Constraint Relaxation and Nonmonotonic Reasoning

Gerhard Brewka*
International Computer Science Institute (ICSI)
1947 Center Street, Suite 600, Berkeley, CA 94704-1105, U.S.A.

Hans Werner Guesgen, Joachim Hertzberg†
German National Research Center for Computer Science (GMD)
Artificial Intelligence Research Division
Schloß Birlinghoven, 5205 Sankt Augustin 1, F.R.G.

TR-92-002

*G. Brewka is on leave from the German National Research Center for Computer Science (GMD), Schloß Birlinghoven, 5205 Sankt Augustin 1, F.R.G.

†H.W. Guesgen and J. Hertzberg are supported by the German Federal Ministry for Research and Technology (BMFT) in the joint project TASSO under grant ITW8900A7. TASSO is also part of the GMD *Leitvorhaben* Assisting Computer (AC).

Abstract

The purpose of this paper is to bring together the two AI areas of constraint-based and nonmonotonic reasoning. In particular, we analyze the relation between different forms of constraint relaxation and a particular approach to nonmonotonic reasoning, namely, preferred subtheories. In effect, we provide formal semantics for the respective forms of constraint relaxation.

1 Introduction

During recent years, *constraint-based reasoning* techniques have proven to be very useful and widely applicable in AI. However, soon after the advent of constraints as a knowledge representation formalism, it has turned out that there is a particular problem in using them for representing practical problems: real world problems, when taken literally, tend to be inconsistent. Usually, there are so many features which one would like or even require the solution of a problem to have, that there really *is* no solution—the problem is overspecified.

Motivated by practical applications as described in [1, 4, 5], an idea developed that has later been called *constraint relaxation* [3] or *partial constraint satisfaction* [7]. The idea is that in fact not every constraint in a constraint network must be satisfied in what one would accept as a solution of the problem, but that there are more or less "hard" constraints, where the hard ones are required to be satisfied, but not necessarily the soft ones. In this paper, these and similar ideas of handling inconsistent constraints are analyzed. In particular, we investigate the applicability of approaches in nonmonotonic reasoning to formalize different forms of constraint relaxation.

The idea of distinguishing soft and hard constraints bears some resemblance to recent approaches of formalizing default reasoning [2, 11] where defaults are represented as standard first order formulas. To achieve the desired behavior of default reasoning, inconsistencies in the set of premises are resolved by disregarding some formulas (defaults) considered to be less important than others. We will see that the techniques used for default reasoning can directly be applied to formalize constraint relaxation. However, this formalization has its limitations. It turns out that to handle inconsistent constraint problems in a satisfactory way, often more is needed than a mere ordering of the constraints according to their importance.

There seem to be two principal ways of handling inconsistent constraint networks. We can leave the classical notion of a constraint unchanged and try to pick out a good non-solution, or we can introduce a new definition of constraints which guarantees the existence of solutions but which differ in quality. These approaches are closely related, and it is mainly a practical issue where to put the information needed to resolve inconsistencies. Nevertheless, we will distinguish these approaches for clarity and discuss them separately.

After presenting in section 2 the necessary formal background of constraints and preferred subtheories, we pursue the first approach in section 3. Constraint networks are extended by different orderings containing information about what to do in cases of inconsistencies. We discuss three such orderings. The first one is an ordering on maximal consistent subsets of the constraints and is a direct adaptation of the preferred subtheory approach in [2]. The next, more general approach introduces an ordering on the solutions of maximal consistent subsets of the constraints. The third further generalization admits arbitrary subsets of the constraints, not just maximal

ones. These formulations of constraint relaxation may also be viewed as further developments of the theoretical formulations in [7, 9, 10].

It turns out that, in the most general case, the role of constraints in the first approach almost vanishes. It therefore seems appropriate to introduce a new notion of constraints where constraints are partial orderings on value assignments. This approach will be presented in section 4 where we also show how it can be "implemented" using standard constraint techniques when numerical values representing the degree of violation of a constraint are used to represent partial orderings. Section 5 concludes.

2 Background

In this section, we will introduce the basic notation and definitions underlying our treatment of inconsistent constraint networks. We will first present the standard constraint terminology restricting ourselves to finite constraint networks. We then reconsider the notion of preferred subtheories as defined in [2]. This approach to default reasoning will be used repeatedly in the next sections.

