


# QRP+Gen: A Framework for Checking Q-Resolution Proofs with Generalized Axioms

Mark Peyrer 

Institute for Symbolic Artificial Intelligence, Johannes Kepler University, Linz, Austria

Martina Seidl 

Institute for Symbolic Artificial Intelligence, Johannes Kepler University, Linz, Austria

---

## Abstract

Q-resolution is a proof system for quantified Boolean formulas (QBFs) that forms the foundation for search-based QBF solvers with clause and cube learning. To derive stronger clauses and cubes, Q-resolution was extended with so-called generalized axioms. The derivation of such generalized axioms relies on solving oracles that could be, for example, SAT solvers or even QBF solvers.

While the correctness of results obtained with classical QCDCL-based solving can be efficiently certified by an independent checker, until now, proof generation had to be turned off to benefit from generalized axioms. Consequently, the results obtained with reasoning under generalized axioms could not be certified independently. To overcome this restriction, we present **QRP+Gen**, a novel framework to automatically generate and check Q-resolution proofs that contain generalized axioms. To this end, we extended the Q-resolution format **QRP** such that all necessary information is included to verify the correctness of generalized axioms. Our extension allows to integrate certificates produced by any oracle which can produce automatically checkable proofs. Furthermore, we developed a proof checker that orchestrates the proof checking of the core Q-resolution proof and the proofs produced by the oracles. As a case study, we equipped the search-based QBF solver **DepQBF** with proof-producing oracles for the SAT-based techniques trivial truth and trivial falsity.

**2012 ACM Subject Classification** Theory of computation → Automated reasoning

**Keywords and phrases** Automated Reasoning, Quantified Resolution Proof, Generalized Axioms

**Digital Object Identifier** 10.4230/LIPIcs.SAT.2025.25

**Supplementary Material** *Software*: <https://doi.org/10.5281/zenodo.16307432>

**Funding** Parts of this work have been supported by the LIT AI Lab funded by the state of Upper Austria and by the Austrian Science Fund (FWF) [10.55776/COE12].

## 1 Introduction

The solving landscape for quantified Boolean formulas (QBFs) is very heterogeneous [17]. In contrast to SAT, which has conflict-driven clause learning (CDCL) [18] as predominant solving paradigm, the situation is less clear for QBFs. Besides QCDCL, the CDCL variant for QBF, expansion- and abstraction-based approaches have proven to be very powerful and of orthogonal strength [11]. To combine different approaches, different directions are taken: a very loose integration is obtained by portfolio-based solving [9, 8] and by pre- and inprocessing [5, 14]. A very tight integration is based on the idea of learning generalized axioms [15]: during search QCDCL solvers learn clauses (conflicts) and cubes (solutions) that are justified by Q-resolution [10]. A generalized axiom is a clause or cube that is learned by calling an oracle solver. Such an oracle can be any complete or incomplete QBF solver.

So far, the usage of generalized axioms has one major drawback: if used in a Q-resolution proof, the QBF variant of resolution proofs as produced by QCDCL solvers, its correctness has no justification, because it was derived by using an oracle solver. Consequently, Q-resolution proofs that rely on generalized axioms cannot be checked by an independent checking tool and the result of the QBF solver cannot be certified. Therefore, generalized axioms have to be disabled if a certificate is required.



© Mark Peyrer and Martina Seidl;  
licensed under Creative Commons License CC-BY 4.0

28th International Conference on Theory and Applications of Satisfiability Testing (SAT 2025).

Editors: Jeremias Berg and Jakob Nordström; Article No. 25; pp. 25:1–25:10

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

To overcome this limitation and certify the correctness of Q-resolution proofs with generalized axioms, we present the novel certification framework **QRP+Gen** which extends the Q-resolution certification framework **QRPcheck** with generalized axioms. To this end, we furthermore extended the Q-resolution format **QRP** to incorporate their certification. As a proof-of-concept we equipped the QCDCL-based solver **DepQBF** [13] with proof generation for generalized axioms obtained by deciding certain propositional abstractions of the considered QBF. Our implementation is publicly available at

