Debugging Systems for Constraint Programming

ESPRIT 22532

Task T.WP3.4: **Declarative Debugging in Constraint Programming**

Delivrable D.WP3.4.M2.1-3

Simplifiaction of Finite Domain Constraints

Abdelkhalek Goumairi and Alexandre Tessier

LIFO, rue Léonard de Vinci, BP 6759, 45067 Orléans Cedex 2, France
goumairi@lifo.univ-orleans.fr, Alexandre.Tessier@inria.fr
http://www.univ-orleans.fr/SCIENCES/LIFO/Members/tessier/

http://discipl.inria.fr/

This paper is an extented abstract of the
DEA dissertation (in French) in appendix.

Abstract

An important issue in Constraint Logic Programming (CLP) systems is how to output constraints in a usable form. Typically, only a small subset $\tilde{x}$ of the variables in constraints is of interest, and so an informal statement of the problem at hand is: given a conjunction $C(\tilde{x}, \tilde{y})$ of constraints, express $\exists \tilde{y}\ C(\tilde{x}, \tilde{y})$ in the simplest form. [13] showed how a set of constraints over the real domain can be simplified. We present here how we can simplify a set of constraints over finite domains. First, we start by a bref sight on the simplification of the constraints over real domain, then we show how we can extend those techniques of simplification over real domain to the finite domain.

This work takes part in the project: ESPRIT DiSCiPl (Debugging Systems for Contraints Programming).

The work is done in the LOCO project (common to INRIA and University of Orléans).

In several parts of the project, the problem of the display of a set of constraints was stated. This problem is not new, indeed it appears yet in the realisation of CLP systems for the display of solutions to a goal: It's the *presentation problem*. For that matter, in the DiSCiPl project, several solutions were suggested: Such as graphics tools, approximations,...etc. Here, we chose the transformmation into an equivalent set of constraints, more legible.

One of the LOCO task in the DiSCiPl ESPRIT project is the declarative erros diagnostis (task T.WP2.1, declarative debugging).

Duning a declarative error diagnostis session, the tool can be led to ask questions to the user, for example: of the form:

$$\exists\ \tilde{y}\ C \longrightarrow a\ \ ?$$

where

- C is a conjonction of constraints;
- a is a atom;
- and $\tilde{y}$ are the free variables of C wich are not free in a.

The aim is to simplify (meaning making it easier to understand, to read), the formula $\exists \tilde{y} C$. For that, we can eliminate existential variables, eliminate redundant constraints and make symbolic transformations (more precisely, substitution of the formula by another one straihforward and equivalent, using for example techniques of automatic demonstration).

1 Introduction

The unit came out from the CLP(R) was described with details in [13]. The wording of the problem of the constraint simplification over finite domains is the same as the wording on the real domain or more precisely:

Given a conjunction $C(\tilde{x}, \tilde{y})$ of constraints, express $\exists \tilde{y}\, C(\tilde{x}, \tilde{y})$ in the simplest form.

> **Example.** x and y are the target variables:
> (a) the constraints $x = z+1, z = y+1$ can be output as $x = y+2$;
> (b) the constraints $x < z$, $z \leq y$, $z \leq y + 2$ can be output as $x < y$;
> (c) the constraints $x = z * y$, can be output as x is multiple of y.

As in the case of the real domain, we can classify the simplification of those constraints in three directions:

(I) Elimination of auxiliary variables (as in (a), (b));

(II) Elimination of redundant constraints (as in (b));

(III) Remplacement of expressions par simpler equivalent ones (as in (c)).

The problem of elimination of the redundant constraints **(II)** on the finite domain seems to be not difficult since it can be treated in parts by methods similar to those on the domaines real. Without any doubt, the elimination of auxiliary variables **(I)** is the most intricate part for the simplification of constraints over finite domains, indeed, the following example show exactly the difficulty of the problem:

> **Example.** On the real domain, the formula $\exists\ x\ (y = 2x)$ can be output as y *real*.
> On the finite domain, the formula $\exists\ x\ (y = 2x)$ can be output as y *even*.

So the elimination of Fourier become wrong.

The problem of equivalent expression research is also a difficult problem.

We are going to see how we can simplify this wording on finite domains, but before we present a brief sight in the simplification of constraints over the real domain.