Usually, a constraint is defined as a relation R on a set of variables $\{v_1, \ldots, v_m\}$ over domains $\{D_1, \ldots, D_n\}$, i.e., $R \subseteq D_1 \times \cdots \times D_n$, and a constraint network as a set of constraints with shared variables. Since we are here only interested in whether some assignment of values to the variables of the constraints is admissible or not, we can reduce the definition of a constraint network to the following:

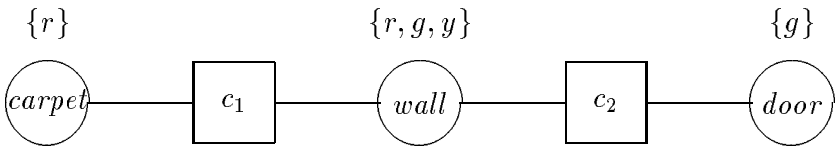
Definition 1 (Constraint Network)
A constraint network is a triple (C, V, D) *where*

1. $V = \{v_1, \ldots, v_n\}$ *is a set of variables,*
2. $D = \bigcup_{1 \leq i \leq n} D_i$ *is a set such that* D_i *is the domain of* v_i*, and*
3. $C = \{c_1, \ldots, c_k\}$ *is a set of constraints, i.e., functions from value assignments to* $\{0, 1\}$ *where a value assignment is a function assigning to each variable* v_i *an element of its domain* D_i*.*

Definition 2 (Solution)
Let $CN = (C, V, D)$ *be a constraint network with* $C = \{c_1, \ldots, c_k\}$*. A value assignment* Val *is a solution of* CN *iff for every* i $(1 \leq i \leq k)$ $c_i(\mathrm{Val}) = 1$*.*

A *constraint satisfaction problem* (CSP) is the problem of finding one, some, or every solution of a constraint network. A constraint network is called *consistent* if it has a solution, inconsistent otherwise.[1]

[1] The notion of consistency used here should not be confused with notions of k-consistency etc. as defined in [6].



Val	c_1(Val)	c_2(Val)
$(carpet \mapsto r, \quad wall \mapsto r, \quad door \mapsto g)$	1	0
$(carpet \mapsto r, \quad wall \mapsto y, \quad door \mapsto g)$	0	0
$(carpet \mapsto r, \quad wall \mapsto g, \quad door \mapsto g)$	0	1

Figure 1: A constraint network for the room painting example and its corresponding functional definition. Both c_1 and c_2 are equality constraints.

We will use the following somewhat minimalistic example throughout the paper. Assume you want to paint your room. You have three different colors available: red (r), green (g), and yellow (y). There are two constraints: c_1, demanding that the color of the wall match the color of the carpet, which is red, and c_2, demanding that the color of the wall match the color of the door, which is green. In figure 1, we show the respective constraint network in a traditional style, adding an excerpt from the functional definition. Obviously, the constraint network is inconsistent.

To make constraint satisfaction techniques more easily comparable to default reasoning, we will sometimes switch to a more logically oriented view of constraints. Due to our restriction to finite constraint networks, a constraint can be viewed as as a disjunction of the form

$$[v_1 = d_{11} \wedge \ldots \wedge v_n = d_{1n}] \vee \ldots \vee [v_1 = d_{l1} \wedge \ldots \wedge v_n = d_{ln}]$$

Note that constraint variables are mapped to logical constants. A constraint network can, in the logical view, simply be seen as a set of constraints. A solution of a set of constraints C is a ground formula s of the form

$$v_1 = d_1 \wedge \ldots \wedge v_n = d_n$$

such that $d_i \in D_i$ and $C \cup \{s\}$ is logically consistent.

The logical version of the constraint c_1 in the room painting example would be

$$carpet = r \wedge wall = r \wedge door = g$$

which would also be a solution for the constraint subset consisting just of the constraint c_1.

Classical constraint networks are monotonic in the sense that the addition of a constraint always reduces the number of solutions, i.e., s is a solution of $C \cup \{C_j\}$

only if s is a solution of C. The networks described in the rest of this paper do not have this property, i.e. they are nonmonotonic.

