A DOMAIN INDEPENDENT EXPLANATION-BASED GENERALIZER

Raymond J. Mooney
Scott W. Bennett

Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
1101 W. Springfield Ave, Urbana, IL 61801

ABSTRACT

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 examples in different domains, and present detailed descriptions of several of these.

I 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 few of the examples to which EGGS has been applied are presented with their resulting generalizations.

II 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) ∧ Liftable(?x) ∧ OpenVessel(?x) → Cup(?x)
Bottom(?y) ∧ PartOf(?y,?x) ∧ Flat(?y) → Stable(?x)
Graspable(?x) ∧ Light(?x) → Liftable(?x)
Handle(?y) ∧ PartOf(?y,?x) → Graspable(?x)
Concavity(?y) ∧ PartOf(?y,?x) ∧ UpwardPointing(?y)
→ 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. 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.

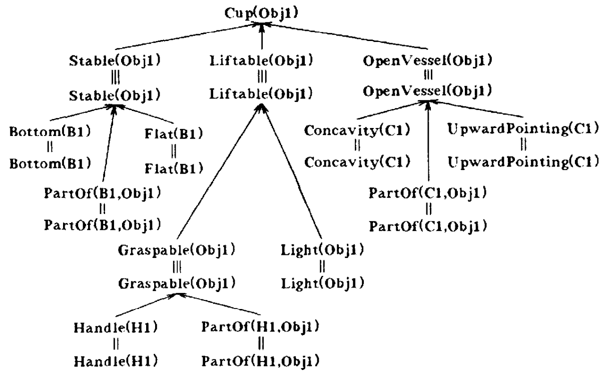


Figure 1: Explanation for Cup(Obj1)
Triple edges indicate equalities between unit patterns.
Double edges indicate equalities to initial assertions.

This research was supported by the National Science Foundation under grant NSF IST 83-17889.

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:

$$\text{Bottom}(?y1) \wedge \text{PartOf}(?y1,?x1) \wedge \text{Flat}(?y1) \wedge \text{Handle}(?y2) \wedge \text{PartOf}(?y2,?x1) \wedge \text{Light}(?x1) \wedge \text{Concavity}(?y3) \wedge \text{PartOf}(?y3,?x1) \wedge \text{UpwardPointing}(?y3) \rightarrow \text{Cup}(?x1)$$

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

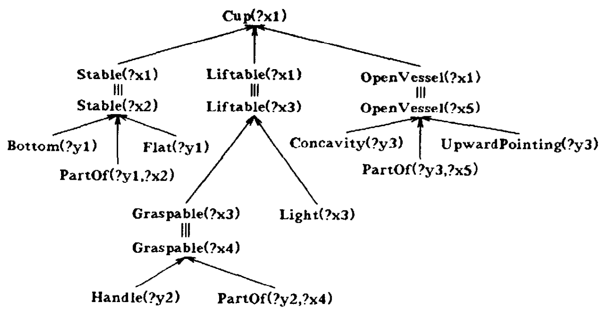


Figure 2: Explanation Structure for Cup Example
Triple edges indicate equalities between unit patterns.

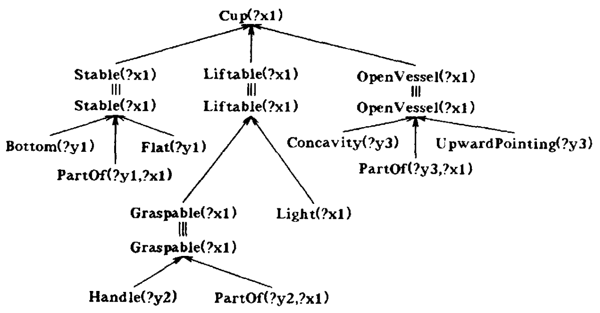


Figure 3: Generalized Explanation for Cup Example
Triple edges indicate equalities between unit patterns.

III 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 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].