2 Output in CLP(R)

In [13], the unit output from the CLP(R) is described in details. The point is settled on the well known problem of the projection in the linear arithmetic constraints. It starts with the Fourier's classic algorithm and increases it thanks to the elimination process of the redundant constraints generated by this algorithm. Then, it studies the other kinds of constraints, equations over trees and nonlinear constraints and it shows how they can be simplified with the lineair contraints. For any kind of details, the reader can refer to [13].

Unfortunately, extension of those simplifications over the real domain to finite domains is not easy. As in the linear programming of constraints case, the simplification of constraints over finite domain is more intricate than over real domain.

3 Output in finite domain

The aim of this paragraph is to present a set of techniques in order to simplify a set of constraints over finite domain. By simplification, we mean reducing as much as possible a set of constraints into another more simple and readable (legible). However, we must make the difference between simplification and resolution which does not mean the same thing (which does not have the same signification).

As it was already evoqued, the simplification of constraints over finite domains can be classified into 3 three directions.

3.1 Elimination of redundant constraints

The problem of elimination is in general a problem which is very difficult, as it is similar to the entailment problem of constraints. We remind that the different categories of redundance defined over the real domain are: Tautology, Syntactic redundancy, Quasi-syntactic redundancy, Hull redundancy, Facet redundancy, Quasi facet redundancy, Independent redundancy, Implicit redundancy.

Our first approach is to extend all theses classes of redondances which were defined over real domain to finite domains. This time, we remind that our polyhedral defined in the real case by $P = \{Ax \leq b, x\ real\}$, will be integer that is to say:

- all the variables of matrix A will be integer;
- all the variables of vector x will be integer;
- all the variables of vector b will be integer.

The definitions will be the same as those for continous domaines, but this time, every, time we must add the above conditions.

However the problem of the detection of this class of redundancy is stated, indeed, we saw that in order to detect the redundancy in the continous case, we had to resolve the maximisation problem: $max\ \{a_i x \mid a_j x \leq \beta_j\ ,\ j \neq i\}$, which indicates that the constraint $a_j x \leq \beta_j$ is redundant independent if the value returned by the linear program is lower or equal to β_i.

Another problem is that of the detection of implicit equalities, indeed, the constraint $a_i x \leq \beta_i$ is an implicit equalitie if β_i is returned as optimal solution of the linear program $min\ \{a_i x \mid Ax \leq b\}$.

In the real case, those problem of optimisation with constraints are not difficult to solve (by using for example simplexe method) which is not the case for finite domain.

We notice that for the Tautology, Syntactic redundancy, Quasi-syntactic redundancy, there is no problem of detection since their detection in the case of finite domains is the same for real domain.

For the other kinds of redondances: Hull redundancy, Facet redundancy, Quasi facet redundancy, independent redundancy, Implicit redundancy, their detection in not easy at all, but as in the continous case, these constraints will be transformed by the hull transformation respectively in Tautology, Syntactic redundancy, Quasi-syntactic redundancy, equalities redundancy.

> *Remark.* We remind that our objective is not to solve a constraints system on finite domains, but it is to simplify as much as possible this system into an easier and more legible one. $\diamond$

In [2], Bockmayr has suggested a method to solve the linear constraint system of Pseudo-Boolean constraints, using approximations for the convexe hull of the integer solutions set, by the cutting plane. One of our approach is to extend our techniques for a linear contraints system on the finite domain. Indeed:

Noticing that, any variable x just can take $k+1$ integer values $0,\ 1,\ 2,\ \ldots, k$, so we can substitute it by a linear combination:

$$x = y_0 + 2y_1 + 4y_2 + \ldots + 2^p y_p$$

of $p+1$ variable: $y_0, \ldots, y_p$, each of them being obliged to take just two value 0 or 1(bivalents variables) (p is the smallest integer such as $k \leq 2^{p+1} - 1$).

So, we can always came back to the case where the linear system on the finite domain variable is a linear system on the bivalent variables.

(NB) The precedented transformation, which is always possible, is not necessarly always worthwile in pratice).

3.2 Elimination of auxiliary variables

As in the case of continous domains, the traditonnal approach for the simplification of constraints is to use the "canonical form" equiped with an efficient algorithm for its calculation.

For linear equation, it is well known that the canonical form is called where equations are represented in the parametric form where the equations are represented in the form $\tilde{x} = t(\tilde{y})$ where the $\tilde{y}$ are distincts of $\tilde{x}$ and t is a tuple of linear expressions.

In this paragraph, we descibe an algorithm for the output of $\exists\ \tilde{y}\ C(\tilde{x}, \tilde{y})$, where C is linear, only according to target variables, treating first equations and then inequalities.