<https://doi.org/10.5281/zenodo.16307432>

## 2 QCDCL with Generalized Axioms

Traditional search-based QBF solving with conflict-driven clause learning and solution-driven cube learning (QCDCL) [23, 7] successively generates variable assignments for a given QBF  $\phi = \Pi : \Psi$  until  $\phi$  is decided. Usually  $\phi$  is in prenex conjunctive normal form (PCNF). Here,  $\Pi = Q_1x_1, \dots, Q_nx_n$  is a quantifier prefix with  $Q_i \in \{\forall, \exists\}$  and the propositional matrix  $\Psi$  is a conjunction of clauses. A clause is a disjunction of literals and a literal is a variable or a negated variable. Note that not every variable that occurs in the matrix has to be part of the prefix (free variables). In contrast to SAT, variables may not be assigned in an arbitrary order, but the dependencies imposed by the quantifier prefix have to be considered. Still, clauses are learned from assignments that falsify the formula to prune the search space. In addition, also cubes (conjunctions of literals) are learned from assignments that satisfy the propositional part of the formula (solutions). While clauses are conjunctively added to the propositional matrix, cubes are disjunctively collected in a cube database. The addition of clauses and cubes preserves the truth/falsity of the original formula, because they are derived via some variant of Q-resolution, the extension of propositional resolution for QBF. On this basis, QCDCL solvers can produce Q-resolution proofs as certificates for the solving results. For false QBFs, clause Q-resolution proofs are generated, and, dually, cube Q-resolution proofs are generated for true formulas. Both types of proofs can efficiently be checked [20]. We want to underline how Q-resolution is in general not expressive enough for QCDCL, which causes the need to disable certain techniques in actual implementations when proof generation is required.

In general, learning smaller clauses and cubes w.r.t the amount of literals is favorable during the search. However, in classical QCDCL implementations, especially large cubes are learned because the cube database is empty in the beginning. Hence, there are no cubes serving as axioms in cube Q-resolution proofs. To initialize the cube database, assignments satisfying the propositional matrix are taken, from which smaller cubes are derived until the empty cube is found [24]. Axioms of clausal Q-resolution proofs are the clauses from the propositional matrix of the original formula.

To learn smaller clauses and cubes faster, the concept of *generalized axioms* has been introduced [15]. Consider a QBF  $\phi = \Pi : \Psi$  and assume that during the search a partial assignment  $\sigma = \{l_1, \dots, l_n\}$  has been found respecting the dependencies imposed by the quantifier prefix. If a variable  $x \in \sigma$ , then  $x$  is set to true, whereas if  $\neg x \in \sigma$ , then  $x$  is set to false. Furthermore, there is no variable  $x$  with  $x, \neg x \in \sigma$ . Now assume that  $\phi' = \phi[\sigma]$  is not decided where  $\phi[\sigma]$  is the formula obtained by assigning all variables according to  $\sigma$ . Using an oracle solver that is able to decide  $\phi'$ , either a clause or a cube can be learned dependent on the value of  $\phi'$ . If  $\phi'$  is false, then the clause  $(\bar{l}_1 \vee \dots \vee \bar{l}_n)$  can be learned. Otherwise, the cube  $(l_1 \wedge \dots \wedge l_n)$  can be learned. The oracle can be any solver that is able to decide a QBF. It could be, for example, a solver that is based on a decision procedure of orthogonal

strength like expansion-based solving [11]. Also incomplete methods can be used as oracles. If no solution is found, then the QCDCL search is resumed under the assignment  $\sigma$ . The following example illustrates the power of the approach.

► **Example 1.** A formula family of false instances that is known to be hard for Q-resolution are the *parity formulas* [1]  $\text{parity}(n) = \exists x_1 \dots x_n \forall z \exists t_2 \dots t_n : \psi(n)$  where  $\psi(n)$  is defined by

$$\psi(n) = \text{xor}(x_1, x_2, t_2) \wedge \bigwedge_{i=3}^n \text{xor}(t_{i-1}, x_i, t_i) \wedge (z \vee t_n) \wedge (\neg z \vee \neg t_n)$$

with  $\text{xor}(a, b, c) = (\neg a \vee \neg b \vee \neg c) \wedge (a \vee b \vee \neg c) \wedge (\neg a \vee b \vee c) \wedge (a \vee \neg b \vee c)$ . To obtain formulas that are hard for Q-resolution, but easy with generalized axioms consider the formula family

$$\text{parity}^*(n) = \exists x \exists x_1 \dots x_n \forall z \exists t_2 \dots t_n : (x \rightarrow \psi(n)) \wedge (\neg x \rightarrow \psi(n)) \wedge (z \leftrightarrow t_n)$$