If there is no solution, we are often interested in solutions of a CSP N' for which a solution exists and that is "as close as possible" to N. In this case, N' will be called a relaxation substitute of N. In the next section, we will discuss various ways of defining such substitutes.

We will next define the nonmonotonic formalism used in the next sections to formalize reasoning with inconsistent constraints, Brewka's preferred subtheory approach [2], which generalizes earlier work of [11]. This approach is based on an ordering of the maximally consistent subsets (subtheories) of a set of premises. Provability can be defined as follows:

Definition 3 (Weak and Strong Provability)
Let T be a set of premises, $<$ a strict partial ordering of $Sub(T)$, the set of maximally consistent subsets (subtheories) of T. A formula p is weakly (strongly) provable from T iff it is provable from at least one (all) of the $<$-minimal elements of $Sub(T)$.

In [2], various ways of generating the ordering of the subtheories from orderings of the premises of a given theory are investigated. For instance, it is shown that Poole's framework for default reasoning can be obtained by defining the preferred subtheories to be those containing a distinguished set of facts. We will restrict our presentation to one approach, the so-called level default theories. Here, the set of available premises P is divided into different reliability levels $P_1, \ldots, P_n$ with the intended meaning that a formula in P_j is of greater reliability than a formula in P_k iff $j < k$. The preferred (i.e., $<$-minimal) subtheories are then defined as follows:[2]

Definition 4 (Preferred Subtheories)
Let $P = P_1, \ldots, P_n$ be a level default theory. $S = S_1 \cup \ldots \cup S_n$ is a preferred subtheory of P iff for all $i (1 \leq i \leq n)$ the set $S_1 \cup \ldots \cup S_i$ is a maximally consistent subset of $P_1 \cup \ldots \cup P_i$.

[2] contains a further generalization to arbitrary partial orderings of the premises. This extension is rather straightforward and will not be discussed in the present paper.

With this formal background, we are in a position to define various approaches to reasoning with inconsistent constraints.

3 Choosing the Best Non-Solution

One of the ways to deal with situations where no solution of a particular constraint problem exists, is to pick out one of the non-solutions of the problem. The question,

[2] This defines the $<$-minimal elements directly without specifying $<$. A corresponding definition of $<$ would be: $M_1 < M_2$ iff there is an i such that $M_1 \cap (P_1 \cup \ldots \cup P_{i-1}) = M_2 \cap (P_1 \cup \ldots \cup P_{i-1})$ and $M_2 \cap P_i$ is a proper subset of $M_1 \cap P_i$.

of course, is: Which of the non-solutions is good? The information about the right choice must be provided somehow. A first idea that comes to mind, is to select one of the solutions of a maximally consistent subset C' of the set of constraints C. This alone is often insufficient, however, as it does not allow to distinguish between important constraints that we require to be satisfied, and less important ones whose violation is more acceptable.

We will therefore adopt the ideas underlying the preferred subtheory approach to this problem. This leads to the following definition:

Definition 5 (Approximate Solution 1)
Let C be a (possibly inconsistent) set of constraints, and $<$ a strict partial ordering on Sub(C)*, the set of maximally consistent subsets of C. We say s is an approximate solution of C iff s is a solution of C' where C' is a $<$-minimal element of* Sub(C).

This definition leaves open how to define the ordering on the maximally consistent subsets. A solution in the spirit of [11] might distinguish between two classes of constraints: the hard constraints that have to be satisfied and the soft constraints that are only satisfied if possible. We will describe a more general approach here, which applies the ideas of level default theories as described in the last section. We split the set of constraints C into levels C_1, C_2, ..., and require that a relaxation substitute for C consist of a maximally consistent number of constraints from C_1, augmented by a maximal number of constraints from C_2 consistent with those chosen so far, etc. This approach is very similar to the ones described in [4, 9].

Definition 6 (Preferred Subsets of Constraints)
Let $C_1, \ldots, C_n$ be a partition of a set of constraints C into levels of importance. $S = S_1 \cup \ldots \cup S_n$ is a preferred subset of C iff for all i $(1 \leq i \leq n)$, the set $S_1 \cup \ldots \cup S_i$ is a maximally consistent subset of $C_1 \cup \ldots \cup C_i$.

