QBFFam: A Tool for Generating QBF Families from Proof Complexity*

Olaf Beyersdorff[1][0000−0002−2870−1648], Luca Pulina[2][0000−0003−0258−3222], Martina Seidl[3][0000−0002−3267−4494], and Ankit Shukla[3][0000−0002−1038−3602]

[1] Friedrich Schiller University Jena, Germany
olaf.beyersdorff@uni-jena.de
[2] University of Sassari, Italy
lpulina@uniss.it
[3] Johannes Kepler University Linz, Austria
{martina.seidl, ankit.shukla}@jku.at

Abstract. We present QBFFam, a tool for the generation of formula families originating from the field of proof complexity. Such formula families are used to investigate the strength of proof systems and to show how they relate to each other in terms of simulations and separations. Furthermore, these proof systems underly the reasoning power of QBF solvers. With our tool, it is possible to generate informative and scalable benchmarks that help to analyse the behavior of solvers. As we will see in this paper, runtime behavior predicted by proof complexity is indeed reflected by recent solver implementations.

Keywords: Quantified Boolean Formulas · Formula Generator · Benchmarking.

1 Introduction

In recent years, much progress has been achieved in the theory and practice of solving quantified Boolean formulas (QBF) [12], offering a rich solving infrastructure, ranging from preprocessing over solving to result validation, strategy extraction, and theoretical lower bounds. As the decision problem for QBF (QSAT) is PSPACE-complete, many practical application problems [35] from fields such as formal verification, artificial intelligence, and reactive synthesis can be efficiently encoded in QBF and handed over to a QBF solver. Because of the PSPACE-completeness of QSAT, however, solving a QBF is a difficult task.

To solve QBFs various solving approaches have been presented (see [12] for a description of recent QBF solving techniques). *Conflict-driven clause/cube learning* (QCDCL) generalizes the successful CDCL paradigm that is dominant in SAT solving. *Expansion-based techniques* that build propositional abstractions

* This work has been supported by the Austrian Science Fund (FWF) under project W1255-N23, the LIT AI Lab funded by the State of Upper Austria, and a grant by the Carl Zeiss Foundation.

and then exploit the power of SAT solvers have been extremely successful in the last QBFEval competitions [34].

Empirical observations indicated that different approaches have a different reasoning power, resulting in a more diverse solving technique landscape than present in SAT. These observations can be confirmed by proof complexity results, offering explanations how the different approaches relate to each other by establishing separation and simulation results of the proof systems underlying the solvers. In many cases, formula families play a crucial role to characterize what is easy/hard for a solver.

In this paper, we present `QBFFam`, a tool for generating prominent formula families from proof complexity. With this tool, we provide a diverse collection of benchmarks that can be arbitrarily scaled and that are used in proof complexity to compare those proof systems that underlie the behavior of the state-of-the-art solvers. In this way, it becomes possible to obtain an improved understanding of solver implementations and their behavior, because for the generated families many theoretical results with respect to lower and upper bounds have been established.

Our tool is available at

`https://github.com/marseidl/qbffam.git`

It is implemented in Python and is called via `qbffam <family> <n>` where `n` is the size of the generated formula according to the definition of the respective family and `family` is one of the following 12 formula families:

`KBKF`	`KBKF_LD`	`KBKF_QU`
`Parity`	`LQParity`	`QUParity`
`EQ`	`EQ-Sq`	`BEQ`
`LONSING`	`TRAPDOOR`	`CR`

Details on the formula families as well as an overview of their applications in proof complexity are given in Section 3. All of the generated formulas are false QBFs in prenex conjunctive normal form (PCNF) and have the structure $QX_1 \dots QX_n.\phi$ where the prefix $Q_1X_1 \dots Q_nX_n$ contains quantifiers $Q_i \in \{\forall, \exists\}$ and the matrix ϕ is a propositional formula in conjunctive normal form (CNF). As usual, a CNF is a conjunction of clauses, a clause is a disjunction of literals, and a literal is a variable or a negated variable. All formulas are closed, i.e., all variables are quantified. Formulas in PCNF are typically represented in the QDIMACS[4] format which is supported by the majority of modern QBF solvers.

