\,(x_{i,1}, \ldots, x_{i,\mu}, y_{k-1,1}, \ldots, y_{k-1,\nu}, s_{k-1}) \\
&\longrightarrow \\
&P(x_{i,1}, \ldots, x_{i,\mu}, y_{i,1}, \ldots, y_{i,\nu}, s_i)
\end{aligned}
\qquad \textbf{(3)}
$$

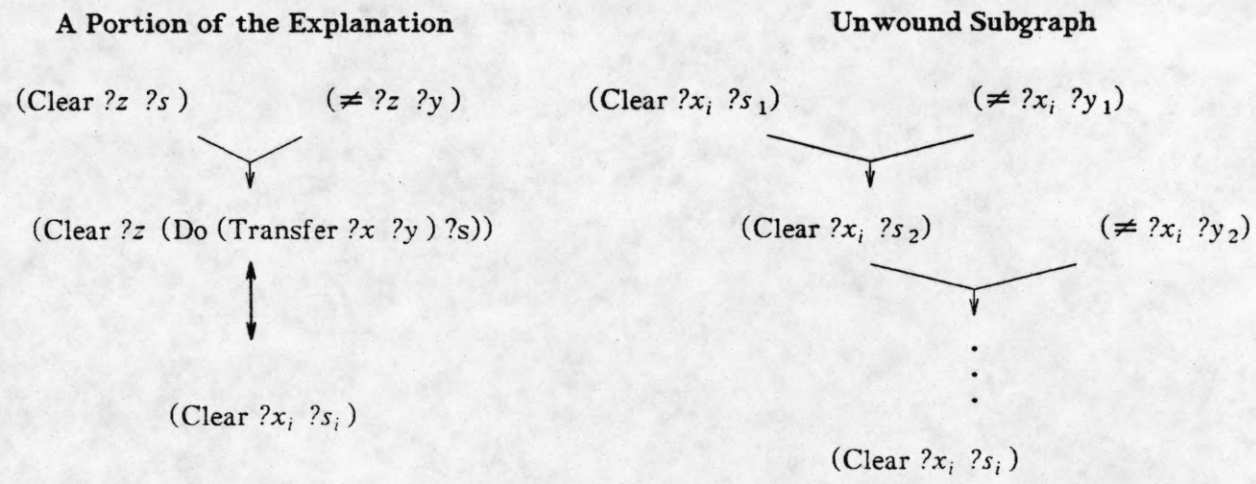

Figure 6. Unwinding a Rule

Similar reasoning is used is used in the unstacking example to insure that, up until the state in which it is moved, a block supports only one other block (and that block is moved in the previous transfer). This means that for the new rule to apply, an initial state block configuration must have successive support relations - in the initial state, the block to be moved in step i must support the one to be moved in step $i-1$ (the first block moved must be clear). As expected, a tower of blocks will be unstacked from the top downward. The new rule applies to the goal of clearing *any* object involved in the tower (including the table, provided there is another table on which to stack). Each block moved is placed on top of the object previously moved because that block is known to be clear at that time. This constraint leads to the building of a new, inverted tower. The first object moved can be placed anywhere that is clear - on the table, on another table, or on another cleared block. In the initial state, every block to be moved must be supported by an object that is supporting no other block. If a supporting object supported more than one block, it would not be clear when it is its turn to be moved, or, for the "goal" object, after the new rule is applied.

Notice that information not contained in the focus rule, but appearing in the example, is incorporated into the extended rule. In the unstacking example, additional rules are used to determine when an object becomes clear. The rule for transferring a block says nothing about the clearness of the block's original support after the block is moved. It applies to objects supporting any number of blocks. Other rules state that the supporting object is now clear if the moved block was the only one it formerly supported. The combination of these rules means that the new rule only applies to towers where each object (other than the top one) only directly supports one block. Unfortunately, while more broadly applicable than a plan for clearing three-block towers, the newly-acquired rule cannot clear objects directly supporting more than one block. The specific example did not address this multiple-support problem. Hence, the explanation-based **BAGGER** system did not solve it.

Once the repeated rule portion of the extended rule is determined, the rest of the explanation is incorporated into the final result. In the unstacking example of figure 2s, this involves the proof

that x is clear in the final state. It is accomplished in a manner similar to the way antecedents are satisfied in the repeated rule portion. The main difference is that the focus rule is now viewed as the N*th* rule application. As before, antecedents must be of one of the four specified types.

The example in figure 6 did not result in any new variables being added to the *RIS*. Other examples of unwinding do add to the variables in that sequence. Often this occurs during the process of specifying the rest of the explanation in terms of the initial state. For example, when building a tower, the *y*-coordinate of the last block added is determined by an unwindable rule. Unwinding this rule adds two terms to each vector in the *RIS*: the height of the block moved (x_i), and the *y*-coordinate of this block following the transfer.

A problem solver that applies **BAGGER**'s learned rules has been implemented. An acquired rule can be applied if its antecedents are satisfied in a state of the world. Satisfying the antecedents will produce an *RIS*. Next, N actions are executed, one for each vector. Note that the problem solver need not evaluate each action's preconditions immediately before performing it. The learned rule guarantees that they will be met.

