Efficient Representation of Relations over Linear Constraints*

Alexander Nareyek and Ulrich Geske

GMD — German National Research Center for Information Technology
GMD-FIRST
Rudower Chaussee 5, D-12489 Berlin
EMail: {alex,geske}@first.gmd.de

Abstract

Dealing with relations over constraints is a difficult matter, because of the possible inherent disjunction. For constraints over finite domains and constraint solver mechanisms, where a direct propagation of the disjunction is impossible, a conjunctive reformulation of the disjunctions has to be found. This contribution describes an easily computable method to transform the constraints with their relations into an efficient conjunctive constraint system.

1 Introduction

Many real-life problems feature non-conjunctive relations between constraints. Figure 1 (a) shows a piecewise linear function as an example. This function has to be formulated by the use of disjunction, as the solution space is non-convex. Figure 1 (b) shows a more complex example with a network of hierarchical relations.

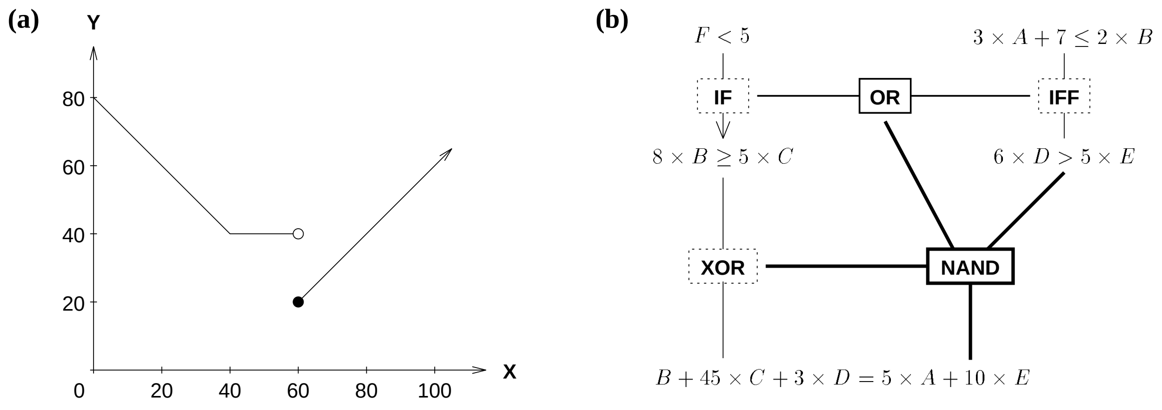


Figure 1: Examples

*This work is part of the project VERMEIL, which is funded by the German Federal Ministry for Education, Science, Research, and Technology (BMBF) under grant no. 01 IN 507 B.

A main problem in modeling those relations is the disjunction. There already exist many approaches to deal with disjunctions. Constraint Logic Programming (see [PVH89]) provides the mechanism of backtracking, whereby a simple but inefficient representation of disjunction is possible. Techniques of Linear Relaxation, as for example variations of Balas' algorithm, lead in finite domains to poor results because of the many additional constraints ([EBal85] and [HBBB93]). The application of an active propagation of the disjunction fails in many cases because of missing constraint solver support.

By a reformulation of the disjunctions it is possible to obtain an entirely conjunctive constraint system. Studies in the field of production planning point out the qualification of this method ([SBHL94] and [AlNa95]).

The reformulation of relations over constraints is a complex matter. At first, for each constraint a **base switch** is generated, which can control the constraint's validity (see figure 2), as not every constraint has to be true.

