Explanations for Whom? *

Richard J. Wallace and Eugene C. Freuder

Constraint Computation Centre, University College Cork
Cork, Ireland
rjw-, ecf@cs.unh.edu

Abstract. We consider some issues raised by recent work on giving explanations and advice in connection with constraint based solvers. Our main focus here is on the manner in which information related to explanations is organized and presented to the user (the *presentation problem*) as opposed to the logical problem of correct characterization of explanations and related ideas. We discuss different means of representing explanations and potential principles relating explanations in their logical form to information presented to the user. We also discuss problems of scaling, especially in regard to the authors' earlier work on explanation trees.

1 Introduction

Recently, the generation of explanations during problem solving has become a topic of interest in the constraints community. A major emphasis has been the explanation of conflicts, where either the original problem or user selections have led to a condition that cannot be extended to a full solution. Systems have been developed in the context of constraint programming that present the constraints involved in a conflict, so the user can then decide how to change the current state of the problem [Jun01] [JB00]. Other work has examined the means of restoring conflict situations to non-conflict states [AFM00]. The work of the present authors and their collaborators has been concerned with the manner in which explanations can be organized and presented to the user [FLW00] [FLW01]. This has also been the concern of earlier work in concurrent engineering [Bow97].

The major motivation at this time for explanations and related features is to inform and guide a human user in the course of interactive problem solving. However, a hiatus is already developing between formal accounts of explanation and the problem of presenting information to a user in a useful way. The key issue here can be designated as the means by which explanations can be given greater perspicuity. Another way of putting this is to say that definitions of explanation, such as those in the references cited above, do not distinguish between good and bad ways of explaining, in the sense of facilitating or detracting from the user's ability to understand how the information in the explanation given actually implies the facts to be explained, as well as enhancing or diminishing his ability to use this information to alter the current problem appropriately. (This paper will avoid discussion of philosophical problems concerning the

* The second author is supported by a Principal Investigator Award from Science Foundation Ireland

definition of explanation, since the definitions that have appeared in the CSP literature seem logically and intuitively sound.)

Thus, the purpose of present paper is to analyze more deeply what we are doing when we present explanations and to compare various approaches with respect to the issues just raised.

In the next section, we consider definitions of explanation that have been given in recent work, as well as the varieties of explanations that can be distinguished. Section 3 considers how presentation is related to the underlying logical form of explanations. Section 4 considers the perspicuity problem directly. Section 5 discusses problems involved in generalizing earlier work, especially that of the present authors. Section 6 briefly adumbrates related issues.

2 Definition and Classification of Explanations

[AFM00] give the following definition of explanation (here in paraphrase): A constraint set, E, is an explanation of L, where L is "a constraint on some variables of X", if the CSP, $\langle X,\ D,\ C\ \cup\ E\rangle$ is consistent, while the CSP $\langle\ X,\ D,\ C\ \cup\ E\ \cup\ \{\neg L\}\rangle$ is inconsistent. X, D, and C refer to the variables, domains and constraints of the original CSP. An example of this is when E is a set of domain restrictions and L is the constraint that only one value can be used to label a particular variable. In a similar vein, [FLW01], [Jun01], and [JB00] have defined explanation in terms of sufficiency or entailment. Thus, a set of problem features that entails a given result can be considered an explanation of that result.

[JB00] describe two categories of explanation, which they call "contradiction explanations" and "eliminating explanations", but these are more related to the kinds of facts to be explained (conflicts and assignments, repectively) than to the character of the explanation itself. However, it is convenient to use one of these labels, "eliminating explanations", as a generic label for explanations that conform to the definition of [AFM00].

What kinds of questions call for eliminating explanations? Here are some examples from [FLW01]:

- why did we get this as a solution?
- why did this choice of labels lead to a conflict?
- why was this value chosen for this variable during processing?

In addition, in the other references cited above, the questions discussed are almost always of the second type, which calls for explanations of conflicts. Not surprisingly, we can conclude that questions that require eliminating explanations are always questions about selections from the elements of the original problem.

However, these are certainly not the only questions possible in the context of solving CSPs. Here are some others:

- is this a viable value? (i.e. does it occur in any solution?)
- why are there so many solutions to this problem?