IV. CONCLUSION

Most research in explanation-based learning involves relaxing constraints on the variables in an explanation, rather than generalizing the structure of the explanation. This paper presents an explanation-based approach to the problem of generalizing to N. To illustrate the approach, situation calculus examples from the blocks world are analyzed. The approach presented leads to efficient plans that can be used to clear an object directly supporting an arbitrary number of other objects, build towers of arbitrary height, and unstack towers containing any number of blocks. A generalized version of DeMorgan's law is also learned.

The fully-implemented **BAGGER** system analyzes explanation structures (in this case, situation calculus proofs) and detects repeated, inter-dependent applications of rules. Once a rule on which to focus attention is found, the system determines how an *arbitrary* number of instantiations of this rule can be concatenated together. This indefinite-length collection of rules is

conceptually merged into the explanation, replacing the specific-length collection of rules, and a standard explanation-based algorithm produces a new rule from the augmented explanation.

The specific example guides the extension of the focus rule into a structure representing an arbitrary number of repeated applications. Information not contained in the focus rule, but appearing in the example, is often incorporated into the extended rule. In particular, "unwindable rules" provide the guidance as to how preconditions of the *ith* application can be specified in terms of the current state.

A concept involving an arbitrary number of substructures may involve any number of substantially different problems. However, a specific solution will have necessarily only addressed a finite number of them. To properly generalize to N, a system must recognize all the problems that exist in the general concept and, by analyzing the specific solution, surmount them. If the specific solution does not provide enough information to circumvent all problems, generalization to N cannot occur because **BAGGER** is designed not to perform any problem-solving search during generalization. When a specific solution surmounts, in an extendible fashion, a sub-problem in different ways during different instantiations of the focus rule, disjunctions appear in the acquired rule.

Rules produced by **BAGGER** have the important property that their preconditions are expressed in terms of the initial state - they do not depend on the results of intermediate applications of the focus rule. If the preconditions are met, the results of multiple applications of the focus rule are immediately determined. There is no need to apply the rule successively, each time checking if the preconditions for the next application are satisfied.

Generalizing structure is an important property currently lacking in most explanation-based systems. This research contributes to the theory and practice of explanation-based learning by developing and testing methods for extending the structure of explanations during generalization.

REFERENCES

[Ahn87] W. Ahn, R. J. Mooney, W. F. Brewer and G. F. DeJong, "Schema Acquisition from One Example: Psychological Evidence for Explanation-Based Learning," Technical Report, AI Research Group, Coordinated Science Laboratory, University of Illinois, Urbana, IL, February 1987.

[Cheng86] P. Cheng and J. G. Carbonell, "The FERMI System: Inducing Iterative Macro-operators from Experience," *Proceedings of the Fifth National Conference on Artificial Intelligence*, pp. 490-495, Philadelphia, PA, August 1986.

[Fikes72] R. E. Fikes, P. E. Hart and N. J. Nilsson, "Learning and Executing Generalized Robot Plans," *Artificial Intelligence 3*, pp. 251-288, (1972).

[Mitchell86] T. M. Mitchell, R. Keller and S. Kedar-Cabelli, "Explanation-Based Generalization: A Unifying View," *Machine Learning 1*, 1, pp. 47-80, (January 1986).

[Mooney86] R. J. Mooney and S. W. Bennett, "A Domain Independent Explanation-Based Generalizer," *Proceedings of the Fifth National Conference on Artificial Intelligence*, pp. 551-555, Philadelphia, PA, August 1986.

[O'Rorke87] P. V. O'Rorke, "Explanation-Based Learning via Constraint Posting and Propagation," Ph. D. Thesis, Department of Computer Science, University of Illinois, Urbana, IL, January 1987.

[Prieditis86] A. E. Prieditis, "Discovery of Algorithms from Weak Methods," *Proceedings of the International Meeting on Advances in Learning*, pp. 37-52, Les Arcs, Switzerland, 1986.

[Rosenbloom86] P. Rosenbloom and J. Laird, "Mapping Explanation-Based Generalization into Soar," *Proceedings of the Fifth National Conference on Artificial Intelligence*, pp. 667-669, Philadelphia, PA, August 1985.

[Shavlik85] J. W. Shavlik, "Building a Computer Model of Learning Classical Mechanics," *Proceedings of the Seventh Annual Conference of the Cognitive Science Society*, pp. 351-355, Irvine, CA, August 1985.

[Shavlik87a] J. W. Shavlik and G. F. DeJong, "Analyzing Variable Cancellations to Generalize Symbolic Mathematical Calculations," *Proceedings of the Third IEEE Conference on Artificial Intelligence Applications*, pp. 100-105, Orlando, FL, February 1987.

[Shavlik87b] J. W. Shavlik and G. F. DeJong, "An Explanation-Based Approach to Generalizing Number," *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Milan, Italy, August 1987.

[Shavlik87c] J. W. Shavlik, "Augmenting and Generalizing Explanations in Explanation-Based Learning," Ph. D. Thesis, Department of Computer Science, University of Illinois, Urbana, IL, forthcoming.