A. 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 is: 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 II. How STRIPS problems are represented with interconnecting units will be clarified with an example later in the paper. 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

```
for each equality between $p_i$ 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

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.

B. 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. This process obtains the appropriate generalized antecedents of the proof; however, as pointed out in [DeJong86], it fails to obtain the appropriately generalized goal pattern and remaining explanation. As indicated in [DeJong86] and as originally specified in [Mahadevan85], the remaining generalized explanation must 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 is shown in the abstract algorithm in Table 2. Once again, the 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.

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

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 p with $p\theta$
            for each pattern $p_j$ supporting p do
                replace $p_j$ with $p_j\theta$
    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)

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 $p\theta$
```

Table 2: EBG Explanation Generalization Algorithm

C. 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, 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.

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

Table 4: EGGS Applied To the Cup Example

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

D. Comparison of Explanation Generalizing 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.

Assuming there are e equalities and p patterns in an explanation ($p > e$), 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. The number of substitution applications required by EBG depends on the number of antecedents for each rule, but in the best case (in which each rule has only one antecedent) it requires one substitution for each pattern in both the back-propagating and forward-propagating steps for a total of $2p$ substitutions. Finally, EGGS requires e unifications to build the global substitution (γ) and p substitutions to apply γ to the explanation structure. Each composition of a substitution with γ also requires a substitution, so there are really $e+p$ overall substitutions.

Therefore, EGGS does less unifications and substitutions than either the abstract STRIPS method or EBG. However, this may be misleading since the complexity of each unification or substitution depends on the nature of the patterns 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, constraints are propagated back through the explanation, and then a second pass is required to propagate constraints forward to obtain the appropriate generalized goal concept. We believe this adds undo complexity to the generalizing algorithm as is obvious from comparing the algorithmic descriptions in Tables 1-3.

IV Application of EGGS to Several Domains

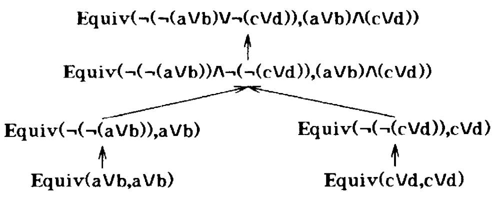
The EGGS technique, which has been fully implemented, has generalized explanations in several domains. This set of examples currently includes narrative understanding [Mooney85], generating physical descriptions of objects from functional information [Winston83], designing logic circuits [Mahadevan85, Mitchell85], solving integration problems [Mitchell83] [Mitchell86], 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 will describe the application of EGGS to the logic circuit design, STRIPS robot planning, and narrative understanding examples. All of the examples are discussed in a longer version of this paper [Mooney86].

A. LEAP example

The LEAP system in [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 computing the function: ¬(¬(a ∨ b) ∨ ¬(c ∨ d)). The system attempts to verify that the given circuit actually computes the desired function. The explanation proving that the function the circuit computes is equivalent to the desired function is shown in Figure 4. Since equated patterns are always identical in specific and generalized explanations, only one of each pair will be shown in this and future figures. In this example, the domain knowledge available to the system is:

Equiv(?x,?y) → Equiv(¬(¬(?x)),?y)
Equiv((¬?x∧¬?y),?a) → Equiv(¬(?x∨?y),?a)
Equiv(?x,?a) ∧ Equiv(?y,?b) → Equiv(?x∧?y,?a∧?b)
Equiv(?x,?x)

The generalized form of this proof is shown in Figure 5. Had constants 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 an a∨b and a c∨d. For example, the generalization would support using two NAND gates and a NOR gate to AND four things together.



Equiv(¬(¬(a∨b)∨¬(c∨d)),(a∨b)∧(c∨d))
Equiv(¬(¬(a∨b))∧¬(¬(c∨d)),(a∨b)∧(c∨d))
Equiv(¬(¬(a∨b)),a∨b)
Equiv(¬(¬(c∨d)),c∨d)
Equiv(a∨b,a∨b)
Equiv(c∨d,c∨d)

Figure 4: LEAP Example -- Specific Explanation

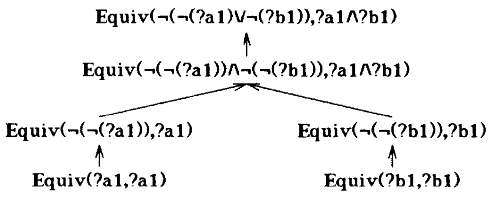


Figure 5: LEAP Example -- Generalized Explanation

B. STRIPS Example

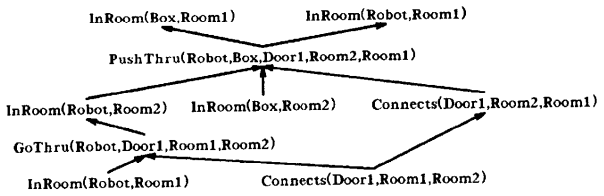
The STRIPS example [Fikes72], as discussed earlier, involves a robot, located in Room1, moving to Room2 through Door1, picking up a box, and moving back to Room1 with the box. An explanation is constructed for the example using the following action definitions:

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 6. The resulting generalization, shown in Figure 7, doesn't constrain the final destination of the robot to be the same as its room of origin. The generalized plan would support having the robot move the box to a Room3 connected to Room2 rather than back to Room1.



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 6: STRIPS Example -- Specific Explanation

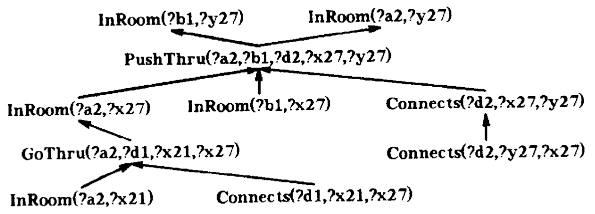


Figure 7: STRIPS -- Generalized Explanation

C. 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 8 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.

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

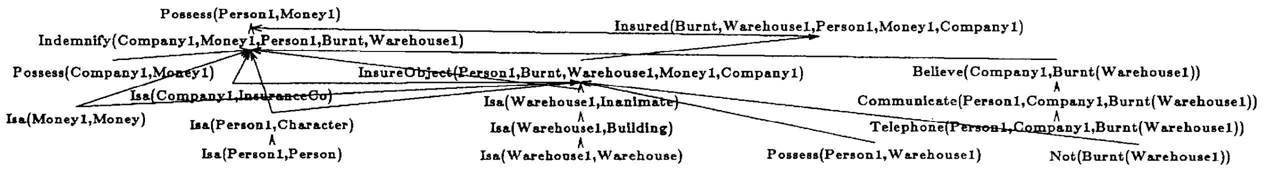

Possess(Person1,Money1)
Indemnify(Company1,Money1,Person1,Burnt,Warehouse1)
Insured(Burnt,Warehouse1,Person1,Money1,Company1)
Possess(Company1,Money1)
InsureObject(Person1,Burnt,Warehouse1,Money1,Company1)
Believe(Company1,Burnt(Warehouse1))
Isa(Company1,InsuranceCo)
Isa(Warehouse1,Inanimate)
Communicate(Person1,Company1,Burnt(Warehouse1))
Isa(Money1,Money)
Isa(Person1,Character)
Isa(Warehouse1,Building)
Telephone(Person1,Company1,Burnt(Warehouse1))
Isa(Person1,Person)
Isa(Warehouse1,Warehouse)
Possess(Person1,Warehouse1)
Not(Burnt(Warehouse1))

Figure 8: GENESIS -- Specific Explanation

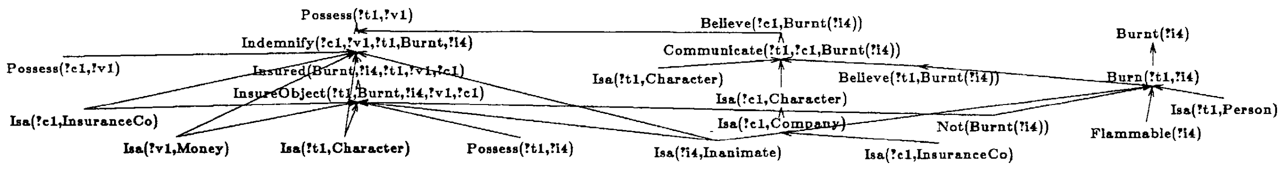


Figure 9: GENESIS -- Generalized Explanation

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.

The generalized explanation can be seen in Figure 9. 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.

V 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 above, 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.

REFERENCES

[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.

[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.

[Mooney86] R. Mooney and S. Bennett, "A Domain Independent Explanation-based Generalizer," Working Paper 71, AI Research Group, Coordinated Science Laboratory, University of Illinois, Urbana, Il., May 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.

[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.