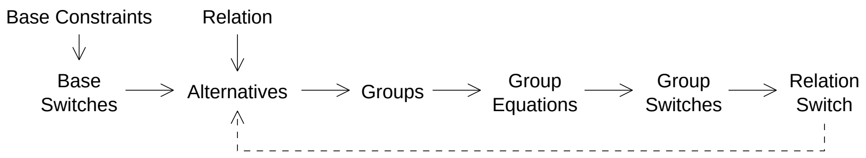
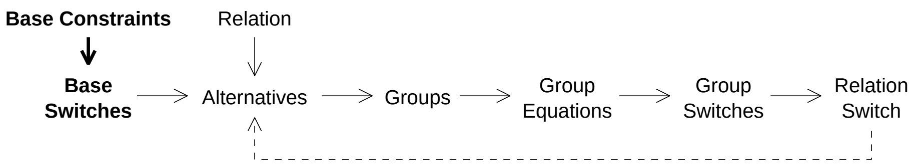


Figure 2: The reformulaton process

To express a relation over constraints (IF, IFF and XOR in figure 1 (b)), assignment **alternatives** of base switches are stated. It is possible to construct a **relation switch**, which is true iff the base switches form one of these alternatives, i.e. the relation is fulfilled. For further relations this relations switch can be used just as the base switches to formulate hierarchical relations (OR and NAND in figure 1 (b)).

2 The Generation of Base Switches



To illustrate the transformation process, the example function from figure 1 (a) shall be studied. There are seven base constraints to describe the three different segments[1]:

First segment:	$C^1: \ X < 40$	$C^2: \ X + Y = 80$	
Second segment:	$C^3: \ 40 \le X$	$C^4: \ X < 60$	$C^5: \ Y = 40$
Third segment:	$C^6: \ 60 \le X$	$C^7: \ 2 \times X = Y + 40$	

Constraints are not valid within the whole problem. For example constraint C^1 is only true within the first segment. Therefore each base constraint has to be reformulated. The reformulation contains base switch variables $S^i \in \{0, 1\}$, whose assignment

[1]The variables of the constraints are assumed to have non-negative finite domains.

represents the validity of the constraints C^i. For the seven base constraints of the example C^1 to C^7, seven base switches S^1 to S^7 are created.

For base constraints of the type $L < R$, where L and R are non-negative linear combinations, the reformulation results in two constraints: one for the case that the base constraint is true, and one if it is false (this means $L \geq R$). By an assignment of the base switch variable S^i, one constraint is fulfilled in a trivial way, the other one becomes valid:

- $C^i \equiv L < R$:[2]

$$\begin{aligned} L + S^i \times (L_{max} + 1) &< R + L_{max} + 1 && (1) \\ L + S^i \times R_{max} &\geq R && (2) \end{aligned}$$

For the type $L \leq R$ this is performed the same way:

- $C^i \equiv L \leq R$:

$$\begin{aligned} L + S^i \times L_{max} &\leq R + L_{max} && (3) \\ L + S^i \times (R_{max} + 1) &> R && (4) \end{aligned}$$

The equality $L = R$ can be expressed as $L \leq R \wedge R \leq L$. For this purpose the equations 3 and 4 are used. Both parts have to have different switch variables S_1^i and S_2^i, which are combined to the base switch variable S^i.

- $C^i \equiv L = R$:

$$\begin{aligned} L + S_1^i \times L_{max} &\leq R + L_{max} && (5) \\ L + S_1^i \times (R_{max} + 1) &> R && (6) \\ R + S_2^i \times R_{max} &\leq L + R_{max} && (7) \\ R + S_2^i \times (L_{max} + 1) &> L && (8) \\ S^i + 1 &= S_1^i + S_2^i && (9) \end{aligned}$$

To formulate $L \neq R$ the equations 5 to 8 can be applied. Only the combination in equation 9 has to be changed to:

$$S^i + S_1^i + S_2^i = 2 \qquad (10)$$

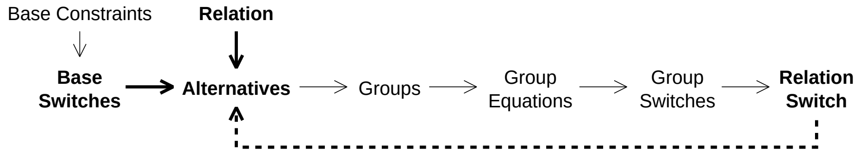
By this method the seven constraints C^i of the example can be transformed. In the case of C^1, this is done as follows:

$$\begin{aligned} X + S^1 \times (X_{max} + 1) &< X_{max} + 41 \\ X + S^1 \times 40 &\geq 40 \end{aligned}$$

This produces the base switch S^1. By a reformulation of the base constraints C^2 to C^7 the rest of the base switches are available.