If the answer to the first question is “Yes”, an explanation for this state of affairs could be a set of values consistent with this value that are also clearly possible assignments. In the limiting case, this would be a solution that includes the value in question. The second question could occur in a context where the system has informed the user that there are more than N solutions. Here, an explanation might involve noting that certain variables with few constraints allow numerous partial solutions. (The latter question raises another point, which is the *precision* of questions, as well as explanations. Unfortunately, at this time we cannot do more than bring this to the reader’s attention.) In both cases, the answer involves problem features that support the condition in question; hence, they can be included in a second class, which could be called “supporting explanations”.

Some further issues have been raised by the work of [FLW00]. From this work, it appears that explanations can vary in extent (specifically, in the number of inferences involved) and that sets of explanations can be organized into meaningful structures.

This work has focused on explanations for assignments. When a value within a larger set of values must be assigned, this means that the remaining values have been eliminated because of selections already made during search. From the set of earlier assignments we can obtain an *immediate explanation* that entails this new assignment. (In terms of the definition given at the beginning of this section, the prior assignments constitute provisional constraints (E) on these domains, and the new assignment reflects the constraint L.) The elements in the immediate explanation may, in turn, have their own explanations (unless they were either chosen by the user or given in the original problem description), and this process can be iterated. This means that explanatory elements can be linked to form a network of elements or an *extended explanation*, which in its fully extended form, where all elements are either themselves explained or givens, is a complete explanation of the assignment in question. At the same time, this structure can be viewed as an organized set of explanations, where assignments at level k have immediate explanations at level $k + 1$.

In the scheme of [FLW00] (and [FLW01]), cycles within explanations are avoided because explanations are always made in terms of existing assignments. In other words, explanations always follow the search path, which is, of course, acyclic. Explanations, therefore, take the form of *explanation trees,* which are rooted at the assignment to be explained. In this case explanations of other assignments within a tree form subtrees.

The methodology of [FLW00] excludes certain potential explanations in order to insure that those explanations that are derived take the form of trees. It could be said that this approach insures that explanations have “causal validity”, since, while the assignment to be explained might, conversely, serve as part of an explanation for a previous assignment (consistent with the basic definition of eliminating explanations), this would not have been the case when the earlier assignment was made. These considerations also appear to apply to the explanations of [Jun01] and [JB00].

The notion of causal validity can be elaborated, in that explanations (for assignments) can be classified according to whether they constitute the actual assignments that led to labeling in question, or whether they constitute assignments, produced according to the rule above, that *could have* led to this labeling. These can be denoted as “effective” and “acceptable” explanations, respectively, where the former is a special subclass of the latter. Although both have causal validity, the former also has "historical validity" in that this is an actual (sufficient) cause of the condition in question. An example is shown in Figure 1, where the problem is the 9-puzzle in which one is given a 9 by 9 board with some cells already labeled, and one must label the empty cells with numbers from 1 to 9 such that every row, column and 9-cell block (delineated by double-lines in Figure 1) has all the numbers from 1 to 9. This means that no two cells in a row, column or block can have the same label. The problem can be solved by examining labeled cells in succession and deleting these labels from the other cells in the same row, column and block. In Figure 1, the first originally empty cell whose value has been specified is (2,3), which must be given a 2. In this figure, the added label is enlarged and italicized and the cells responsible for the domain reduction are in bold-face. These cells, therefore, constitute the effective explanation for this label. At the same time, the 3 in cell (8,3) could be used instead of the one in (2,8), to form an acceptable explanation.

4		**5**			7			1
	8	*2*	**1**	**6**			**3**	
		9		5		7	4	
		7		4		1		2
9	1		5				6	
	2		8		6			5
5			4		3	2		
	7	3			1		5	
8				2		6		3

Fig. 1. Sample 9-puzzle problem, showing initial state of the puzzle.

The account of eliminating explanations given here appears to give no basis for choosing effective over non-effective, acceptable explanations. Moreover, concerns such as being able to change the current problem or even comprehensibility (cf. below), may result in explanations of the latter type being chosen over the former. On the other hand, it may sometimes be valuable to give explanations that have historical validity, and in this case it would be necessary to derive effective explanations. (There may be psychological reasons for showing the actual basis of a result: for example, the user may prefer explanations that conform to his memory of previous choices and processing. But these issues lie beyond the scope of the present paper.)

