SAT-to-SAT in QBFEval 2016

Bart Bogaerts, Tomi Janhunen, Shahab Tasharrofi

Helsinki Institute for Information Technology HIIT
Department of Computer Science, Aalto University, Finland
firstname.lastname@aalto.fi

Abstract. In this short paper, we describe QBF solvers from the latest QBFEval that were based on the SAT-to-SAT solver. We present the main ideas behind the solvers and discuss their strengths and weaknesses.

1 Introduction

SAT-TO-SAT is a recently introduced solver that integrates several SAT solvers to solve problems arbitrarily high in the polynomial hierarchy [7, 3]. In order to evaluate its performance, we submitted three versions of this solver to the latest QBF evaluation:

- QSTS: This version represents our solver without any out-of-the-box preprocessors.
- XB-QSTS: This version represents our solver with both QXBF [12] and BLOQQER [2] preprocessors.
- XB-BID-QSTS: This version integrates our solver with BREAKID, an award-winning symmetry breaker [4], in addition to QXBF and BLOQQER.

Moreover, the above solvers also differ in internal options that were used. Unfortunately, due to being relatively young, some bugs were still present in the versions submitted to the competition. In particular, there were problems with (a) parsing QCIR theories [14], (b) some simplification procedures when translating QDIMACS to SAT-TO-SAT and, (c) integrating SAT-TO-SAT with BREAKID. Hence, our submission of XB-QSTS to Prenex CNF tracks turned out to be our only correct solver in the official competition.

Nevertheless, the competition results show that SAT-TO-SAT is a state-of-the-art QBF solver which finished second in the standard Prenex CNF track, only behind RAREQS [8]. In other Prenex CNF tracks (2QBF and Random QBFs), SAT-TO-SAT finished in the fourth and sixth places, respectively.

Since submitting SAT-TO-SAT to the QBF competition, we have fixed the problems identified above and the results reported in this short paper reflect our experiments with the corrected versions of our solver. In what follows, we provide a brief description of how SAT-TO-SAT works internally as well as our updated experimental results, comparing QSTS and XB-QSTS to RAREQS, DEPQBF [11] and GHOSTQ [10].

2 Background

Vocabularies and Interpretations. A *vocabulary* is a set of symbols, also called *atoms*; we use σ, τ, ν to denote vocabularies. If σ is a vocabulary, a (two-valued) σ-interpretation is a mapping $\sigma \to \{\mathbf{t}, \mathbf{f}\}$ where $\mathbf{t}$ denotes *true* and $\mathbf{f}$ *false*. A *partial*

σ*-interpretation* is a mapping $\sigma \rightarrow \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$, where $\mathbf{u}$ denotes *unknown*. We often identify a partial σ-interpretation J with a set of tuples p^v with $p \in \sigma, v \in \{\mathbf{t}, \mathbf{f}\}$ (with each atom occurring at most once), meaning that J assigns all atoms occurring in this set to their corresponding values, and all others to unknown. This allows us to define the "union" of two interpretations. E.g., if σ and τ are disjoint, I is a (partial) σ-interpretation and J a (partial) τ-interpretation, we use $I \cup J$ to interpret symbols in σ in the same way as I and symbols in τ in the same way as J. If I and J are two σ-interpretations, we will use the expression $I \cup J$ only if $I \cup J$ indeed defines a partial interpretation (i.e., does not contain both $p^{\mathbf{t}}$ and $p^{\mathbf{f}}$).
Formulas. *Propositional formulas* are recursively built from atoms $p, q, r, \ldots$ using connectives $\wedge$, $\vee$ and $\neg$. A *propositional theory* is a collection of propositional formulas that implicitly represents their conjunction. A singleton theory $\{\phi\}$ is sometimes shown as ϕ. A propositional formula is a σ*-formula* if its atoms are in σ. A *literal* is an atom or its negation. A *clause* is a disjunction of literals. A formula is in *Conjunctive Normal Form (*CNF*)* if it is a conjunction of clauses. A sub-formula occurs *positively* (resp. *negatively*) if it is within the scope of an even (resp. odd) number of negations.