[2] L_{max} and R_{max} shall indicate the maxima of the terms L and R. As these values depend on the applied tools, no concrete values can be given here.

3 A Relation as Alternatives

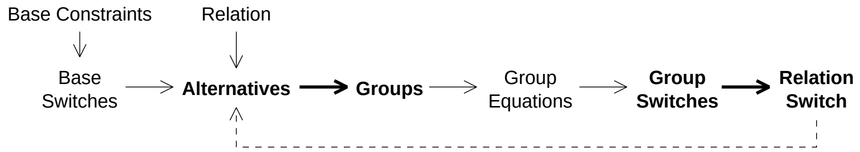


The context of the example base constraints C^1 to C^7 shall be described as only one relation, since the system is not very big. The relation is expressed by alternatives A_a, which specify all possible assignments for the appertaining base switches[3]. For the first segment of the function, an alternative A_1 has to be arranged, where S^1, S^2 and S^4 (respectively C^1, C^2 and C^4) are true, and the rest of the base switches are false. The second segment needs two alternatives, one for the point of contact (A_2) and one for the rest (A_3). The third segment needs two alternatives as well (A_4 and A_5):

	S^1	S^2	S^3	S^4	S^5	S^6	S^7
A_1	1	1	0	1	0	0	0
A_2	0	1	1	1	1	0	0
A_3	0	0	1	1	1	0	0
A_4	0	0	1	0	1	1	1
A_5	0	0	1	0	0	1	1

A relation does not have to rely only on base switches. Relation switches can also be included to express hierarchies or networks of relations. These are necessary for the relations OR and NAND in figure 1 (b).

4 The Creation of a Relation Switch



Switch variables can be introduced to represent the applicability of one alternative. To reduce the number of needed constraints and variables, the alternatives are divided into groups[4]. Then only one group switch variable per group is necessary, instead of one variable per alternative. A group switch I_g indicates the applicability of one alternative of the group G_g. As only one alternative can be true, the relation switch R can be equated with the sum of these group switches:

$$R = \sum_{g=1}^{|G|} I_g \qquad (11)$$

[3] It is possible to connect the base switches in a simpler way, because of the implicit excluding structure of the function segment constraints. Nevertheless this example is studied further since it is easy to understand.

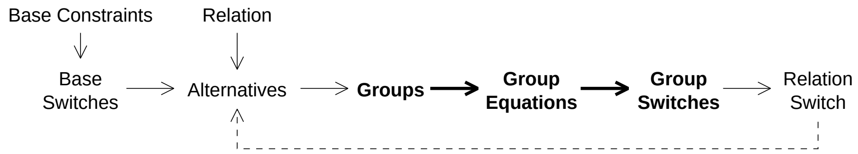
[4] The restrictions for the formation of those groups become clear later.

The relation switch R expresses the relation's validity, but some relations have to be valid, for example if they are on top of a hierarchy (e.g. NAND in figure 1 (b)). In this case the creation of the relation switch R can remain undone, as one of the alternatives has to be true:

$$\sum_{g=1}^{|G|} I_g \quad = \quad 1 \tag{12}$$

The alternatives of the example function can be divided into two groups with $G_1 = \{A_1, A_2, A_5\}$ and $G_2 = \{A_3, A_4\}$ with the group switches I_1 and I_2. There are no further superior relations. Therefore equation 12 can be applied: $I_1 + I_2 = 1$.

5 The Creation of the Group Switches



Now the modeling of the group switches I_g shall be studied. Taking the validity of the group switch for granted ($I_g = 1$), for each group a linear equation can be established which ensures an assignment for the base switches according to one alternative of the group. The appertaining base switches S^i are multiplied with coefficients α_a^i and equated with a constant Σ_g:

$$\sum_{i=1}^{|S|} \left(\alpha_g^i \times S^i\right) \quad = \quad \Sigma_g \tag{13}$$

For each group g the combination sum Σ_g and one coefficient α_g^i per base switch S^i have to be determined. Figure 4 shows the computation of these values for the example function. Pay heed to this figure for the following explanations.