As mentioned before, [2] also describes a more general approach based on partial orderings of the premises instead of reliability levels. It uses total linearizations of the partial orderings. Its application to constraints is similar to the one above and will be omitted here.

Unfortunately, this direct adaptation of ideas from default reasoning to reasoning with inconsistent constraints has its limitations. In the approach defined so far, a formula (default or constraint) is either valid, or invalid and disregarded. For default reasoning purposes this is entirely reasonable. For constraints, a somewhat more flexible approach is desirable and possible. Instead of entirely disregarding a constraint, one might want to replace it by a weaker one, where a constraint c is weaker than d if c accepts more value asignments than d, or, in logical terms, if c contains all the disjuncts of d plus some additional ones. Assume, in our room painting example, that you want to relax the constraints in order to make the network consistent by admitting that *carpet* is considered matching the color of the wall if

wall $= g$ and *carpet* $= r$. This would mean to introduce an additional disjunct, namely *carpet* $= r \wedge$ *wall* $= g \wedge$ *door* $= g$ into the logical formulation of c_1.

In general, this is equivalent to disregarding unsatisfiable constraints and preferring some of the solutions of the now consistent problem to others. We therefore order the *solutions* of maximally consistent subsets of the constraints instead of the maximally consistent subsets themselves in the following definition:

Definition 7 (Approximate Solution 2)
Let C be a set of constraints, $<$ an ordering of the solutions of Sub(C)*, denoted* Sol(Sub(N))*. s is an approximate solution of C if it is a $<$-minimal element of* Sol(Sub(N))*.*

In our example, we might order the solutions of maximally consistent subsets of the constraints such that solutions where *wall* $= g$ are preferred to such solutions where this is not the case.

This definition is already quite general. It still requires, however, that the relaxation substitutes be maximally consistent subsets of the constraints. In some cases, even this requirement might be too strong. Imagine a situation where we have to choose between "heavily" violating a small set of constraints and "weakly" violating a superset of these constraints. There may be situations where the harmless violation of more constraints is preferable to the harmful violation of fewer constraints.

There are two maximally consistent subsets, each consisting of exactly one of the two constraints. Our last definition requires that our approximate solution be among the solutions of these subsets, i.e., we only have the choice between r and g. But y might be the better solution in this case because although it violates both constraints this violation might be more tolerable than any of the other solutions.

To be able to handle such examples in our framework, we have to take solutions of nonmaximal subsets of the constraints into account. Our room painting example shows that even the case where no constraint at all is satisfied, has to be considered. This amounts to ordering all possible value assignments and leads to the following definition:

Definition 8 (Approximate Solution 3)
Let C be a set of constraints, $<$ an ordering of the value assignments satisfying the condition: if C is consistent then s is $<$-minimal iff s is a solution of C. s is an approximate solution of C iff it is a $<$-minimal value assignment.

In our example, we now can give the value assignment that assigns the value y to the variable *color* preference over all other assignments and thus obtain the desired result. Obviously, the role of the constraints almost vanishes in this last and so far most general definition. Almost all the necessary information resides in the ordering of value assignments. The only role left for the constraints is to guarantee that this ordering prefers a solution of the constraints, if the constraints are consistent.

In the next section, we will therefore introduce a new notion of constraint networks that generalizes the classical one. The information of how to handle inconsistencies will become part of the constraints, thus giving them back their original importance.

4 A New Notion of Constraints

We first introduce a generalized definition of constraint networks. The intuitive idea underlying this definition can be described as follows: a constraint is considered as a judge with certain preferences for value assignments. A constraint network is a committee of judges determining the quality of a value assignment based on the preferences of the individual judges. A specific combination rule is used to determine the decision of the committee from the individual preferences. In a classical network, the judges can only describe their preferences by classifying the value asignments as either acceptable or unacceptable. The combination rule underlying classical networks is the rule of veto, i.e., if there is a single judge disagreeing with a value assignment, then this assignment is not considered as a solution. In the general case, we admit arbitrary strict partial orderings of value assignments as the votes of a judge, and we admit more interesting combination functions. Let us make these intuitions precise:

Definition 9 (Generalized Constraint Network)
A constraint network is a quadruple (C, V, D, Comb) *where*