After assigning  $x$  to either value, a solver needs to solve a parity problem of a given size  $n$  with the additional equivalence  $(z \leftrightarrow t_n)$ . Obviously, this contradicts with  $(z \not\leftrightarrow t_n)$  that is in  $\psi(n)$ . However, if the variables are assigned according to the quantifier prefix as it is done by a QCDCL-solver, this obvious contradiction is detected very late. Assuming that a QCDCL-solver uses Q-resolution as a proof format, the algorithm will be restricted by such formulas causing the  $\text{parity}^*(n)$  to be hard for rather small values of  $n$ .

However, if an oracle call to a SAT solver is made, this will almost immediately conclude unsatisfiability due to being able to neglect the quantifier dependencies. It is easy to prove that the propositional formula is false for both values of  $x$ . In consequence, also the QBF is false. By deriving  $x$  and  $\neg x$  as generalized axioms, short proofs can be found for the QBF.

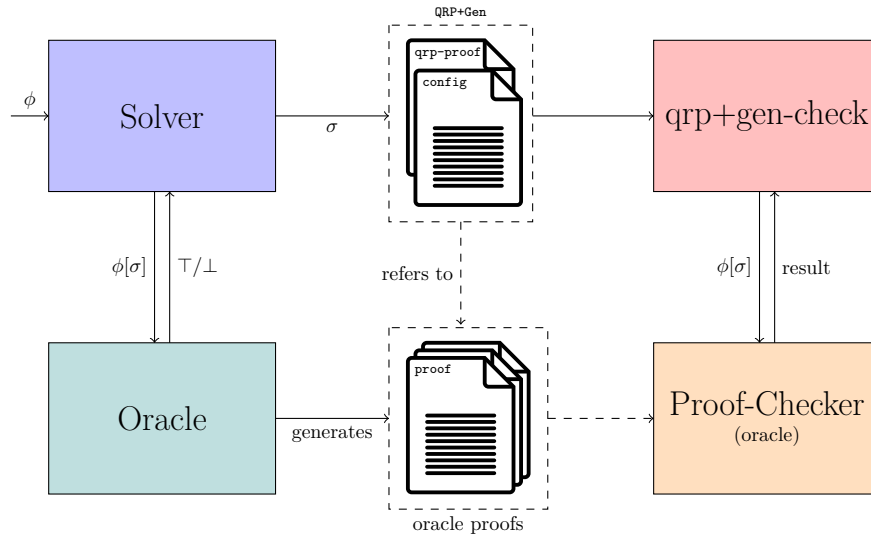
The only problem with this approach is that the proof containing generalized axioms is not a valid certificate for the solving result, because the axioms are not justified. For this reason, QCDCL solvers do not provide certificates when generalized axioms are enabled so far. Therefore, no independent checking is possible and the result of the solver (and the internally called oracles) has to be trusted without witness.

### 3 The QRP+Gen Certification Framework

To enable certification for QCDCL solvers that use solver oracles to learn generalized axioms either in terms of clauses or cubes, we developed the **QRP+Gen** framework. It is based on the **QRPcheck** certification tool chain [19] which provides a checker for Q-resolution proofs in the **QRP** format. To close the holes in the proofs introduced by generalized axioms, the oracles have to also provide witnesses which are used to justify the soundness of the learned axioms. The general architecture of our framework is shown in Figure 1 which allows for an automatic integration of the different proofs giving a full witness for the input formula.

#### The QRP+Gen Proof Format

The **QRP** format was extended to the **QRP+Gen** format. This extension allows to link the proof produced by an oracle with the respective generalized axiom. Proofs in the **QRP+Gen** format may contain classical axioms in the form of clauses that occur in the input formula, or in the form of covering cubes which satisfy the propositional matrix. Furthermore, it may contain clauses/cubes that are derived via resolution by referring to their antecedents as well as clauses/cubes simplified by universal/existential reduction. The soundness of deriving



■ **Figure 1** Architecture of QRP+Gen.