Separation

After the grouping of the alternatives, some base switches have to be separated from the rest[5]. On the one hand, this has to be done for base switches, where all possible assignments of the group alternatives are zero. In the function example's second group this applies for S^1 and S^2 (dotted encirclement).

On the other hand, for each alternative of a group, there has to be one base switch separated, where only the alternative's assignment is one, and the assignments for the other alternatives are zero. For the example's first group these are S^1, S^5 and either S^6 or S^7 (solid encirclement). For the second group S^4 and either S^6 or S^7 have to be separated. The precondition for the grouping of the alternatives is given by this. A grouping is only allowed, if this separation is possible. That is why the alternatives of the example can not be in only one group altogether.

[5]The sorting of figure 4 serves only the purpose of a clear arrangement.

Determining Antecedent Coefficients

The coefficients for the not-separated base switches can be determined now. The coefficients are built sequentially, beginning with a value of one (first group: $\alpha_1^2 = 1$; second group: $\alpha_2^3 = 1$). If there are base switches, where all possible assignments are identical, their coefficients get the same values (first group: $\alpha_1^2 = \alpha_1^4$; second group: $\alpha_2^3 = \alpha_2^5$). The other coefficients are built iterative, in the way that the next coefficient at a time is the sum of the hitherto determined coefficients, added by one (first group: $\alpha_1^3 = \alpha_1^2 + \alpha_1^4 + 1 = 3$ and $\alpha_1^7 = \alpha_1^2 + \alpha_1^4 + \alpha_1^3 + 1 = 6$; second group: $\alpha_2^7 = \alpha_2^3 + \alpha_2^5 + 1 = 3$).

Calculating Combination Sum

The combination sum Σ_g is calculated by the doubled sum of the hitherto determined coefficients, added by one (see figure 4).

Determining Remaining Coefficients

Base switches, where all possible assignments of the group alternatives are zero, have been separated. The activation of those base switches has to be avoided. Therefore their coefficients are set to the combination sum, increased by one (second group: $\alpha_2^1 = \alpha_2^2 = 11 + 1 = 12$).

Now there is left one coefficient per alternative. This one is used to raise the left side of equation 13 to the combination sum Σ_g. For the first group of the example this is done as follows:

$$\begin{array}{rcccccc}
\alpha_1^1 & = & \Sigma_1 - \alpha_1^2 - \alpha_1^4 & = & 23 - 1 - 1 & = & 21 \\
\alpha_1^5 & = & \Sigma_1 - \alpha_1^2 - \alpha_1^4 - \alpha_1^3 & = & 23 - 1 - 1 - 3 & = & 18 \\
\alpha_1^6 & = & \Sigma_1 - \alpha_1^3 - \alpha_1^7 & = & 23 - 3 - 6 & = & 14
\end{array}$$

Building Group Switch

A computed group equation is valid iff the group switch is valid, i.e. one alternative of the group is applicable. Therefore the group equation is reformulated according to section 2, equations 5 to 9, with the group switch I_g instead of a base switch S^i.

6 Flattening

It is possible to flatten the network of relations. In the extreme only one relation – a disjunctive normal form – remains, as in the function example. But in most cases this causes an enormous increase of alternatives. On the other hand these alternatives can be grouped more effectively. Therefore it might be useful to flatten the relation network only partially. Smaller systems can be flattened up to the disjunctive normal form.

If the whole network is flattened to a disjunctive normal form, the transformation process can be simplified again. As the disjunction is the top relation, equation 12 instead of 11 is brought into action. This means, that one of the alternatives has to be valid at any rate. Because of the disjunctive normal form, invalid alternatives do not have to force any base switches to take a specific value anymore, as the valid alternative already assigns values to **all** base switches. Therefore the conditions for invalid alternatives can be assuaged. Regarding equation 13 the creation of a group switch can be simplified as follows (instead of applying equations 5 to 9):

- $L = R$:

$$
\begin{aligned}
L + S_1 \times L_{max} &\leq R + L_{max} & (14) \\
S_2 \times R &\leq L & (15) \\
2 \times I_g &\leq S_1 + S_2 & (16)
\end{aligned}
$$

Further improvement might be reached through the integration of techniques for variable or constraint elimination (see e.g. [SDK83] and [JLIm95]).