Although it is not the focus of this paper, a few words should be said about computing explanations. For eliminating explanations, the definition of [AFM00] provides a means of deciding that any given E is, in fact, an explanation of L. However, in the worst case this is an intractable problem (*ibid.*). On the other hand, the explanation trees of [FLW01] are produced by methods that guarantee correct explanations. So, it is not necessary to decide that an E derived in this way is an explanation for the assignment in question. With this approach, some trees must be updated whenever assignments are retracted or altered, but this has been tractable in practice (cf. [JB00]). For supporting explanations, there are no previous results to cite, but since this type of explanation depends on partial or complete solutions, finding such explanations will be intractable in the worst case. (The remainder of this paper will be restricted to eliminating explanations, although it would be surprising if some of the ideas discussed were not also relevant to supporting explanations.)

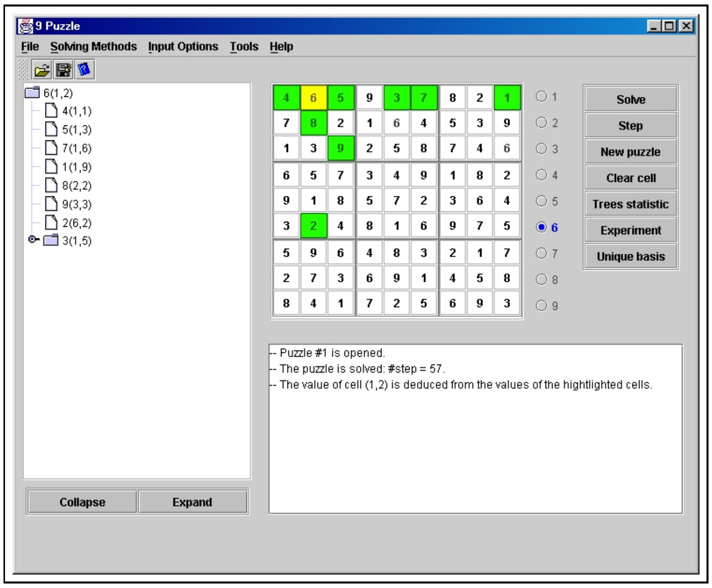


Fig. 2. Screendump from the testbed for the 9-puzzle problem, showing the solution to a problem of this type and an immediate explanation for label 6 in cell (1,2). Elements of this explanation are highlighted on puzzle board, while first level of the explanation tree is shown in lefthand panel.

3 Explanations and Explanatory Signs

The testbeds described in [FLW01] show very clearly that when we actually present explanations to the user, there are special problems involved in the presentation itself. In other words, we are faced with the problem of how to present the various parts or aspects of an explanation and how to conceptualize this process. This can be referred to overall as the "presentation problem", as opposed to the logical/ computational problem of defining and deducing explanations. Figure 2 shows a testbed for the 9-puzzle, with an immediate explanation shown on the board and in the panel on the left. In the former case it is obvious that the user must be able to comprehend the conventions involved. He must understand that each highlighted square represents a necessary element in the explanation. In addition, although the original constraints are not shown explicitly, they are also a necessary part of the explanation, and must, therefore, be 'kept in mind' while viewing the presentation.

Another example, also drawn from [FLW01] is the 9-queens problem, shown in Figure 3. Similar conventions are used in that the cells (queens or empty cells) are highlighted (here, with a light gray background). In addition, the dark queens represent user selections while the others were derived by arc consistency processing.

At this point it is useful to introduce some definitions. An *explanatory element* is a distinguishable component of the constraint representation that is used in an explanation, e.g., a single assignment or, more generally, a single constraint. An *explanatory sign* is a perceptually distinct component of the information presented to the user, that is needed to convey the explanation. ("Perceptually distinct" is, of course, vague; a tentative definition is that it is something that can be perceived as an entity by someone who does *not* know what the display means.) Presenting explanations to the user, therefore, involves presenting a collection of signs that represents or refers to an explanation.

In this context, the important questions are,

- what information must be conveyed explicitly (i.e. conveyed by explantory signs)?
- are there rules for conveying such information?

As the demo in Figure 2 shows, information can be conveyed in (possibly many) different forms. In addition, not all the requisite information needs to be conveyed explicitly; in this case, the original constraints are implicit, although at the same time the layout of the presentation serves to indicate what constraints are involved in each case.