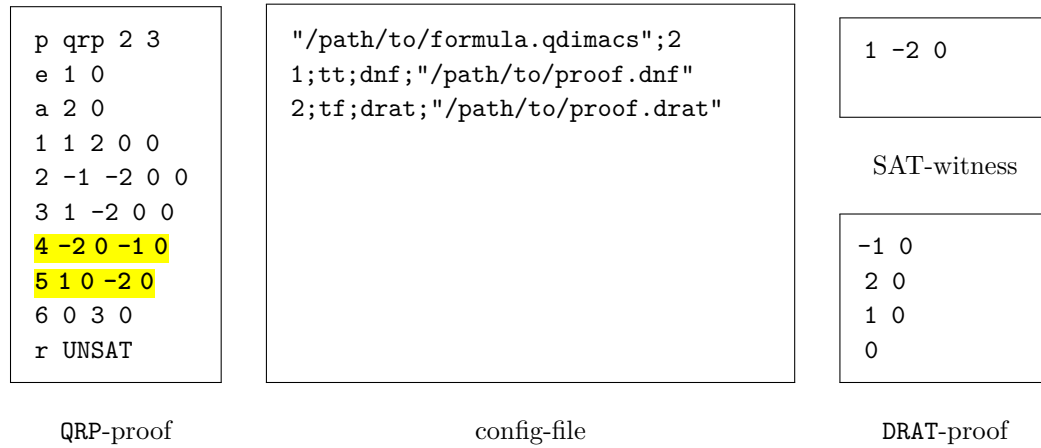
those clauses/cubes can be efficiently checked based on the clauses of the input formula and clauses/cubes that have previously been checked. In addition, *generalized axioms* are allowed in the QRP+Gen format. These are neither derived by Q-resolution, nor occur in the input formula, nor are covering cubes. Here, we need the proof of the oracle. A generalized axiom is indicated by one *negative* antecedent in the format, which absolute value describes the ID of the corresponding oracle proof. Information on the respective proof is given in a configuration file which has to be provided to the checker. The configuration file has the following format:

```
CONFIG = HEADER \n {ENTRY \n}
HEADER = " Path " ; number
ENTRY = number ; transformer-specifier ; format-specifier ; " Path "
```

The *HEADER* consists of the path to the original qdimacs-formula as well as the amount of proofs that are referenced in the following (the number of entries). Each *ENTRY* is identified by a unique proof ID. Next, a transformer is specified, which applies simple syntactical modifications on the input QBF that can efficiently be checked. For example, it could be necessary to drop the prefix of the input QBF. Next, the proof format of the proof produced by the oracle is indicated. Finally, the path to the proof file is provided. An example of a config-file is shown in the middle of Figure 2.

### Generation of Proofs

A QBF  $\phi$  in PCNF is passed to a QCDCL solver that calls oracle solvers to learn generalized axioms. It is assumed that these oracle solvers are able to produce certificates that can be efficiently checked. During the classical QCDCL search, a Q-resolution trace with clauses and cubes is produced as usual. If a generalized axiom is learned based on a partial assignment  $\sigma$ , then the oracle solver has to provide a certificate for the truth value of  $\phi[\sigma]$ . An entry in the configuration file is generated and the generalized axiom is added to the Q-resolution trace with the respective pointer to the configuration file. By storing the generalized axiom,



■ **Figure 2** Example of a QRP+Gen proof.

$\phi$  under assignment  $\sigma$  can be reconstructed. The Q-resolution proof with generalized axioms refers to the oracle proofs which are stored in separate files. The solving oracle does not need any information about the QCDCL solver and its proof generation does not have to be modified. The left part of Figure 2 shows a Q-resolution proof with two generalized axioms (the highlighted clause/cube). On the right, there are two oracle proofs: one is a DRAT proof and the other one is a satisfying assignment.

