Finite Domain Constraints in Constraint Logic Programming

Philippe Codognet and Daniel Diaz

INRIA - Rocquencourt
B.P. 105, 78 153 Le Chesnay Cedex, FRANCE
Phone: +33 1 39 63 57 08   Fax: + 33 1 39 63 53 30

Abstract

We describe the techniques used in finite domain contraint solvers in the Constraint Logic Programming (CLP) paradigm and present the language and system `clp(FD)` developped at INRIA. These schemes are based on local propagation techniques developed for finite domain constraints. Such techniques stem from Constraint Satisfaction Problems in Artificial Intelligence and have been introduced in Constraint Logic Programming by the CHIP language. Very close to those methods are the interval arithmetic constraints of BNR-Prolog or Newton. The basic idea is to manage a network of constraints between a set of variables which can take values in some finite domains by ensuring local consistency propagation through the constraints linking the variables. Constraint Logic Programming usually only implements, for efficiency reasons, arc-consistency, i.e. propagation of unary constraints (domains of variables), rather than full path-consistency, i.e. propagation of binary constraints, or more general full k-consistency, i.e. constraints involving k variables. Also a popular technique is to enforce a relaxed, or partial, form of k-consistency (also called partial lookahead), which consists in considering and propagating through the constraint network not the full domains of variables but only some approximations of domains, such as the minimal and maximal values. The efficiency of this scheme have been assessed for handling linear equations and inequation in various numerical problems and industrial applications. Thus some constraints can be consistent with the approximations of current domains of variables but not completely satisfied and should be reconsidered when the domains of the variables they involve are further reduced. Therefore the constraint popagation phase is followed by a so-called labeling phase where variables not yet fixed are incrementally instanciated to some value in their domains (which has usually been reduced in the previous phase). Various heuristics can be incorporated in this labeling phase in order to choose the next variable to instanciate. An instanciation can lead to the (re)activation of some constraints that are not completely satisfied, possibly reducing the domains of other variables. This process continues until some solution is found, i.e. until no suspended constraint needs to be reconsidered anymore. Such techniques are called *local* consistency because they do not ensure global consistency in general, although for instance arc-consistency is complete for some subsets of binary constraints and n-consistency will obviously ensure global consistency of a system of n variables. Note that global consistency means that the problem is solved.

The CLP paradigm however is not only based on constraint solvers, but also on the idea of a declarative logical language based on Horn clauses (cf. Prolog). The advantages of having solvers integrated in a declarative language are as follows. First, the underlying logic language can be used as a metalanguage for stating the numerical constraints, instead of giving an explicit encoded formulation, which is in general quite complex and rather unreadable. Second, heuristics can be added in the program itself, as opposed to a closed algorithm with (a finite set of) built-in heuristics.