Organisation. The rest of the paper is structured as follows. We first review related work in Section 2. In Section 3 we discuss the 12 formula families supported by `QBFFam`. Here we also give an overview of relevant results from proof complexity for these formulas in several QBF proof systems. In particular, we report which formula family has/does not have short proofs in what proof systems. In Section 4 we describe a case study, where we evaluate modern QBFs

[4] `http://www.qbflib.org/qdimacs.html`

solvers on two formula families. We conclude with an outlook to future work in Section 5.

2 Related Work

The tool most closely related to QBFFam is the tool CNFGen [29] which is a generator for crafted SAT instances from propositional proof complexity. Among others, it supports the generation of formula families such as the *pigeonhole formulas* or the *Tseitin formulas*. Many of the provided formula families are known to be exponentially hard for propositional resolution and therefore for plain resolution-based CDCL solvers, as propositional resolution and (non-deterministic) CDCL are known to be equivalent [3,33]. This is also underpinned by experimental evaluations. Together with the rigorous lower bounds obtained in proof complexity such experiments help to understand the solving behavior of SAT solvers, identify their limitations, and also point towards directions for improvement.

To the best of our knowledge, there is no similar generator in the context of QBF solving so far. There are tools and frameworks for generating hard *random formulas* with a CNF matrix [15] or non-CNF matrix [18]. These random generators are used to empirically support theoretical characterizations of random formulas (cf. for example [17]). On the practical side they form the foundation for fuzzing, a testing technique that aims to find defects in solvers by massively solving random instances, thus achieving high code coverage, which is important to detect conceptual errors and only sporadically triggered corner cases [14].

3 Formula Families

Currently, our tool QBFFam supports the generation of 12 different formula families which are summarized in Table 1 together with a characterization in terms of number of quantifier alternations, number of variables, and number of clauses. Additionally, we also provide information on their proof complexity indicating for which proof systems short proofs or lower bounds are known.

Q-resolution (QRes) is the simplest among the considered proof systems, providing rules for resolution over existential variables and universal reduction [28]. In QRes-QU [20] resolution over universals is allowed as well. In long-distance resolution QRes-LD [1] certain resolution steps, forbidden in Q-Resolution, generating tautologous clauses are allowed. The system QRes-LQU$^+$ [2] combines long-distance resolution with resolution over universals, yielding a very powerful proof system. Another extension of QRes is QRes-SYM [26] which is able to exploit symmetries of formulas [27].

The proof system ∀Exp-Res [25] is the formal basis for expansion-based QBF solving. In addition to the resolution rule it has a rule that captures the expansion of universal variables and the renaming of existential variables in terms of annotations. The more powerful proof systems IR-calc and IRM-calc provide more flexibility than ∀Exp-Res in the way how and when annotations are obtained [10].

In the following, we briefly discuss the supported formula families.

Table 1. Characteristics of the families and overview of some results from proof theory.

formula family	#alt	#vars	#cl	QRes	QRes-LD	QRes-QU	QRes-LQU$^+$	∀Exp-Res	IR-calc	IRM-calc	QRes-SYM
KBKF	$n+1$	$4n$	$4n+1$	✗	✓	✓	✓	✗	✗	✓	✓
KBKF_LD	$n+1$	$4n$	$4n+1$	✗	✗	✓	✓	✗	✗	✗	✓
KBKF_QU	$n+1$	$5n$	$4n+1$	✗	✓	✗	✓	✗	✗	✓	✓
Parity	2	2n	$4n-2$	✗	✓	✗	✓	✓	✓	✓	✓
LQParity	2	2n	$8n-6$	✗	✗	✗	✓	✓	✓	✓	✓
QUParity	2	2n+1	$8n-6$	✗	✗	✗	✗	✓	✓	✓	✓
EQ	3	$3n$	$2n+1$	✗	✓	✗	✓	✗	✗	✓	✓
EQ-Sq	3	n^2+4n	$5n^2$	✗	✓	✗	✓	✗	✗	✓	✓
BEQ	4	$6n+2$	$5n+2$	✗	✓	✗	✓	✗	✗	✓	✗
CR	2	n^2	$2n$	✓	✓	✓	✓	✓	✓	✓	✓
TRAPDOOR	3	$O(n^2)$	$O(n^2)$	✓	✓	✓	✓	✓	✓	✓	✓
LONSING	2	$O(n^2)$	$O(n^2)$	✓	✓	✓	✓	✓	✓	✓	✓

✓ ... short proofs (poly size) ✗ ... no short proofs (exponential lower bounds)
#alt ... number of quantifier alternations
#vars ... number of variables #cl ... number of clauses

