EXPLANATION-BASED GENERALIZATION AND CONSTRAINT PROPAGATION WITH INTERVAL LABELS

Kai Zercher

Siemens AG, ZFE IS INF 33,
Otto-Hahn-Ring 6, D-8000 München 83

TU München, Institut für Informatik
Orleanstr. 34, D-8000 München 80
Germany

zercher@ztivax.uucp

Abstract

Two ways of applying EBG to constraint propagation with interval labels are presented. The first method, CP-EBG-1, is described by a straightforward use of a Prolog EBG implementation. The second, CP-EBG-2, performs two phases: First, constraint propagation is done and, using EBG, a generalized final labelling is derived but no extra conditions are learned. Second, constraint propagation is again performed using the final labellings of phase 1 as the initial labelling. This time, conditions are learned which form the desired concept description.
It is shown that CP-EBG-2 learns more general concept descriptions than CP-EBG-1. A proof is outlined that CP-EBG-2 produces correct concept descriptions for the class of constraints using linear equations and interval arithmetic. Central to this proof - and to possible proofs for other constraint classes - is the notion of a *moderate* generalization. It guarantees that a generalization which was learned from one instance and which is now used in a new situation, does not lead to the exclusion of any solution for this new situation.

Keywords

Explanation-based generalization, constraint propagation, interval labels, moderate generalization.

1. Introduction

Constraint propagation is a popular and widely used inference technique in AI (Davis, 1987). It operates on a constraint network, i.e., a set of variables (nodes) interconnected by constraints which express relations among variables. The most common type of constraint propagation is label inference: Every variable is assigned a set of possible values, and constraints are evaluated in order to restrict this set. This sort of constraint propagation is also called local constraint propagation (Güsgen & Hertzberg, 1988) or simply, as we will do throughout this paper, constraint propagation.

Given a constraint network, typical questions are whether a locally consistent solution (labeling of all variables) exists or what a particular value of such a solution is. Sometimes, the constraint network is fixed but some inputs vary and we have to determine the answer to these questions for various possible inputs. This is a situation which we faced in an application of model-based diagnosis for robot operations. Since constraint propagation turned out to be too slow for this application, we applied explanation-based generalization (EBG) to it. Thereby we learned general rules which, if applicable, derived the same results as constraint propagation would do. When we compared the execution time of constraint propagation for a specific instance with the time needed to match a rule learned by EBG from constraint propagation and this example, the latter was faster by two orders of magnitude. With the help of conditions learned from constraint propagation, we could construct rules for error diagnosis whose usage reduced the average diagnosis time considerably (Zercher, 1988, 1990 a).

In the next section we present a straightforward way to apply EBG to constraint propagation and illustrate it with a small example; this is the method used in the above mentioned literature. In Section 3, we propose a new, two phase approach, which learns a more general but still correct concept description. Properties of both methods are discussed in Section 4. We show in Section 5 that one can also learn concept descriptions from inconsistencies detected by constraint propagation. Section 6 introduces the notion of a *moderate* generalization, and we will show that, together with the monotonicity property of constraints, this is sufficient to guarantee that the new method yields correct generalizations. In the final section, we present our conclusion.

2. EBG and constraint propagation: First approach

EBG is a powerful, knowledge intensive learning technique (Mitchell et al., 1986; DeJong & Mooney, 1986). It is able to learn a correct generalization from a single training instance. Utilizing an available domain theory, EBG constructs an explanation why the given example belongs to the target concept. This explanation is then generalized and a conjunction of conditions, which forms a sufficient condition for the target concept, is extracted from it. EBG guarantees that the learned description fulfills an operationality criterion (Keller, 1987) that tries to ensure that the description can be efficiently evaluated. Some authors put EBG in a theorem proving framework and consequently speak about proofs, proof trees, and so forth; other authors put it in a problem solving framework where an explanation is a sequence of operators which transform an initial state into a goal state. As (Mooney & Bennett, 1986) have shown, both are two views of the same coin since the proposed, different generalization algorithms yield basically the same results.