A *quantified Boolean formula* (QBF) is built using the same recursive rules, but also using quantifiers $\forall$ and $\exists$ over propositional atoms. Similar to propositional theories, we define a QBF *theory* to be a collection of quantified Boolean formulas. A σ-QBF is a QBF with free symbols belonging to σ. A σ-QBF with $\sigma = \{\}$ is called a QBF *sentence*. We use $\exists\tau : \varphi$ to abbreviate $\exists p_1 \ldots \exists p_n : \varphi$ if $\tau = \{p_1, \ldots, p_n\}$. For QBF φ, we use $\varphi(\sigma)$ to denote that φ's free symbols are all in σ, i.e., φ is a σ-QBF. A *prenex* QBF is a QBF in which all quantifiers are in the front, i.e., a sequence of quantifiers followed by a propositional formula. The QDIMACS format is a numerical format to describe prenex QBFs in which the propositional formula is in CNF. QDIMACS is the de-facto standard for representing QBF instances.
Satisfiability Relation for QBFs. The satisfiability relation between σ-interpretations I and a σ-QBFs φ, denoted by $I \models \varphi$, is defined recursively in the standard way: (i) $I \models p$ if $I(p) = \mathbf{t}$; (ii) $I \models \neg\varphi$ if $I \not\models \varphi$; (iii) $I \models \varphi \wedge \varphi'$ (resp. $I \models \varphi \vee \varphi'$) if $I \models \varphi$ and (resp. or) $I \models \varphi'$; (iv) $I \models \forall x : \varphi$ (resp. $I \models \exists x : \varphi$) if $(I \cup \{x^{\mathbf{t}}\}) \models \varphi$ and (resp. or) $(I \cup \{x^{\mathbf{f}}\}) \models \varphi$. For a (partial) σ-interpretation I and σ-QBF φ, we call φ *I-satisfiable* if φ has a model that extends I and *I-unsatisfiable* otherwise.

3 SAT-TO-SAT

This section discusses the high-level ideas behind SAT-TO-SAT solver. The first subsection describes the inner workings of the core solver. The second subsection describes preprocessing techniques used to optimize the input for our solver.

3.1 SAT-TO-SAT

Nested SAT Solvers. SAT-TO-SAT starts from a simple idea that has been around for years in the QBF community, i.e., nesting of SAT solvers [16, 9, 15]. The idea is easiest to explain in the case of a 2QBF. For a theory of the form $\psi = \forall\sigma : \varphi_1 \Rightarrow \exists\tau : \varphi_2$ with φ_1 and φ_2 in CNF, we can rewrite ψ as $\neg\exists\sigma : [\varphi_1 \wedge \neg\exists\tau : \varphi_2]$. The idea is that the first SAT solver tries to find an assignment I for σ to satisfy φ_1 (i.e., essentially, searching for a witness to the falsify ψ) and, as soon as such an assignment I is found, the second solver searches for an assignment I' to τ such that $I \cup I'$ is a model for φ_2. In case no assignment I' is found, I indeed witnesses falsity of ψ. But, if I' is found, we analyze φ_2 to learn a new clause C over σ that falsifies I and is a consequence of $\neg\exists\tau : \varphi_2$. We then add C to the theory of the first solver and continue searching for a new assignment I. This idea is very effective as (1) it allows reuse of existing SAT techniques such as conflict-driven clause learning [13]; (2) it treats SAT solvers as black boxes and, hence, all state-of-the-art SAT solvers can be plugged in; and (3) it fits the lazy clause generation paradigm [5]. This idea easily generalizes beyond 2QBF by allowing deeper nestings of SAT solvers.

