FERPModels: a Certification Framework for Expansion-Based QBF Solving

Vedad Hadžić, Roderick Bloem, Ankit Shukla and Martina Seidl

January 27, 2023

FERPModels: A Certification Framework for Expansion-Based QBF Solving

Vedad Hadzic, Roderick Bloem
Technical University Graz
Graz, Austria
{vedad.hadzic, roderick.bloem}@iaik.tugraz.at

Ankit Shukla, Martina Seidl
Johannes Kepler University Linz
Linz, Austria
{ankit.shukla, martina.seidl}@jku.at

***Abstract*—Modern expansion-based solvers for quantified Boolean formulas (QBFs) are successful in many applications. However, no such solver supports the generation of proofs needed to independently validate the correctness of the solving result and for the extraction of winning strategies which encode concrete solutions to the application problems.**

In this paper, we present a complete tool chain for proof generation, result validation, and for universal winning strategy extraction in the context of expansion-based solving. In particular, we introduce a proof format for the ∀Exp-Res calculus on which expansion-based solving is founded, implement proof generation in a recent QBF solver, provide a checker for these proofs, and develop a new strategy extraction algorithm.

***Index Terms*—QBFs, ∀Exp-Res, proof generation, strategy extraction**

I. Introduction

Quantified Boolean Formulas (QBF) provide an attractive framework for encoding and solving reasoning problems ranging from symbolic reasoning in artificial intelligence [14] to formal verification [9] and automatic synthesis of computing systems [7], [8] (see [18] for a survey). By translating such application problems into QBF decision problems, QBF solvers can be employed as black-boxes for solving these problems. Ideally, QBF solvers give not only a true/false answer but they also generate an efficiently checkable *proof* to independently verify the correctness of the result. Furthermore, users also need *winning strategies* from the QBF solver to get concrete solutions for the original problems. Winning strategies, for example, can encode the error trace in verification, the synthesized program in synthesis, a feasible plan in planning, or a witness that there is none.

In recent years, expansion-based QBF solving, combined with counter-example guided abstraction refinement [10], was shown to be extremely powerful [13], [16]. The theoretical foundation of this approach is the ∀Exp-Res [11] proof system. While this proof system is theoretically well understood, no recent expansion-based solver supports the generation of efficiently checkable proofs and the extraction of winning strategies so far. In contrast, solvers based on other solving paradigms support the generation of such proofs and provide winning strategies [15], [17].

his work has been supported by the Austrian Science Fund (FWF) under project W1255-N23 and the LIT AI Lab funded by the State of Upper Austria.

This paper presents the framework FERPModels for generating, checking, and extracting winning strategies from ∀Exp-Res-*refutation proofs* (proofs of *falsity*). As there is no proof format for ∀Exp-Res proofs, we first propose a proof format called FERP. Second, we extend the expansion-based QBF solver `Ijtihad` [6] to generate FERP proofs. Third, we implement a proof checker for FERP proofs. Finally, we develop a new symbolic algorithm for the extraction of universal winning strategies from FERP proofs. We evaluate our framework FERPModels on several benchmarks and compare it to other certification frameworks.

II. Preliminaries

We consider QBFs $Q_1x_1 \dots Q_nx_n.\phi$ in prenex conjunctive normal form (PCNF) where $Q_i \in \{\forall, \exists\}$ and ϕ is a propositional formula in conjunctive normal form (CNF) over variables x_i. A formula in CNF is a conjunction of clauses and a clause is a disjunction of literals. A literal is a variable or a negated variable. A QBF $\forall x_1 \Pi.\phi$ is false iff $\Pi.\phi[x_1/\top]$ or $\Pi.\phi[x_1/\bot]$ is false where $\phi[x/t]$ denotes the CNF obtained by setting x to truth value t. As a running example we consider the false QBF $\Phi = \exists x \forall a \exists y.((x \vee a \vee y) \wedge (\neg x \vee \neg a \vee y) \wedge (\neg y))$.

In expansion-based QBF solving, one kind of variable (either the existential or the universals) is eliminated according to the quantifier semantics. To preserve the dependencies imposed by the quantifiers, new copies of the variables that occur to the right of the expanded variables have to be introduced. If we expand a in QBF Φ, we obtain the PCNF $\Phi[a/\bot, y/y^{\bar{a}}] \wedge \Phi[a/\top, y/y^{a}]$ which contains only existentially quantified variables. Hence it can be solved by a SAT solver. Obviously, careless expansion leads to exponential space consumption. Only in combination with powerful abstraction and pruning techniques does expansion-based solving work well in practice. Expansion-based solving is based on the ∀Exp-Res calculus [11] that provides two kinds of rules: (1) the axiom rule for expanding universal variables and introducing the necessary copies of the existential variables and (2) the propositional resolution rule. The correctness of ∀Exp-Res proofs is checkable in polynomial time.

III. The Certification Framework at a Glance

We provide a complete framework to certify and validate the results of the expansion-based QBF solver `Ijtihad` [6].

