June 1986 UILU-ENG-86-2216

COORDINATED SCIENCE LABORATORY
College of Engineering

A DOMAIN INDEPENDENT EXPLANATION-BASED GENERALIZER

Raymond J. Mooney
Scott W. Bennett

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: nclassified	1b. RESTRICTIVE MARKINGS: None
2a. SECURITY CLASSIFICATION AUTHORITY: N/A	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A	
4. PERFORMING ORGANIZATION REPORT NUMBER(S): UILU-ENG-86-2216	5. MONITORING ORGANIZATION REPORT NUMBER(S): N/A
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. Srpingfield Ave. Urbana, IL 61801	7b. ADDRESS *(City, State and ZIP Code)*: 1800 G Street Washington, D.C. 20550
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: National Science Foundation; 8b. OFFICE SYMBOL *(If applicable)*: N/A	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: N00014-86-K-0309
8c. ADDRESS *(City, State and ZIP Code)*: 1800 G Street Washington, D.C. 20550	10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.
N/A	N/A	N/A	N/A

11. TITLE *(Include Security Classification)* A Domain Independent Explanation-Based Generalizer

12. PERSONAL AUTHOR(S) Raymond J. Mooney and Scott W. Bennett

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT *(Yr., Mo., Day)*	15. PAGE COUNT
Technical	FROM ______ TO ______	June 1986	22

16. SUPPLEMENTARY NOTATION N/A

17. COSATI CODES

FIELD	GROUP	SUB. GR.

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)* MACROP STRIPS EGGS GENESIS equalities, patterns

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

A domain independent technique for generalizing a broad class of explanations is described. This method is compared and contrasted with other approaches to generalizing explanations, including an abstract version of the algorithm used in the STRIPS system and the EBG technique recently developed by Mitchell, Keller, and Kedar-Cabelli. We have tested this generalization technique on a number of examples in different domains, and present detailed descriptions of these.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION: Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER *(Include Area Code)* / 22c. OFFICE SYMBOL: NONE

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

Technical Report
UILU-ENG-86-2216

A Domain Independent Explanation-Based Generalizer

Raymond J. Mooney
Scott W. Bennett

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

June 1986

ABSTRACT

This report is an extended and revised version of a paper appearing in the *Proceedings of the National Conference on Artificial Intelligence*, August 1986.

A domain independent technique for generalizing a broad class of explanations is described. This method is compared and contrasted with other approaches to generalizing explanations, including an abstract version of the algorithm used in the STRIPS system and the EBG technique recently developed by Mitchell, Keller, and Kedar-Cabelli. We have tested this generalization technique on a number of examples in different domains, and present detailed descriptions of these.

A Domain Independent Explanation-Based Generalizer

1. Introduction

If one considers many of the different Explanation-Based Learning (EBL) systems under construction [Mitchell83, Mitchell85, Mooney85, O'Rorke84, Winston83], certain commonalities become evident in the generalization phase of the learning process. Such systems work by first constructing an explanation for an example being processed. Next, this explanation is generalized. This latter process can be characterized in a domain independent way.

Recent work on the generalization phase of EBL is underway at Rutgers [Mitchell86] and here at the University of Illinois [DeJong86]. In this paper, we present a technique called Explanation Generalization using Global Substitution (EGGS) which we believe provides a natural way for conducting this generalization. This method is quite similar to both the EBG technique introduced in [Mitchell86] and to the MACROP learning process used in STRIPS [Fikes72]. Consequently, the generalization technique used in STRIPS and the regression technique used in EBG are outlined and contrasted with EGGS. Lastly, a number of examples to which EGGS has been applied are presented with their resulting generalizations.

2. Explanations, Explanation Structures, and Generalized Explanations

In different domains, various types of explanations are appropriate. In [Mitchell86], an explanation is defined as a logical proof which demonstrates how an example meets a set of sufficient conditions defining a particular concept. This type of explanation is very appropriate for learning classic concept definitions, such as learning a structural specification of a cup, an example introduced in [Winston83] and discussed in [Mitchell86]. However, when learning general plans in a problem solving domain (as in STRIPS [Fikes72] or GENESIS [Mooney85]), it is more appropriate to consider an explanation to be a set of causally connected actions which demonstrate how a goal state was achieved.

Consequently, in this paper, we will take a very broad definition of the term *explanation* and consider it to be a connected set of *units*, where a unit is set of related patterns. A unit for an inference rule has patterns for its antecedents and its consequent, while a unit for an action or operator has patterns for effects, preconditions, etc.. Connections between units in an explanation, such as the consequent of one inference rule matching the antecedent of another, or the effect of one action matching the precondition of another action, are represented as *equalities* between patterns in the respective units. The *goal* is a distinguished pattern in the explanation which represents the final conclusion in an inference chain or the final desired state in a plan. In correspondence with the terminology in [Mitchell86], an *explanation structure* is defined as an explanation with each instantiated unit replaced by its general patterns (with unique variables). For instance, consider the cup example from [Winston83]. Given facts like the following:

Light(Obj1)
PartOf(Handle1,Obj1)
Handle(Handle1)

and the following inference rules:

Stable(?x) $\wedge$ Liftable(?x) $\wedge$ OpenVessel(?x) $\rightarrow$ Cup(?x)
Bottom(?y) $\wedge$ PartOf(?y,?x) $\wedge$ Flat(?y) $\rightarrow$ Stable(?x)
Graspable(?x) $\wedge$ Light(?x) $\rightarrow$ Liftable(?x)
Handle(?y) $\wedge$ PartOf(?y,?x) $\rightarrow$ Graspable(?x)
Concavity(?y) $\wedge$ PartOf(?y,?x) $\wedge$ UpwardPointing(?y) $\rightarrow$ OpenVessel(?x)

a proof tree (explanation) can be constructed for the goal Cup(Obj1) as shown in Figure 1. The explanation structure for this proof is shown in Figure 2. The edges between patterns in a unit are assumed to be directed so that an explanation forms a directed acyclic graph (DAG). These directed edges define certain patterns in a unit as the *support* for other patterns in the unit. For example, the support for the consequent of an inference rule is its antecedents and the support for the effects of an action are its preconditions.

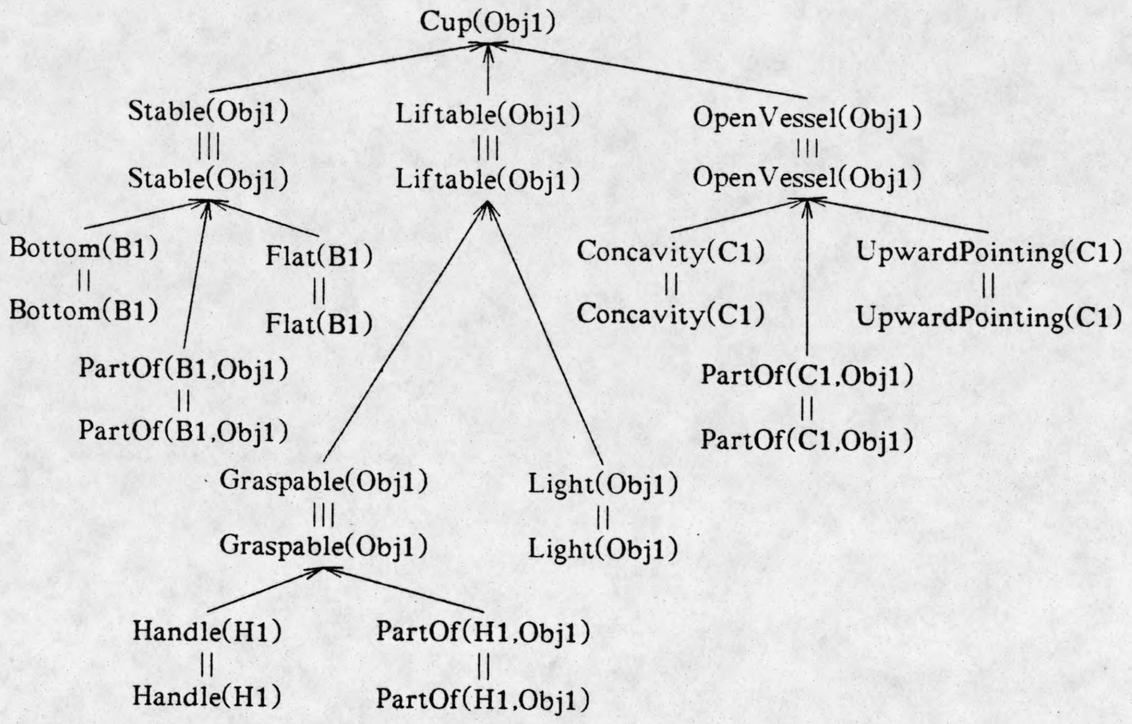


Figure 1: Explanation for Cup(Obj1)
Triple edges indicate equalities between unit patterns.
Double edges indicate equalities to initial assertions.