Other systems appear to present explanations in the form of list of alphanumeric strings expressing mathematical relations (e.g., [JB00]). This includes basic constraints such as $X < Y$ and assignments such as $x = a$. Another tack is taken by [Bow97], where certain explanations "include a natural language paraphrase of the constraint", which itself is encoded as a sentence in the predicate calculus. It is of interest that in these cases, constraint information is conveyed explicitly. An important difference is that in these situations, unlike those studied by [FLW00] and [FLW01], constraints can be added and retracted, so that they cannot always be assumed.

4 Issues of Perspicuity

One of the interesting things about the testbeds described in [FLW00] and [FLW01] is their perspicuity. This seems to be based on several factors, including the small size of the problems as well as the fact that critical components of these problems, specifically their variables and even their domains as well as each separate constraint, can be given a direct visual representation. In these demos, the representation, i.e., the collection of explanatory signs, has an extraordinary brevity or efficiency. For example, it is interesting that, for the 9-puzzle, unlike the queens demo, it was not necessary to represent the entire domain explicitly, but this does not seem to be 'missing' from the representation. If the principles involved in these demos can be generalized, then it may be possible to generate comprehensible and, therefore, useful explanations in a large number of interactive settings.

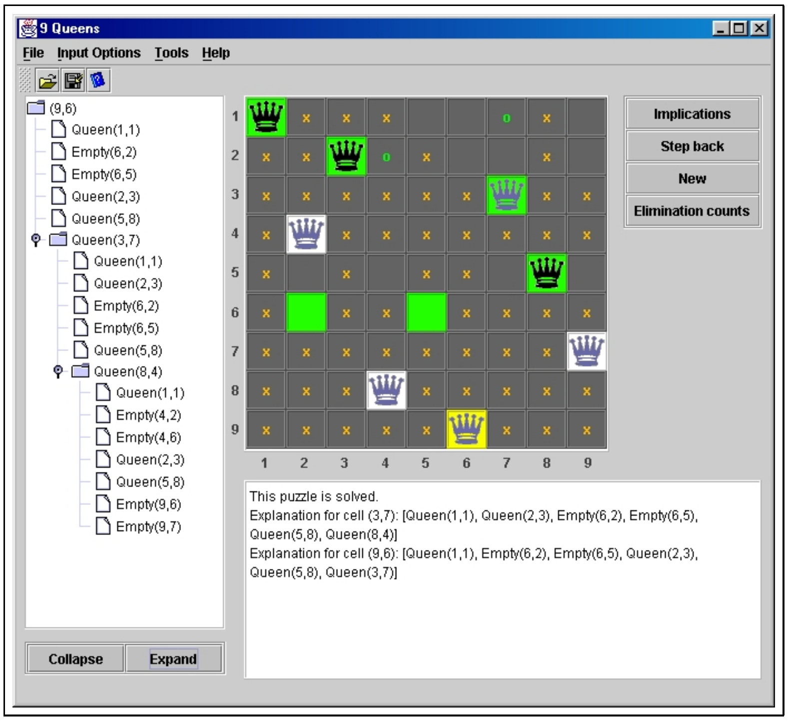


Fig. 3. Screendump from the testbed for the 9-queens problem showing an explanation for the queen in cell (9,6). Lefthand panel shows the extended explanation tree, representing a complete explanation for this assignment.

It should not be overlooked that one probable reason for the perspicuity of these demos is that the problems themselves are quite homogeneous. Because explanations are based on problem elements, this means the explanatory elements are likely to be homogeneous also. This suggests that perspicuity or non-perspicuity can be referred to two distinct levels: the level of the explanation itself, specifically, to relations among the explanatory elements, and the level of explanatory signs. This means that the presentation problem involves both levels and, therefore, cannot be restricted to the selection of signs. This point is also implied in discussions of minimality of explanations ([AFM00], [Bow97], [FLW00]), since minimal explanations are *ceteris paribus* likely to have greater perspicuity.

The demos of [FLW01] suggest some possible principles of presentation, for example:

1. Equivalent signs should have equivalent semantic content; here this means they should refer to the same kinds of explanatory elements
2. Explanatory elements should be non-redundant

(Note that each of these 'principles' has its basic reference at a different one of the two levels just described, which supports the argument of the last paragraph.) And, in addition, there is the minimality principle just referred to.