7 Conclusion

With the introduced algorithm an easily computable reformulation of problems with relations over constraints is possible. In this manner problems of qualitatively very high dimensions can be treated. Figure 3 shows a comparison of an implementation of the proposed method with a realization based on backtracking[6]. A relation with 16 alternatives and the worst possible reduction rate of 50% was chosen, i.e. a division in eight groups was possible. To enlarge the system, this relation was related by AND to similar relations. The base switches within the relations were all different, so that each additional relation multiplies the size of the whole solution space by 16.

By means of representative values a mean value is calculated, which states the average time to find a solution, which is given as an optimum of a goal function. Variant (a) and (b) show the results for the proposed method, where in variant (b) no reduction is performed at all, i.e. the 16 alternatives need 16 groups.

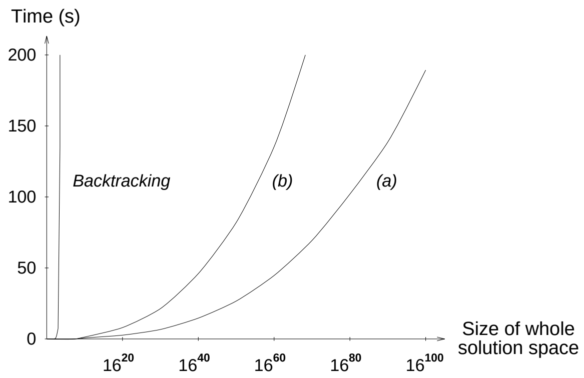


Figure 3: Comparison with 16 alternatives per dimension

The superiority of the proposed algorithm is obvious. The exponential growth of backtracking is contrasted with a polynomial growth (the second derivation decreases). The group reduction variant (a) supplies the best results. Although the reduction rate is only 50% in this case, there is a threefold acceleration.

[6] The algorithms were implemented in CHIP (see [HeSi95]).

At present the method is to be included in programs for production planning and the configuration of process control systems.

References

[PVH89] Pascal Van Hentenryck. *Constraint Satisfaction in Logic Programming*, MIT Press, 1989

[EBal85] E. Balas. *Disjunctive programming and a hierarchy of relaxation for discrete optimization problems*. SIAM journal on algebraic and discrete methods, 6(3), 1985.

[HBBB93] Henri Beringer and Bruno de Backer. *Satisfiability of Boolean formulas over linear constraints*. Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence, 1993.