1. $V = \{v_1, \ldots, v_n\}$ *is a set of variables,*
2. $D = \bigcup_{1 \leq i \leq n} D_i$ *is a set such that* D_i *is the domain of* v_i*, and*
3. $C = \{c_1, \ldots, c_k\}$ *is a set of constraints where each constraint is a strict partial order on value assignments, and*
4. Comb *is a combination function that, given* C*, produces a single partial order on value assignments.*

Definition 10 (Solution)
Let $CN = (C, V, D, \mathrm{Comb})$ *be a constraint network. A value assignment* Val *is a solution of* CN *iff* Val *is a minimal element of the partial order* $\mathrm{Comb}(C)$*.*

Note that these definitions guarantee the existence of a solution; in other words, there are no inconsistent constraint networks any more. To be able to work with these definitions in practice, we have to define ways of specifying constraints, i.e., partial orderings on value assignments, and the combination rule. Interestingly, the level default theory approach can be used here as well if the preference orderings of the constraints can be represented as levels.

Let C be a set of constraints, each constraint c_j assigning a preference level $P_{c_j(\mathrm{Val})}$ to every value assignment Val. We can generate a level default theory T in the

following way: for each c_j in C and each P_i put a formula p into level T_i of the level default theory such that p is the disjunction of all value assignments (represented as conjunctions) that are assigned to the level P_i by constraint c_j, i.e., all value assignments Val for which $c_j(\mathrm{Val}) = i$.

The definition of preferred subtheories implicitly specifies a combination rule, which picks out a solution that is among the most preferred of at least one constraint, and—if possible—among the second best solutions of another constraint, and—if possible—among the third best solutions of yet another constraint, and so on. It should be noted that a constraint in this approach does not, as in the last section, correspond to a formula in one level, but contributes a formula to each level.

But there is again a limitation. If we reconsider our painting example, it is not difficult to see that, again, the preferred subtheory approach does not provide a way of preferring the solution $wall = y$ to all other solutions. We therefore have to investigate other ways of representing orderings.

One particularly convenient way of expressing priority orderings in constraint networks, is to introduce some kind of metric (see also [7]). We can, for instance, specify the quality of a value assignment by using integers interpreted as penalties, indicating the degree to which a particular value assignment violates the constraint. This can easily be modeled using the Comb function: the numbers associated with the value assignments are used to determine a numerical measurement of the overall quality of a certain solution. The combination rule Comb can, for instance, be given as a weighted sum of the numerical values obtained from each constraint, i.e., for a set of constraints $C = \{c_1, \ldots, c_k\}$ and a value assignment Val we can define

$$\mathrm{Comb}(c_1, \ldots, c_k)(\mathrm{Val}) = w_1 c_1(\mathrm{Val}) + \ldots + w_k c_k(\mathrm{Val})$$

where w_j is a weight indicating the importance of constraint c_j. Following the convention that a constraint assigns the penalty 0 to a value assignment, whenever it belongs to its most preferred ones, then this definition guarantees that value assignments violating no constraint, if they exist, get the value 0 and are thus preferred to solutions violating at least one constraint. Violating a constraint here means not belonging to its most preferred value assignments.

Note that in the special case of defining priority orderings by penalty functions, the third item of definition 9 may be notated as follows, which is in analogy to definition 1:

3'. $C = \{c_1, \ldots, c_k\}$ is a set of constraints where each constraint is a function from value assignments to $\mathbb{N}$.

However, the value 0 changes its meaning here: in definition 1, a value 0 as result of c_i meant that the constraint was violated; now, as the value of c_i is to be interpreted as penalty, the value 0 signifies that the respective constraint is satisfied optimally.

Applied to the room painting example of figure 1, the penalty functions might be defined as follows:

Val	c_1(Val)	c_2(Val)
$(carpet \mapsto r, \quad wall \mapsto r, \quad door \mapsto g)$	0	8
$(carpet \mapsto r, \quad wall \mapsto y, \quad door \mapsto g)$	3	3
$(carpet \mapsto r, \quad wall \mapsto g, \quad door \mapsto g)$	8	0

When defining each weight w_i in Comb to be equal 1, we obtain the combination rule

$$\text{Comb}(c_1, c_2)(\text{Val}) = c_1(\text{Val}) + c_2(\text{Val})$$

which prefers the assignment $wall = y$ over $wall = r$ as well as $wall = g$.

