A Geometric Constraint over $k$-Dimensional Objects and Shapes Subject to Business Rules

Mats Carlsson$^1$, Nicolas Beldiceanu$^2$, and Julien Martin$^3$

$^1$ SICS, P.O. Box 1263, SE-164 29 Kista, Sweden
Mats.Carlsson@sics.se

$^2$ École des Mines de Nantes, LINA UMR CNRS 6241, FR-44307 Nantes, France
Nicolas.Beldiceanu@emn.fr

$^3$ INRIA Rocquencourt, BP 105, FR-78153 Le Chesnay Cedex, France
Julien.Martin@inria.fr

Abstract. This paper presents a global constraint that enforces rules written in a language based on arithmetic and first-order logic to hold among a set of objects. In a first step, the rules are rewritten to Quantifier-Free Presburger Arithmetic (QFPA) formulas. Secondly, such formulas are compiled to generators of $k$-dimensional forbidden sets. Such generators are a generalization of the indexicals of cc(FD). Finally, the forbidden sets generated by such indexicals are aggregated by a sweep-based algorithm and used for filtering.

The business rules allow to express a great variety of packing and placement constraints, while admitting effective filtering of the domain variables of the $k$-dimensional object, without the need to use spatial data structures.

1 Introduction

This paper extends a global constraint $\text{geost}(k, \mathcal{O}, \mathcal{S}, \mathcal{R})$ for handling the location in space of $k$-dimensional objects $\mathcal{O}$ ($k \in \mathbb{N}^+$), each of which taking a shape among a set of shapes $\mathcal{S}$, subject to rules $\mathcal{R}$ in a language based on arithmetic and first-order logic. This language can also be seen as a natural target constraint of the Rules2CP modeling language [1].

In order to model directly a lot of side constraints, which always show up in the context of real-life applications, many global constraints have traditionally been extended with extra options or arguments. This is why, in a closely related area, the $\text{diffn}$ constraint [2] of CHIP provides, beside non-overlapping, a variety of other geometrical constraints (in fact more than 10 side constraints). Even if this makes sense when one wants to efficiently solve specific real-life applications, this proliferation of arguments and options has two major drawbacks:

- Having a lot of ad-hoc side constraints is too specific and can sometimes be quite frustrating since it does not allow to express a variant of an existing side constraint.
- Designing a filtering algorithm for each side constraint independently is not enough and managing the interaction of several side constraints becomes more and more challenging as the number and variety of side constraints increase.
The approach presented in this paper addresses these two issues in the following way:

- Firstly, having a rule language for expressing side constraints is obviously more flexible than having a large set of predefined side constraints.
- Secondly, as we will see later on, our filtering algorithms allow to directly take into account the interaction between all rules.

In $\text{geost}(k, \mathcal{O}, \mathcal{S}, \mathcal{R})$, each shape from $\mathcal{S}$ is defined as a finite set of shifted boxes, where each shifted box is described by a box in a $k$-dimensional space at the given offset with the given sizes. More precisely a shifted box $s \in \mathcal{S}$ is an entity defined by its shape id $s.\text{sid}$, shift offset $s.\text{t}[d]$, $1 \leq d \leq k$, and sizes $s.\text{l}[d]$ (where $s.\text{l}[d] > 0$ and $1 \leq d \leq k$). All attributes of a shifted box are integer values. A shape is a collection of shifted boxes all sharing the same shape id.

Each object $o \in \mathcal{O}$ is an entity defined by its unique object id $o.\text{oid}$ (an integer), shape id $o.\text{sid}$ (an integer if the object has a fixed shape, or a domain variable for polymorphic objects, which have alternative shapes), and origin $o.\text{x}[d]$, $1 \leq d \leq k$ (integers, or domain variables that do not occur anywhere else in the constraint). Objects and shifted boxes may also have additional, integer (but see also Section 6) attributes, such as weight, customer, or fragility, used by the rules.

Each rule in $\mathcal{R}$ is a first-order logical formula over the attributes of objects and shifted boxes. From the point of view of domain filtering, the main contribution of this paper is that multi-dimensional forbidden sets can be automatically derived from such formulas and used by the sweep-based algorithm of $\text{geost}$ [3]. This contrasts with the previous version of $\text{geost}$, where an ad-hoc algorithm computing the multi-dimensional forbidden sets had to be worked out for each side constraint. $\mathcal{R}$ may also contain macros, providing abbreviations for expressions occurring in formulas or in other macros.

The rule language. The language that makes up the rules to be enforced by the $\text{geost}$ constraint is based on first-order logic with arithmetic, as well as several features including macros, bounded quantifiers, folding and aggregation operators. We will show how all but a core fragment of the language can be eliminated by equivalence-preserving rewriting. The remaining fragment is a subset of Quantifier-Free Presburger Arithmetic (QFPA), which has a very simple semantics and, as we also will show, is amenable to efficient compilation.

Constraint satisfaction problems using quantified formulas (QCSP) have for instance been studied by Benedetti et al. [4], mostly in the context of modeling games. QCSP does not provide disjunction but actively uses quantifiers in the evaluation, whereas we eliminate all quantifiers in the process of rewriting to QFPA.

Example 1. This running example will be used to illustrate the way we compile rules to code used by the sweep-based algorithm [3] for filtering the nonground attributes of each object. Suppose that we have five objects $o_1$, $o_2$, $o_3$, $o_4$ and $o_5$ such that:

1. A domain variable $v$ is a variable ranging over a finite set of integers denoted by $\text{dom}(v)$; $\underline{v}$ and $\overline{v}$ denote respectively the minimum and maximum possible values for $v$.
2. The sweep-based algorithm performs recursive traversals of the placement space for each coordinate increasing as well as decreasing lexicographic order and skips unfeasible points that are located in a multi-dimensional forbidden set.