Table 1 shows a simple Prolog implementation of constraint propagation along with some of the predicates needed for interval arithmetic. We normally work with a CommonLisp implementation, however, we use Prolog here since it is the de facto standard language for describing EBG research. Constraint propagation works by repeatedly evaluating constraints. Whenever a value used by a constraint is changed, this constraint must be reprocessed. This can mean that the same constraint is executed several times. Constraint propagation stops - reaches quiescence - when no constraint can change the value of a variable. Termination of constraint propagation is guaranteed for some classes of constraints, e.g., linear equations with unit coefficient, but not for all (Davis, 1987).

For the purpose of EBG, an explanation (proof) is a sequence of constraint propagation steps which transforms an initial variable labelling into a final labelling which is consistent, i.e., every constraint is fulfilled. Such a sequence is exactly what cp_consistent/3 (the '/3' is a Prolog notation which means that the predicate has 3 arguments) will produce provided the predicate succeeds. In (Kedar-Cabelli & McCarty, 1987) a clear and concise EBG implementation is given. Their predicate prolog_ebg/3 has as arguments, the goal clause, the generalized goal, and a list of learned conditions (the operational concept definition). By applying it to cp_consistent/3, we have the desired combination of EBG and constraint

```
/*  cp__consistent( +Constraints, +StartLabelling, -FinalLabelling ) :-          */
/*     succeeds if constraint propagation terminates with a consistent final labelling.  */
cp__consistent( [Constraint|RConstraints], StartLabelling, FinalLabelling ) :-
    constraint(Constraint, StartLabelling, NewLabelling, AffectedConstraints),
    /* Applies a constraint and produces a new labelling. If a value of a variable  */
    /* was changed then all constraints which must be reused are contained in       */
    /* AffectedConstraint. Fails if an empty interval was derived.                  */
    append(RConstraints, AffectedConstraints, UpdatedConstraints),
    cp__consistent(UpdatedConstraints, NewLabelling, FinalLabelling).

cp__consistent( [], FinalLabelling, FinalLabelling ).

constraint(c4, [x(X__Int), y(Y__Int)), a(A__Int), b(B__Int), c(C1__Int) ],
               [x(X__Int), y(Y__Int)), a(A__Int), b(B__Int), c(C2__Int) ], [] ) :-
    /* The simple constraint C ← X + [-2, 2] + A taken from Table 2.                */
    interval__plus(X__Int, [-2, 2], H__Int), interval__plus(H__Int, A__Int, CNew__Int),
    interval__intersection(C1__Int, CNew__Int, C2__Int).

interval__intersection([A1,A2], [B1,B2], [C1,C2]) :-
    maximum(A1, B1, C1), minimum(A2, B2, C2), C1 ≤ C2.

interval__plus([A1,A2], [B1,B2], [C1,C2]) :-
    C1 is A1 + B1,  C2 is A2 + B2.

maximum(-∞, Y, Y).
maximum(X, Y, X) :- Y ≤ X.
maximum(X, Y, Y) :- X ≤ Y.
```

Table 1: Pieces of the Prolog code for constraint propagation

propagation; it will be called CP-EBG-1. However, we perform a few modifications, which we will explain after the following examples.

We first look at what happens if we apply prolog__ebg/3 to the predicate interval__intersection/3. Provided that arithmetic operations and comparisons are declared as being operational, calling

```
prolog__ebg( interval__intersection([1,5], [2,6], Int3),
             interval__intersection([X,Y], [U,V], GInt3),
             LearnedConditions ).
```

will succeed with the following bindings:

Int3 = [2,5], GInt3 = [U,Y], LearnedConditions = [X ≤ U, Y ≤ V, U ≤ Y]

The first condition guarantees that the lower bound of the result interval is the lower bound of the second interval, the second condition guarantees that the upper bound of the result interval is the upper bound of the first interval, and the third condition guarantees that the result interval is well defined.