The task of explanation-based generalization is to take an explanation and its associated explanation structure and generate a *generalized explanation*, which is the most general version which still maintains the structural validity of the original explanation. This means that substitutions must be applied to the patterns in the explanation structure so that it is constrained in such a way that equated patterns unify directly without requiring any substitutions. The generalized explanation of the cup example is shown in figure 3. This generalized explanation can then be used to extract the following general definition of a cup:

Bottom(?y1) ∧ PartOf(?y1,?x1) ∧ Flat(?y1) ∧ Handle(?y2) ∧ PartOf(?y2,?x1)
∧ Light(?x1) ∧ Concavity(?y3) ∧ PartOf(?y3,?x1) ∧ UpwardPointing(?y3) → Cup(?x1)

In problem solving domains, the generalized explanation represents a general plan schema or MACROP for achieving a particular class of goals.

3. Explanation Generalizing Algorithms

Several algorithms have been developed for generalizing various types of explanations. The STRIPS system [Fikes72] incorporated a method for generalizing blocks-world plans into MACROPS. The EBG method [Mitchell86] uses a modified version of goal-regression [Waldinger77] to generalize proofs of concept membership. Concurrently with Mitchell *et. al*'s development of EBG, we developed a method [DeJong86] (which we now call EGGS) which generalizes the broad class of explanations defined in the previous section. However, the general techniques used by the

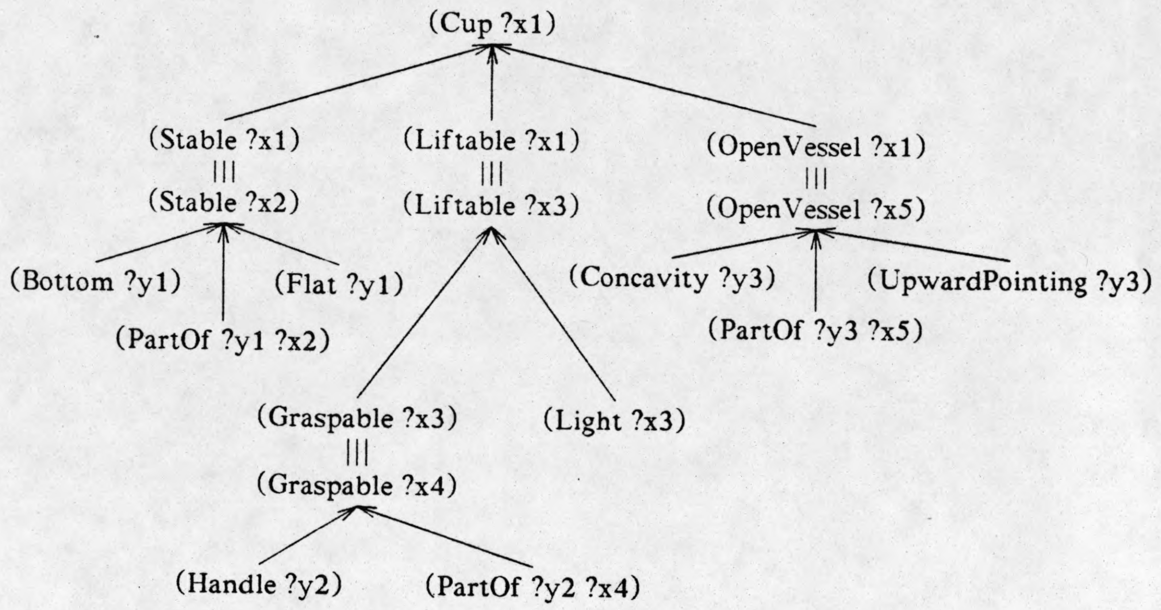


Figure 2: Explanation Structure for the Cup Example
Triple edges indicate equalities between unit patterns.

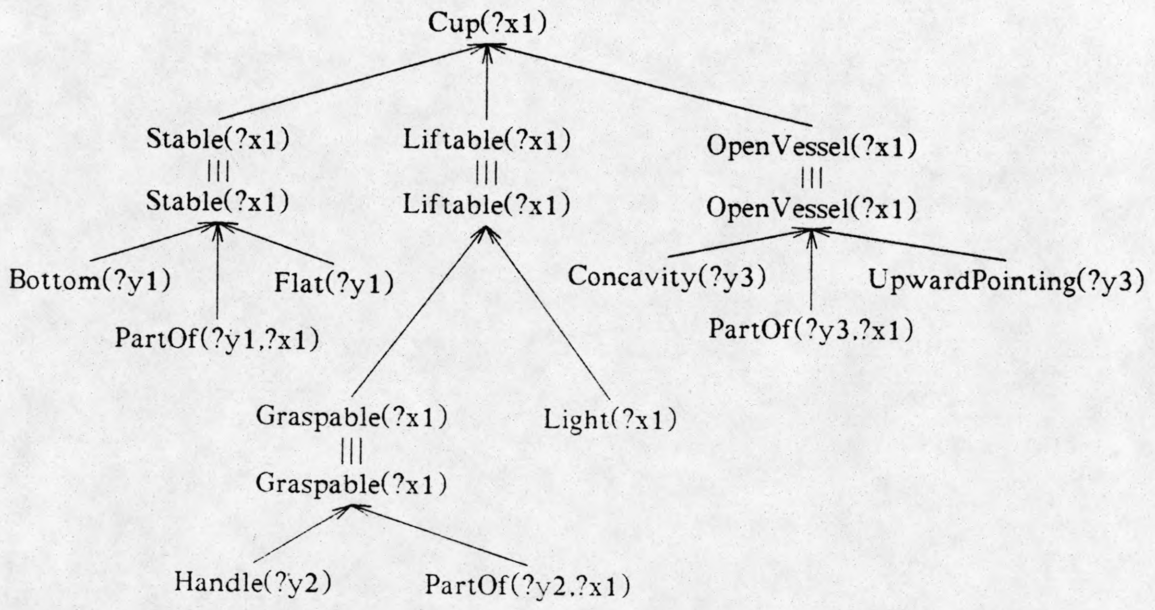


Figure 3: Generalized Explanation for the Cup Example
Triple edges indicate equalities between unit patterns.

other two methods can be abstracted to apply to the class of explanations defined above. Consequently, this section is devoted to presenting and comparing algorithmic descriptions of all three methods as applied to this class of explanations. All of the algorithms rely on unification pattern matching and we will use the unification notation described in [Nilsson80]. The MGU (most general unifier) of two patterns is defined as the most general substitution which will make the two patterns identical. The expression $p\theta$, where p is a pattern and θ is a substitution, denotes the pattern resulting from applying θ to p. The expression $\gamma\theta$, where both γ and θ are substitutions, denotes the substitution resulting from the composition of γ and θ.

3.1. STRIPS MACROP Learning

The first work on generalizing explanations was the learning of robot plans in STRIPS [Fikes72]. STRIPS worked in a "blocks world" domain and after its problem solving component generated a plan for achieving a particular state, it generalized the plan into a problem solving schema (a MACROP) which could be used to efficiently solve similar problems in the future. Work on the STRIPS system was the first to point out that generalizing a causally connected set of actions or inferences could not be done by simply replacing each constant by a variable. This method happens to work on the cup example given above. The proper generalized explanation can be obtained by replacing Obj1 by ?x1, B1 by ?y1, H1 by ?y2, and C1 by ?y3. However, in general, such a simplistic approach can result in a structure which is either more general or more specific than what is actually supported by the system's domain knowledge.

The following examples are given in [Fikes72] to illustrate that simply replacing constants with variables can result in improper generalizations. The following operators are used in these examples:

GoThru(?d,?r1,?r2):	Go through door ?d from room ?r1 to room ?r2.
PushThru(?b,?d,?r1,?r2):	Push box ?b through door ?d from room ?r1 to room ?r2.
SpecialPush(?b):	Specific operator for pushing box ?b from Room2 to Room1.

Given the plan:

```
GoThru(Door1,Room1,Room2)
SpecialPush(Box1)
```

simply replacing constants by variables results in the plan:

```
GoThru(?d,?r1,?r2)
SpecialPush(?b)
```

This plan is too general since SpecialPush is only applicable when starting in Room2, so having a variable ?r2 as the destination of the GoThru is too general and ?r2 should be replaced by Room2. Given the plan:

```
GoThru(Door1,Room1,Room2)
PushThru(Box1,Door1,Room2,Room1)
```

simply replacing constants by variables results in the plan:

```
GoThru(?d,?r1,?r2)
PushThru(?b,?d,?r2,?r1)
```

This plan is too specific since the operators themselves do not demand that the room in which the robot begins (r1) be the same room into which the box is pushed. The correct generalization [illegible]:

```
GoThru(?d,?r1,?r2)
PushThru(?b,?d,?r2,?r3)
```

The exact process STRIPS uses to avoid these problems and correctly generalize an example is dependent on its particular representations and inferencing techniques; however, the basic technique is easily captured using the representations discussed in section 2. How STRIPS problems are represented with interconnecting units will be clarified with an example in section 4.8. However, assuming they are represented in this fashion, a description of the explanation generalizing algorithm is shown in Table 1. It should be noted that the generalization process in STRIPS was constructed specifically for generalizing robot plans represented in triangle tables and using resolution to prove preconditions. There was no attempt to present a general learning method based on generalizing explanations in any domain. However, the algorithm in Table 1 is a straight-forward generalization of the basic process used in STRIPS. The basic technique is to unify each pair of matching patterns in the explanation structure and apply each resulting substitution to all of the patterns in the explanation structure. After all of the unifications and substitutions have been made, the result is the generalized explanation since each pattern has been replaced by the most general pattern which allows all of the equality matches in the explanation to be satisfied.