Quantifier-Independent Decision Ordering. To the best of our knowledge, a limitation that all contemporary QBF solvers share is that, when solving QBF theories, the decision ordering on quantified variables is bound by the quantification order. That is, inner level variables are decided on only after all outer level variables are assigned a value. For example, in the formula $\exists x \exists y\, \forall p \forall q\ \phi(x, y, p, q)$, the variables p and q cannot be decided upon unless the truth values of both x and y have been assigned. This restriction is present in nested SAT solvers (the second SAT solver can only be called if a complete assignment for σ is found), non-CNF solvers such as GHOSTQ [10], counter-example guided abstraction refinement (CEGAR) solvers such as RAREQS [8], and QDPLL-based solvers such as DEPQBF [11].

Such a dependence between variable decision ordering and quantification level is problematic because it means that relevant conflicts and/or solutions in QBF theories cannot be found unless all (possibly irrelevant) outer-level variables have a value. Solvers such as DEPQBF partly mitigate this problem by extracting (static) variable dependency information out of QBF theories. While a valuable approach, such solutions neither address the root cause of the problem nor are they general enough to apply to all QBF theories. The SAT-TO-SAT approach is based on *a new method to make variable decision ordering independent of the quantification structure* of those variables. The intuition is that when the outermost solver arrives in a situation with a partial assignment for σ (following the notation introduced in the paragraph on nested solvers above), it can already call the nested solver. The nested solver should then search for an assignment to τ that models φ_2 for *every* interpretation of the unassigned variables in σ. If such an assignment is found, a conflict clause can be added to the theory of the outermost solver. The search for such an assignment would in general require major modifications to the solver. Instead, we implemented this in a different way. We show that certain theory transformation, that include duplication of the atoms in σ, achieve the same effect. This method has been detailed for the case of 2QBF by Janhunen et al. [7] and extended to the general case by Bogaerts et al. [3].

3.2 Optimizations

Definition Extraction. We detect *definitions* and pull them to the outermost level possible. This technique is similar to reverse Tseitin engineering in GhostQ [6] and is based on the observation that the following two formulas are always equivalent:

$$\varphi(\overline{x}) = \exists\overline{y} : [\psi_1 \wedge \neg\exists\overline{z} : \exists z' : (z' \Leftrightarrow \psi_2(\overline{x}, \overline{y})) \wedge \psi_3)].$$
$$\varphi(\overline{x}) = \exists\overline{y} : \exists z' : [\psi_1 \wedge (z' \Leftrightarrow \psi_2(\overline{x}, \overline{y})) \wedge \neg\exists\overline{z} : \ \psi_3].$$

Transforming the first formula to the second reduces the level of variable z' by moving it to outer levels of a QBF theory. Repeated applications of this transformation guarantee that all definitions are moved to the outermost level possible. Our current implementation only extracts definitions of the form $z' \Leftrightarrow \bigwedge\{l_1, \dots l_k\}$ or $z' \Leftrightarrow \bigvee\{l_1, \dots l_k\}$. Identifying other Boolean definitions is a topic for future research.

Extracting definitions leads to QBF theories with a richer vocabulary for the outer-level solvers. Since conflict clauses are the only means of passing information between SAT solvers in SAT-TO-SAT, having a rich vocabulary in the outer solver means that information can be passed along more succinctly. For instance, when $z = \bigwedge\{l_1, \dots, l_n\}$ and $z' = \bigwedge\{l'_1, \dots, l'_m\}$, the conflict clause $z \vee z'$ summarizes $n \times m$ different conflict clauses $l_i \vee l'_j$ (for $1 \leq i \leq n$ and $1 \leq j \leq m$). Hence, enriching the vocabulary of outer-level solver with variables z and z' allows us to reduce the number of conflict clauses passed from the inner-level solver to the outer-level solver.

Unit and Equality Propagation. When extracting definitions $z \Leftrightarrow \bigvee\{l_1, \dots, l_k\}$ or $z \Leftrightarrow \bigwedge\{l_1, \dots, l_k\}$, we look for special cases when $k = 0$ or $k = 1$. When $k = 0$, we propagate the forced value of z to obtain a simpler QBF (unit propagation) and, when $k = 1$, we replace all occurrences of z with l_1 (equality propagation). Propagating unit and/or equal literals sometimes leads to vast simplifications in a QBF instance.