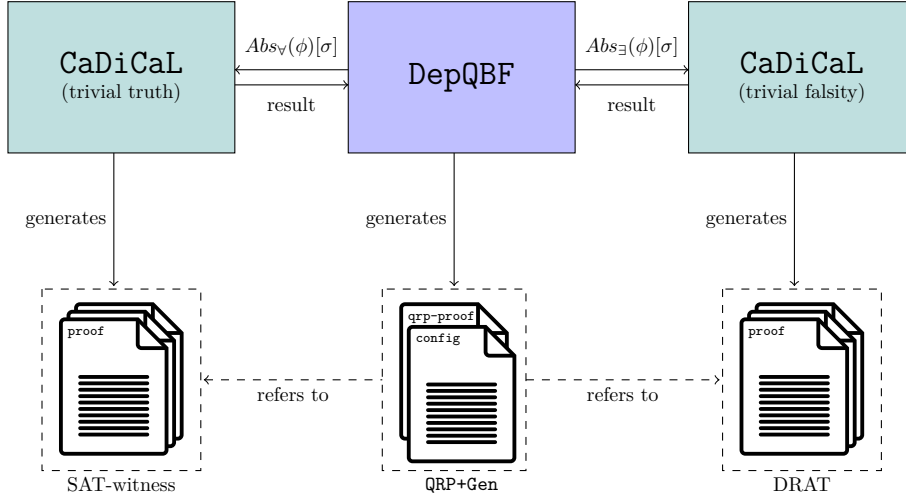
### Proof Checking

We extended the tool **QRPcheck** [19] to be able to check QRP+Gen-proofs. For checking, a config-file has to be specified as a parameter and is loaded before starting with verifying the QRP+Gen-proof. An example of such a file can be seen in the middle of figure 2. This way, all information necessary to find and check the oracle proofs is provided. In case a generalized axiom is found in the proof, an external checker as specified in the config file is called. Before calling the external checker, some transformation steps might be necessary to ensure that  $\phi[\sigma]$  has the correct format. Furthermore, the correctness of all clauses/cubes derived by standard Q-resolution is checked.

## 4 Certifying Generalized Axioms in DepQBF

In the following, we present how we extended the QBF solver **DepQBF** to obtain certificates for generalized axioms that use two SAT solvers as oracles. The SAT solvers are called with certain propositional abstractions of the input QBF under the current assignment  $\sigma$ . Clauses and cubes learned as generalized axioms are justified by the techniques *Trivial Truth* and *Trivial Falsity* [6, 15]. To certify the correctness of such generalized axioms, different types of proofs are combined as shown in Figure 3. In addition, **DepQBF** learns generalized axioms by calling the expansion-based solver **Nenofex** [12] which is not able to produce proofs. Extending **Nenofex** with proof generation capabilities is beyond the scope of this work, hence we disabled learning generalized axioms with **Nenofex**. In the original implementation, **DepQBF** employed **PicoSAT** [3] as SAT solver. Instead of **PicoSAT**, we use the latest version of the SAT solver **CaDiCaL** [4] which is able to produce DRAT proofs [22].<sup>1</sup>

<sup>1</sup> We thank Maximilian Heisinger for integrating **CaDiCaL** into **DepQBF**.



■ **Figure 3** Combination of proofs in QRP+Gen.

### Trivial Falsity

Trivial falsity is based on the following observation: if the matrix  $\Psi$  of a given QBF  $\phi = \Pi : \Psi$  is already propositionally unsatisfiable, then the QBF is false. Given a partial assignment  $\sigma$  that is consistent with the Q-resolution rules, then the clause  $\neg\sigma$  can be learned as generalized axiom if  $\Psi[\sigma]$  is unsatisfiable. The **DRAT** proof of the unsatisfiability of  $\Psi[\sigma]$  produced by **CaDiCaL** serves as witness for the generalized axiom and is referenced in the **QRP+Gen** proof. In the proof checking phase, our checker **QRP+GenCheck** can easily check the correctness of the generalized axiom  $\neg\sigma$  by stripping the original formula of its prefix (done by a transformer in **QRP+GenCheck**) and pass it to **DRAT-trim** [22] together with the corresponding **DRAT** proof.

### Trivial Truth

Trivial truth is based on the following observation: a QBF  $\Pi : \Psi$  is true if the propositional formula  $\Psi'$  is satisfiable where  $\Psi'$  is obtained from  $\Psi$  by removing all universal literals from each clause. Given a partial assignment  $\sigma$  that is consistent with the quantifier prefix  $\Pi$ , then the cube  $\sigma$  can be learned as generalized axiom if  $(\Psi[\sigma])'$  (i.e. the first  $\sigma$  is applied on  $\Psi$  and then the remaining universal literals are dropped) is satisfiable. As a witness of  $\sigma$ , a satisfying assignment  $\tau$  of  $(\Psi[\sigma])'$  is stored. For checking the correctness of generalized axioms justified by trivial truth, we implemented a checker called **ttCheck** that takes  $\Psi$  as well as  $\sigma$  and  $\tau$  as input. It checks the consistency of  $\tau$  and  $\sigma$ , i.e., it ensures that there is no literal  $l$  with  $l \in \tau$  and  $\bar{l} \in \sigma$ . Furthermore, it checks that for each clause  $C \in \Psi$ , there exists literal  $l \in C$  such that  $l \in \tau$  and  $l$  is existential or that  $l \in \sigma$ . This allows **ttCheck** to certify the correct application of trivial truth in a very efficient manner.