KBKF Formulas and Extensions KBKF_LD, KBKF_QU. Already in their first paper on Q-resolution from 1995 [28] Kleine Büning, Karpinski, and Flögel introduced a formula family that is nowadays known as the KBKF formula family. Since their inception, the KBKF formulas have triggered lots of research in QBF proof complexity. The original motivation of [28] was to provide quantified extended Horn formulas that have no short QRes proofs. Interestingly, the formulas also provide exponential separations between QRes and QRes-QU [20] as well as between QRes and QRes-LD [19]. The formulas KBKF have unbounded quantifier complexity, and much later it became clear [7], that such formulas are indeed needed for separating QRes and QRes-QU. The KBKF formulas remain hard in expansion-based systems ∀Exp-Res and IR-calc, but become easy in IRM-calc [10].

Extensions of KBKF have been introduced to obtain hard formulas for more powerful proof systems. In particular, the formula family KBKF_QU duplicates universal variables in the prefix and in clauses and becomes hard for QRes-QU, but remains easy for QRes-LD [2]. Another modification KBKF_LD [2] adds variables from the innermost existential quantifier block to some clauses. These formulas are hard for the systems QRes-LD [2] and IRM-calc [10]. All three formula families exhibit many symmetries, making them simple if reasoning on symmetries is supported [26].

A simple self-contained proof of the hardness of KBKF in QRes is given in [5]. Most further hardness results mentioned above lift QRes hardness to stronger proof systems.

Parity Formulas Parity and Extensions LQParity, QUParity. The formulas of the parity family Parity are Tseitin-transformed CNF representations of QBFs with structure $\exists x_1, \ldots, x_n \forall z.(z \vee \phi_n) \wedge (\neg z \vee \neg \phi_n)$ where $\phi_n = x_1 \oplus \ldots \oplus x_n$. The unique strategy for falsifying the formula is to set the only universal variable z to $x_1 \oplus \ldots \oplus x_n$. Hence, the unique Herbrand function for z must compute the parity function, which is exponentially hard for bounded-depth circuits AC^0 [22]. As strategy extraction in QRes and QRes-QU is in AC^0 [1], the QRes and QRes-QU proofs of Parity must be of exponential size. An alternative proof of hardness for Parity in QRes, not relying on the complex machinery of AC^0 lower bounds, is given in [7].

In contrast, Parity is easy for QRes-LD [16] and ∀Exp-Res [10]. The extensions LQParity and QUParity are constructed to obtain hard formulas for QRes-LD and QRes-LQU$^+$, respectively [10].

Equality Formulas EQ and Extensions EQ-Sq and BEQ. The equality formulas [6] have a quantifier prefix of the form $\exists x_1 \ldots x_n \forall u_1 \ldots u_n \exists t_1 \ldots t_n$ and encode that $x_i \leftrightarrow u_i$ for $1 \leq i \leq n$. The t_i variables are Tseitin variables for obtaining a PCNF, collected in one clause of size n. Arguably, the equality formulas are the simplest formulas hard formulas for QRes. In [6] a semantic technique via cost is developed to show their hardness (as well as many more hardness results). A related technique [4] is applicable to show their hardness for the expansion systems ∀Exp-Res and IR-calc. However, they become easy in QRes-LD [8].

The EQ-Sq formulas [8] are a 'squared' version of the EQ formulas with n additional variables in each of the first two blocks and n^2 innermost Tseitin variables. They are used to show an exponential separation between QRes-LD without universal reduction (exponential lower bounds for EQ-Sq) and the proof system M-Res (short proofs for EQ-Sq) [8].

Finally the blocked equality formulas BEQ introduce a blocker such that symmetries are destroyed and cannot be exploited to find short proofs [13]. This technique does not only work for the equality formulas, but it is a general approach to eliminate symmetries from a formula without changing its meaning.

Completion Principle, Trapdoor, and Lonsing Formulas. The last block of formulas from Table 1 comprises of three formula families that are easy for all of the described proof systems. Though QCDCL is associated with the proof systems QRes and QRes-LD (QCDCL runs can be efficiently translated into QRes-LD refutations as clauses learned in QCDCL can be derived in QRes-LD), this correspondence is not an exact one as demonstrated by recent research [5, 23]. In particular, [23] has shown that practical QCDCL does not simulate QRes. This builds on the completion principle formulas CR, first described in [25], which describe an easy 'completion' game, played on an $n \times n$ matrix by two players

(cf. [25]). These formulas are easy for QRes, but hard for practical QCDCL using UIP learning [23].

