On the reification of global constraints

Nicolas Beldiceanu · Mats Carlsson · Pierre Flener · Justin Pearson

Published online: 13 November 2012
© Springer Science+Business Media New York 2012

Abstract We introduce a simple idea for deriving reified global constraints in a systematic way. It is based on the observation that most global constraints can be reformulated as a conjunction of total function constraints together with a constraint that can be easily reified.

Keywords Global constraint · Reification · Reformulation · Functional dependency

1 Introduction

Conventional wisdom has it that many global constraints cannot be easily reified, i.e., augmented with a 0–1 variable reflecting whether the constraint is satisfied (value 1) or not (value 0). Reified constraints are useful for expressing propositional formulas over constraints and for expressing that a certain number of constraints hold (e.g., the cardinality operator [14]). Using standard algorithms from automata theory, we have previously shown [4, Section 3.7.196 of 2010 version] how to reify a global constraint that can be expressed in terms of a counter-free finite automaton [3, 12]. However, many global constraints, such as alldifferent and cumulative, cannot
be expressed by an automaton whose size is polynomial in the number of variables of the constraint. The importance of the negation of global constraints has recently increased, e.g., in the context of a constraint seeker with negative samples [5] and for proving the equivalence of constraint models [1, 10].

Many early constraint programming systems, such as CHIP, GNU Prolog, Ilog Solver, and SICStus Prolog, provide reification for arithmetic constraints. However, when global constraints started to get introduced (e.g., \texttt{alldifferent} and \texttt{cumulative}), reification was not available for global constraints. We believe that, in the early 1990s, reification was not considered for global constraints since it was believed that reification could only be obtained by modifying the filtering algorithms attached to each global constraint. Nowadays, Minion [8, 9] features \textit{constraint trees}, which constitute a very efficient mechanism for executing Boolean combinations of primitive as well as global constraints.

In this letter, we present a \textit{portable} reification method that is useful on solvers that do not have such features, and so this work is orthogonal to specific implementation approaches.

2 How to derive reified global constraints

A global constraint $GC(A)$ can be defined by restrictions $R(A)$ on its arguments $A$, e.g., restrictions on the bounds of its arguments, and by a condition $C(A)$ on its arguments, i.e., we have $GC(A) \equiv R(A) \land C(A)$. For instance, for a constraint defined by a finite automaton (e.g., \texttt{global\_contiguity} [4, page 1058]), a typical restriction is that the variables take values in a given alphabet (e.g., values 0 and 1 for \texttt{global\_contiguity}). See [4, pages 9–17] for other examples of such restrictions. Note that the set of restrictions may be empty, that is $R(A)$ may be always satisfied. We define the \textit{reified} version of $GC(A)$ as $R(A) \land (C(A) \leftrightarrow b)$, where $b$ is a 0–1 variable reflecting whether $GC(A)$ holds or not. In particular, we require the negation of $GC(A)$ to satisfy the same restrictions $R(A)$.

Let a \textit{core reifiable constraint} be a constraint of the form of a Boolean combination of linear arithmetic equalities and inequalities and 0–1 variables. We assume that such constraints are already reifiable, without resorting to the methods being developed in this letter. This is the case in all constraint programming systems that we are aware of.

A constraint $R(v_1, \ldots, v_n)$ is a \textit{total function} (TF) if and only if its variables can be partitioned into two non-empty sets, $X$ and $Y$, such that for any assignment to the variables of $X$ there is a \textit{unique} assignment to the variables of $Y$ satisfying $R$. For instance, \texttt{nvalue}(nv, \langle v_1, \ldots, v_n \rangle) [4, page 1466] is a TF since variable $nv$ is uniquely determined by the number of distinct values of the set $\{v_1, \ldots, v_n\}$ of variables. However, \texttt{alldifferent}(\langle v_1, \ldots, v_n \rangle) is not a TF, because no subset of the variables $\langle v_1, \ldots, v_n \rangle$ uniquely determines the other variables. The set $Y$ may contain more than one variable, witness the \texttt{sort}(\langle v_1, \ldots, v_n \rangle, \langle w_1, \ldots, w_n \rangle) constraint [4, page 1772], where $\langle v_1, \ldots, v_n \rangle$ uniquely determine $\langle w_1, \ldots, w_n \rangle$. The global constraint catalogue [4] contains a significant number (23 %) of TF constraints.

We now provide the key observation that allows us to reify most global constraints in a straightforward way. Given a global constraint $GC(A)$ defined by $R(A) \land C(A)$, it turns out that the condition $C(A)$ can often be reformulated as a conjunction