## 5 Experimental Evaluation

We conducted two experiments showing the impact of **QRP+Gen** applied on the previously described case study. Both experiments were executed on a Ubuntu 24.04 machine with dual-socket AMD EPYC 7313@3.7 GHz CPUs. We assigned 8 GB RAM as well as 1 h per task as resource limitations.

## Crafted Formulas

Going back to example 1, the introduced crafted formulas  $\text{parity}^*(n)$  are hard for Q-resolution but easy when generalized axioms are used. We implemented a generator for the  $\text{parity}^*$  formulas on the base of **qbffam** [2] and tested the results. **depqbf-qrp** does not use generalized axioms, whereas **depqbf-qrp-gen** uses trivial truth and trivial falsity. Both are producing a proof trace. This first experiment used the range  $n = 2 \dots 25$ . The needed resources per formula size can be seen in figure 4. Note that the space-data represent the size of the proof, not the RAM used.

While **DepQBF** without generalized axiom shows an exponential increase in both solving time and proof size, the version using generalized axioms only has a linear increase in space. The reason for that is **CaDiCaL**, which is able to solve the trivial falsity represented by the formula directly whereas the Q-resolution of **DepQBF** has to face the difficulty of the parity family [1]. This underlines the potential of using generalized axioms, allowing the oracle to exploit structures unknown to the QBF-solver.

## QBF-Gallery 23

In the second experiment, we used the QBF-Gallery 2023 [21] as a test set to evaluate the runtime of different configurations. As a secondary set, we used the preprocessor **Bloqqer** [5] to generate simplified versions of the formulas from the Gallery. Per test set, four different configurations were investigated. Techniques like advanced dependency management and lazy QPUP [16] were deactivated in general as they do not support proof generation. The rest of the configuration was dependent on the run:

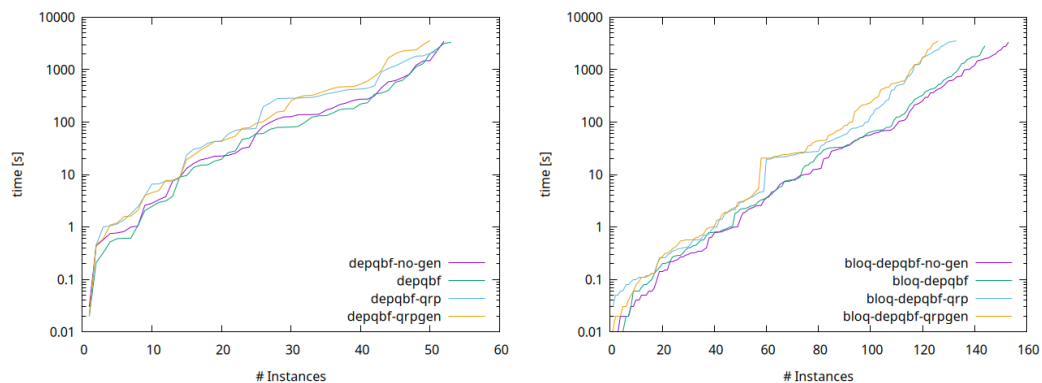
- **depqbf-no-gen** is without tracing and no generalized axioms.
- **depqbf** is the base configuration without tracing but with generalized axioms.
- **depqbf-qrp** is with normal QRP-tracing (thus without generalized axioms)
- **depqbf-qrp-gen** is with full QRP+Gen-tracing

The results can be seen in figure 5. Additionally, we tested how many samples can be verified using **QRP+GenCheck** by granting an additional 1 h of resources. For the QBF-gallery 2023 (left part of figure 5), out of 50 solved examples, 48 instances were verified w.r.t. the resource limitations. For the preprocessed samples (right part of figure 5), all 126 could be verified. In terms of proof sizes, QRP and QRP+Gen are similar for all instances of both test sets.