This result was further strengthened in [9], where QRes and QCDCL (with arbitrary learning schemes) are shown to be incomparable. This is witnessed by the Parity formulas, which are hard in QRes, but easy in QCDCL (with the right heuristics, possibly difficult to find in practice).[5]

In the opposite direction, the TRAPDOOR and LONSING formulas (first defined in [9] and [30], respectively) are easy for QRes, but require exponential running time in QCDCL (even with arbitrary learning schemes). Both principles build QBFs that incorporate the well-known propositional pigeonhole principle (PHP). Using the right quantifier prefix, which needs to be obeyed by QCDCL decision heuristics, they 'trap' QCDCL into refuting the PHP formulas (which are exponentially hard for propositional resolution [21] and hence for (Q)CDCL), while easy (even constant size) QRes proofs of TRAPDOOR and LONSING exist.

4 Case Study

Our tool QBFFam opens up many possibilities to conduct interesting experiments. In particular, it can be used to investigate whether the solver implementations indeed follow the behavior predicted by proof complexity and to compare their strength.

As a first case study, we consider 30 formulas of the KBKF family as well as 30 formulas of the LQParity family. We selected those families because they are well investigated in proof complexity and correspond to incomparable proof systems as discussed above. In both cases we selected the values 10, 15-40, 50, 60, 70, 80 for n.

In our experiments, we considered five solvers in six configurations. The QCDCL solver DepQBF (version 6.1) [31] was run with and without long-distance enabled. We included the solver Qute [32] as a second QCDCL solver which supports dynamic dependency learning. As expansion-based solvers, we included Rareqs [24] which recursively processes the quantifier alternations to build the propositional abstraction of a formula as well as the non-recursive expansion-based solver Ijtihad. Finally, we included the solver Caqe that implements causal abstraction [36] and which dominated the QBFEval competitions [34][6] over the last years. All experiments were run on Intel Xeon E5-2620 v4 CPU machines with the timeout set to 300 seconds and the memory restricted to 7GB.

The results of our experiments are shown in Figure 1. The plot on the left shows the runtimes for the KBKF family. For four of the six solvers, the formulas are very hard, especially for the expansion-based solvers Rareqs, which does not solve any formula, and Ijtihad, which solves only one formula within the time limit. Also, for Qute and DepQBF the formulas get difficult with increasing n. Both get until $n = 20$. For this formula, DepQBF needs 167 seconds and Caqe

[5] However, formulas hard for QRes-LD such as LQParity are hard for QCDCL (with arbitrary heuristics) from a theory point of view.

[6] http://www.qbfeval.org

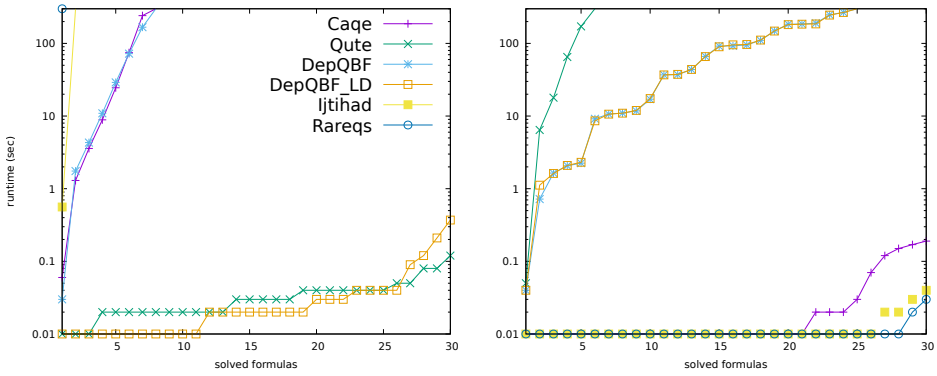

Fig. 1. Runtime comparison on `KBKF` formulas (left) and `LQParity` formulas (right).

needs 241 seconds. For DepQBF with long-distance resolution and Qute these formulas are very easy: all of them are solved in less than one second, confirming results from proof complexity.

The situation is different for the `LQParity` formulas (see Fig. 1 on the right). Here Caqe, Ijtihad, and Rareqs solve all of the formulas quickly. This is also in accordance with the results from proof complexity. The formulas are hard for the QCDL-based solvers, which could only solve 24 formulas (both configurations of DepQBF) and five formulas (Qute).