Table 2 shows an extremely simple constraint propagation problem. It is atypical in the sense that there is no feedback, i.e., no constraint must be executed twice, but it is sufficient for demonstration purposes. The training example is defined by X = 5 and Y = 7. If we call

```
prolog_ebg( cp_consistent([c1, ... ], [ x([5,5]), y([7,7]), a([-∞, ∞]), ... ], Labelling),
            cp_consistent([c1, ... ], [ x([X,X]), y([Y,Y]), a([-∞, ∞]), ... ], GenLabelling),
            LearnedConditions ).
```

it will succeed with the following bindings:

```
Labelling = [ x([5,5]), y([7,7]), a([1,2]), b([2,3]), c([6,9]) ],
GenLabelling = [ x([X,X]), y([Y,Y]), a([1, 2]), b([2, 3]), c([Y-1, X + 4]) ],
LearnedConditions = [ X ≤ 6,  6 ≤ Y,  -5 ≤ X-Y]     /* simplified result */
```

This means, that we have learned the general rule (we use '←' instead of ':-' in order to distinguish Prolog code from learned rules):

```
cp_consistent([c1, c2, c3, c4, c5], [x([X,X]), y([Y,Y]), a([-∞, ∞]), b([-∞, ∞]), c([-∞, ∞]) ],
                    [ x([X,X]), y([Y,Y]), a([1, 2]), b([2, 3]), c([Y-1, X + 4]) ])
    ← X ≤ 6,  6 ≤ Y,  -5 ≤ X-Y.
```

In some cases, we might not be interested in computing the consistent final labelling but only in determining whether it exists depending on the values of X and Y. With the definition

```
cp_consistent_problem1(X, Y) :-
    cp_consistent([c1, c2, c3, c4, c5], [ x([X,X]), y([Y,Y]), a([-∞, ∞]), b([-∞, ∞]), c([-∞, ∞]) ], _ ).
```

we would get the rule:

```
cp_consistent_problem1(X, Y)  ←  X ≤ 6,  6 ≤ Y,  -5 ≤ X-Y.
```

Table 2 shows in detail what happens; it lists the constraint propagation steps, their grounded and generalized results and the learned conditions. Some expressions have been simplified in order to enhance readability. The conditions are constructed when C_{i-1} and $C_{new,i}$ are intersected in order to get C_i . This is done in the same fashion as demonstrated above in the example for the predicate interval_intersection/3. In Figure 1, the dark shaded area shows the space covered by the learned concept definition.

To the the standard EBG approach, we have applied the following modifications:

- Predicates which are used for control purposes do not cause any conditions to enter the learned concept description. We took append/3 as an extremely simple strategy to control constraint propagation. Other strategies (Davis, 1987) could also be used as long as they are "fair" (Güsgen & Hertzberg 1988).
- The list of learned conditions is simplified. After conditions have been individually simplified, e.g., X + Y-2 ≤ Y + 5 becomes X ≤ 7, redundant conditions are

A Simple Constraint Propagation Problem:

$A \leftarrow [1, 2]$, $B \leftarrow [2, 3]$, $C \leftarrow [5, 16]$, $C \leftarrow X + [-2, 2] + A$, $C \leftarrow Y + [-3, 3] + B$

Training example: $X = 5$, $Y = 7$

Trace of CP-EBG-1 for the variable C

Name	Interval	Generalized Interval	Learned Conditions or Comment
C_0	$[-\infty, \infty]$	$[-\infty, \infty]$	-
$C_{new,1}$	[5, 16]	[5, 16]	constraint $C \leftarrow [5, 16]$
C_1	[5, 16]	[5, 16]	$5 \leq 16$
$C_{new,2}$	[4, 9]	[X-1, X + 4]	constraint $C \leftarrow X + [-2, 2] + A$
C_2	[5, 9]	[5, X + 4]	$X-1 \leq 5$, $X+4 \leq 16$, $5 \leq X+4$
$C_{new,3}$	[6, 13]	[Y-1, Y + 6]	constraint $C \leftarrow Y + [-3, 3] + B$
C_3	[6, 9]	[Y-1, X + 4]	$5 \leq Y-1$, $X+4 \leq Y+6$, $Y-1 \leq X+4$