The results of Lonsing et al. [15] are reflected in both benchmark-sets. While **DepQBF** with Dynamic Quantified Blocked Clause Elimination (dyn-QBCE) and generalized axioms is able to outperform its previous version in general, using a version applying only generalized

n	time [s]		space [byte]	
	depqbf-qrp-gen	depqbf-qrp	depqbf-qrp-gen	depqbf-qrp
2	0.02	0.02	308	924
3	0.02	0.02	448	2.3k
...	...	...	...	...
22	0.02	208	3.8k	8.2G
23	0.02	544	4.0k	16G
24	0.02	1442	4.2k	38G
25	0.02	N/A	4.4k	N/A

■ **Figure 4** The first experiment conducted with the crafted formulas.



■ **Figure 5** Experiments conducted on QBF gallery 23 and preprocessed QBF gallery 23.

axioms is not. Q-resolution is not powerful enough to certify the deletion of clauses which are blocked under a specific assignment. Providing a solution (e.g. by extending the proof system) would be beyond the scope of this show case. However, we have shown that even without dyn-QBCE generalized axioms can be beneficial. Moreover, as there is only little drawback enabling **QRP+Gen**-certifications, we lay the foundation towards a more general proof system enabling the full potential of the techniques applied by **DepQBF** with proper certification.

## 6 Conclusion

We presented **QRP+Gen**, the – to the best of our knowledge – first framework allowing certificates for reasoning with generalized axioms. We extended the well-established **QRP** proof format with justifications for clauses and cubes derived by generalized axioms based on results obtained from oracle solvers. In our framework, any oracle solver able to produce proofs for which checking tools are available can be considered.

As a case study, we implemented proof generation and checking for generalized axioms obtained by the abstraction-based techniques trivial truth and trivial falsity in the solver **DepQBF**. We evaluated the framework on different types of benchmarks and observed only little overhead caused by proof generation. With our framework, it is now possible to independently confirm the result of a QBF solver that relies on general axioms.

The presented approach is, however, not limited to generalized axioms. Our framework can be employed in other scenarios where different solving systems based on different proof systems are combined. Potential applications include interfacing proofs produced by preprocessing tools and complete solvers. Our framework could also be applied in the context of cube&conquer-based solving [8].

---

## References

- 1 Olaf Beyersdorff, Leroy Chew, and Mikoláš Janota. New resolution-based QBF calculi and their proof complexity. *ACM Transactions on Computation Theory*, 11(4):1–42, 2019. doi:10.1145/3352155.
- 2 Olaf Beyersdorff, Luca Pulina, Martina Seidl, and Ankit Shukla. Qbffam: A tool for generating QBF families from proof complexity. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 21–29. Springer, 2021. doi:10.1007/978-3-030-80223-3\_3.