3.2. EBG

Mitchell, Keller, and Kedar-Cabelli [Mitchell86] outline a technique for generalizing a logical proof that a particular example satisfies the definition of a concept. An example of such a proof is the one in Figure 1 explaining how a particular object satisfies the functional requirements of a cup. Unlike the STRIPS MACROP learning method, EBG is meant to be a general method for learning by generalizing explanations of why an example is a member of a concept. The EBG algorithm is based on regression [Waldinger77] and involves back-propagating constraints from the goal pattern through the explanation back to the leaves of the explanation structure. However, as initially pointed out in [DeJong86], it fails to obtain the appropriately generalized goal pattern in certain situations (see the example in section 4.7). As indicated in [DeJong86] and as originally specified in [Mahadevan85], the proper goal concept can be obtained by starting with the generalized antecedents obtained from regression and rederiving the proof. This propagates constraints forward from the generalized antecedents to the final generalized goal concept. Hence, the correct EBG algorithm involves both a back-propagate and a forward-propagate step as shown in the abstract algorithm in Table 2. The original specification of this corrected version of EBG (outlined in [DeJong86] and given explicitly in [Mooney86]) involved first back-propagation and then forward-propagation. Problems with particular examples (see section 4.7) have since indicated the need for performing the forward-propagation step first to collect constraints at the root of the explanation before constraints are back-propagated to the leaves. In any case, as with the STRIPS algorithm, the eventual result of the EBG algorithm is the generalized explanation since each pattern has been replaced by the most general pattern which allows all of the equality matches in the explanation to be satisfied.

3.3. EGGS

Finally, there is the EGGS algorithm which we developed to generalize explanations of the very abstract form defined and used in this paper. The algorithm is quite similar to the abstract STRIPS algorithm and is shown in Table 3. The difference between EGGS and the abstract STRIPS algorithm is that instead of applying the substitutions throughout the explanation at each step, all the substitutions are composed into one substitution γ. After all the unifications have been done,

```
for each equality between patterns $p_i$ and $p_j$ in the explanation structure do
    let $\theta$ be the MGU of $p_i$ and $p_j$
    for each pattern $p_k$ in the explanation structure do
        replace $p_k$ with $p_k\theta$
```

Table 1: STRIPS Explanation Generalization Algorithm

```
let g be the goal pattern in the explanation structure
ForwardPropagate(g)
BackPropagate(g)

procedure ForwardPropagate(p)
    for each pattern $p_i$ supporting p do
        if $p_i$ is equated to some pattern
            then
                let e be the pattern equated to $p_i$
                ForwardPropagate(e)
                let $\theta$ be the MGU of $p_i$ and e
                replace $p_i$ with $p_i\theta$
                replace p with $p\theta$

procedure BackPropagate(p)
    for each pattern $p_i$ supporting p do
        if $p_i$ is equated to some pattern
            then
                let e be the pattern equated to $p_i$
                let $\theta$ be the MGU of e and $p_i$
                replace e with $e\theta$
                for each pattern $p_j$ supporting e do
                    replace $p_j$ with $p_j\theta$
                BackPropagate(e)
```

Table 2: EBG Explanation Generalization Algorithm

```
let $\gamma$ be the null substitution {}
for each equality between patterns $p_i$ and $p_j$ in the explanation structure do
    let $\theta$ be the MGU of $p_i\gamma$ and $p_j\gamma$
    let $\gamma$ be $\gamma\theta$
for each pattern $p_k$ in the explanation structure do
    replace $p_k$ with $p_k\gamma$
```

Table 3: EGGS Explanation Generalization Algorithm

one sweep through the explanation applying the accumulated substitution γ results in the generalized explanation. Table 4 demonstrates this technique as applied to the cup example above. It shows how γ changes as it is composed with the substitutions resulting from each equality. Applying the final substitution γ to the explanation structure in Figure 2 results in the generalized explanation in Figure 3.

3.4. Comparison of Explanation Generalization Algorithms

It is reasonably clear that all of the above algorithms compute the same desired generalized explanation. They all perform a set of unifications and substitutions to constrain the explanation structure into one in which which equated patterns unify directly without requiring any substitutions. The difference between them lies in the number of unifications and substitutions required and the order in which they are performed.

Table 4: EGGS Applied To the Cup Example

Equality	θ	γ
Stable(?x1) ≡ Stable(?x2)	{?x1/?x2}	{?x1/?x2}
Liftable(?x1) ≡ Liftable(?x3)	{?x1/?x3}	{?x1/?x2, ?x1/?x3}
Graspable(?x3) ≡ Graspable(?x4)	{?x1/?x4}	{?x1/?x2, ?x1/?x3, ?x1/?x4}
OpenVessel(?x1) ≡ OpenVessel(?x5)	{?x1/?x5}	{?x1/?x2, ?x1/?x3, ?x1/?x4, ?x1/?x5}

Assuming there are e equalities and p patterns in an explanation ($p > 2e$), the STRIPS method requires e unifications each resulting in p applications of a substitution (i.e. ep substitutions). The EBG method does a unification for each equality in both the back-propagating and forward-propagating steps for a total of $2e$ unifications. Two substitutions are done for each unification in ForwardPropagate ($2e$ substitutions) and each pattern in the explanation structure requires a substitution in BackPropagate (p substitutions) for a total of $2e+p$ substitutions. Finally, EGGS requires e unifications and $3e$ substitutions (two substitutions with γ for each equality plus one substitution per equality for the composition) to build the global substitution (γ) and p substitutions to apply γ to the explanation structure. Consequently, EGGS requires e unifications and $3e+p$ substitutions.

Therefore, in the limit, both EBG and EGGS perform fewer substitutions than STRIPS. However, EBG requires twice as many unifications as either STRIPS or EBG, and EGGS requires e more substitutions than EBG. However, these figures are not very informative with regard to overall computational complexity since the complexity of each unification or substitution depends on the nature of the patterns and substitutions involved. Consequently, these figures are not absolute complexity results, but only rough indications of overall complexity.

As described in [O'Rorke85], generalizing explanations can be viewed as a process of posting and propagating constraints. Neither the abstract STRIPS algorithm nor EGGS impose any order on the posting of constraints (equalities between patterns) and both simultaneously propagate constraints in all directions. EBG, on the other hand, imposes an unnecessary ordering on the posting and propagation of constraints. First it propagates constraints forward to the goal and then backwards to the leaves of the explanation. We believe this adds undo complexity to the generalizing algorithm as is obvious from comparing the algorithmic descriptions in Tables 1-3.

A final advantage which we believe the EGGS method has is that it can be integrated with the explanation building process. Every time a rule or action is added to an explanation under construction, the substitution resulting from unifying its patterns with other patterns in the existing explanation structure can be composed with the current global substitution γ to obtain the updated γ. When the explanation is fully constructed, the generalized form can be immediately obtained by simply applying the accumulated global substitution to the final explanation structure. Although the process of integrating the explanation construction and generalization processes does not result in increased efficiency, we believe it has a certain aesthetic appeal. Neither the STRIPS nor EBG methods allow this sort of integration of processes since they require the existence of the complete explanation before generalizing.

3.5. Pruning Explanation Structures

Often, the explanation structure for a particular example is too specific to support a reasonably useful generalization. In these cases, the *operationality criterion* [Mitchell86] is met by nodes higher in the explanation tree than the leaves and it is advisable to *prune* units from the explanation structure which are more specific than required for operationality. If this pruning is done prior to generalization (using any of the above generalizing algorithms), it will result in a more abstract generalized explanation which is applicable to a broader range of examples. For example, if the rule for inferring Graspable is removed from the explanation structure shown in Figure 2, the following more general (but less operational) definition of Cup is acquired:

> Bottom(?y1) ∧ PartOf(?y1,?x1) ∧ Flat(?y1) ∧ Graspable(?x1) ∧ Light(?x1)
> ∧ Concavity(?y3) ∧ PartOf(?y3,?x1) ∧ UpwardPointing(?y3) → Cup(?x1)

More appropriate examples of pruning will be given later in the paper.

If EGGS generalization is integrated with explanation construction as described above, the global substitution (γ) is computed before the explanation is complete and hence before appropriate pruning can normally be determined. Therefore, pruning requires that the unifications for the removed equalities be retracted from γ before applying it to the final explanation structure. Retracting unifications from a substitution is an involved process. Consequently, if pruning is required, integrating explanation construction and generalization results in undoing previous computation and is probably not a good idea.