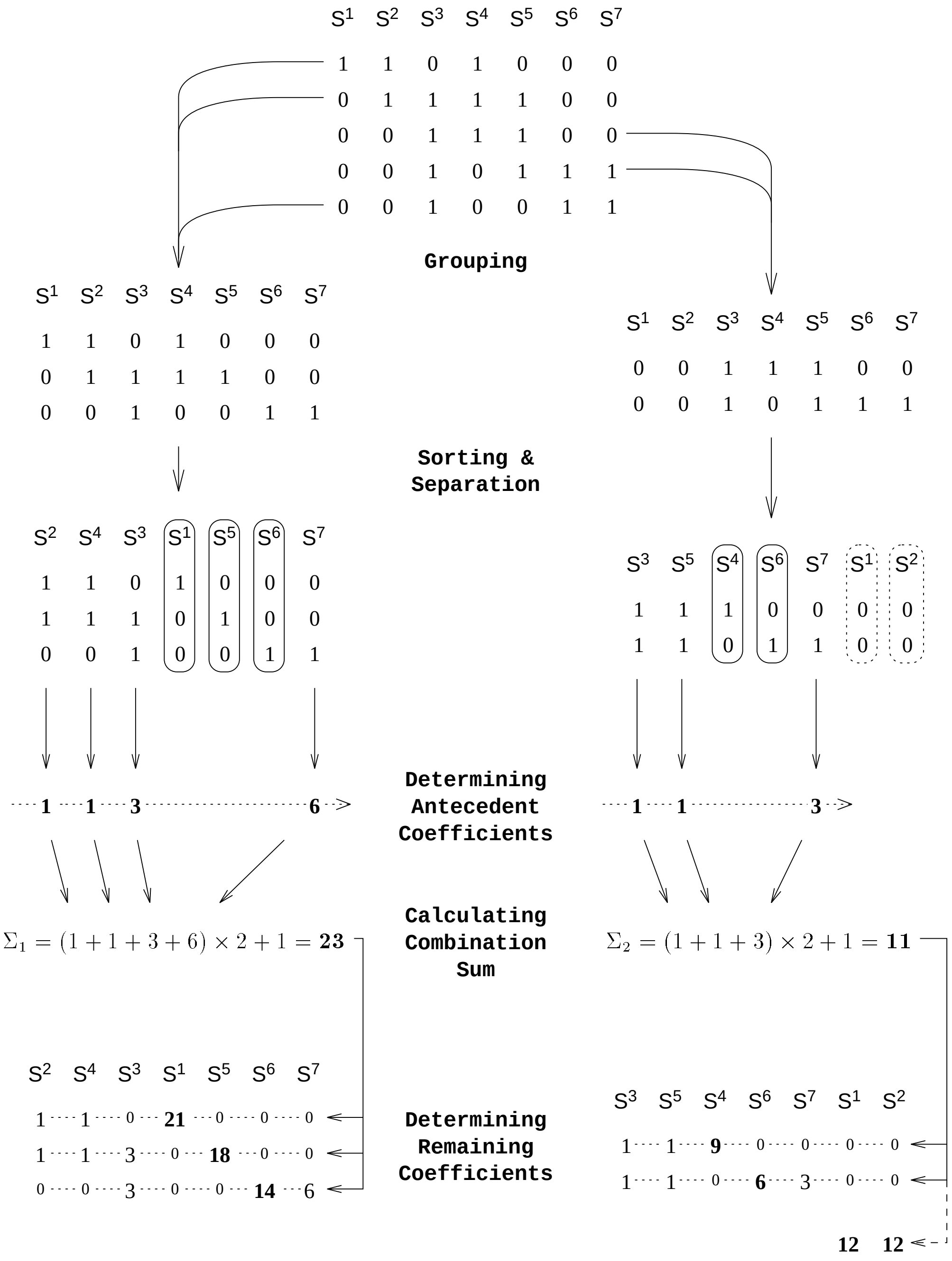
[SBHL94] Silvia Breitinger and Hendrik Lock. *Improving Search for Job-Shop Scheduling with CLP(FD)*. Programming Language Implementation and Logic Programming, Springer-Verlag, 1994.

[AlNa95] Alexander Nareyek. *Modellierung der Disjunktion in der constraintlogischen Programmierung*. Arbeitspapiere der GMD, Nummer 932, Sankt Augustin, 1995.

[SDK83] Maciej M. Syslo, Narsingh Deo and Janusz S. Kowalik, *Discrete Optimization Algorithms*, Prentice-Hall, 1983.

[JLIm95] Jean-Louis Imbert, *Fourier's Elimination: Which to Choose?*, Principles and Practice of Constraint Programming, MIT Press, 1995.

[HeSi95] Helmut Simonis, *The CHIP System and Its Applications*, Principles and Practice of Constraint Programming, Springer-Verlag, 1995.



Resulting Group Equations:

$$21 \times S^1 + S^2 + 3 \times S^3 + S^4 + 18 \times S^5 + 14 \times S^6 + 6 \times S^7 = 23$$

$$12 \times S^1 + 12 \times S^2 + S^3 + 9 \times S^4 + S^5 + 6 \times S^6 + 3 \times S^7 = 11$$

Figure 4: Computation of the group equations