Learned conditions (simplified): $X \leq 6$, $6 \leq Y$, $-5 \leq X-Y$

Trace of phase 2 of CP-EBG-2 for the variable C

Name	Interval	Generalized Interval	Learned Conditions or Comment
C_0	[6, 9]	[Y-1, X + 4]	result of phase 1
$C_{new,1}$	[5, 16]	[5, 16]	constraint $C \leftarrow [5, 16]$
C_1	[6, 9]	[Y-1, X + 4]	$5 \leq Y-1$, $X+4 \leq 16$, $Y-1 \leq X+4$
$C_{new,2}$	[4, 9]	[X-1, X + 4]	constraint $C \leftarrow X + [-2, 2] + A$
C_2	[6, 9]	[Y-1, X + 4]	$X-1 \leq Y-1$, $X+4 \leq X+4$, $Y-1 \leq X+4$
$C_{new,3}$	[6, 13]	[Y-1, Y + 6]	constraint $C \leftarrow Y + [-3, 3] + B$
C_3	[6, 9]	[Y-1, X + 4]	$Y-1 \leq Y-1$, $X+4 \leq Y+6$, $Y-1 \leq X+4$

Learned conditions (simplified): $X \leq 12$, $6 \leq Y$, $-5 \leq X-Y$, $X-Y \leq 0$

Table 2: Constraint propagation and EBG

removed. A condition is redundant if it is either always true like $0 \leq 2$ or if it is implied by the remaining conditions. For instance, $X \leq 7$ is implied by $X \leq 5$. For the general case of linear inequalities, this test can be performed with the well known Simplex method (e.g., Papadimitriou & Steiglitz, 1982). In our experiments, simplification drastically decreased the size of the learned concept description. This is one of the major causes why the learned concept

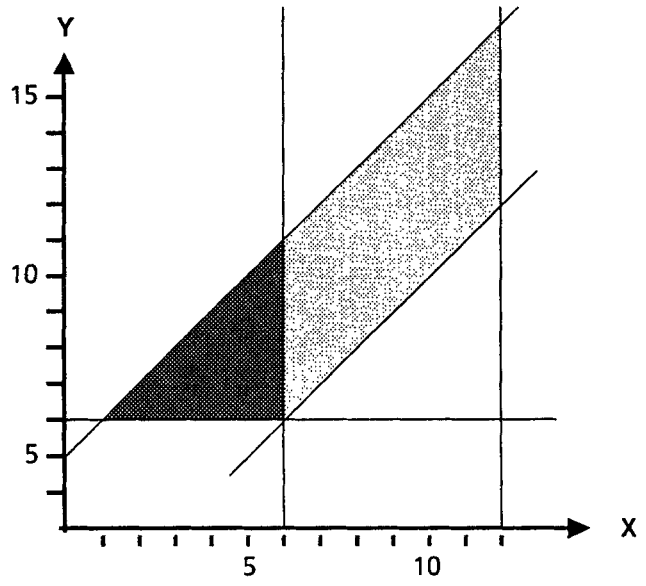

Figure 1: Coverage of learned concept definitions

description can be executed much faster than constraint propagation on the same example. See also (Minton, 1988) on the importance of simplification for EBG.

- We handle the predicate is/2 differently. Instead of getting the generalized interval [Y-1, X + 4], standard EBG would produce something like [U, V] plus the conditions U is Y-1 and V is X + 4. We achieve our type of generalization with a simple change: When prolog_ebg/3 performs is/2 on the grounded data, =/2 is done for the generalized data (in addition, the use of clause/2 must be altered slightly). This modification causes in fact only a syntactic change in the EBG result. However, it has several small advantages, the main being that it makes the presentation of EBG and its results simpler and easier to read.

