sCOP: SAT-based Constraint Programming System

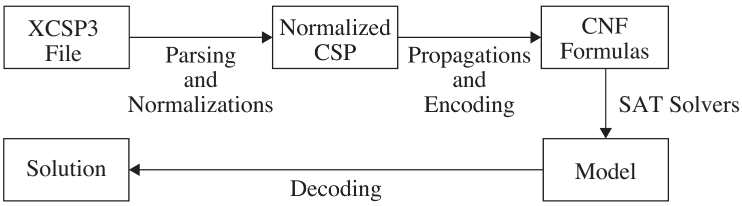
XCSP3 Competition in 2018

Takehide Soh[1], Daniel Le Berre[2], Mutsunori Banbara[1], and Naoyuki Tamura[1]

[1] Information Science and Technology Center, Kobe University, Japan
{soh@lion.,banbara@,tamura@}kobe-u.ac.jp
[2] CRIL-CNRS, Université d'Artois, France
leberre@cril.fr

1 Overview

sCOP is a SAT-based constraint programming system written in Scala. Like Sugar [3] and Diet-Sugar [2], sCOP encodes XCSP3 instances into conjunctive normal form (CNF) formulas using the order encoding [5, 4] and the log encoding for Pseudo-Boolean (PB) constraints [2]. Then, sCOP launches a SAT solver which will return a model if any. In last, a solution of the XCSP3 instance is decoded from the model computed.



This figure shows the framework of sCOP. In the following, we briefly explain each part of this framework.

2 Parsing and Normalizations

Parsing is done by using an official tool XCSP3-Java-Tools [3]. Currently, sCOP accepts constraints in the XCSP3-core language[4].

Normalizations in sCOP are almost same as ones in Sugar [3] and are follows:

Global Constraints. global constraints are translated into intensional constraints by a straightforward way but we use extra pigeon hole constraints for alldifferent constraints.

Extensional Constraints. extensional constraints are translated into intensional constraints by using a variant of multi-valued decision diagrams. This is a difference to ones in Sugar.

Intensional Constrains. using Tseitin transformation, intentional constraints are normalized to be in the form of CNF over linear comparisons $\sum_i a_i x_i \geq k$ where a_i's are integer coefficients, x_i's are integer variables and k is an integer constant.

[3] https://github.com/xcsp3team/XCSP3-Java-Tools
[4] http://www.xcsp.org/specifications

3 Propagations and Encoding

Constraint propagations are executed to the normalized CSP (clausal CSP, i.e., in the form of CNF over linear comparisons $\sum_i a_i x_i \geq k$) to remove redundant values, variables, and linear comparisons. Currently, it is done by using an AC3 like algorithm.

Encoding methods used in sCOP are the followings:

Order Encoding [5, 4]. the order encoding uses propositional variables $p_{x \geq d}$'s meaning $x \geq d$ for each domain value d of each integer variable x. To encode linear comparisons, Algorithm 1 of the literature [4] is used in sCOP.

Log Encoding. the log encoding uses a binary representation of integer variables. There are several ways to encode linear comparisons by using those propositional variables. In sCOP, we replace all integer variables with its binary representation—it gives us a set of PB constraints. We then encode PB constraints into CNF formulas by using the BDD encoding [1].

sCOP basically uses the order encoding but uses the log encoding in case that the huge number of clauses is expected to be encoded. For this expectation, the idea of domain product criteria [2] is used.

4 SAT Solvers

For sequential solving, sCOP uses a SAT solver MapleCOMSPS [5] which is a winning solver on the main track of the SAT competition 2016. It also shows a good performance for solving CSP instances encoded by sCOP. For parallel solving, sCOP uses a SAT solver glucose-syrup [6] which is a winning solver on the parallel track of the SAT competition 2017.

Acknowledgements. We would like to thank to the competition organizers for their huge efforts. This work was supported by JSPS KAKENHI Grant Numbers JP16K16036 and JP16H02803.

References

1. Eén, N., Sörensson, N.: Translating pseudo-Boolean constraints into SAT. Journal on Satisfiability, Boolean Modeling and Computation 2(1-4), 1–26 (2006)
2. Soh, T., Banbara, M., Tamura, N.: Proposal and evaluation of hybrid encoding of CSP to SAT integrating order and log encodings. International Journal on Artificial Intelligence Tools 26(1), 1–29 (2017)
3. Tamura, N., Banbara, M.: Sugar: a CSP to SAT translator based on order encoding. In: Proceedings of the 2nd International CSP Solver Competition. pp. 65–69 (2008)
4. Tamura, N., Banbara, M., Soh, T.: PBSugar: Compiling pseudo-boolean constraints to SAT with order encoding. In: Proceedings of the 25th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2013), IEEE. pp. 1020–1027 (Nov 2013)
5. Tamura, N., Taga, A., Kitagawa, S., Banbara, M.: Compiling finite linear CSP into SAT. Constraints 14(2), 254–272 (2009)

[5] https://sites.google.com/a/gsd.uwaterloo.ca/maplesat/
[6] http://www.labri.fr/perso/lsimon/glucose/