4. Application of EGGS to Several Domains

The EGGS technique, which has been fully implemented, has generalized explanations in several domains. This set of examples currently includes narrative understanding [Mooney85], generating physical descriptions of objects from functional information [Winston83], designing logic circuits [Mahadevan85, Mitchell85], solving integration problems [Mitchell83, Mitchell86], solving equations [Silver83], proving theorems in mathematical logic [O'Rorke84], the Safe-To-Stack problem from [Mitchell86], the suicide example from [DeJong86], and STRIPS robot planning [Fikes72].

The Cup example was discussed in detail in section 2. In this section, we describe the application of EGGS to the remaining examples.

4.1. LEAP example

The LEAP system [Mitchell85] is a learning apprentice in VLSI design which observes the behavior of a circuit designer. It attempts to learn in an explanation-based fashion from circuit examples it observes. Given the task of implementing a circuit which computes the logical function:

> (a ∨ b) ∧ (c ∨ d)

a circuit designer creates a circuit consisting of three NOR gates like that shown in Figure 4. The system attempts to verify that the given circuit actually computes the desired function. The explanation proving that the circuit computes a function which is equivalent to the desired function is shown in Figure 5. Since equated patterns are always identical in specific and generalized explanations, only one of each pair of equated patterns will be shown in this and future figures. In this example, the domain knowledge available to the system includes:

> *Remove Double Negation:* Equiv(?x,?y) → Equiv(¬(¬(?x)),?y)

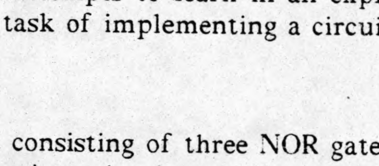


Figure 4: LEAP Training Example

$$\begin{tikzcd}
& \mathrm{Equiv}(\neg(\neg(a\vee b)\vee\neg(c\vee d)),(a\vee b)\wedge(c\vee d)) & \\
& \mathrm{Equiv}(\neg(\neg(a\vee b))\wedge\neg(\neg(c\vee d)),(a\vee b)\wedge(c\vee d)) \arrow[u] & \\
\mathrm{Equiv}(\neg(\neg(a\vee b)),a\vee b) \arrow[ur] & & \mathrm{Equiv}(\neg(\neg(c\vee d)),c\vee d) \arrow[ul] \\
\mathrm{Equiv}(a\vee b,a\vee b) \arrow[u] & & \mathrm{Equiv}(c\vee d,c\vee d) \arrow[u]
\end{tikzcd}$$

Figure 5: LEAP Example -- Specific Explanation

DeMorgan's Law: $\mathrm{Equiv}((\neg ?x\wedge\neg ?y),?a) \rightarrow \mathrm{Equiv}(\neg(?x\vee ?y),?a)$
Substitution of Equals: $\mathrm{Equiv}(?x,?a) \wedge \mathrm{Equiv}(?y,?b) \rightarrow \mathrm{Equiv}(?x\wedge ?y,?a\wedge ?b)$
Definition Of Equivalence: $\mathrm{Equiv}(?x,?x)$

The generalized form of this proof is shown in Figure 6. The general fact learned from this example is:

$\mathrm{Equiv}(\neg(\neg ?a\vee\neg ?b),\ ?a\wedge ?b)$

Had the constants a, b, c and d simply been replaced by variables, the result would have been overly specific. As a result of the explanation-based approach, the resulting generalization is not sensitive to the fact that the first stage of the circuit involved two NOR gates. For example, the generalization would support using two NAND gates and a NOR gate to AND four inputs together.

4.2. MA Example

Another explanation-based learning system in the domain of logic is MA [O'Rorke84] which learns proof schemata from sample natural deduction proofs. When the system cannot complete a proof for a particular theorem, a teacher steps in and completes the proof. MA then generalizes the teacher's proof in an explanation-based manner to generate a proof schema which can be used to

$$\begin{tikzcd}
& \mathrm{Equiv}(\neg(\neg(?a1)\vee\neg(?b1)),?a1\wedge ?b1) & \\
& \mathrm{Equiv}(\neg(\neg(?a1))\wedge\neg(\neg(?b1)),?a1\wedge ?b1) \arrow[u] & \\
\mathrm{Equiv}(\neg(\neg(?a1)),?a1) \arrow[ur] & & \mathrm{Equiv}(\neg(\neg(?b1)),?b1) \arrow[ul] \\
\mathrm{Equiv}(?a1,?a1) \arrow[u] & & \mathrm{Equiv}(?b1,?b1) \arrow[u]
\end{tikzcd}$$

Figure 6: LEAP Example -- Generalized Explanation

solve future problems. Consider the example discussed in [O'Rorke84] of proving a particular case of the law of excluded middle: Deducible(NIL,P∨¬P) (i.e. P∨¬P can be deduced from the empty set of assumptions). A natural deduction proof of this example is shown in Figure 7. The following rules of natural deduction [Manna74] are employed in this proof:

Assumption Axiom: Deducible((?x . ?y),?x)
Or Introduction: Deducible(?x,?y) → Deducible(?x,?y∨?z)
Or Introduction: Deducible(?x,?y) → Deducible(?x,?z∨?y)
Elimination Of Assumption: Deducible((?x . ?y),?z) ∧ Deducible((¬?x . ?y),?z) → Deducible(?y,?z)

Deducible(?x,?y) means that the wff ?y is deducible from the list of assumptions (wffs) ?x. LISP *dot notation* is used to represent lists of assumptions. The generalized proof EGGS generates for this example is shown in Figure 8. From a specific instance of proving P∨¬P from no assumptions, a general proof is learned for proving the disjunction of any wff and its negation from any set of assumptions.

4.3. LEX2 Example

LEX2 [Mitchell83] is an explanation-based learning system in the domain of integration problem solving. Unlike the other systems addressed in this paper which learn new inference rules or plans, LEX learns search control knowledge. Specifically, it learns heuristics for when to apply particular integration operators by determining, in an explanation-based manner, why the application of a particular operator led to the solution of a specific problem. However, MACROPS for solving particular types of integration problems can also be learned in an explanation-based manner, and the conditions which allowed each operator to lead to the solution are easily extracted from the general MACROP. For example, consider solving the following problem discussed in [Mitchell86]: $\int 7x^2dx$. This problem can be formulated as logical inference using the following inference rules:

Integer(?r) ∧ ?r≠-1 → Solution($\int ?x^{?r}d?x,\ ?x^{?r+1}/(?r+1)$)

$$\begin{tikzcd}
 & \mathbf{Deducible(NIL,P\lor\neg P)} & \\
\mathbf{Deducible((P),P\lor\neg P)} \arrow[ur] & & \mathbf{Deducible((\neg P),P\lor\neg P)} \arrow[ul] \\
\mathbf{Deducible((P),P)} \arrow[u] & & \mathbf{Deducible((\neg P),\neg P)} \arrow[u]
\end{tikzcd}$$

Figure 7: MA Example -- Specific Explanation

$$\begin{tikzcd}
 & \mathrm{Deducible(?y17,?y16\lor\neg?y16)} & \\
\mathrm{Deducible((?y16\ .\ ?y17),?y16\lor\neg?y16)} \arrow[ur] & & \mathrm{Deducible((\neg?y16\ .\ ?y17),?y16\lor\neg?y16)} \arrow[ul] \\
\mathrm{Deducible((?y16\ .\ ?y17),?y16)} \arrow[u] & & \mathrm{Deducible((\neg?y16\ .\ ?y17),\neg?y16)} \arrow[u]
\end{tikzcd}$$

Figure 8: MA Example -- Generalized Explanation

$\text{Constant}(?c) \wedge \text{Solution}(\int ?f d?x, ?z) \rightarrow \text{Solution}(\int ?c{*}?f d?x, ?c{*}?z)$

The expression Solution(?x,?y) means ?y is the solution to the problem ?x. The proof for the solution to the given problem is shown in Figure 9, and the generalized proof is shown in Figure 10. The general rule learned from this example is:

$$\text{Constant}(?c1) \wedge \text{Integer}(?r1) \wedge ?r1 \neq -1 \rightarrow \text{Solution}(\int ?c1{*}?x1^{?r1} d?x1, ?c1{*}(?x1^{?r1+1}/(?r1+1)))$$

This rule can then be used to immediately solve similar problems like: $\int 5x^3 dx$. The conditions which allowed the first operator to lead to a solution are also implicitly represented in this rule. The problem must match the form in the conclusion: $\int ?c1{*}?x1^{?r1} d?x1$ and satisfy the antecedents of the rule: Constant(?c1), Integer(?r1), and $?r1 \neq -1$. In a problem with more steps, the "success conditions" for other operators in the solution can be obtaining by *pruning* (i.e. removing from the explanation structure) the steps for all the previous operators in the solution prior to generalizing the explanation structure.