Extended Learning Vocabulary. Sometimes when the internal solver in SAT-TO-SAT finds a model, there are several alternatives to constructing a conflict clause. This happens when some clauses at level k depend on more than one literal from levels below k (i.e., outer levels). In such cases, for each clause C with more than one variable from outer levels, we can introduce a new auxiliary variable v_C which is equivalent to the disjunction of all literals in C that belong to the outer level. Similar to definition extraction, the richer learning vocabulary for outer level solver is expected to summarize many conflict clauses into one. This, in turn, is a limited type of *extended resolution* [17] that leads to stronger learned clauses. In SAT-TO-SAT, introduction of such auxiliary variables happens statically, in a preprocessing phase.

This idea is not new and has been recently studied in two independent papers [9, 15]. Janota et al. [9] calls this idea *clause selection*.

Weakened Definitions. Our previous QBF pre-processing techniques introduced several definitions. Depending on how the defined variables are used, some definitions can be weakened, essentially only adding one of the two implications. By analyzing the *dependency graph* of the definition structure, we can determine which formulas depend positively or negatively on given variables and, in a sound way, drop one direction of the introduced definitions.

Symmetry Breaking. After the preprocessing techniques described above, we detect and break symmetries of the QBF theory similar to Audemard et al. [1]. That is, for each quantifier alternation (which also corresponds to each SAT solver that we create), we fix all variables from other levels and, using symmetry breaking tool BREAKID [4], we check if there exist some automorphisms over the set of variables at the current level.

If such automorphisms are found, we add the symmetry breaking clauses generated by BREAKID to the solver for that level. Contrary to the work by Audemard et al. [1], our symmetry breaking techniques do not require any modifications to the actual solver: they are based merely on preprocessing.

4 Experiments

4.1 Configurations

Not all optimizations were applied in all versions of our solvers submitted to QBF-Eval 2016. We experimentally tested which combinations of optimizations work well together, as well with other existing QBF preprocessors and submitted the following three configurations to the QBF Evaluation:

- QSTS: Our QSTS submission is comprised of our bare-bone solver (i.e., the nested SAT solver architecture of SAT-TO-SAT plus its quantifier-independent decision ordering feature) as well as following optimizing transformations: definition extraction, unit and equality propagation, and weakened definitions.
- XB-QSTS: Our XB-QSTS submission is similar to QSTS except that it disables the quantifier-independent decision ordering but uses QXBF and BLOQQER preprocessors in order to simplify a given QBF instance.
- XB-BID-QSTS: Our XB-BID-QSTS submission is similar to XB-QSTS except that it also uses symmetry breaking.

4.2 Results

For experimental results, we ran QBFEval 2016 instances (which consists of 825 formulas in Prenex CNF format) on a machine with a 32-core Intel® Xeon® CPU E5-4650 clocked at 2.70GHz, with 256GB of memory, and running Ubuntu Linux with kernel version 3.13.0-91. We limited the resources available for each process to one CPU core, at most 4GB of memory, and at most 600 seconds of time.

Under the above conditions, we ran the three different configurations of our solver (i.e., QSTS, XB-QSTS and XB-BID-QSTS) and compared them with RAREQS+BLOQQER (i.e., RAREQS QBF solver plus BLOQQER preprocessor), DEPQBF (under its best configuration, i.e., DEPQBF-V2, as it participated in the QBFEval) and GHOSTQ-CEGAR. We used all 825 instances that were made available in the Prenex CNF track. Table 1 summarizes our results.

