AN EXPLANATION-BASED APPROACH TO GENERALIZING NUMBER *

Jude W. Shavlik*
Gerald F. DeJong

Coordinated Science Laboratory
University of Illinois
Urbana, IL 61801 USA

ABSTRACT

An approach to generalizing number in *explanation-based learning* is presented. Generalizing number can involve generalizing such things as the number of entities involved in a concept or the number of times some action is performed. This issue has been largely ignored in previous explanation-based learning research. Instead, other research has focused on changing constants into variables and determining the general constraints on those variables. In the approach presented, *generalization to N* is triggered by the detection of inference rules of a specified syntactic form. When one is found, it is extended into the rule that results from an arbitrary number of repeated applications of the original rule. If the preconditions of the extended rule are met, the results of multiple applications of the original rule are immediately determined. There is no need to apply the underlying rule successively, each time checking if the preconditions for the next application are satisfied.

I INTRODUCTION

This paper addresses the important issue in explanation-based learning of generalizing number. Generalizing number can involve generalizing such things as the number of entities involved in a concept or the number of times some action is performed. This issue has been largely ignored in previous explanation-based learning research. Instead, other research has focused on changing constants into variables and determining the general constraints on those variables.

Consider the LEAP system [l]. The system is shown an example of using NOR gates to compute the boolean AND of two OR's. It discovers that the technique generalizes to computing the boolean AND of any two inverted boolean functions. However, LEAP cannot generalize this technique to allow constructing the AND of an arbitrary number of inverted boolean functions using a multi-input NOR gate. This is the case even if LEAP's initial background knowledge were to include the general version of DeMorgan's Law and the concept of multi-input NOR gates. Generalizing the number of functions requires alteration of the original examples explanation. This generalization cannot be performed using their *goal regression* algorithm alone.

Ellmans system [2] also illustrates the need for generalizing number. From an example of a four-bit circular shift register, his system constructs a generalized design for an arbitrary four-bit permutation register. A design for an N-bit circular shift register cannot be produced. As Ellman points out. such generalization, though desirable, cannot be done using the technique of changing constants to variables.

Many other explanation-based generalization algorithms [3-6] also cannot alter the structure of their explanations. No additional objects nor inference rules can be incorporated into the explanation. These algorithms work by changing constants in the observed example to variables with constraints. Another algorithm [7] allows for the elimination of easily-reconstructed details. However, extensive augmentation of the explanation can be often required to produce the appropriate concept.

Many important concepts require generalizing number. For example, physical laws such as momentum and energy conservation apply to arbitrary numbers of objects, building blocks-world towers requires an arbitrary number of repealed stacking actions, and setting a table involves an arbitrary number of guests. This paper presents an explanation-based approach to the problem of "generalizing to *N*."

II AN APPROACH

Observations of repeated application of a rule or operator may indicate that generalizing the number of rules in the explanation may be appropriate. However, alone this is insufficient. To be conducive to number generalization there must be a certain recursive structural pattern. That is. each application must achieve preconditions for the next. For example, consider stacking blocks. The same sort of repositioning of blocks occurs repeatedly, each building on the last. We adopt the vocabulary of predicate calculus to investigate this notion of structural recursion. 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. Figure 1 illustrates the concept of repeated rule application.

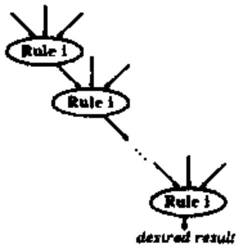

Figure 1. Repeated Rule Application

Consider inference rule 1. Notice that the consequent of this inference rule can be used to partially satisfy the antecedents of another application of the rule (via the predicate

$$
\begin{aligned}
&P(x_{i-1,1}, \ldots, x_{i-1,p}, z_{\alpha}, \ldots, z_{\beta}, y_{i-1,1}, \ldots, y_{i-1,q})\\
&\quad \textbf{and}\\
&R(x_{i-1,1}, \ldots, x_{i-1,p}, x_{i,1}, \ldots, x_{i,p}, z_{\delta}, \ldots, z_{\eta})\\
&\quad \textbf{and}\\
&S(x_{i-1,1}, \ldots, x_{i-1,p}, z_{\gamma}, \ldots, z_{\kappa})\\
&\quad \textbf{and}\\
&T(x_{i,1}, \ldots, x_{i,p}, z_{\lambda}, \ldots, z_{\mu})\\
&\quad \textbf{and}\\
&\forall j \in 1, \ldots, q \quad y_{i,j} = f_j(x_{i,1}, \ldots, x_{i,p}, x_{i-1,1}, \ldots, x_{i-1,p},\\
&\qquad\qquad z_{\nu}, \ldots, z_{\omega}, y_{i-1,1}, \ldots, y_{i-1,q})\\
&\quad \longrightarrow\\
&P(x_{i,1}, \ldots, x_{i,p}, z_{\alpha}, \ldots, z_{\beta}, y_{i,1}, \ldots, y_{i,q}) \qquad (1)
\end{aligned}
$$

* This research was partially supported by the National Science Foundation under grant NSF 1ST 85 11542.