Unlike the process for learning integration problem solving described in [Mitchell86], this method obtains both a general rule and heuristic conditions from a single explanation using a single generalizing (constraint propagating) mechanism. The method described in [Mitchell86] learns only search heuristics and uses the EBG method as well as a separate regression step using rules of the domain theory to regress operators through problem states. However, the above method requires a specialized technique for extracting the search heuristics from the generalized proof.

4.4. Rewrite Rule Versions of LEAP and LEX2 Examples

Like many mathematical problems, the LEAP and LEX2 examples can be more concisely represented and solved using *rewrite rules* [Bundy83] instead of logical deduction. Chains of

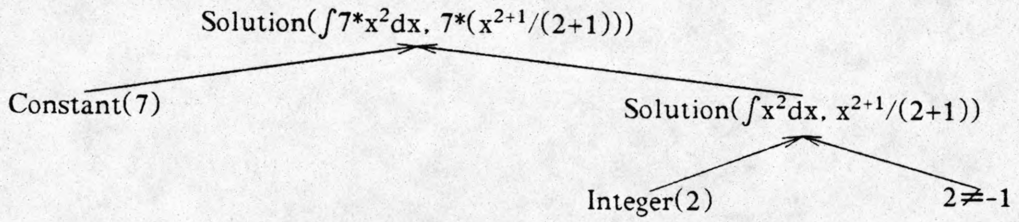

$\text{Solution}(\int 7{*}x^2 dx, 7{*}(x^{2+1}/(2+1)))$
Constant(7)
$\text{Solution}(\int x^2 dx, x^{2+1}/(2+1))$
Integer(2)
$2 \neq -1$

Figure 9: LEX2 Example -- Specific Explanation

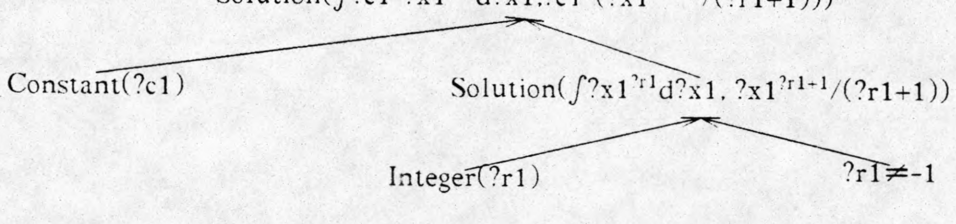

$\text{Solution}(\int ?c1{*}?x1^{?r1} d?x1, ?c1{*}(?x1^{?r1-1}/(?r1+1)))$
Constant(?c1)
$\text{Solution}(\int ?x1^{?r1} d?x1, ?x1^{?r1+1}/(?r1+1))$
Integer(?r1)
$?r1 \neq -1$

Figure 10: LEX2 Example -- Generalized Explanation

rewrite rule applications can be represented as explanations in which each rewritten expression is a pattern in the explanation. For example, consider solving the LEAP example above by rewriting the expression $\neg(\neg(a\vee b)\vee\neg(c\vee d))$ to $(a\vee b)\wedge(c\vee d)$ using the following rewrite rules:

$\neg\neg ?x \rightarrow ?x$
$\neg(?x \vee ?y) \rightarrow \neg ?x \wedge \neg ?y$

The explanation, explanation structure, and generalized explanation for this example are shown in Figures 11, 12, and 13 respectively. Notice that if a rule rewrites only a subterm of an expression, dummy variables are added to the pattern in the explanation structure to fill out the expression so it can be unified with the previous expression. The new rewrite rule which is learned from the LEAP example is:

$\neg(\neg ?f4 \vee \neg ?x3) \rightarrow ?f4 \wedge ?x3$

The LEX2 example of solving the integral: $\int 7x^2$ can also be done using rewrite rules. The following two rules are used in this example:

$\int(?c*?f)d?x \rightarrow ?c*\int ?f?d?x$ if $\neg$OccursIn(?x,?c)
$\int ?x^{?r}d?x \rightarrow ?x^{?r+1}/(?r+1)$ if $?r \neq -1 \wedge$ Integer(?r)

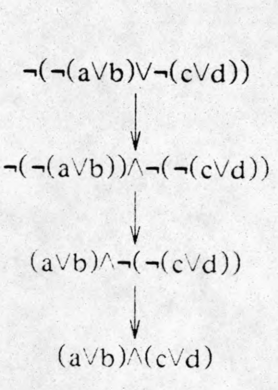
The *if* conditions attached to the rewrite rules are additional conditions which must be met for the rule to apply. The specific and generalized explanations for the rewrite version of this problem are shown in Figures 14 and 15 respectively. The new rewrite rule learned from this version is:

$\int(?f3*(?x2^{?r1}))d?x2 \rightarrow ?f3*((?x2^{?r1+1})/(?r1+1))$ if $\neg$OccursIn(?x2,?f3) $\wedge$ $?r1 \neq -1$ $\wedge$ Integer(?r1)

4.5. Equation Solving Example

EGGS has also been tested in the domain of solving algebraic equations. The LP system [Silver83] learns from single equation solutions in an explanation-based manner; however, it learns very abstract "strategies" which are not guaranteed to solve problems to which they can be applied. EGGS learns more specific rewrite rules which are guaranteed to solve a particular class of problems.

The background knowledge used in this domain is also most conveniently given in terms of rewrite rules. The following set of rules can be used to solve the equation: $\log_e(x+1) + \log_e(x-1) = c$ (from [Bundy83]).



$$\begin{tikzcd}
\neg(\neg(a\vee b)\vee\neg(c\vee d)) \arrow[d] \\
\neg(\neg(a\vee b))\wedge\neg(\neg(c\vee d)) \arrow[d] \\
(a\vee b)\wedge\neg(\neg(c\vee d)) \arrow[d] \\
(a\vee b)\wedge(c\vee d)
\end{tikzcd}$$

Figure 11: LEAP Rewrite -- Specific Explanation

$$\begin{tikzcd}
\text{(not (or ?x1 ?y1))} \arrow[d] \\
\text{(and (not ?x1) (not ?y1))} \arrow[d, phantom, "|||"] \\
\text{(?f1 (not (not ?x2)) . ?f2)} \arrow[d] \\
\text{(?f1 ?x2 . ?f2)} \arrow[d, phantom, "|||"] \\
\text{(?f3 ?f4 (not (not ?x3)) . ?f5)} \arrow[d] \\
\text{(?f3 ?f4 ?x3 . ?f5)}
\end{tikzcd}$$

Figure 12: LEAP Rewrite -- Explanation Structure

$$\begin{tikzcd}
\neg(\neg(?f4)\vee\neg(?x3)) \arrow[d] \\
\neg(\neg(?f4))\wedge\neg(\neg(?x3)) \arrow[d] \\
?f4\wedge\neg(\neg(?x3)) \arrow[d] \\
?f4\wedge ?x3
\end{tikzcd}$$

Figure 13: LEAP Rewrite -- Generalized Explanation

$$\begin{tikzcd}
\int(7*(x^2))dx \arrow[dr] & & \neg\mathrm{OccursIn}(x,7) \arrow[dl] & & \\
& 7*\int(x^2)dx \arrow[dr] & & 2\neq -1 \arrow[dl] & \mathrm{Integer}(2) \arrow[dll] \\
& & 7*((x^{2+1})/(2+1)) & &
\end{tikzcd}$$

Figure 14: LEX2 Rewrite Example – Specific Explanation

$$\begin{tikzcd}
\int(?f3*(?x2^{?r1}))d?x2 \arrow[dr] & & \neg\mathrm{OccursIn}(?x2,?f3) \arrow[dl] & & \\
& ?f3*\int(?x2^{?r1})d?x2 \arrow[dr] & & ?r1\neq -1 \arrow[dl] & \mathrm{Integer}(?r1) \arrow[dll] \\
& & ?f3*((?x2^{?r1+1})/(?r1+1)) & &
\end{tikzcd}$$

Figure 15: LEX2 Rewrite Example – Generalized Explanation

$\log_{?b}?u=?v \rightarrow ?u=?b^{?v}$ if $\mathrm{OccursIn}(x,?u) \wedge \neg\mathrm{OccursIn}(x,?v)$
$?u-?v=?z \rightarrow ?u=?z+?v$ if $\mathrm{OccursIn}(x,?u) \wedge \neg\mathrm{OccursIn}(x,?v)$
$?u^2=?v \rightarrow ?u=\mathrm{sqrt}(?v)$ if $\mathrm{OccursIn}(x,?u) \wedge \neg\mathrm{OccursIn}(x,?v))$
$(?u+?v)*(?u-?v) \rightarrow (?u^2)-(?v^2)$ if $\mathrm{OccursIn}(x,?u)$
$\log_{?w}?u+\log_{?w}?v \rightarrow \log_{?w}(?u*?v)$ if $\mathrm{OccursIn}(x,?u) \wedge \mathrm{OccursIn}(x,?v) \wedge \neg\mathrm{OccursIn}(x,?w)$

The OccursIn conditions constitute control information which assure that the application of a rule is a step towards the solution. This control information forms a part of the explanation and is learned along with the body of the new compiled rewrite rule. These conditions are proven deductively using the following inference rules:

$\mathrm{OccursIn}(?x,?x)$
$\mathrm{OccursIn}(?x,?y) \rightarrow \mathrm{OccursIn}(?x,(?y\ .\ ?z))$
$\mathrm{OccursIn}(?x,?z) \rightarrow \mathrm{OccursIn}(?x,(?y\ .\ ?z))$
$\mathrm{Atom}(?y) \wedge \neg\mathrm{Eq}(?x,?y) \rightarrow \neg\mathrm{OccursIn}(?x,?y)$
$\neg\mathrm{OccursIn}(?x,?y) \wedge \neg\mathrm{OccursIn}(?x,?z) \rightarrow \neg\mathrm{OccursIn}(?x,(?y\ .\ ?z))$

Unfortunately, the graph of the explanation for the solution to this problem is too large to be included in the paper; however, below are traces of the rewrite rules applied in the specific and general cases:

Specific Solution:
$\log_e(x+1)+\log_e(x-1)=c$
$\log_e((x+1)*(x-1))=c$
$(x+1)*(x-1)=e^c$
$(x^2)-(1^2)=e^c$

$x^2=(e^c)+(1^2)$
$x=\mathrm{sqrt}((e^c)+(1^2))$

General solution:
$\log_{?b1}(x+?v3)+\log_{?b1}(x-?v3)=?v2$
$\log_{?b1}((x+?v3)*(x-?v3))=?v2$
$(x+?v3)*(x-?v3)=?b1^{?v2}$
$(x^2)-(?v3^2)=?b1^{?v2}$
$x^2=(?b1^{?v2})+(?v3^2)$
$x=\mathrm{sqrt}((?b1^{?v2})+(?v3^2))$

New Rule:
$\log_{?b1}(x+?v3)+\log_{?b1}(x-?v3)=?v2 \rightarrow x=\mathrm{sqrt}((?b1^{?v2})+(?v3^2))$
if $\mathrm{Atom}(?b1) \wedge \neg\mathrm{Eq}(x,?b1) \wedge \mathrm{Atom}(?v2) \wedge \neg\mathrm{Eq}(x,?v2) \wedge \mathrm{Atom}(?v3) \wedge \neg\mathrm{Eq}(x,?v3)$

However, maintaining the *exact* structure of the proofs for the conditions has required ?b1, ?v2, and ?v3 to be atoms as they were in the example. That is, in the example, we proved that ?b1 (the base of the logarithm) did not contain x because it was an Atom which was not Eq to x. By *pruning* the explanation structure a little, we can gain generality without sacrificing much operationality. That is, by removing the final steps used to prove the conditions before generalizing we can obtain the following more general rule:

$\log_{?b1}(?u3+?v3)+\log_{?b1}(?u3-?v3)=?v2 \rightarrow ?u3=\mathrm{sqrt}((?b1^{?v2})+(?v3^2))$
if $\neg\mathrm{OccursIn}(x,?b1) \wedge \neg\mathrm{OccursIn}(x,?v2) \wedge \neg\mathrm{OccursIn}(x,?v3) \wedge \mathrm{OccursIn}(x,?u3)$

This rule is at the appropriate level of generality to cover a relatively large class of problems. For example, this rule can now be used to solve the following example in one step.

$\log_{3*e}(x+2*a) + \log_{3*e}(x-2*a)=c+d$
$x=\mathrm{sqrt}((3*e)^{c+d}+(2*a)^2)$

4.6. Safe-To-Stack Example

The Safe-To-Stack example was introduced in [Mitchell86] to illustrate the EBG explanation-based generalization method. It involves learning an operational rule for when it is safe to stack something on an endtable. The specific example involves the following facts:

Isa(Obj1,Box)
Isa(Obj2,Endtable)
Color(Obj1,Red)
Volume(Obj1,1)
Density(obj1,.1)

The following domain rules are used to prove that Obj1 can be safely stacked on Obj2:

$\mathrm{Volume}(?x,?v) \wedge \mathrm{Density}(?x,?d) \rightarrow \mathrm{Weight}(?x,?v*?d)$
$\mathrm{Weight}(?x,?w) \wedge \mathrm{Weight}(?y,?u) \wedge \mathrm{Less}(?w,?u) \rightarrow \mathrm{Lighter}(?x,?y)$
$\mathrm{Isa}(?x,\mathrm{Endtable}) \rightarrow \mathrm{Weight}(?x,5)$

The specific proof for this problem is shown in Figure 16 and the generalized proof is shown in Figure 17. The general rule learned from this example is:

$\mathrm{Volume}(?x1,?v1) \wedge \mathrm{Density}(?x1,?d1) \wedge \mathrm{Isa}(?y1,\mathrm{Endtable}) \wedge \mathrm{Less}(v1*d1,5) \rightarrow \mathrm{Safe\text{-}To\text{-}Stack}(?x1,?y1)$

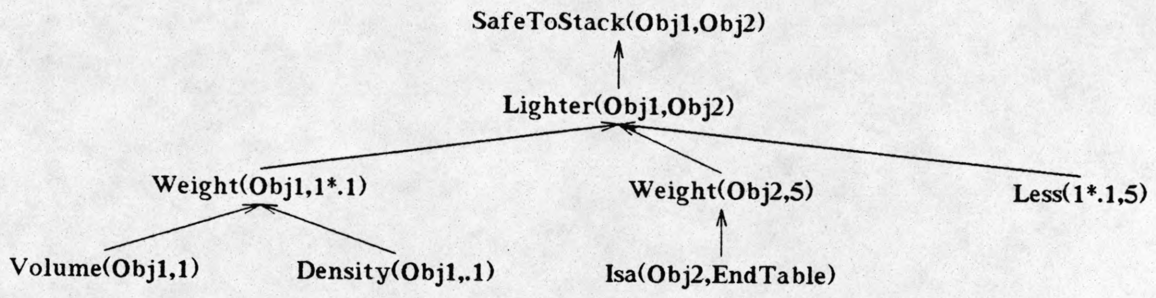


Figure 16: Safe-To-Stack Example -- Specific Explanation

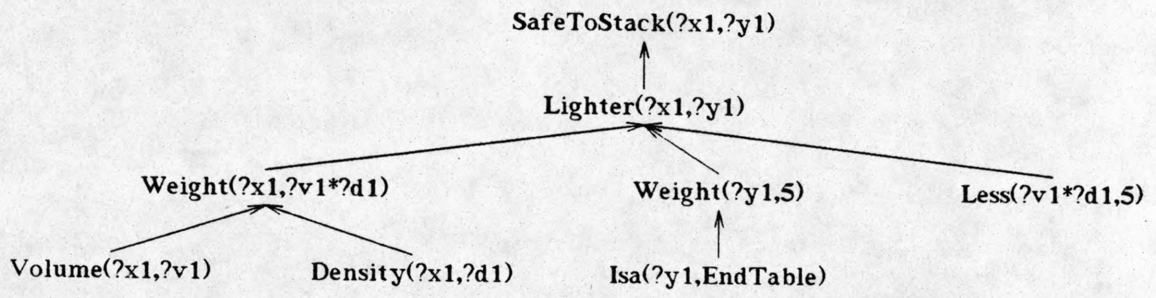


Figure 17: Safe-To-Stack Example -- Generalized Explanation

4.7. Suicide Example

The Suicide example was introduced in [DeJong86] to illustrate some problems with the original specification of the EBG algorithm given in [Mitchell86]. It involves inferring that an individual will commit suicide if he is depressed and buys a gun. The specific facts of the problem are:

Depressed(John)
Buy(John,Obj1)
Isa(Obj1,Gun)

The domain rules are:

Depressed(?x) → Despize(?x,?x)
Despize(?x,?y) → Hate(?x,?y)
Hate(?x,?y) ∧ Possess(?x,?z) ∧ Isa(?z,Gun) → Kill(?x,?y)
Buy(?x,?y) → Possess(?x,?y)

The proof that John will commit suicide is shown in Figure 18 and the general proof that anyone who is depressed and buys a gun will commit suicide is shown in Figure 19. The general rule learned from the generalized explanation is:

Depressed(?y1) ∧ Buy(?y1,?c1) ∧ Isa(?c1,Gun) → Kill(?y1,?y1)

This simple example was originally constructed to indicate the necessity of the forward-propagation step in EBG. If the forward-propagation step is not performed, the proper description of the goal concept supported by the explanation is never determined (i.e. the constraint that the

