Implementing Bit-Vector Variables, with an Application to the Generation of S-Boxes

Specification of a Master Thesis in Computer Science
at the Department of Information Technology at Uppsala University
by Kellen Dye (19810421-1217)

14th February 2014

This master thesis project (D-level, 20 weeks, 30 credits) will be done at the ASTRA research group (`http://www.it.uu.se/research/group/astra`) at Uppsala University. It will be supervised by Dr Jean-Noël Monette and reviewed by Prof. Pierre Flener.

Background: Constraint Programming

Constraint programming (CP) [3] is a set of techniques to find values for some given decision variables within their given domains, so that some given constraints on these variables are satisfied and possibly that some cost (or benefit) expression on these variables is minimised (or maximised).

The core concept of CP is the notion of **constraint**, which is a declarative encapsulation of specialised algorithms, discussed below. Beside simple comparison ($<$, $\leq$, $=$, $\neq$, $\geq$, $>$) and arithmetic (linear equation, disequation, and inequation) constraints, a large number of combinatorial constraints have been identified and equipped with algorithms. Those constraints capture recurring problem substructures, such as $\textsc{AllDifferent}(\{x_1, \ldots, x_n\})$, which means that the x_i decision variables take pairwise distinct values.

CP solves such combinatorial problems by **systematic search** with backtracking, and performs at every node of the search tree a particular kind of inference called **propagation**. For each constraint, a **propagation algorithm** narrows the domains of its decision variables by eliminating impossible values according to some desired level of **consistency**. Propagation at a node of the search tree then amounts to computing the greatest simultaneous fixpoint of the propagators for all the constraints.

Classically, decision variables take their values in a subset of the integers. However other domains have been proposed, e.g., floating point numbers, sets, and graphs. Recent work has been performed on the domain of **bit-vector** variables in CP. Such variables take their values in a subset of all possible vectors of bits (of a given length). Such a domain has applications in many areas of computer science, e.g., in cryptography or program verification.

Task Description

The goal of this thesis is to provide an implementation of the bit-vector domain in CP and to apply it on relevant problems.

More precisely, the work can be roughly divided into the following goals:

- Implement the bit-vector variables as described in [1].

- Design and implement propagation algorithms, for both primitive and global constraints applied on bit-vectors.
- Use the bit-vector variables and constraints to design cryptographic S-boxes following the approach outlined in [2].

Procedure

- Meetings will be held by appointment or on request by the student, supervisor, or reviewer.
- All implementation will be done in *Gecode*, a free software (available at `http://www.gecode.org/`), under the terms of the MIT license.
- The thesis will be written in English using LaTeX.
- A paper will be written with the supervisor and reviewer if the results warrant it. It will be submitted to some relevant CP event.
- All documents and software produced during this work will be archived at ASTRA.

Time Plan

The equivalent of 20 full-time weeks will be spent on this work, beginning in week 4 of 2014. We aim at completion by late June 2014, upon a public presentation of the results and revision of the thesis according to the comments by the audience and the reviewer. The time will be spent according to the following time plan:

1. Background study on *Gecode* and bit-vector solvers (2 weeks).
2. Implementation of the bit-vector variables and primitive constraints (3 weeks).
3. Experimental comparison of the implementation with existing approaches (3 weeks)
4. Design and implementation of a CP model for the generation of S-boxes (3 weeks).
5. Design and implementation of global constraints that improve the CP model for S-boxes (3 weeks).
6. Experimental evaluation of the CP model for S-boxes (2 weeks).
7. Writing of the thesis (4 weeks).

References

[1] L. D. Michel and P. Van Hentenryck. Constraint satisfaction over bit-vectors. In M. Milano, editor, *Principles and Practice of Constraint Programming CP 2012*, volume 7514 of *LNCS*, pages 527–543. Springer, 2012.

[2] V. Ramamoorthy, M. Silaghi, T. Matsui, K. Hirayama, and M. Yokoo. The design of cryptographic s-boxes using CSPs. In J. Lee, editor, *Principles and Practice of Constraint Programming CP 2011*, volume 6876 of *LNCS*, pages 54–68. Springer, 2011.

[3] F. Rossi, P. van Beek, and T. Walsh, editors. *Handbook of Constraint Programming.* Elsevier, 2006.