3. EBG and constraint propagation: Second approach

We will call the new approach CP-EBG-2. From a given example it is able to learn a more general concept description than CP-EBG-1. CP-EBG-2 consists of two phases: First, we do constraint propagation and use EBG to construct a

generalized final labelling but we do not learn any ≤/2 conditions. Second, with the consistent labelling and its generalization produced in phase 1, we again perform constraint propagation and apply EBG to it. Off course, the labelling and its generalization will not be changed by any constraint (after all, this is the property of a consistent labelling). This time, however, we will learn conditions and form the desired concept description. For our example these steps are illustrated in Table 2. We have learned the general rule

cp_consistent([c1, c2, c3, c4, c5], [x([X,X]), y([Y,Y]), a([-∞, ∞]), b([-∞, ∞]), c([-∞, ∞])],
[x([X,X]), y([Y,Y]), a([1, 2]), b([2, 3]), c([Y-1, X + 4])])
← X ≤ 12, 6 ≤ Y, -5 ≤ X-Y, X-Y ≤ 0.

or as an alternative

cp_consistent_problem1(X, Y) ← X ≤ 12, 6 ≤ Y, -5 ≤ X-Y, X-Y ≤ 0.

The area covered by the learned concept description is depicted in Figure 1 by the light and dark shaded areas. As we can see, the concept description found by the new method is more general (covers a larger area) than the one learned by the old method (only the dark shaded area). In Section 6 we will show that CP-EBG-2 always produces correct concept descriptions. The Prolog code of CP-EBG-1, CP-EBG-2, and the examples presented here is available from the author upon request.

Why can CP-EBG-2 learn a more general concept description than CP-EBG-1 for the same generalized labelling? A formal argument is presented in Section 6, but the following trivial example should help to get an intuitive understanding: Assume we want to find the maximum of the list [1,2,3]. We do this in the usual sequential fashion and after we have determined that $1 \leq 2$ and $2 \leq 3$ we conclude that 3 is the maximum. A generalization of these reasoning steps would be that Z is the maximum of [X,Y,Z] if $X \leq Y$ and $Y \leq Z$. However, the antecedent '$X \leq Z$ and $Y \leq Z$' would obviously be more general and still be correct. We could get this antecedent if we generalize the verification that 3 is indeed the maximum, i.e., $1 \leq 3$ and $2 \leq 3$. By generalizing the verification of the solution instead of generalizing the steps which led to the solution, a more general concept description was found.

As an effect of our modified EBG, all functional expressions, e.g., Y-1, needed to compute the final labelling are included in the final generalized labelling. If we use standard EBG this is not the case; we just get generalized intervals like [U,V]. Consequently, starting phase 2 of CP-EBG-2 only with such a generalized labelling would not produce a useful result; the connection to the parameters of

the initial labelling - here X and Y- would be lost. To perform CP-EBG-2 correctly with standard EBG, we keep after phase 1 all predicates which compute the final labelling, i.e., the is/2 predicates, and remove all others, i.e., the ≤/2 predicates. Phase 2 is then performed as before. Both methods will produce equivalent results. In this paper one should read 'generalized labelling' as either the result of our modified EBG or as the result of standard EBG plus the conditions which are needed to compute the labelling. When we talk about 'generalized solution' this should be understood the same way.

At first one might think that the result of CP-EBG-2 could also be achieved by first computing the final consistent labelling and then presenting this as a training example to EBG. EBG would proof that the labelling is consistent and would then derive a general condition for this. But this idea is flawed since all connections to the initial labelling and its parameters are lost. The two phases of CP-EBG-2 are really required.

4. Properties of CP-EBG-1 and CP-EBG-2

The training example used by CP-EBG-1 and CP-EBG-2 guides the generalization process. It thereby determines which generalized final labelling is constructed and when constraint propagation and its generalization is stopped . The latter point is not apparent in our example, but as soon as there are feedback loops, e.g., introduced by the constraint B ← C - A, this becomes important. When feedback loops are present, it is difficult to imagine how generalization could be done without the help of examples. This is similar to the problem partial evaluation faces with recursive rules.
Phase 2 of CP-EBG-2 is conveniently described as doing EBG. Note however, in this phase the example is not really required since the outcome of all constraint propagation steps, i.e., no change, is known in advance.

