An Interval Lattice-Based Constraint Solving Framework for Lattices

Antonio J. Fernández1* and Patricia M. Hill2

1 Departamento de Lenguajes y Ciencias de la Computación, E.T.S.I.I., 29071 Teatinos, Málaga, Spain
afdez@lcc.uma.es

2 School of Computer Studies, University of Leeds, Leeds, LS2 9JT, England
hill@scs.leeds.ac.uk

Abstract. We present a simple generic framework to solve constraints on any domain (finite or infinite) which has a lattice structure. The approach is based on the use of a single constraint similar to the indexicals used by CLP over finite domains and on a particular definition of an interval lattice built from the computation domain. We provide the theoretical foundations for this framework, a schematic procedure for the operational semantics, and numerous examples illustrating how it can be used both over classical and new domains. We also show how lattice combinators can be used to generate new domains and hence new constraint solvers for these domains from existing domains.

Keywords: Lattice, constraint solving, constraint propagation, indexicals.

1 Introduction

Constraint Logic Programming (CLP) systems support many different domains such as finite ranges of integers, reals, finite sets of elements or the Booleans. The type of the domain determines the nature of the constraints and the solvers used to solve them. Existing constraint solvers (with the exception of the CHR approach [7]), only support specified domains. In particular, the cardinality of the domain determines the constraint solving procedure so that existing CLP systems have distinct constraint solving methods for the finite and the infinite domains. On the other hand, CHR [7] is very expressive, allowing for user-defined domains. Unfortunately this flexibility has a cost and CHR solvers have not been able to compete with the other solvers that employ the more traditional approach. In this paper we explore an alternative approach for a flexible constraint solver that allows for user and system defined domains with interaction between them.

* This work was partly supported by EPSRC grants GR/L19515 and GR/M05645 and by CICYT grant TIC98-0445-C03-03.
Normally, for any given domain, a solver has many constraints, each with its own bespoke implementation. The exception to this rule is CLP(FD) [4] which is designed for the finite domain of integers and based on a single generic constraint often referred to as an indexical. The implementation of indexicals uses a simple interval narrowing technique which can be smoothly integrated into the WAM [2,6]. This approach has been shown to be adaptable and very efficient and now integrated into mainstream CLP systems such as SICStus Prolog.

This paper has two contributions. First, we provide a theoretical framework for the indexical approach to constraint solvers. This is formulated for any ordered domain that is a lattice. We have observed that most of the existing constraint solvers are for domains that are lattices. Thus our second contribution is to provide a theoretical foundation for more generic constraint solvers where a single solver can support any system or user-defined domain (even if its cardinality is infinite) provided it is a lattice. One advantage of our framework is that, as it is based on lattice theory, it is straightforward to construct new domains and new constraint solvers for these domains from existing ones. In this paper, we describe different ways of performing these constructions and illustrate them by means of examples.

The paper is structured as follows. Section 2 recalls algebraic concepts used in the paper. In Section 3 the computation domain, the execution model and a schema of an operational semantics are described. Section 4 shows the genericity of the theoretical framework by providing several instances which include both the common well-supported domains as well as new domains. Section 5 describes with examples how the framework can be used on the combination of domains. The paper ends with some considerations about related work and the conclusions.

2 Preliminaries

2.1 Ordered Sets

Definition 1. (Ordering) Let \( C \) be a set with equality. A binary relation \( \preceq \) on \( C \) is an ordering relation if it is reflexive, antisymmetric and transitive. The relation \( < \) can be defined in terms of \( \preceq \)

\[
\begin{align*}
    c < c' & \iff c \preceq c' \land c \neq c', \\
    c \preceq c' & \iff c < c' \lor c = c'.
\end{align*}
\]

We write \( c \preceq_C c' \) (when necessary) to express that \( c \preceq c' \) where \( c, c' \in C \). Let \( C \) be a set with ordering relation \( \preceq \) and \( c, c' \in C \). Then we write \( c \sim c' \) if either \( c \preceq c' \) or \( c' \preceq c \) and \( c \neq c' \) otherwise. Any set \( C \) which has an ordering relation is said to be ordered. Evidently any subset of an ordered set is ordered.

Definition 2. (Dual of an ordered set) Given any ordered set \( C \) we can form a new ordered set \( \hat{C} \) (called the dual of \( C \)) which contains the same elements as \( C \) and \( b \preceq_C a \) if and only if \( a \preceq_C b \). In general, given any statement \( \Phi \) about ordered sets, the dual statement \( \hat{\Phi} \) may be obtained by replacing each expression of the form \( x \preceq y \) by \( y \preceq x \).