Interestingly, the (meta)-problem of finding a solution with its degree of quality can be represented in a standard constraint network. Every solution of this new network corresponds to a solution of a relaxation of the original problem, and additionally contains the necessary numbers to compute its quality.

One straightforward way to achieve this, is to construct for a given numerical generalized constraint network

$$CN = (V, D, C, \text{Comb})$$

a corresponding (classical) constraint network

$$CN' = (V', D', C')$$

in the following way:

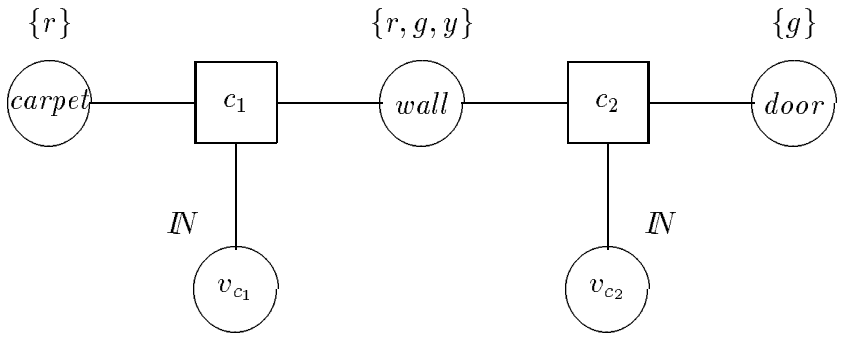
1. $V' = V \cup \{v_{c_1}, \ldots, v_{c_k}\}$ where $k = |C|$, i.e., k is the number of constraints in CN.

2. $D'_{c_j} = I\!N$, for $1 \leq j \leq k$. The domains of all variables in V remain unchanged.

3. $C' = \{c'_1, \ldots, c'_k\}$ and $c'_i(\text{Val}) = 1$ iff 1) $c_i(\text{Val}|V) = n$, and 2) $\text{Val}(v_{c_i}) = n$.

In figure 2, we show a relaxed version for the room painting example with additional variables and the corresponding functional description.

Standard constraint satisfaction algorithms like the one in [8] can be applied to the new constraint network. If all variables v_{c_j} have the value 0, we have found a solution where no constraint in the original problem is violated. If some of these variables are greater than 0, we can use these numbers to compute the overall quality of a solution according to the rule Comb. It thus can be ensured that a "good" solution is found.



Val					c_1(Val)	c_2(Val)
$(carpet \mapsto r,$	$wall \mapsto r,$	$\cdots,$	$v_{c_1} \mapsto 0,$	$\cdots)$	1	$\cdots$
$(carpet \mapsto r,$	$wall \mapsto r,$	$\cdots,$	$v_{c_1} \mapsto 1,$	$\cdots)$	0	$\cdots$
$(carpet \mapsto r,$	$wall \mapsto r,$	$\cdots,$	$v_{c_1} \mapsto 2,$	$\cdots)$	0	$\cdots$
$\vdots$	$\vdots$		$\vdots$			
$(carpet \mapsto r,$	$wall \mapsto y,$	$\cdots,$	$v_{c_1} \mapsto 0,$	$\cdots)$	0	$\cdots$
$\vdots$	$\vdots$		$\vdots$			
$(carpet \mapsto r,$	$wall \mapsto y,$	$\cdots,$	$v_{c_1} \mapsto 3,$	$\cdots)$	1	$\cdots$
$(carpet \mapsto r,$	$wall \mapsto y,$	$\cdots,$	$v_{c_1} \mapsto 4,$	$\cdots)$	0	$\cdots$
$\vdots$	$\vdots$		$\vdots$			
$(carpet \mapsto r,$	$wall \mapsto g,$	$\cdots,$	$v_{c_1} \mapsto 0,$	$\cdots)$	0	$\cdots$
$\vdots$	$\vdots$		$\vdots$			
$(carpet \mapsto r,$	$wall \mapsto g,$	$\cdots,$	$v_{c_1} \mapsto 8,$	$\cdots)$	1	$\cdots$
$(carpet \mapsto r,$	$wall \mapsto g,$	$\cdots,$	$v_{c_1} \mapsto 9,$	$\cdots)$	0	$\cdots$
	$\vdots$					
$(\cdots,$	$wall \mapsto r,$	$door \mapsto g,$	$\cdots,$	$v_{c_2} \mapsto 0)$	$\cdots$	0
	$\vdots$	$\vdots$		$\vdots$		
$(\cdots,$	$wall \mapsto r,$	$door \mapsto g,$	$\cdots,$	$v_{c_2} \mapsto 8)$	$\cdots$	1
$(\cdots,$	$wall \mapsto r,$	$door \mapsto g,$	$\cdots,$	$v_{c_2} \mapsto 9)$	$\cdots$	0
	$\vdots$	$\vdots$		$\vdots$		