In the demos the most obvious example of the first principle is the equivalence of cells in each board; in each problem, a cell always represents the same kind of entity, either a variable (Figure 2) or a value (Figure 3). This principle is also followed in the tree representations in the lefthand panels. At the same time, in the queens demo (Figure 3) perspicuity is compromised somewhat by the inclusion of "empty cell" entities. This is because in the tree representations these are shown as separate elements in the same way that different assignments (queens in different rows) are; however, they do not act independently to eliminate values from the domain of interest.

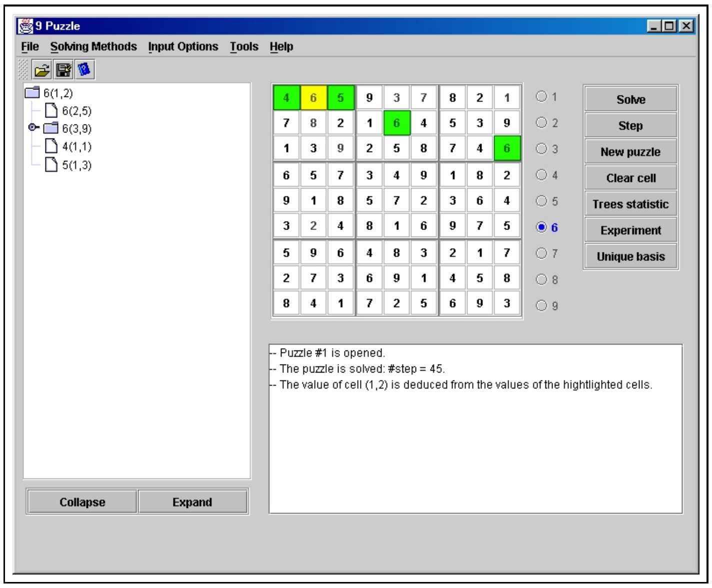


Fig. 4. Another screendump from the 9-puzzle testbed with a solution and immediate explanation for label 6 in cell (1,2). This explanation is based on a different method for solving the puzzle from that used in Figure 2.

The methods that were used to produce explanation trees are also consistent with the second principle, since explanatory elements were usually stored when they served to delete one or domain elements. (Acceptable, non-effective explanations were also generated so as to limit redundancy among explanatory elements.) Again, however, there were violations of this principle in the queens testbed (cf. Figure 3).

In [FLW00], an alternative method was used to solve the 9-puzzle, and eliminating explanations organized as explanation trees were built based on this method. The method was based on the idea that, if it can be shown that all but one cell within a block cannot be assigned a particular label, then the remaining cell must be given this value (assuming that a solution exists). In this case, an explanation for a label is composed of a set of cells, which taken together exclude that label from the other cells in the same block. An example is shown in Figure 4. In terms of the present discussion, this example is of interest because, while it is consistent with the principle of minimality (in relation to explanations based on the more direct method), the relations between explanatory elements and the assignment to be explained are in some ways less homogeneous. Thus, cells (1,1) and (1,3), labeled "4" and "5", are included in the explanation because these are prior labels and, therefore, these cells cannot be given other labels, including "6". On the other hand, cells (2,5) and (3,9), both of which have the label "6", are included by virtue of the row constraints which prevent other cells in these rows from being given the same label. This example shows quite clearly that there are tradeoffs among these perspicuity 'principles', so it will not always be easy to decide how to build the best explanations.

Some further issues arise when one considers explanation trees and extended explanations. In particular, perspicuity is gained by having the different levels (degrees of extension) of an explanation represented by different levels in the tree. This suggests that explanation trees in and of themselves have properties that result in perspicuous presentations.

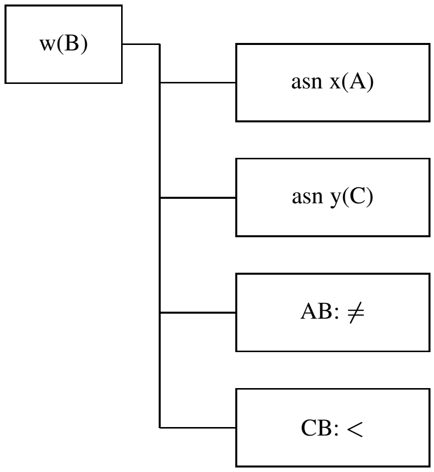
Still other issues arise in connection with the question of what to present explictly. The demos of [FLW00] and [FLW01] suggest that only certain elements need to be represented overtly, here variables, values and assignments. In other words, it appears that good explanations involve segregating some elements that the user will focus on, while the remaining elements form a kind of background. Moreover, this and other work suggests that these 'focal' elements are the ones that undergo change during the course of problem solving.