The algorithm must produce this time an output containing the particular target variable x wich appears in C, for example if we eliminate y from $\exists\ y \mid x = 2y$ (else the information x is even will be lost). So in the contrary of the case of continous domain, the algorithm will only modify the contraints of the form: $\tilde{x} = t'(\tilde{y})$ where t' is a tuple of linear expressions formed by coefficients equal to 1 or to -1 i.e: all variable x_i is in the form: $x_i = \sum_i \alpha_i y_i$ where $\alpha_i \in \{1, -1\}$.

3.2.1 Linear equations

The equations are always maintainted in the *parametric form* , $\tilde{u} = t(\tilde{v})$ where variables of $\tilde{u}$ are called objects variables and are distincts from variables of $\tilde{v}$, called, parametres.

The algorithm has the same form as that of continous domains, but this time, it will only modify the equation shown above. the other kinds of linear equations will be simplified with other constraints.

> *Remark.* We can think to turn the constraints $x = 2y$ into the form $x = y + y$, wich become the kind of the constraints seen above and then apply the algorithm described above, unfortunately those constraints must be maintained under the parametric form. $\diamond$

3.2.2 Linear Inequalities

In this paragraphe, we adopte the same approach than that of continous domains i.e: Fourier's algorithm + Cernikov + elimination of strict redundant constraints. Since our aim is to simplify a system of linear inequalities and not to solve it, we can extend the algorithm given in the continous case to the case of the finite domain.

Example. We want to simplify the formula $\exists\ y \mid x + y \leq 1\ et\ x - y \leq 2$
If we apply the Fourier's algorithm (adding both inequalites), we obtain the similar formula $x \leq 3/2$.
Geometrically, the constraint $x \leq 3/2$ is just the projection of the polyhedron $P = \{x + y \leq 1, x - y \leq 2\}$ in the abscissa axis. (refer to figure 1)

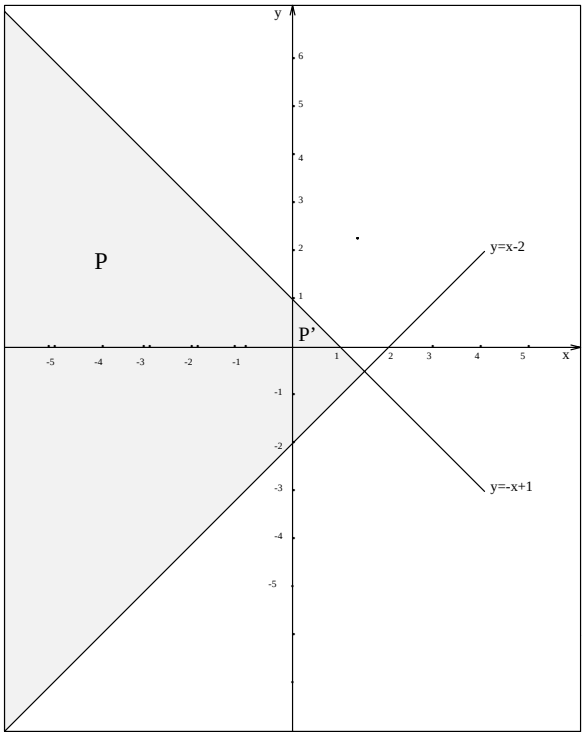

Figure 1: Simplification of linear inequalities

In the finite domain case, the projection of the integer polyhedron $P' = \{x+y \leq 1, -x-y \leq 2, x\ integer\ \geq 0, y\ integer\ \geq 0\}$ on the abscissa axis is the constraint $x \leq 1$. (refer to figure 1)

The simplification of linear inequalities over finite domains is the same as over the real domain, except that an inequalities of the form $x \leq 3/2$ may be substitute by $x \leq E(3/2)$, where E(x) refers to the integer part of x.

3.3 Nonlinear Constraints

In the section 3.2, we had to face the problem of the simplification of the constraint $\exists y \mid x = 2y$. We choose to keep this contraints as it is, that is to say that those kind of constraints are not going to be simplified with the lineair constraints. They will be simplified with the constraints of the form $y \times z$, which are non linear constraints.

4 Conclusion

In this work, we were led to study the solutions put foward for the simplification of the constraint over real domain and suggest methods to solve the problem of simplification of the constraints over finite domain.

This task stanted with a synthesis of works on the output (Simplification, introduction) of constraints over the real domain.