One question is whether two phases are sufficient. The rule learned by CP-EBG-2 describes the largest possible coverage of the determined generalized labelling. Consequently, another phase which would try to change the learned ≤/2 conditions could not be of any help.

Whenever an interval intersection is generalized three ≤/2 conditions are produced. Consequently, CP-EBG-1 will initially produce a number of conditions equal to three times the number of constraint propagation steps. On the other

hand, with CP-EBG-2 it is just three times the number of existing constraints. The number only depends on the problem size and not on the number of inference steps performed. Usually, this means that CP-EBG-2 initially produces far fewer conditions than CP-EBG-1. Therefore, less time must be spent by CP-EBG-2 on simplification. Consequently, CP-EBG-2 will usually - despite the need for a second phase - be much faster than CP-EBG-1.

When constraint propagation does not terminate, CP-EBG-1 or CP-EBG-2 are obviously not possible. In such a situation one must resort to other inference techniques. If the task is to decide the consistency of constraints which are linear equalities or inequalities, a sound technique is the Simplex method. We can even apply EBG to it (Zercher, 1990b).

Finally, a note on implementation seems in place. The predicate prolog_ebg/3 is a general EBG implementation based on a Prolog meta-interpreter and is therefore quite slow. A much better speed is achieved with a specialized implementation: All predicates are augmented by extra parameters for generalized values. The generalization operations are explicitly coded. We experienced that Lisp is better suited than Prolog. One reason is that Lisp has efficient array operations which are needed to implement the Simplex method (required for simplification).

5. Learning from inconsistencies

Sometimes, we want to learn a concept description for the inconsistency of a constraint network depending on the input variables. An inconsistency is present when the intersection of two intervals, e.g., [1,2] and [4,5], is empty. CP-EBG-1 learns conditions for such a case by doing the same as before with the one exception that the detection of an inconsistency, i.e., the upper bound of one interval is lower than the lower bound of the other, produces a corresponding general $</2$ condition and immediately stops constraint propagation. With the training example $X = 5, Y = 15$ we learn the rule

$$cp\_inconsistent\_problem1(X, Y) \leftarrow X\text{-}Y < -5,\ 1 \leq X,\ X \leq 6.$$

In the case of CP-EBG-2, we stop phase 1 as soon as an inconsistency has been found and in phase 2, we just generate the one condition capturing the inconsistent interval intersection. In effect this means that we take just the one

$</2$ condition from phase 1 and ignore all $\leqq/2$ conditions. For the same training example, CP-EBG-2 learns the rule

cp_inconsistent_problem1(X, Y) ← X-Y < -5.

CP-EBG-2 will always produce just one single condition which implies an inconsistency. This is quite surprising at first. If we look at the space defined by the extra input variables (here X and Y), one can easily show that the subspace of all consistent variable values is convex. Consequently, an inconsistent subspace can be described by just one hyperplane (linear inequality).

For a given instance, we expect that the one condition concept description learned from this example can be executed much faster than constraint propagation on this example. We can even proof this statement with precise complexity results. If linear equations and unit coefficients are used as constraints the time complexity of constraint propagation is already O(nE) where n is the number of variables and E is the total size of all constraints (Davis, 1987). In contrast to this, executing one learned condition has a complexity of just O(n).

6. Correctness of CP-EBG-2

We will try to keep this section as brief and as informal as possible. We will heavily use (Güsgen & Hertzberg, 1988) for which we will henceforth take the short form (G&H); for those who want to get a deeper understanding we highly recommend this paper. Since prolog_ebg is a correct method and cp_consistent implements a fair constraint propagation strategy (Def. 8 G&H), our first approach CP-EBG-1 is guaranteed to learn a correct concept description. Correct means, that if an instance fulfills the learned conditions then the labelling, which we get by instantiating and evaluating the generalized labelling, is identical to the final labelling produced by constraint propagation.

The concept descriptions constructed by both approaches guarantee that the final generalized labelling is consistent. However, the CP-EBG-1 concept description in addition assures that, for a new instance, all constraint propagation steps yield results analog to the one produced by the training instance. Consequently, CP-EBG-2 learns a more general concept description than CP-EBG-1. We must however show that the CP-EBG-2 concept description is also correct.