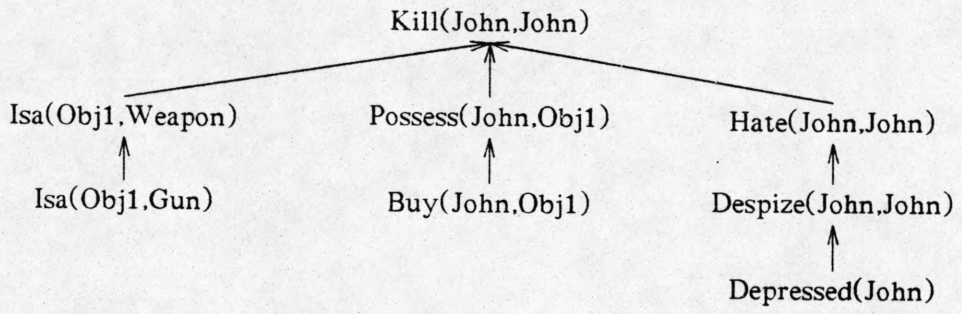


Figure 18: Suicide Example -- Specific Explanation

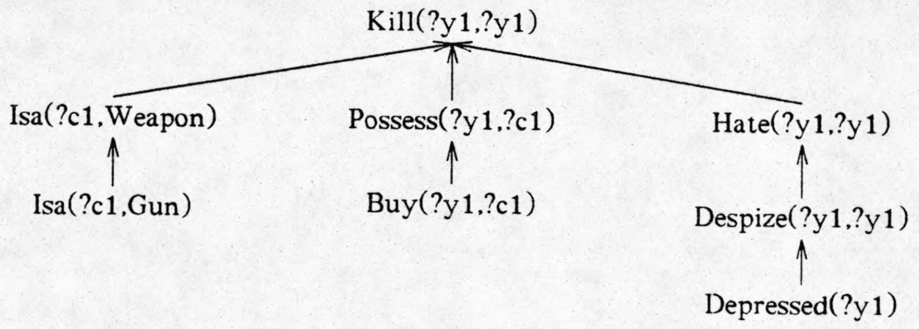


Figure 19: Suicide Example -- Generalized Explanation

killer be the same as the person killed is never imposed) and EBG constructs the following erroneous rule:

Depressed(?y) ∧ Buy(?x,?c) ∧ Isa(?c,Gun) → Kill(?x,?y)

This rule states that someone who buys a gun kills all depressed people. In addition, if the back-propagation step is performed first before forward-propagation, as originally suggested in [DeJong86, Mooney86], the constraint that the killer be the same as the person killed is imposed on the goal concept but not on the Buy branch of the explanation, and the following erroneous rule is learned:

Depressed(?y) ∧ Buy(?x,?c) ∧ Isa(?c,Gun) → Kill(?y,?y)

This rule states that if someone buys a gun all depressed people will kill themselves. Since EGGS propagates constraints uniformly in all directions, these problem are never encountered in our approach.

4.8. STRIPS Example

The STRIPS example [Fikes72], as discussed earlier, involves a situation as shown in Figure 20. The example involves a robot, located in room 1, moving to room 2 through door 1, picking up a box, and moving back to room 1 with the box. An explanation is constructed for the example using the following domain knowledge:

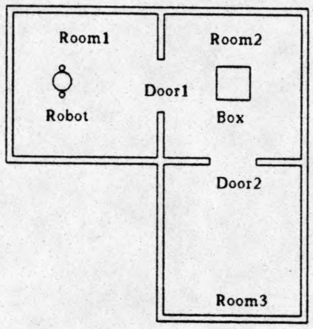


Figure 20: STRIPS: World Picture

STRIPS Actions		
Action	Preconditions	Effects

GoThru(?a,?d,?r1,?r2)	InRoom(?a,?r1)
	Connects(?d,?r1,?r2)
		InRoom(?a,?r2)

PushThru(?a,?o,?d,?r1,?r2)	InRoom(?a,?r1)
	InRoom(?o,?r1)
	Connects(?d,?r1,?r2)
		InRoom(?a,?r2)
		InRoom(?o,?r2)

An inference rule used in this example is:

Connects(?d,?r1,?r2) → Connects(?d,?r2,?r1)

The specific explanation for this plan is shown in Figure 21. The resulting generalization, shown in Figure 22, doesn't constrain the final destination room of the robot to be the same as its room of origin. The generalized plan, shown below, would support having the robot move the box to room 3 rather than back to room 1.

Learned MACROP		
Action	Preconditions	Effects

MoveBox(?a2,?b1,?x21,?d1,?x27,?d2,?y27)	InRoom(?a2,?x21)
	InRoom(?b1,?x27)
	Connects(?d1,?x21,?x27)
	Connects(?d2,y27,?x27)
		InRoom(?a2,?y27)
		InRoom(?b1,?y27)

4.9. GENESIS Example

The arson example from the GENESIS system[Mooney85] is a more complicated one in which domain specific generalization rules are used to augment the normal EGGS procedure. The specific explanation structure shown in Figure 23 is constructed from the following story which is presented to the narrative understanding system:

Stan owned a warehouse. He insured it against fire for $100,000. Stan burned the warehouse. He called Prudential and told them it was burnt. Prudential paid him $100,000.

This story is translated into the following assertions:

Possess(P1,W1)
Isa(P1,Person)

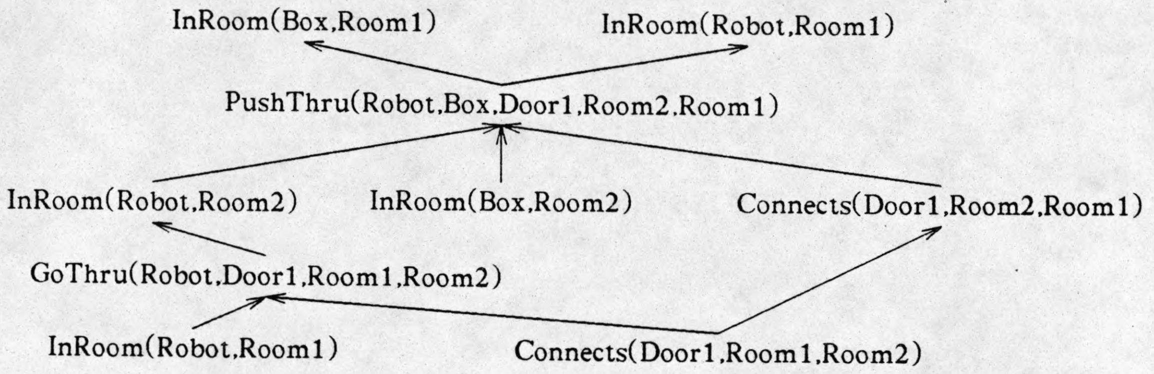

InRoom(Box,Room1)
InRoom(Robot,Room1)
PushThru(Robot,Box,Door1,Room2,Room1)
InRoom(Robot,Room2)
InRoom(Box,Room2)
Connects(Door1,Room2,Room1)
GoThru(Robot,Door1,Room1,Room2)
InRoom(Robot,Room1)
Connects(Door1,Room1,Room2)

Figure 21: STRIPS Example – Specific Explanation

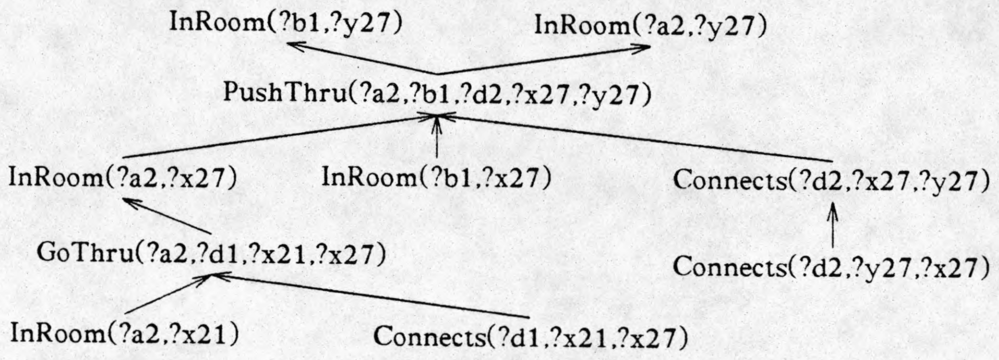


Figure 22: STRIPS – Generalized Explanation

Name(P1,Stan)
Isa(W1,Warehouse)
InsureObject(P1,Burnt,W1,M1,C1)
Isa(M1,Money)
Amount(M1,100000,Dollars)
Isa(C1,InsuranceCo)
Name(C1,Prudential)
Burn(P1,W1)
Telephone(P1,C1,Burnt(W1))
Indemnify(C1,M1,P1,Burnt,W1)

The domain knowledge available to the system includes the following inferences and the knowledge about actions represented in Table 5.

Isa(?x,Warehouse) → Isa(?x,Building)
Isa(?x,Building) → Isa(?x,Inanimate)
Isa(?x,Person) → Isa(?x,Character)
Isa(?x,InsuranceCo) → Isa(?x,Company)

Isa(?x,Company)→ Isa(?x,Character)
Telephone(?x,?y,?z) → Communicate(?x,?y,?z)

The explanation is that, since Stan's goal was to get money, he insured an unburnt warehouse he owned with Prudential. He then proceeded to burn down the flammable warehouse and to telephone Prudential to tell them about it. Since Prudential believed the warehouse was burnt, the building was insured with them, and they had the requisite money to reimburse Stan, they paid him the indemnity. Certain facts in the story, such as the name of the insurance company, are found not to be part of the explanation for how Stan acquired the money.