We reminded the main classes of redundancy and the methods of elimination of auxiliary variables as well as tecniques to identify them.

As for finite domains, the techniques suggested for continous domains constitute a part of the solution. Indeed, a redundant constraints over continous polyhedron has also effects over integer polyhedron. Only the problem of the identification of some redundant constraints is stated. For that purpose, we put foward a method to solve this problem which becomes a problem of optimisation.

The main difficulty seem to be the elimination of auxiliary variables where Fourier's method can't be used. We suggest modification to adapt the elimination to finite domains.

This work should enable an introduction of a unit of presentation. (simplification, display) of constraints over finite domain in the DiSCiPl ESPRIT project.

The other kinds of constraints such as disequalities and strict inequalities were not evoqued. This work could be the objective of an other work.

Another point should be to find ad hoc method for variable elimination in finite domain constraints.

References

[1] Alexander Bockmayr and Thomas Kasper, *Pseudo-Boolean and Finite Domain Contraint Programming: A Case Study.* Im Stadtwald, D-66123 Saarbrücken, Germany.

[2] Alexander Bockmayr, *Cutting Planes in Constraint Logic Programming.* Technical Report MPI-I-94-207, Max-Plank-Institut für Informatik, Saarbrücken, February 1994.

[3] Alexander Bockmayr, Peter Barth, *Finite Domain and Cutting Plane Techniques in CLP(PB).* Im Stadtwald, D-66123 Saarbrücken, Germany.

[4] A. Colmerauer. *Introduction to PROLOG III.* In 4th Annual ESPRIT Conference, Bruxelles. North Holland, 1987.

[5] Bruno De backer, *Méthodes de résolution de disjonctions de constraints linéaires. Application à la Programmation Logique avec Contraintes..* Thèse de doctorat en Informatique. Université d'Orléans, 1995.

[6] Daniel Diaz, Björn Carlson and Mats Carlsson, *Entailment of Finite Domain Contraints.* INRIA-Rocquencourt, domaine de Voluceau, 78153 Le Chesnay, France.

[7] Daniel Diaz, *Etude de la compilation des langages logiques de programmation par constraints sur les domaines finis: le système CLP(FD).* Thèse de doctorat en Informatique. Université d'Orléans, 1995.

[8] J. Jaffar and M.J. Maher, *Contraint logic programming:* A survey. Journal of Logic Programming, 1994.

[9] J. Jaffar and J.L. Lassez, *Contraint logic programming.* In Proc. 14th ACM Symp. Principales of Programming Languages, Munich, 1987.

[10] J.L. Lassez and M.J. Maher, *On Fourier's Algorithm for Linear Arithmetic Contraints.* Journal of Automated reasonning 9: 373-379, 1992.

[11] J.L. Lassez and K. McAloon, *Simplification and Elimination of Redundant Linear Arithmetic Contraints.* In Proc. North American Conference on Logic Programming. Cleveland. 1989. pp. 35 51.

[12] J.L. Lassez and K. McAloon, *Generalized Canonical Forms for Linear Contraints and Applications.* In Proc. Int. Conf. On Fifth Generation Computer Systems. ICOT. Tokyo,1988. pp. 703 710.

[13] M.J. Maher and J. Jaffar, *output in CLP(R).* Proceedings of the international conference on fifth generation computer systems, 1992.

[14] M. Minoux, *Programmation Mathématique - Théorie et algorithmes.* Dunod, Paris, 1983.

[15] M. Dincbas, P. van Hentenryck, H. Simonis, A. Aggoun, and T. Graf, *The contraint logic programming language CHIP.* In Fifth Generation Computer Systems, Tokyo, 1988. Springer, 1988.

[16] O. Lhomme and M. Rueher, *Application des techniques CSP au Raisonnement sur les intervalles.* Rapport de recherche numéro 94-59.

[17] P.G. Ciarlet, *Introduction à l'analyse numérique matricielle et à l'optimisation.* Masson, Paris, 1990.

[18] R.J. Duffin, *On Fourier's Analysis of Linear Inequality System.* Mathematical Programming study. Vol. 1 (1974), pp. 71-95.

[19] S.N. Cernikov, *Contraction of Finite Systems of Linear Inequalities.* Doklady Akademiia Nauk SSSR. Vol. 152. No. 5 (1963). pp.1075-1078. (English translation in soviet Mathematics Doklady. Vol. 4, No.5 (1963). pp. 1520-1524.)

This research was supported in part by LOCO Project, common to University of Orléans and INRIA Rocquencourt.