Finite Domain Constraints in SICStus Prolog

Mats Carlsson, SICS

Introduction

In this note, I was given the opportunity to give the community a glimpse of the CLP(FD) library of SICStus Prolog. It has evolved since 1995 and consists of some 10k lines of Prolog and 60k lines of C code. An overview of the library is available elsewhere [1], so after giving a brief summary of the architecture, I will focus on recent and on-going developments.

Brief Solver Anatomy

The solver provides three ways of defining propagators: (a) in Prolog, via a public API; (b) in C, using an internal API; and (c) via *indexicals*. An indexical p_i for constraint p is a function from $\{D(x_j) \mid 1 \leq j \leq n \wedge i \neq j\}$ to $D(x_i)$, the domain of x_i. So it takes k indexicals to propagate a k-ary constraint. Indexicals can be hand-written, but are normally compiled from SMT formulas (see below). They are executed by a custom bytecode emulator.

Propagators of types (a,b) are stateful whereas indexicals are stateless. The solver comes with a library of propagators of types (b,c). Application code can define new propagators of types (a,c).

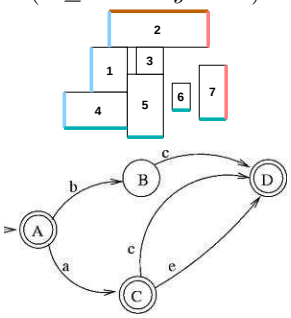
In order to seamlessly integrate with the host system, the solver heavily uses features such as coroutining, attributed variables and mutable terms. Data structures such as domains and suspension lists are Prolog terms.

What's New in Release 4

When SICStus Prolog 4 [2] was released in 2007, as well as later, we took the opportunity to incorporate several new constraints, most of which were invented or studied as part of CP research activities:

- **New in 4.2:** *SMT Formulas.* A very common class of constraint formulas are Boolean combinations of linear arithmetic, *element*, and *table* constraints. We support three ways of propagating them: (a) by flattening to primitives, (b) by compilation to indexicals, and (c) by compilation to an extended version of the *case/[3,4]* constraint, where normally (c) is stronger than (b) and (b) stronger than (a).

$$(x \in [0,99] \wedge y = 10x) \vee (x \in [100,999] \wedge y = 9x) \vee (x \geq 1000 \wedge y = 8x)$$



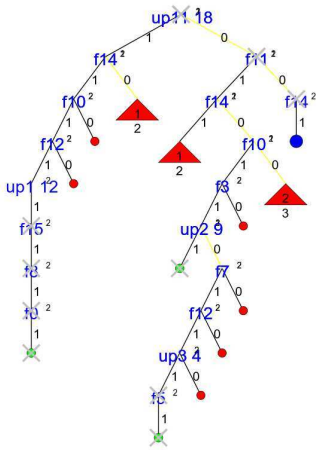
- *geost/[2,3,4].* Constrains the location in space of non-overlapping multi-dimensional objects. [3]
- *automaton/[3,8,9].* In [4], we showed how to obtain filtering for any constraint whose checker of ground instances can be expressed as a finite automaton, optionally with counters. In [5], we showed how to automatically annotate such automata with counters reflecting string properties, like the number of occurrences of given strings.
- *minimum/2 and maximum/2.* Constrain a variable to equal the minimum (maximum) of a list of values.

- *nvalue/2.* Constrains a variable to equal the number of distinct values taken by a list of values.
- *table/[2,3].* Provides a convenient front-end for defining k-ary constraints by extension.

In The Pipeline

- The FDBG tracer [6] allows users to trace domain changes performed by propagators as well as by labeling choices. The code has fallen slightly out of date with the solver, and is about to get a face-lift.
- The CP-VIZ generic visualization platform [7] provides multiple views showing the search tree, and the state of constraints and variables for a postmortem analysis of a constraint program. It is system independent and uses a lightweight, intermediate XML format to exchange information between the solver and the visualization tools. A prototype plugin for SICStus was done in 2010. It will be properly integrated and released with support in the SPIDER IDE for SICStus.

```
3*<fdvar_5>+5*<fdvar_6>+2#=<fdvar_7>
    fdvar_5 = 1
    fdvar_6 = 1..3
    fdvar_7 = 10..26 -> 10..20
```



References

[1] M. Carlsson, G. Ottosson, and B. Carlson. An open-ended finite domain constraint solver. In H. Glaser, P. Hartel, and H. Kuchen, editors, *PLILP'97*, volume 1292 of *LNCS*, pages 191–206. Springer, 1997.

[2] M. Carlsson and P. Mildner. SICStus Prolog – the first 25 years. *Theory and Practice of Logic Programming*, 2011. In press. Also available as arXiv:1011.5640.

[3] N. Beldiceanu, M. Carlsson, E. Poder, R. Sadek, and C. Truchet. A generic geometrical constraint kernel in space and time for handling polymorphic k-dimensional objects. In C. Bessiere, editor, *CP'2007*, volume 4741 of *LNCS*, pages 180–194. Springer, 2007.

[4] N. Beldiceanu, M. Carlsson, R. Debruyne, and T. Petit. Reformulation of global constraints based on constraint checkers. *Constraints*, 10(4), 2005.

[5] N. Beldiceanu, M. Carlsson, P. Flener, and Justin Pearson. On matrices, automata, and double counting. In *CPAIOR'2010*, volume 6140 of *LNCS*, pages 10–24. Springer, 2010.

[6] D. Hanák, T. Szeredi, and P. Szeredi. FDBG, the CLPFD debugger library of SICStus Prolog. In B. Demoen and V. Lifschitz, editors, *ICLP'04*, volume 3132 of *Lecture Notes in Computer Science*, pages 458–459. Springer, 2004.

[7] H. Simonis, P. Davern, J. Feldman, D. Mehta, L. Quesada, and M. Carlsson. A generic visualization platform for CP. In D. Cohen, editor, *CP'2010*, volume 6308 of *LNCS*, pages 460–474. Springer, 2010.