A labelling L assigns every variable of a constraint network an interval, i.e., a description of a set of possible values. A labelling L_1 is a subset ($\subseteq$) of a labelling

L_2 , iff the subset relation holds for every variable value. A constraint c is a mapping from a labelling to a new labelling. A constraint includes computing a new value and intersecting it with the old one. We only deal with monotonic constraints (Def. 2 G&H), i.e., $c(L_1) \subseteq L_1$ and $[\, L_1 \subseteq L_2 \rightarrow c(L_1) \subseteq c(L_2)\,]$. Almost all types of constraints, including the one we are using (linear equations with intervals), are monotonic. An instantiation h is a function which maps a generalized labelling to a grounded labelling by replacing certain variables with a value and evaluating all expressions. For instance, if h instantiates X with 5 and Y with 7, then h([Y-1,X + 4]) = [6,9].

Def. 1: Let c be a constraint, h an instantiation, L_1 a labelling, GL_1 a generalized labelling such that $L_1 = h(GL_1)$. A generalization gen is a function which produces a new generalized labelling GL_2 given c, L_1, and GL_1 such that $GL_2 = gen(c,L_1,GL_1)$ and $c(L_1) = h(GL_2)$.

Def. 2: Let h_1 be one particular instantiation, $L_1 = h_1(GL_1)$, $GL_2 = gen(c,L_1,GL_1)$, and H be the set of all possible instantiation functions.
A generalization gen is *moderate*, iff $\forall\, h_i \in H$: $c(h_i(GL_1)) \subseteq h_i(GL_2)$.

Moderateness means that if we derive a general labelling from a constraint execution of a specific instance and then use this labelling for a different instance (instantiation), the resulting instantiated labelling will be a superset of what executing the constraint will give us. Moderateness prevents us from ruling out possibly consistent values from a labelling.

For linear equations and interval arithmetic, the generalizations we perform are all moderate. The rational is the following: When we generalize arithmetic operations like + /2 there is no loss of precision. When we generalize interval_intersection/3 we have to generalize maximum/3 (responsible for the lower bound) and minimum/3 (responsible for the upper bound). For instance if we compute the minimum of 9 and 13 we get 9; if the generalized values are X + 4 and Y + 6 the generalized result is X + 4. Even if this result is not correct for a different instantiation (e.g., X = 8, Y = 5), it can only cause an interval to be larger than it should be (but never smaller).

Prop. 1: Let cp be a mapping from an initial labelling to a final consistent labelling (if it exists) as it is computed by constraint propagation. Let GL_S be a general start labelling. If GL_F is a generalized final labelling constructed by moderate generalizations of the constraint propagation for some h_1, then $\forall\, h_i \in H$: $cp(h_i(GL_S)) \subseteq h_i(GL_F)$.

This can be easily proved by induction. It already gives us the correctness of CP-EBG-2 in the case of inconsistency: If a superset of the labelling achievable with

constraint propagation is already inconsistent, then constraint propagation would also detect an inconsistency.

Prop. 2: *Same requirements as Proposition 1.*

If $h_i(GL_F)$ is consistent, i.e., $h_i(GL_F) = cp(h_i(GL_F))$,

then $h_i(GL_F) = cp(h_i(GL_S))$.

This can be proved using the uniqueness property of constraint propagation (Prop.2 G&H). Proposition 2 guarantees that if the CP-EBG-2 concept description is fulfilled then constraint propagation would terminate with a consistent labelling identical to the instantiated generalized final labelling constructed by CP-EBG-2. Consequently, this proposition gives us the correctness of CP-EBG-2.