As Table 1 shows, RAREQS+BLOQQER shows the best performance among all these solvers. This result is consistent with competition results of QBFEval 2016 in which RAREQS+BLOQQER was ranked first. Also, the total number of solved QBF instances for RAREQS+BLOQQER in our experiments is almost indistinguishable from competition results (hence, further confirmation for the results we report here). That is, RAREQS+BLOQQER solved 640 QBF instances in QBFEval 2016 and 639 instances in our experiments. Our experiments also show consistency with QBFEval 2016 in the case of XB-QSTS (our only bug-free submission to QBFEval 2016), DEPQBF-V2 and

Solver	#SAT	#UNSAT	#Solved
RAREQS+BLOQQER	308	331	639
XB-BID-QSTS	300	327	627
XB-QSTS	297	318	615
DEPQBF-V3	292	303	595
GHOSTQ-CEGAR	297	287	584
QSTS	210	342	552

Table 1. Experimental results comparing different configurations of SAT-TO-SAT with other state-of-the-art QBF solvers on instances from QBFEval 2016.

GHOSTQ-CEGAR solvers. That is, the difference between the total number of solved instances in our experiments and those reported by QBFEval 2016 is at most eight.

As indicated by Table 1, our corrected version of XB-BID-QSTS performs better than our XB-QSTS solver, and they both perform better than all other solvers except RAREQS+BLOQQER. Hence, we conclude that we could have received both the 2nd and the 3rd ranks in QBFEval 2016 if we had managed to submit a correctly working version of XB-BID-QSTS.

5 Conclusion

We briefly discussed our contribution, SAT-TO-SAT, to the latest QBF evaluation. The main difference between SAT-TO-SAT and other QBF solvers is its ability to make variable decision ordering independent from the quantifier prefix. We conclude that, while still in an early development stage, SAT-TO-SAT is a promising solver capable og competing with the best QBF solvers on Prenex CNFs. Additionally, our experiments with bug-fixed versions of our solver showed the effectiveness of symmetry breaking. That is, a bug-free submission of XB-BID-QSTS would have been ranked 2nd in the Prenex-CNF track of QBFEval 2016 after RAREQS+BLOQQER and before XB-QSTS.

Acknowledgments. This work is generously supported by the Finnish Center of Excellence in Computational Inference Research (COIN) funded by the Academy of Finland (under grant #251170).

References

1. Audemard, G., Mazure, B., Sais, L.: Dealing with symmetries in quantified Boolean formulas. In: SAT 2004 - The Seventh International Conference on Theory and Applications of Satisfiability Testing, 10-13 May 2004, Vancouver, BC, Canada, Online Proceedings (2004), `http://www.satisfiability.org/SAT04/programme/101.pdf`
2. Biere, A., Lonsing, F., Seidl, M.: Blocked clause elimination for QBF. In: Bjørner, N., Sofronie-Stokkermans, V. (eds.) Automated Deduction - CADE-23 - 23rd International Conference on Automated Deduction, Wroclaw, Poland, July 31 - August 5, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6803, pp. 101–115. Springer (2011), `http://dx.doi.org/10.1007/978-3-642-22438-6_10`