This also indicates that there is a close connection between the theoretical properties of the underlying proof systems and the practical implementations of the solvers.

5 Conclusion

We presented `QBFFam`, a tool for the generation of instances related to prominent formula families from proof complexity. We briefly described these families and surveyed recent results from proof complexity which help to understand the power of proof systems, and thus the power of QBF decision procedures and their implementations in QBF solvers. In a small case study we evaluated recent QBF solvers on two formula families and could indeed observe that the properties predicted by proof complexity are reflected by the solving runtimes. This opens the way to many further interesting experiments.

In future, `QBFFam` can be extended to support graph-based formulas [11] or random formulas [6]. Both also play an important role in the context of proof complexity. Another extension of `QBFFam` that seems to be of practical interest is the generation of true formulas. True QBFs are currently not investigated in proof complexity with the argument that in QBF, proof systems for satisfiability are dual to those of unsatisfiability. Having such formula families, however, seems to be useful for evaluating solver implementations as well.

References

1. Balabanov, V., Jiang, J.R.: Unified QBF certification and its applications. Formal Methods Syst. Des. **41**(1), 45–65 (2012)
2. Balabanov, V., Widl, M., Jiang, J.R.: QBF resolution systems and their proof complexities. In: Proc. of the 17th Int. Conf. on Theory and Applications of Satisfiability Testing (SAT'14). Lecture Notes in Computer Science, vol. 8561, pp. 154–169. Springer (2014)
3. Beame, P., Kautz, H.A., Sabharwal, A.: Towards understanding and harnessing the potential of clause learning. J. Artif. Intell. Res. (JAIR) **22**, 319–351 (2004)
4. Beyersdorff, O., Blinkhorn, J.: Dynamic QBF dependencies in reduction and expansion. ACM Trans. Comput. Log. **21**(2), 8:1–8:27 (2020)
5. Beyersdorff, O., Blinkhorn, J.: A simple proof of QBF hardness. Information Processing Letters **168** (2021)
6. Beyersdorff, O., Blinkhorn, J., Hinde, L.: Size, cost, and capacity: A semantic technique for hard random qbfs. Log. Methods Comput. Sci. **15**(1) (2019)
7. Beyersdorff, O., Blinkhorn, J., Mahajan, M.: Hardness characterisations and size-width lower bounds for QBF resolution. In: Proc. ACM/IEEE Symposium on Logic in Computer Science (LICS). pp. 209–223. ACM (2020)
8. Beyersdorff, O., Blinkhorn, J., Mahajan, M.: Building strategies into QBF proofs. J. Autom. Reasoning **65**(1), 125–154 (2021)
9. Beyersdorff, O., Böhm, B.: Understanding the relative strength of QBF CDCL solvers and QBF resolution. In: Proc. Innovations in Theoretical Computer Science (ITCS). pp. 12:1–12:20 (2021)
10. Beyersdorff, O., Chew, L., Janota, M.: New resolution-based QBF calculi and their proof complexity. ACM Transactions on Computation Theory **11**(4), 26:1–26:42 (2019)
11. Beyersdorff, O., Chew, L., Mahajan, M., Shukla, A.: Feasible interpolation for QBF resolution calculi. Logical Methods in Computer Science **13**(2) (2017)
12. Beyersdorff, O., Janota, M., Lonsing, F., Seidl, M.: Quantified Boolean formulas. In: Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.) Handbook of Satisfiability, 2nd edition. Frontiers in Artificial Intelligence and Applications, IOS press (2021)
13. Blinkhorn, J., Beyersdorff, O.: Proof complexity of QBF symmetry recomputation. In: Proc. of the 22nd Int. Conf. on Theory and Applications of Satisfiability Testing (SAT'19). Lecture Notes in Computer Science, vol. 11628, pp. 36–52. Springer (2019)
14. Brummayer, R., Lonsing, F., Biere, A.: Automated testing and debugging of SAT and QBF solvers. In: Proc. of the 13th Int. Conf. on Theory and Applications of Satisfiability Testing (SAT'10). Lecture Notes in Computer Science, vol. 6175, pp. 44–57. Springer (2010)
15. Chen, H., Interian, Y.: A model for generating random quantified boolean formulas. In: Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI'05). pp. 66–71. Professional Book Center (2005)
16. Chew, L.: QBF proof complexity. Ph.D. thesis, University of Leeds, Leeds (2017)
17. Creignou, N., Daudé, H., Egly, U., Rossignol, R.: New results on the phase transition for random quantified boolean formulas. In: Proc. of the 11th Int. Conf. on Theory and Applications of Satisfiability Testing (SAT'08). Lecture Notes in Computer Science, vol. 4996, pp. 34–47. Springer (2008)