5 Questions of Generality

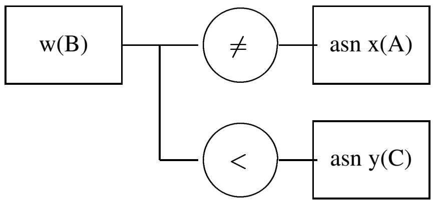
This is perhaps the most important outstanding question, especially when we consider the presentation problem and issues of perspicuity. We can divide it into two parts:

- how do we handle heterogeneous problems, in particular heterogeneous constraints?
- how do our presentation methods scale up? That is, can we develop methods for explanation that can be used with much larger problems?

In dealing with heterogeneous constraints, it is evident that we cannot rely on implicit indications as in the demos discussed here. The way in which these are represented can be guided by some of the principles adduced in the previous section. This can be shown by example. Figure 5 shows two ways of indicating the constraints in the explanation of an assignment (of value w to variable B). In the first case (Figure 5 a.), constraints are indicated by further nodes in the explanation tree. But this violates the principle of equivalence between explanatory elements and similar explanatory signs. And, if we imagine larger explanation trees of this type, we can see that they will be harder to decipher than trees in which the semantics is more homogeneous. In the second case (Figure 5 b.), constraints are indicated by labeling the arcs that link the assignment to be explained to previous assignments. We would argue that, at least in part because this form of explanation tree is consistent with the principle of equivalence, the resulting tree, while of course more complicated than those in the demos, is still reasonably easy to comprehend.



a.



b.

Fig. 5. Portions of explanation trees for problem with heterogeneous constraints. Lower case letters are assignments, upper case variables. a. Both assignments and constraints are nodes in an explanation tree. b. Only assignments are nodes, while constraints are labeled arcs between appropriate nodes.

The problem of scale can also be construed as a perspicuity problem. It is not clear from the testbeds described in [FLW00] and [FLW01] that the methods used will scale. For example, what about the 100 queens problem? An obvious difficulty here is that larger problems cannot be presented in toto on a single screen. But for that matter, what if they could be? Would this be of any use or would the user simply be overwhelmed? It appears that this will be a problem for any method; hopefully, the ideas gained from the previous demos will provide some guidance in dealing with larger problems. In particular, ideas related to focus and background must be employed to handle large problems.

6 Concluding Remarks

In closing, it should be emphasized that the ideas outlined in this paper must be subjected to empirical evaluation, although this may take us into other fields of science. More pointedly, at some point it has to be admitted that folk psychology or just plain obliviousness to these problems is insufficient. This raises a number of issues that boil down to how to test for perspicuity, and how to measure usefulness.

Another possible issue is whether there are any interesting relations between the concerns of this paper, in particular the issue of perspicuity and computational complexity. Unfortunately, at this point we can do no more than raise the question.

References

[AFM00] J. Amilhastre, H. Fargier, and P. Marquis. Consistency restoration and explanations in dynamic CSP - application to configuration. In *ECAI-2000 Workshop on Modeling and Solving Problems with Constraints*, 2000.

[Bow97] J. Bowen. Using dependency records to generate design coordination advice in a constraint-based approach to concurrent engineering. *Computers in Industry*, pages 191–199, 1997.

[FLW00] E. C. Freuder, C. Likitvivatanavong, and R. J. Wallace. A case study in explanation and implication. In *CP2000 Workshop on Analysis and Visualization of Constraint Programs and Solvers*, 2000.

[FLW01] E. C. Freuder, C. Likitvivatanavong, and R. J. Wallace. Deriving explanations and implications for constraint satisfaction problems. In T. Walsh, editor, *Principles and Practice of Constraint Programming - CP-2001. Lect. Notes in Comp. Sci.*, in press. Springer, Berlin, 2001.

[JB00] N. Jussien and V. Barichard. The palm system: explanation-based constraint programming. In *CP2000 Workshop on Techniques for Implementing Constraint Programming Systems*, pages 118–133, 2000.

[Jun01] U. Junker. Quickxplain: Conflict detection for arbitrary constraint propagation algorithms. In *IJCAI-2001 Workshop on Modeling and Solving Problems with Constraints*, pages 75–82, 2001.