[1] University of Illinois Cognitive Science/Artificial Intelligence Fellow.

P), as illustrated in figure 1. The antecedents of this rule involve three qualitatively different types of variables. (The differences between the three types will become clear when the extended version of this rule is described.) Predicate P involves all three types of variables, while predicate R specifies a necessary relationship between the (i-1)th and ith collection of x 's.* Predicate S constrains the (i-l)th collection of x's and predicate T constrains the uh collection. Lastly, the uh collection of y s are partially defined by terms in the (i-1)th application.

Chaining together several applications of rule 1 produces rule le.

$$
\begin{array}{l}
P(x_{0,1}, \ldots, x_{0,p}, z_{\alpha}, \ldots, z_{\beta}, y_{0,1}, \ldots, y_{0,q}) \\
\quad \textbf{and} \\
\forall j \in 1, \ldots, n \\
\qquad R(x_{j-1,1}, \ldots, x_{j-1,p}, x_{j,1}, \ldots, x_{j,p}, z_{\delta}, \ldots, z_{\eta}) \\
\qquad\quad \textbf{and} \\
\qquad S(x_{j-1,1}, \ldots, x_{j-1,p}, z_{\gamma}, \ldots, z_{\kappa}) \\
\qquad\quad \textbf{and} \\
\qquad T(x_{j,1}, \ldots, x_{j,p}, z_{\lambda}, \ldots, z_{\mu}) \\
\quad \textbf{and} \\
\forall j \in 1, \ldots, q \quad y_{n,j} = F_j(n) \\
\quad \longrightarrow \\
P(x_{n,1}, \ldots, x_{n,p}, z_{\alpha}, \ldots, z_{\beta}, y_{n,1}, \ldots, y_{n,q}) \qquad\qquad \textbf{(1e)} \\
\quad \textbf{where} \\
F_j(1) \equiv f_j(x_{0,1}, \ldots, x_{0,p}, x_{1,1}, \ldots, x_{1,p}, z_{\nu}, \ldots, z_{\omega}, y_{0,1}, \ldots, y_{0,q}) \\
F_j(i) \equiv f_j(x_{i-1,1}, \ldots, x_{i-1,p}, x_{i,1}, \ldots, x_{i,p}, z_{\nu}, \ldots, z_{\omega}, \\
\qquad\qquad F_1(i-1), \ldots, F_q(i-1)) \qquad \text{for } i > 1
\end{array}
$$

In this extended inference rule all references to the y_{ij} for i >0 are eliminated and the z0 terms remain unchanged from one application of the original rule to the next. Hence, besides the initial situation, all that needs to be specified for an arbitrary number of applications of rule 1 is a sequence of x, j terms. The predicates R. S. and T place constraints on possible sequences of x s. In particular, the predicates S and T constrain which terms can be members of the sequence, while predicate R specifies the relationship between successive members of the sequence.

The general form of a sequence is shown below. It consists of an ordered collection of p-ary vectors.

$$
<x_{0,1}, \ldots, x_{0,p}>, <x_{1,1}, \ldots, x_{1,p}>, \ldots, <x_{n,1}, \ldots, x_{n,p}>
$$

III SOME EXAMPLES

Two simple examples that concretely illustrate the above procedure are presented below.

A. Blocks World

Imagine an explanation-based learning system that deals with the blocks world. Assume that in the course of its operation this system has to determine the position of the top of block which is resting on a table. Also assume that in the course of doing this it produces an explanation structure that can be transformed into rule 2 below. (Figure 2 illustrates this rule.) This rule is in a form that matches rule 1. Rule 2e is the extended form of rule 2. (In these rules, all terms beginning with a ? are universally quantified variables.)

```
(AND (Yposition ?object_{i-1} ?y_{i-1})
     (On ?object_i ?object_{i-1})
     (= ?y_i (+ (height ?object_i) ?y_{i-1})))
  ⟶
(Yposition ?object_i ?y_i)                        (2)
```

' Although not done here for reasons of clarity, the approach presented can be extended to situations where there are relations among the (i-k)w through the ith collections.