18. Creignou, N., Egly, U., Seidl, M.: A framework for the specification of random SAT and QSAT formulas. In: Proc. of the 6th Int. Conf. on Tests and Proofs (TAP'12). Lecture Notes in Computer Science, vol. 7305, pp. 163–168. Springer (2012)
19. Egly, U., Lonsing, F., Widl, M.: Long-distance resolution: Proof generation and strategy extraction in search-based QBF solving. In: Proc. Logic for Programming, Artificial Intelligence, and Reasoning (LPAR). pp. 291–308 (2013)
20. Gelder, A.V.: Contributions to the theory of practical quantified boolean formula solving. In: Proc. of the 18th Int. Conf. on Principles and Practice of Constraint Programming (CP'18). Lecture Notes in Computer Science, vol. 7514, pp. 647–663. Springer (2012)
21. Haken, A.: The intractability of resolution. Theoretical Computer Science **39**, 297–308 (1985)
22. Håstad, J.: Computational Limitations of Small Depth Circuits. MIT Press, Cambridge (1987)
23. Janota, M.: On Q-Resolution and CDCL QBF solving. In: Proc. International Conference on Theory and Applications of Satisfiability Testing (SAT). pp. 402–418 (2016)
24. Janota, M., Klieber, W., Marques-Silva, J., Clarke, E.M.: Solving QBF with counterexample guided refinement. Artif. Intell. **234**, 1–25 (2016)
25. Janota, M., Marques-Silva, J.: Expansion-based QBF solving versus Q-resolution. Theor. Comput. Sci. **577**, 25–42 (2015)
26. Kauers, M., Seidl, M.: Short proofs for some symmetric quantified boolean formulas. Inf. Process. Lett. **140**, 4–7 (2018)
27. Kauers, M., Seidl, M.: Symmetries of quantified boolean formulas. In: Theory and Applications of Satisfiability Testing - SAT 2018 - 21st International Conference, SAT 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 9-12, 2018, Proceedings. Lecture Notes in Computer Science, vol. 10929, pp. 199–216. Springer (2018)
28. Kleine Büning, H., Karpinski, M., Flögel, A.: Resolution for quantified boolean formulas. Inf. Comput. **117**(1), 12–18 (1995)
29. Lauria, M., Elffers, J., Nordström, J., Vinyals, M.: CNFgen: A Generator of Crafted Benchmarks. In: Proc. of the 20th Int. Conf. on Theory and Applications of Satisfiability Testing (SAT'17). Lecture Notes in Computer Science, vol. 10491, pp. 464–473. Springer (2017)
30. Lonsing, F.: Dependency Schemes and Search-Based QBF Solving: Theory and Practice. Ph.D. thesis, Johannes Kepler University Linz (2012)
31. Lonsing, F., Egly, U.: Depqbf 6.0: A search-based QBF solver beyond traditional QCDCL. In: CADE. Lecture Notes in Computer Science, vol. 10395, pp. 371–384. Springer (2017)
32. Peitl, T., Slivovsky, F., Szeider, S.: Qute in the QBF evaluation 2018. J. Satisf. Boolean Model. Comput. **11**(1), 261–272 (2019)
33. Pipatsrisawat, K., Darwiche, A.: On the power of clause-learning SAT solvers as resolution engines. Artif. Intell. **175**(2), 512–525 (2011)
34. Pulina, L., Seidl, M.: The 2016 and 2017 QBF solvers evaluations (qbfeval'16 and qbfeval'17). Artif. Intell. **274**, 224–248 (2019)
35. Shukla, A., Biere, A., Pulina, L., Seidl, M.: A survey on applications of quantified boolean formulas. In: Proc. of the the 31st IEEE Int. Conf. on Tools with Artificial Intelligence, (ICTAI'19). pp. 78–84. IEEE (2019)
36. Tentrup, L.: CAQE and quabs: Abstraction based QBF solvers. J. Satisf. Boolean Model. Comput. **11**(1), 155–210 (2019)