7. Conclusions

We have presented two ways of combining EBG with constraint propagation. The first, CP-EBG-1, is described by a straightforward use of prolog_ebg/3 and it is obviously a correct method. The second, CP-EBG-2, performs two phases: First, a generalized final labelling is derived but no conditions are learned. Second, constraint propagation is again performed using the final labellings of phase 1 as the initial labelling. Now conditions are learned which form the desired concept description. This concept description is more general than the one constructed by CP-EBG-1. The reason is that although both descriptions guarantee that the final labelling is consistent, the CP-EBG-1 description in addition assures that the final labelling is reached with equivalent intermediate labellings. Or, to express it differently, a CP-EBG-2 description only guarantees that a particular solution is correct, whereas a CP-EBG-1 description requires that the solution is reached on a specific path with specific intermediate results.

We have proved the correctness of CP-EBG-2 for the class of constraints using linear equations and interval arithmetic. Central to this proof - and to possible proves for other constraint classes - is the notion of a *moderate* generalization. It guarantees that a generalization which was learned from one instance and which we now used in a new situation, does not lead to the exclusion of any solution for this new situation. The propositions outlined in Section 6 are very general and do not depend on the use of interval labels; they are applicable whenever variables are labelled with (the description of) a set of possible values.

We have applied CP-EBG-2 in an example application where we had originally used CP-EBG-1 (Zercher 1990a). CP-EBG-2 led to clear improvements, i.e., for a

given set of training examples, the number of rules needed to be learned was decreased, the coverage of the learned rule set increased, and the average matching time for the rule set also decreased.

We think that a two phase EBG approach, i.e., first derive a generalized solution and then justify this solution again in order to learn a concept description, will also be useful in other domains. In fact, we first discovered the two phase principle when we tried to apply EBG to the Simplex method (Zercher 1990b). Here it was obviously the correct and better approach. This then triggered a reconsideration of our first combination of EBG and constraint propagation. Clearly, whenever we want to apply the two phase approach to a new inference technique, we have to put in extra efforts in order to determine whether we will still learn correct concept descriptions.

Acknowledgements

I would like to thank Angelika Hecht, Peter Struss, and my advisor Prof. Bernd Radig for many fruitful discussions and useful suggestions on earlier versions of this paper. Special thanks go to the anonymous referees for their helpful comments. The author thankfully acknowledges the support by a Ph.D. grant from Siemens AG.

References

Davis, E., (1987). Constraint propagation with interval labels. *Artificial Intelligence, 32*,281-331.

DeJong, G., & Mooney,R. (1986). Explanation-based learning: An alternative view. *Machine Learning, 1*, 145-176.

Güsgen, H.-W., & Hertzberg, J. (1988). Some fundamental properties of local constraint propagation. *Artificial Intelligence, 36*, 237-247.

Kedar-Cabelli, S. T., & McCarty, L. T. (1987). Explanation-based generalization as resolution theorem proving. *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 383-389). Irvine,CA: Morgan Kaufmann.

Keller, R. M. (1987). Defining operationality for explanation-based learning. *Proceedings of the Sixth National Conference on Artificial Intelligence* (pp. 482-487).

Minton, S., (1988). Quantitative results concerning the utility of explanation-based learning. *Proceedings of the Seventh National Conference on Artificial Intelligence* (pp. 564-569).

Mitchell, T. M., Keller, R., & Kedar-Cabelli, S. (1986). Explanation-based generalization: a unifying view. *Machine Learning, 1*, 47-80.

Mooney, R. J., & Bennett, S. W. (1986). A domain independent explanation-based generalizer. *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 551-555). Philadelphia, PA: Morgan Kaufmann.

Papadimitriou, C. H., & Steiglitz, K. (1982). *Combinatorical Optimization: Algorithms and Complexity*. Englewood Cliffs, NJ: Prentice-Hall.

Zercher, K., (1988). Model-based learning of rules for error diagnosis. In Hoeppner, W. (Ed.), *Proceedings of the 12th German workshop on artificial intelligence (GWAI 88)* (pp. 196-205). Springer.

Zercher, K., (1990 a). Constructing decision trees from examples and their explanation-based generalizations. In Marburger, H., (Ed.), *Proceedings of the 14th German workshop on artificial intelligence (GWAI 90)* (pp. 267-276). Springer.

Zercher, K., (1990 b). Learning efficient rules and decision trees for error diagnosis of robot operations. Unpublished working paper.