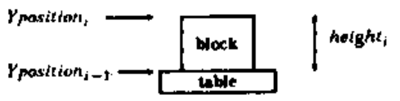

Figure 2. Determining the Y-Position of a Block

In rule 2e the sequence is made explicit. For all consecutive pairs of sequence elements, the first must be on the second. In addition, the function +' is introduced. This recursive function has two arguments: a sequence of numbers and a "seed" number. It maps these into a single number - the sum of all the numbers. The function derived-sequence takes a sequence and a unary function and maps them into another sequence - the one which results from applying the function to each member of the original sequence.

```
(Sequence ?s) and (InitialElement ?object_0 ?s)
   and
(Yposition ?object_0 ?y_0) and (FinalElement ?object_n ?s)
   and
((Member ?j ?s) and (Member ?k ?s) and (Successor ?j ?k ?s)
  → (On ?k ?j))
   and
(= ?y_n (+' (derived-sequence (sub-sequence ?s 1 n) height) ?y_0))
   ⟶
(Yposition ?object_n ?y_n)                                  (2e)
```

The extended rule can be used to find the y -position of a block supported by several other blocks when the y -positions of the underlying blocks are not directly known. All that needs to be known is the heights of each block and the y-position of the table top (or the y-position of one intervening block). In this case the extended rule is obtained from an example that involved no repeated actions nor structures.

An important task for a system that generalizes number is to loosen the preconditions of a rule as far as possible while still maintaining the veracity of the rule. Also, as much guidance as possible should be provided so that a problem solver can most easily determine when a rule is both applicable and appropriate.

Imagine using rule 2e in a backward-chaining fashion. If a problem solver is to find the y -position of an object it needs to choose a sequence that satisfies the specified constraints. This task is simplied if the preconditions are specified in terms of sets or bags'. rather than sequences. In this case, there is no need to test each permutation of a given collection of elements. If a bag satisfies the rules preconditions, then any sequence derived from that bag suffices. Other derivable properties, such as the cardinality of the bag or the length of the sequence, might also usefully constrain a problem solver.

One case where it is easy to specify the preconditions in bag terms occurs when there are no inter-element constraints (i.e., predicate R in rule 1 is not used). If an inter-element predicate does appear in the preconditions, the properties of that predicate determine how loosely the preconditions can be expressed. For example, if R is an equivalence relation (that is. R is reflexive, symmetric, and transitive), then the elements must form an equivalence class, a property that is order independent.

In the above example, R is (the atransitive version of) On. In bag terms, rule 2e requires a collection of elements where. (i) except for one element {object0,). every element is on one and only one other element, (ii) except for possibly one element (?object„). every element has one and only one element on it, and (iii)lhe sum of the heights of all elements other than object „ plus the y -position of object $_0$ equals ?y_n . If a bag with these properties is obtained, the necessary sequence can easily be constructed.

A hag (or multi set) is an unordered collection of elements in which an element can occur more than once.

B. Digital Circuit Design

The second example involves a simplified version of circuit design. Figure 3 shows two flip-flops. When the clock is pulsed, the input of a flip-flop is passed to its output, provided its *select* line is on. Assume that from observing the connection of two flip-flops, a learning system derives rule 3. That is, it determines

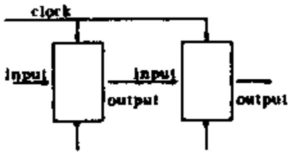

Figure 3. Two Flip-Flops

```
(Output $?FF_{i-1}$ $?value$ $?time_{i-1}$)
and
(Connected $?FF_{i-1}$ $?FF_i$)
and
(Selected $?FF_i$)
and
(= $?time_i$ (increment $?time_{i-1}$))
→
(Output $?FF_i$ $?value$ $?time_i$)   (3)
```

that if two flip-flops are connected together and the select line of the first is on, after the next clock pulse the current output of the first flip-flop becomes the second's output. Rule 3 is also of the form of rule 1. The extended form of rule 3 is rule 3e. (Notice that all of the select lines are required to be on.) In constructing this rule, knowledge about repeated incrementing is used to define $?y_n$ in terms of the number of flip-flops connected. The new rule can be used to build such things as a delay line and can lead to the concept of an N-bit shift register.

```
(Sequence $?s$) and (InitialElement $?FF_0$ $?s$)
    and
(Output $?FF_0$ $?value$ $?time_0$) and (FinalElement $?FF_n$ $?s$)
    and
((Member $?j$ $?s$) and (Member $?k$ $?s$) and (Successor $?j$ $?k$ $?s$)
  → (Connected $?k$ $?j$))
    and
((Member $?j$ $?s$) → (Selected $?j$))
    and
(= $?time_n$ (+ (length $?s$) $?time_0$))
    →
(Output $?FF_n$ $?value$ $?time_n$)                (3e)
```

