Cardinal: A Finite Sets Constraint Solver

Francisco Azevedo

Published online: 17 February 2007
© Springer Science + Business Media, LLC 2007

Abstract In this paper we present Cardinal, a general finite sets constraint solver just made publicly available in ECLiPSe Constraint System, suitable for combinatorial problem solving by exploiting inferences over sets cardinality. In fact, to deal with set variables and set constraints, existing set constraint solvers are not adequate to handle a number of problems, as they do not actively use important information about the cardinality of the sets, a key feature in such problems. Cardinal is formally presented as a set of rewriting rules on a constraint store and we illustrate its efficiency with experimental results. We show the importance of propagating constraints on sets cardinality, by comparing Cardinal with other solvers. Another contribution of this paper is on modelling: we focus essentially on digital circuits problems, for which we present new modelling approaches and prove that sets alone can be used to model these problems in a clean manner and solve them efficiently using Cardinal. Results on a set of diagnostic problems show that Cardinal obtains a speed up of about two orders of magnitude over Conjunto, a previous available set constraint solver, which uses a more limited amount of constraint propagation on cardinalities. Additionally, to further extend modelling capabilities and efficiency, we generalized Cardinal to actively consider constraints over set functions other than cardinality. The Cardinal version just released allows declaring union, minimum and maximum functions on set variables, and easily constraining those functions, letting Cardinal especial inferences efficiently take care of different problems. We describe such extensions and discuss its potentialities, which promise interesting research directions.

Keywords CLP, Constraint Logic Programming · PI, Primary Input · TG, Test Generation

1 Introduction

A set is naturally used to collect distinct elements sharing some property. Combinatorial search problems over these data structures can thus be naturally modelled by high level languages with set abstraction facilities, and efficiently solved if constraint reasoning prunes search space when the sets are not fully known a priori (i.e. they are variables ranging over a set domain).
Set constraints have deserved in the last years special attention by the Constraint Programming community and have been addressed in recent literature for set-based program analysis systems for infinite sets [25] and, as finite set constraint relations, in languages for general set-based combinatorial search problems [6, 10]. Many interesting theoretical and practical results were obtained [11, 16, 20, 24, 27, 29] making it a very rich and promising research topic [1, 28].

Many complex relations between sets can be expressed with constraints such as set inclusion, disjointness and equality over set expressions that may include such operators as intersection, union or difference of sets. Also, as it is often the case, one is not interested simply on these relations but on some attribute or function of one or more sets (e.g. the cardinality of a set). For instance, the goal of many problems is to maximise or minimise the cardinality of a set. Even for satisfaction problems, some sets, although still variables, may be constrained to a fixed cardinality or a stricter cardinality domain than just the one inferred by the domain of a set variable (for instance, the cardinality of a set may have to be restricted to be an even number).

Finite set constraints were introduced in PECOS [29] and Conjunto [20] (formalized in [21]). These were the first languages to represent set variables by set intervals with a lower and an upper bound considering set inclusion as a partial ordering. Consistency techniques are then applied to set constraints by interval reasoning [8]. In Conjunto (available as an ECLiPSe [17] library), a set domain variable $S$ is specified by an interval $[a, b]$ where $a$ and $b$ are known sets ordered by set inclusion, representing the greatest lower bound and the lowest upper bound of $S$, respectively.

To deal with optimisation problems, Conjunto includes the cardinality of a set as a graded function in the system, and generalises a graded function as $f: P(H_U) \rightarrow N$ mapping a non-quantifiable term of the power-set of the Herbrand universe to a unique integer value denoting a measure of the term, and satisfying $S_1 \subseteq S_2 \Rightarrow f(S_1) \leq f(S_2)$ for two sets $s_1, s_2$.

The cardinality of a set $S$, given as a finite domain variable $C$ ($\#S=C$), is not a bijective function since two distinct sets may have the same cardinality. Still, due to the properties of a graded function, it can be constrained by the cardinalities of the set bounds. Conjunto allows graduated constraints over cardinalities but this cardinality information is largely disregarded until it is known to be equal to the cardinality of one of the set bounds, in which case an inference rule is triggered to instantiate the set.

Although Conjunto represented a great improvement over previous CLP languages with set data structures [20], it lacked some inferences on the cardinality level, which are crucial for a number of CSPs. In fact, Conjunto makes a very limited use of the information about the cardinality of set variables. The reason for this lies in the fact that it, in general, too costly to derive all the inferences one might do over the cardinality information in order to tackle the problems Conjunto had initially been designed for (i.e. large scale set packing and partitioning problems) (Gervet, 1999, personal communication). Nonetheless, and given their nature, we anticipated that some use of this information could be quite useful and speed up the solving of these problems.

Recently, set solvers with domains using reduced ordered binary decision diagrams (ROBDDs) have been proposed [24, 27] with more efficient domain propagators, but that are not so efficient when handling cardinality constraints.

Inferences using cardinalities can be very useful to deduce more rapidly the non-satisfiability of a set of constraints, thus improving efficiency of combinatorial search problem solving. As a simple example, if $Z$ is known to be the set difference between $Y$ and $X$, both contained in set $\{a, b, c, d\}$, and it is known that $X$ has exactly two elements, it should be inferred that the cardinality of $Z$ can never exceed two elements (i.e. from