- 3 Armin Biere. PicoSAT essentials. *Journal on Satisfiability, Boolean Modeling and Computation*, 4(2-4):75–97, 2008. doi:10.3233/SAT190039.
- 4 Armin Biere, Tobias Faller, Katalin Fazekas, Mathias Fleury, Nils Froleyks, and Florian Pollitt. CaDiCaL 2.0. In *36th International Conference on Computer Aided Verification*, pages 133–152. Springer, 2024. doi:10.1007/978-3-031-65627-9\_7.
- 5 Armin Biere, Florian Lonsing, and Martina Seidl. Blocked clause elimination for QBF. In *23rd International Conference on Automated Deduction*, pages 101–115. Springer, 2011. doi:10.1007/978-3-642-22438-6\_10.
- 6 Marco Cadoli, Andrea Giovanardi, and Marco Schaerf. An algorithm to evaluate quantified boolean formulae. *AAAI/IAAI*, 98:262–267, 1998. URL: <http://www.aaai.org/Library/AAAI/1998/aaai98-036.php>.
- 7 Enrico Giunchiglia, Massimo Narizzano, and Armando Tacchella. Clause/term resolution and learning in the evaluation of quantified boolean formulas. *Journal of Artificial Intelligence Research*, 26:371–416, 2006. doi:10.1613/JAIR.1959.
- 8 Maximilian Heisinger, Martina Seidl, and Armin Biere. Paraqooba: A fast and flexible framework for parallel and distributed QBF solving. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 426–447. Springer, 2023. doi:10.1007/978-3-031-30823-9\_22.
- 9 Holger H Hoos, Tomáš Peitl, Friedrich Slivovsky, and Stefan Szeider. Portfolio-based algorithm selection for circuit qbfs. In *24th International Conference of Principles and Practice of Constraint Programming*, pages 195–209. Springer, 2018. doi:10.1007/978-3-319-98334-9\_13.
- 10 Mikoláš Janota. On Q-Resolution and CDCL QBF solving. In *19th International Conference on Theory and Applications of Satisfiability Testing*, pages 402–418. Springer, 2016.
- 11 Mikoláš Janota and Joao Marques-Silva. Expansion-based QBF solving versus Q-Resolution. *Theoretical Computer Science*, 577:25–42, 2015. doi:10.1016/J.TCS.2015.01.048.
- 12 Florian Lonsing and Armin Biere. Nenofex: Expanding nnf for qbf solving. In *11th International Conference on Theory and Applications of Satisfiability Testing*, pages 196–210. Springer, 2008. doi:10.1007/978-3-540-79719-7\_19.
- 13 Florian Lonsing and Uwe Egly. DepQBF 6.0: A search-based QBF solver beyond traditional QCDCL. In *26th International Conference on Automated Deduction*, pages 371–384. Springer, 2017. doi:10.1007/978-3-319-63046-5\_23.
- 14 Florian Lonsing and Uwe Egly. QRATPre+: effective QBF preprocessing via strong redundancy properties. In *22nd International Conference on Theory and Applications of Satisfiability Testing*, pages 203–210. Springer, 2019. doi:10.1007/978-3-030-24258-9\_14.
- 15 Florian Lonsing, Uwe Egly, and Martina Seidl. Q-Resolution with Generalized Axioms. In *19th International Conference on Theory and Applications of Satisfiability Testing*, pages 435–452. Springer, 2016. doi:10.1007/978-3-319-40970-2\_27.
- 16 Florian Lonsing, Uwe Egly, and Allen Van Gelder. Efficient clause learning for quantified Boolean formulas via QBF pseudo unit propagation. In *16th International Conference on Theory and Applications of Satisfiability Testing*, pages 100–115. Springer, 2013. doi:10.1007/978-3-642-39071-5\_9.
- 17 Paolo Marin, Massimo Narizzano, Luca Pulina, Armando Tacchella, and Enrico Giunchiglia. Twelve years of QBF evaluations: QSAT is PSPACE-hard and it shows. *Fundamenta Informaticae*, 149(1-2):133–158, 2016. doi:10.3233/FI-2016-1445.
- 18 Joao Marques-Silva, Inês Lynce, and Sharad Malik. Conflict-driven clause learning SAT solvers. In *Handbook of satisfiability*, pages 133–182. ios Press, 2021. doi:10.3233/FAIA200987.
- 19 Aina Niemetz. QRPcheck. URL: <https://fmv.jku.at/qrpcheck/>.
- 20 Aina Niemetz, Mathias Preiner, Florian Lonsing, Martina Seidl, and Armin Biere. Resolution-based Certificate Extraction for QBF: (tool presentation). In *15th International Conference on Theory and Applications of Satisfiability Testing*, pages 430–435. Springer, 2012. doi:10.1007/978-3-642-31612-8\_33.

- 21   Luca Pulina, Martina Seidl, and Simone Heisinger. QBFGallery 2023. URL: <https://qbf23.pages.sai.jku.at/gallery/>.
- 22   Nathan Wetzler, Marijn JH Heule, and Warren A Hunt Jr. DRAT-trim: Efficient checking and trimming using expressive clausal proofs. In *17th International Conference on Theory and Applications of Satisfiability Testing*, pages 422–429. Springer, 2014.
- 23   Lintao Zhang and Sharad Malik. Conflict driven learning in a quantified Boolean satisfiability solver. In *IEEE/ACM international conference on Computer-aided design*, pages 442–449, 2002. doi:10.1145/774572.774637.
- 24   Lintao Zhang and Sharad Malik. Towards a symmetric treatment of satisfaction and conflicts in quantified boolean formula evaluation. In *International Conference on Principles and Practice of Constraint Programming*, pages 200–215. Springer, 2002. doi:10.1007/3-540-46135-3\_14.