3. Bogaerts, B., Janhunen, T., Tasharrofi, S.: Solving QBF instances with nested SAT solvers. In: Darwiche, A. (ed.) Beyond NP, Papers from the 2016 AAAI Workshop, Phoenix, Arizona, USA, February 12, 2016. AAAI Workshops, vol. WS-16-05. AAAI Press (2016), http://www.aaai.org/ocs/index.php/WS/AAAIW16/paper/view/12603
4. Devriendt, J., Bogaerts, B., Bruynooghe, M., Denecker, M.: Improved static symmetry breaking for SAT. In: Creignou, N., Berre, D.L. (eds.) Theory and Applications of Satisfiability Testing - SAT 2016 - 19th International Conference, Bordeaux, France, July 5-8, 2016, Proceedings. Lecture Notes in Computer Science, vol. 9710, pp. 104–122. Springer (2016), http://dx.doi.org/10.1007/978-3-319-40970-2_8
5. Feydy, T., Stuckey, P.J.: Lazy clause generation reengineered. In: Gent, I.P. (ed.) Principles and Practice of Constraint Programming - CP 2009, 15th International Conference, CP 2009, Lisbon, Portugal, September 20-24, 2009, Proceedings. Lecture Notes in Computer Science, vol. 5732, pp. 352–366. Springer (2009), http://dx.doi.org/10.1007/978-3-642-04244-7_29
6. Goultiaeva, A., Bacchus, F.: Recovering and utilizing partial duality in QBF. In: Theory and Applications of Satisfiability Testing - SAT 2013 - 16th International Conference, Helsinki, Finland, July 8-12, 2013. Proceedings. pp. 83–99 (2013), http://dx.doi.org/10.1007/978-3-642-39071-5_8
7. Janhunen, T., Tasharrofi, S., Ternovska, E.: Sat-to-sat: Declarative extension of SAT solvers with new propagators. In: Schuurmans, D., Wellman, M.P. (eds.) Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA. pp. 978–984. AAAI Press (2016), http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12323
8. Janota, M., Klieber, W., Marques-Silva, J., Clarke, E.M.: Solving QBF with counterexample guided refinement. In: Cimatti, A., Sebastiani, R. (eds.) Theory and Applications of Satisfiability Testing - SAT 2012 - 15th International Conference, Trento, Italy, June 17-20, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7317, pp. 114–128. Springer (2012), http://dx.doi.org/10.1007/978-3-642-31612-8_10
9. Janota, M., Marques-Silva, J.: Solving QBF by clause selection. In: Yang, Q., Wooldridge, M. (eds.) Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015. pp. 325–331. AAAI Press (2015), http://ijcai.org/Abstract/15/052
10. Klieber, W., Sapra, S., Gao, S., Clarke, E.M.: A non-prenex, non-clausal QBF solver with game-state learning. In: Theory and Applications of Satisfiability Testing - SAT 2010, 13th International Conference, Proceedings. pp. 128–142 (2010), http://dx.doi.org/10.1007/978-3-642-14186-7_12
11. Lonsing, F., Biere, A.: DepQBF: A dependency-aware QBF solver. Journal of Satisfiability (JSAT) 7(2-3), 71–76 (2010), http://jsat.ewi.tudelft.nl/content/volume7/JSAT7_6_Lonsing.pdf
12. Lonsing, F., Biere, A.: Failed literal detection for QBF. In: Sakallah, K.A., Simon, L. (eds.) Theory and Applications of Satisfiability Testing - SAT 2011 - 14th International Conference, SAT 2011, Ann Arbor, MI, USA, June 19-22, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6695, pp. 259–272. Springer (2011), http://dx.doi.org/10.1007/978-3-642-21581-0_21
13. Marques-Silva, J.P., Sakallah, K.A.: GRASP: A search algorithm for propositional satisfiability. IEEE Transactions on Computers 48(5), 506–521 (1999), http://dx.doi.org/10.1109/12.769433
14. QBF Gallery: QCIR-G14: A non-prenex non-CNF format for quantified Boolean formulas. Tech. rep. (04 2014), http://qbf.satisfiability.org/gallery/qcir-gallery14.pdf

15. Rabe, M.N., Tentrup, L.: CAQE: A certifying QBF solver. In: Kaivola, R., Wahl, T. (eds.) Formal Methods in Computer-Aided Design, FMCAD 2015, Austin, Texas, USA, September 27-30, 2015. pp. 136–143. IEEE (2015)
16. Ranjan, D.P., Tang, D., Malik, S.: A comparative study of 2QBF algorithms. In: SAT 2004 - The Seventh International Conference on Theory and Applications of Satisfiability Testing, 10-13 May 2004, Vancouver, BC, Canada, Online Proceedings (2004), `http://www.satisfiability.org/SAT04/programme/113.pdf`
17. Tseitin, G.S.: On the complexity of derivation in the propositional calculus, *Zapiski nauchnykh seminarov*. LOMI 8, 234–259 (1968), english translation of this volume: Studies in Constructive Mathematics and Mathematical Logic, Part 2, A. O. Slisenko, eds. Consultants Bureau, N.Y., 1970, pp. 115-125