A Survey on DQBF: Formulas, Applications, Solving Approaches

Gergely Kovásznai

IoT Research Center,
Eszterhazy Karoly University of Applied Sciences,
Eger, Hungary
kovasznai.gergely@ekf.hu

1 Introduction

Dependency Quantified Boolean Formulas (DQBF) are obtained by adding Henkin quantifiers to Boolean formulas and have seen growing interest in the last years. In contrast to QBF, the dependencies of a variable in DQBF are explicitly specified instead of being implicitly defined by the order of the quantifier prefix. This enables us to also use partial variable orders as part of a formula instead of only allowing total ones. As a result, problem descriptions in DQBF can possibly be exponentially more succinct. While QBF is PSPACE-complete, DQBF was shown to be NEXPTIME-complete [14].

2 Applications

Many practical problems are known to be NEXPTIME-complete. This includes, e.g., partial information non-cooperative games [13] or certain bit-vector logics [12, 15] used in the context of Satisfiability Modulo Theories (SMT). More recently, also applications in the area of partial equivalence checking (PEC) problems [7, 8] have been discussed and DQBF has been shown to offer a natural encoding for PEC problems. For running experiments, one can access publicly available PEC problem instances and their DQBF encodings [3, 6, 9].

3 Solving Approaches

The first known direct solving approach for DQBF is an adaptation of QDPLL [5], which did not end up being very efficient.

Expansion-based techniques for DQBF were also investigated [1, 2] and yielded in a (not publicly available) expansion-based solver in [8] that uses an underlying SAT solver.

In [4], a refutational approach is proposed that is based on QBF abstraction, thus an uderlying QBF solver is used. This approach is incomplete since it only allows refutation of unsatisfiable formulas.

Based on the fact that Effectively Propositional Logic (EPR) is another logic which is NExpTime-complete, we investigated how to adapt an instantiation-based EPR solving approach, the Inst-Gen calculus [10, 11] to DQBF. We published those results and proposed a new instantiation-based DQBF solver, iDQ in [6]. As the experiments showed, iDQ turned out to be an efficient solver.

In [9], an elimination-based solving strategy is proposed and is implemented in the DQBF solver called HQS. Besides the powerful elimination strategy, HQS utilizes several optimizations, such as pure and unit literal detection, yields to an even more efficient DQBF solver than iDQ.

Apart from the solving technique we use, preprocessing techniques can speed up the solving significantly. Therefore it is worth to investigate if well-known SAT and QBF preprocessing techniques can be adapted to DQBF.

References

1. V. Balabanov, H. K. Chiang, and J. R. Jiang. Henkin quantifiers and boolean formulae. In *Proc. SAT'12*, 2012.
2. V. Balabanov, H. K. Chiang, and J. R. Jiang. Henkin quantifiers and boolean formulae: A certification perspective of DQBF. *Theoretical Computer Science*, 2013.
3. B. Finkbeiner and L. Tentrup. Fast DQBF refutation. In *Proc. SAT 2014*, pages 243–251, 2014.
4. B. Finkbeiner and L. Tentrup. Fast DQBF refutation. In *Proc. SAT'14*, 2014.
5. A. Fröhlich, G. Kovásznai, and A. Biere. A DPLL algorithm for solving DQBF. In *Proc. POS'12*, 2012.
6. A. Fröhlich, G. Kovásznai, A. Biere, and H. Veith. iDQ: Instantiation-based DQBF solving. In *Proc. POS 2014, aff. to SAT 2014*, pages 103–116, 2014.
7. K. Gitina, S. Reimer, M. Sauer, R. Wimmer, C. Scholl, and B. Becker. Equivalence checking for partial implementations revisited. In *Proc. MBMV'13*, pages 61–70, 2013.
8. K. Gitina, S. Reimer, M. Sauer, R. Wimmer, C. Scholl, and B. Becker. Equivalence checking of partial designs using dependency quantified boolean formulae. In *Proc. ICCD'13*, pages 396–403, 2013.
9. K. Gitina, R. Wimmer, S. Reimer, M. Sauer, C. Scholl, and B. Becker. Solving DQBF through quantifier elimination. In *Proc. DATE 2015*, pages 1617–1622. EDA Consortium, 2015.
10. K. Korovin. Instantiation-based automated reasoning: From theory to practice. In *Proc. CADE'09*, pages 163–166, 2009.
11. K. Korovin. Inst-Gen - a modular approach to instantiation-based automated reasoning. In *Programming Logics*, pages 239–270, 2013.
12. G. Kovásznai, A. Fröhlich, and A. Biere. On the complexity of fixed-size bit-vector logics with binary encoded bit-width. In *Proc. SMT'12*, 2012.
13. G. Peterson, J. Reif, and S. Azhar. Lower bounds for multiplayer noncooperative games of incomplete information, 2001.
14. G. L. Peterson and J. H. Reif. Multiple-person alternation. In *Proc. FOCS'79*, pages 348–363, 1979.
15. C. M. Wintersteiger, Y. Hamadi, and L. de Moura. Efficiently solving quantified bit-vector formulas. In *Proc. FMCAD'10*, 2010.