Table 5: GENESIS Actions for Arson

Action	Preconditions	Effects

Burn(?a,?o)	Flammable(?o)
	Not(Burnt(?o))
		Burnt(?o)
		Believe(?a,Burnt(?o))

Communicate(?a,?o,?i)	Believe(?a,?i)	Believe(?o,?i)

InsureObject(?a,?s,?o,?v,?c)	Possess(?a,?o)
	Not(?s(?o))
		Insured(?s,?o,?a,?v,?c)

Indemnify(?a,?o,?t,?s,?i)	Possess(?a,?o)
	Insured(?s,?i,?t,?o,?a)
	Believe(?a,?s(?o))
		Possess(?t,?o)

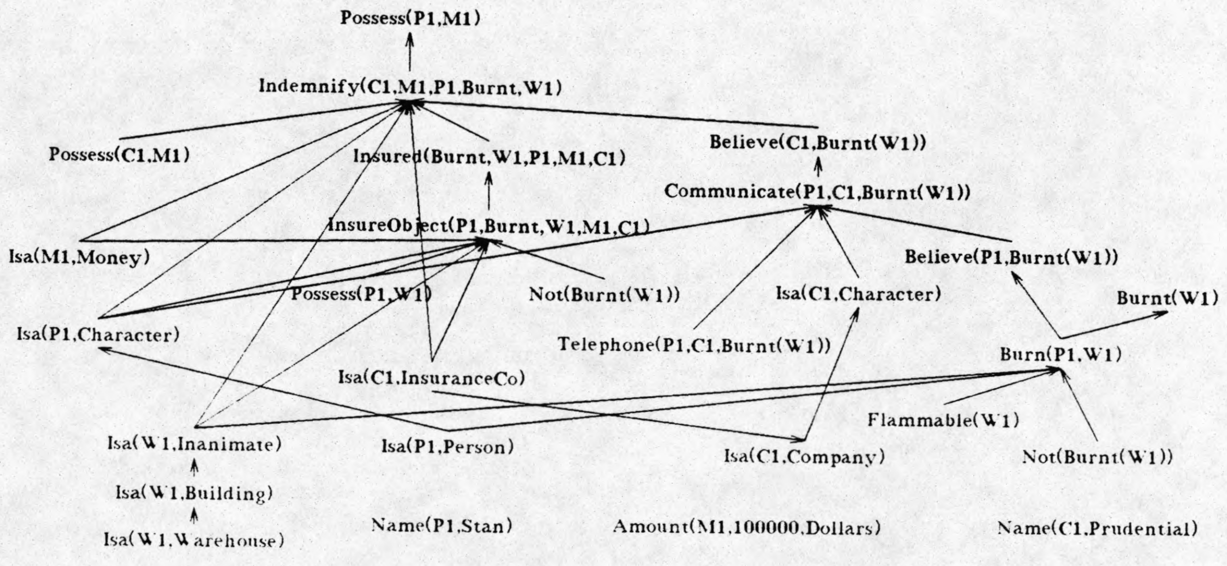


Figure 23: GENESIS Example -- Specific Explanation

The generalized explanation can be seen in Figure 24. In addition to the normal EGGS generalization process, hierarchical class inferences (ISA inferences) have been pruned to arrive at an appropriate generalization. The rule is to prune any facts supporting only ISA inferences. For instance, in this example, including the fact that the object burned was a warehouse would make the resulting generalization overly specific and less useful in understanding future narratives. However, it was important to include warehouse in the specific explanation in order to infer that it could be burned. Since the fact that the object was a warehouse only supports the fact that it is a building, this fact is removed from the generalized explanation. Likewise, the fact that the object is a building is also pruned. The fact that the object is an inanimate object cannot be pruned because it is a precondition for burn, insure-object, and indemnify. Consequently, it becomes part of the generalized structure. Also, in the generalized explanation, patterns with the same generalized specification have been combined.

5. Conclusion

In an attempt to formulate a general framework for explanation-based generalization, we have developed a representation and an algorithm which we believe are well suited for learning in a wide variety of domains. The representation of explanations defined in this paper has allowed easy representation of a wide variety of examples from various domains. The EGGS algorithm is an efficient and concise algorithm which we have used to generalize each of these examples with the same generalizing system. Future research issues include techniques for improving generality, such as the pruning of hierarchical class inferences discussed in section 4.3, and methods for dealing with imperfect and intractable domain theories and other problems outlined in [Mitchell86].

Acknowledgements

This research benefitted greatly from discussions with Paul O'Rorke and the direction of Gerald DeJong. Electronic mail correspondence with Smadar Kedar-Cabelli and Prasad Tadepalli was very helpful in developing the EBG algorithm presented in this report.

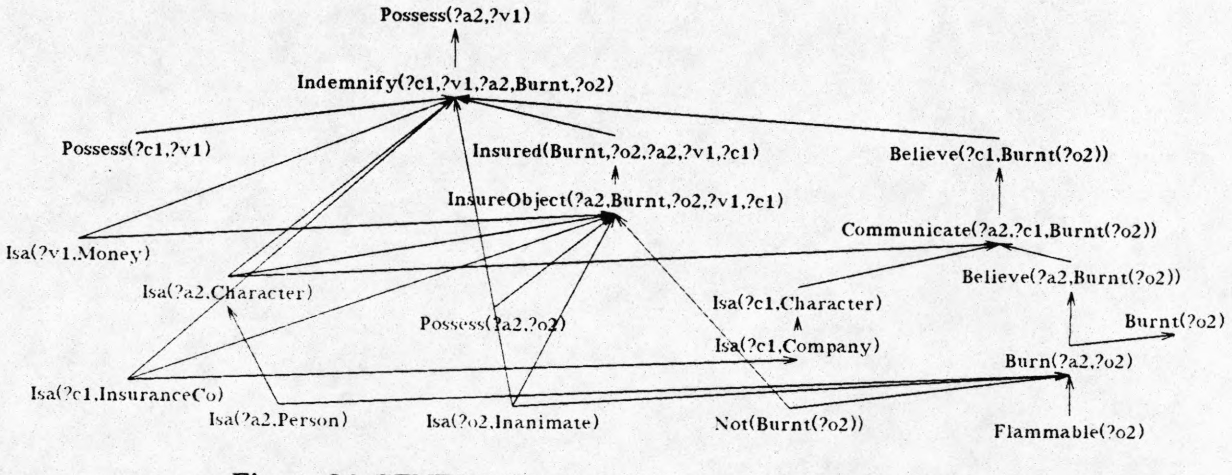


Figure 24: GENESIS Example -- Generalized Explanation

References

[Bundy83] A. Bundy, *The Computer Modelling of Mathematical Reasoning*, Academic Press, New York, NY, 1983.

[DeJong86] G. F. DeJong and R. J. Mooney, "Explanation-Based Learning: An Alternative View," *Machine Learning 1*, 2 (April 1986), .

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

[Mahadevan85] S. Mahadevan, "Verification-Based Learning: A Generalization Strategy for Inferring Problem-Reduction Methods," *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, August 1985, pp. 616-623.

[Manna74] Z. Manna, *Mathematical Theory of Computation*, McGraw Hill, New York, 1974.

[Mitchell83] T. M. Mitchell, "Learning and Problem Solving," *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, West Germany, August 1983, pp. 1139-1151.

[Mitchell85] T. M. Mitchell, S. Mahadevan and L. I. 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.

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

[Mooney85] R. J. Mooney and G. F. DeJong, "Learning Schemata for Natural Language Processing," *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, August 1985. (Also appears as Working Paper 67, AI Research Group, Coordinated Science Laboratory, University of Illinois at Urbana-Champaign.)

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

[Nilsson80] N. J. Nilsson, *Principles of Artificial Intelligence*, Tioga Publishing Company, Palo Alto, CA, 1980.

[O'Rorke84] P. V. O'Rorke, "Generalization for Explanation-based Schema Acquisition," *Proceedings of the National Conference on Artificial Intelligence*, Austin, TX, August 1984, pp. 260-263.

[O'Rorke85] P. V. O'Rorke, "Constraint Posting and Propagation in Explanation-Based Learning," Working Paper 70, AI Research Group, Coordinated Science Laboratory, University of Illinois, Urbana, IL, November 1985.

[Silver83] B. Silver, "Learning Equation Solving Methods from Worked Examples," *Proceedings of the 1983 International Machine Learning Workshop*, Urbana, IL, June 1983, pp. 99-104.

[Waldinger77] R. Waldinger, "Achieving Several Goals Simultaneously," in *Machine Intelligenge 8*, E. Elcock and D. Michie (ed.), Ellis Horwood Limited, London, 1977.

[Winston83] P. H. Winston, T. O. Binford, B. Katz and M. Lowry, "Learning Physical Descriptions from Functional Definitions, Examples, and Precedents," *Proceedings of the National Conference on Artificial Intelligence*, Washington, D.C., August 1983, pp. 433-439.