Figure 2: Relaxed version of the room painting example of figure 1. v_{c_1} and v_{c_2} represent the qualities of assignments to *carpet*, *wall*, and *door*.

5 Conclusions

We have discussed two ways of dealing with inconsistent constraints. In the first approach, the notion of a constraint remained unchanged. In case of an inconsistency, a good non-solution was chosen. To define the quality of non-solutions, various preference relations had to be introduced. In the most general case, this lead to a situation, where the role of the constraints almost vanished. We therefore proposed a second approach in which the notion of constraints has been generalized to be just a partial order on value assignments.

Our presentation intended to distinguish between these two approaches for the sake of clarity. In fact, they are more closely related than our presentation might suggest. Of course, it is mainly a matter of convenience whether we want to add the necessary priority information separately, as in the first approach, or whether we want to make it part of the definition of a constraint, as in the second approach. It is often easy to switch forth and back between these two views. Note, for instance, that the numerical representation of partial orderings presented at the end of the last section as a way of implementing the second approach, can as well be interpreted as a way of computing the ordering of non-solutions, i.e., as an implementation of the first approach.

In both approaches, ideas developed to formalize defeasible reasoning in general proved to be applicable, in particular the preferred subtheory approach to default reasoning. However, their application to constraint problems had its limitations. More general ways of defining the necessary priority orderings proved to be useful for reasoning with inconsistent constraints.

References

[1] A. Borning, R. Duisberg, B. Freeman-Benson, A. Kramer, and M. Woolf. Constraint hierarchies. In *Proc. 1987 ACM Conference on Object-Oriented Programming Systems, Languages and Applications*, pages 48–60, Orlando, Florida, 1987.

[2] G. Brewka. Preferred subtheories: An extended logical framework for default reasoning. In *Proc. of the 11 th IJCAI*, pages 1043–1048, Detroit, Michigan, 1989.

[3] R. Dechter and J. Pearl. Network-based heuristics for constraint-satisfaction problems. *J. Artificial Intelligence*, 34:1–38, 1987.

[4] Y. Descotte and J.C. Latombe. Making compromises among antagonist constraints in a planner. *J. Artificial Intelligence*, 27:183–217, 1985.

[5] B.N. Freeman-Benson, J. Maloney, and A. Borning. An incremental constraint solver. *Communications of the ACM*, 33:54–63, 1990.

[6] E.C. Freuder. Synthesizing constraint expressions. *Communications of the ACM*, 21:958–966, 1978.

[7] E.C. Freuder. Partial constraint satisfaction. In *Proc. of the 11 th IJCAI*, pages 278–283, Detroit, Michigan, 1989.

[8] H.W. Guesgen. *CONSAT: A System for Constraint Satisfaction.* Research Notes in Artificial Intelligence. Morgan Kaufmann, San Mateo, California, 1989.

[9] J. Hertzberg, H.W. Guesgen, and H. Voss A. Voss, M. Fidelak. Relaxing constraint networks to resolve inconsistencies. In *Proc. of the 12 th GWAI*, pages 61–65, Eringerfeld, Germany, 1988.

[10] W. Hower. On conflict resolution in inconsistent constraint networks. Diplomarbeit, Universität Kaiserslautern, Kaiserslautern, Germany, 1989.

[11] D. Poole. A logical framework for default reasoning. *J. Artificial Intelligence*, 36:27–47, 1988.