A Solver for Quantified Circuits based on their Primal and Dual Encodings[⋆]

Uwe Egly, Florian Lonsing, and Johannes Oetsch

Vienna University of Technology,
Institute of Information Systems,
Knowledge-Based Systems Group 184/3,
Favoritenstraße 9-11, A-1040 Vienna, Austria
{uwe,lonsing,oetsch}@kr.tuwien.ac.at

Overview

In a nutshell, our approach to solve *quantified circuits in prenex-normal form* relies on running two instances of a QBF solver on a primal and a dual version of the problem encoding in parallel, thereby following ideas of Van Gelder [4]. While the primal encoding refers to the original circuit, the dual version is the negation of the primal encoding, and thus computes the opposite truth value at its output. For quantified circuits (in prenex-normal form), the dual encoding is rather easily obtained from the primal version without increasing the size of the representation by flipping all quantifiers and negating the variable that represents the circuit output. The hypothesis is that for most solvers, either the primal or the dual version will be easier to solve. Hence, our approach is to solve both encodings in parallel. We also make use of preprocessing by means of the tool `Bloqqer` [1], and we use `DepQBF` [2] as backend PCNF-QBF solver. Thus, we require an intermediate translation of the encodings into PCNF form, where we basically translate gates into sets of clauses following Tseitin [3].

The workflow can be summarised as follows:

1. Starting from the quantified circuit as input, a primal and dual version of the encoding in PCNF is generated.
2. We run `Bloqqer` on both QBFs.
3. The resulting formulas a further processed by the QBF solver `DepQBF`.

Preprocessing and solving is done in parallel. We monitor which process finishes first. If one process finishes, the other process is killed, and the the result or its negation of `DepQBF` is returned, depending on whether the primal or the dual encoding has been solved first. Our submission therefore qualifies as *prenex non-CNF solver* for the *parallel track* of the evaluation.

Input: We except, according to the guidelines of QBFEVAL'16[1], the file containing the quantified circuit instance to be processed and the granted CPU time limit (in seconds) as the only command-line input. The quantified circuit has to be represented in the syntactically restricted cleansed version of the QCIR[2] format in prenex-normal form.

⋆ Supported by the Austrian Science Fund (FWF) under grant S11409-N23.

[1] http://www.qbflib.org/qbfeval16.php

[2] http://qbf.satisfiability.org/gallery/qcir-gallery14.pdf

Output: As required by the QBF evaluation, the solver returns exit code 10 if the instance is satisfiable, and exit code 20 if the instance is unsatisfiable.

References

1. Biere, A., Lonsing, F., Seidl, M.: Blocked Clause Elimination for QBF. In: Bjørner, N., Sofronie-Stokkermans, V. (eds.) CADE. LNCS, vol. 6803, pp. 101–115. Springer (2011)
2. Lonsing, F., Biere, A.: DepQBF: A Dependency-Aware QBF Solver. JSAT 7(2-3), 71–76 (2010)
3. Tseitin, G.S.: On the Complexity of Derivation in Propositional Calculus. Studies in Constructive Mathematics and Mathematical Logic (1968)
4. Van Gelder, A.: Primal and Dual Encoding from Applications into Quantified Boolean Formulas. In: Schulte, C. (ed.) CP. LNCS, vol. 8124, pp. 694–707. Springer (2013)