IV RELATED WORK

Several other approaches to generalizing number have been recently proposed. Prieditis [8] has outlined plans for a system which learns macro-operators representing sequences of repeated STRIPS-like operators. While we agree very much with the spirit of Prieditis' work, we feel that STRIPS-like operators impose unwarranted restrictions. For one thing, our use of predicate calculus allows generalization of repeated structure and repeated actions in a uniform manner. In the FERMI system [9], cyclic patterns are recognized using empirical methods and the detected repeated pattern is generalized using explanation-based learning techniques. However, unlike the techniques presented in this paper, the rules acquired by FERMI are not guaranteed to always work. After a significant amount of work, a learned problem-solving strategy may terminate unsuccessfully. A third system. Physics 101 [10. 11], differs from the above two approaches in that the need for augmenting explanation structures is motivated by an analytic justification of an example's solution and general domain knowledge. In a sample problem, information about number, localized in a single physics formula, leads to a global restructuring of a specific solution's explanation. However. Physics 101 takes advantage of properties of mathematical calculations. To be a broad solution of the generalization to N problem. non mathematically-based domains must also be handled.

V CONCLUSION

Most research in explanation-based learning involves relaxing constraints on the entities in a situation, rather than generalizing the number of entities themselves. This paper presents an approach to generalizing to N in explanation-based learning. Generalization is triggered by the detection of rules of a certain syntactic form (i.e.. rule 1), and a technique for extending these rules is presented. The extended versions are modified so that a problem solver can efficiently apply them. This involves attempting to expression the preconditions for these rules in terms of order-independent data structures such as sets and bags. If the preconditions of the extended rule are met. the results of multiple applications of the underlying 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.

A first computer implementation of the ideas presented here has been developed. The BAGGER system [12] analyzes explanation structures and attempts to construct inference rules of the form of rule 1. When one is found, it is extended into the rule that results from an arbitrary number of repeated applications of the original rule. This system is being tested on problems from various domains, including the blocks world, digital circuit design, and mathematical problem solving.

REFERENCES

1. T. M. Miichell, S. Mahadevan and L. 1. Steinberg, "LEAP: A Learning Apprentice for VLSI Design," *Proceedings of the Ninth International Joint Conference on Artificial Intelligence,* Los Angeles, CA, August 1985, pp. 573-580.
2. T. Ellman, "Generalizing Logic Circuit Designs by Analyzing Proofs of Correctness," *Proceedings of the Ninth International Joint Conference on Artificial Intelligence,* Los Angeles, CA, August 1985, pp. 643-646.
3. R. F. Fikes, P. F. Hart and N. J. Nilsson, "Learning and Executing Generalized Robot Plans," *Artificial Intelligence 3,* (1972), pp. 251-288.
4. T. M. Mitchell, R. Keller and S. Kedar-Cabelli, "Explanation-Based Generalization: A Unifying View," *Machine learning I,* 1 (January 1986), pp. 47-80.
5. R. J. Mooney and S. \V. Bennett, "A Domain Independent Explanation-Based Gencralizer," *Proceedings of the National Conference on Artificial Intelligence,* Philadelphia, PA, August 1986, pp. 551-555.
6. P. Rosenbloom and J. Laird, "Mapping. Explanation-Based Generalization into Soar," *Proceedings of the National Conference' on Artificial Intelligence,* Philadelphia, PA, Augusl 1986, pp. 561-567.
7. G. F. Dejong and R. J. Mooney. "Explanation-Based Learning: An Alternative View," *Machine Learning I, 2* (April 1986), pp. 145-176.
8. A. E. Prieditis, "Discovery of Algorithms from Weak Methods," *Proceedings of the International Meeting on Advances in learning,* Les Arcs, Switzerland, 1986, pp. 37-52.
9. P. Cheng and J. G. Carbonell, "The FERMI System. Inducing Iterative Macro-operators from Experience," *Proceedings of the National Conference on Artificial Intelligence,* Philadelphia, PA, August 1986, pp. 490-495.
10. J. W. Shavlik and G. F. Dejong, "Building a Computer Model of Learning Classical Mechanics," *Proceedings of the Seventh Annual Conference of the Cognitive Science Society,* Irvine, CA, August 1985. pp. 351-355.
11. 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,* Orlando, FL, February 1987.
12. J. W . Shavlik and G. F. Dejong, "BAGGER: An EBL System that Extends and Generalizes Explanations," *Proceedings of the National Conference on Artificial Intelligence,* Seattle, WA, July 1987.