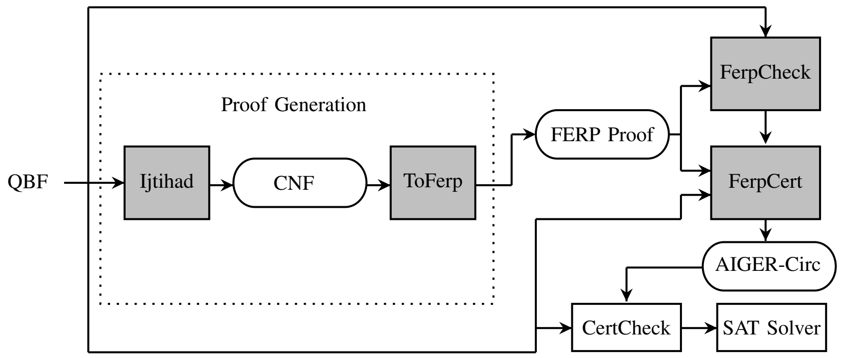


Figure 1: The FERPModels framework. The gray components are new or modified.

The overall certification workflow of FERPModels is shown in Figure 1. FERPModels is publicly available at

https://github.com/SFMV/ferp-models

To install all necessary third-party tools like SAT solvers and to build our tools, the script `requirements.py` is provided. The full tool chain is called via `pipeline.py <qbf> <f>` where `<qbf>` is the input QBF in `QDIMACS` format and `<f>` is the name of the file in which the generated winning strategy is stored. In the following we describe different components of FERPModels.

a) Proof Generation: Given a QBF Φ in `QDIMACS` format, the expansion-based QBF solver `Ijtihad` decides whether Φ is true or false. If Φ is false, `Ijtihad` dumps the partial expansion, the unsatisfiable CNF formula ψ, in DIMACS format. In DIMACS and in QDIMACS variables are represented as integers. Note that the same integer in the DIMACS and QDIMACS file usually not refers to the same variable. The mapping between the variables of ψ and Φ is stored in comments in the DIMACS file. From this annotated DIMACS formula, the tool `ToFerp` generates a $\forall$Exp-Res proof in the FERP format. Figure 2 shows an example.

The grammar for the `FERP` proof format is as follows:

⟨*trace*⟩	::=	{⟨*annotation*⟩} {⟨*clause*⟩}
⟨*annotation*⟩	::=	‘x’ ⟨*vars*⟩ ⟨*vars*⟩ ⟨*literals*⟩
⟨*vars*⟩	::=	{⟨*idx*⟩} ‘0’
⟨*literals*⟩	::=	{⟨*lit*⟩} ‘0’
⟨*antecedents*⟩	::=	⟨*idx*⟩ ‘0’ | ⟨*idx*⟩ ⟨*idx*⟩ ‘0’
⟨*clause*⟩	::=	⟨*idx*⟩ ⟨*literals*⟩ ⟨*antecedents*⟩
⟨*lit*⟩	::=	⟨*idx*⟩ | ⟨*neg*⟩
⟨*idx*⟩	::=	‘1’ | ‘2’ | … | ⟨*max-idx*⟩
⟨*neg*⟩	::=	‘−’ ⟨*idx*⟩

Symbol `0` is a delimiter. The proof lines starting with `x` contain information about the variable mapping. The first section lists integer names of variables from CNF ψ. The second section lists the integer names of existential variables

```
QBF
p cnf 3 3
e 1 0
a 2 0
e 3 0
1 2 3 0
-1 -2 3 0
-3 0

CNF
p cnf 3 4
1 3 0
-1 2 0
-3 0
-2 0
```

∀Exp-Res proof

```
c variable mappings
x 1 0 1 0 0
x 2 0 3 0 2 0
x 3 0 3 0 -2 0
c resolution
c line idx bold
1  1 3 0 1 0
2 -1 2 0 2 0
3 -3 0 3 0
4 -2 0 3 0
5  2 3 0 1 2 0
6  2 0 5 3 0
7  0 6 4 0
```

Figure 2: QBF in QDIMACS from Sec. II and CNF expansion (left), and $\forall$Exp-Res proof in FERP format (right).

from QBF Φ. In the last section, there are universal literals which are to the left of the existential variables in the prefix. The polarity of the literal indicates the assignment. In our example, variables x, a, y are represented by integers $1, 2, 3$ in the QDIMACS file. In the CNF file x maps to 1, while y^a and $y^{\bar{a}}$ map to 2 and 3. The remaining lines provide information about the clauses of the $\forall$Exp-Res proof. After a unique line number, the first section shows the literals of a clause. The second section contains either one or two numbers. If it contains one number n, then it says that the clause was obtained by applying the axiom rule of $\forall$Exp-Res for the n-th clause of Φ. For example, the clause with index 1 in Figure 2 is obtained from clause $(x \vee a \vee y)$ of Φ, the first clause in the QDIMACS file. Similarly, two numbers indicate the application of the resolution rule and refer to the two parent clauses.

b) Proof Checking: The tool `FerpCheck` checks the generated `FERP` proofs by validating the expansion and resolution steps performed by the `Ijtihad`. `FerpCheck` traverses the FERP proof in the reverse topological order, starting with an empty constraint and checks the validity of each step. This way, all parts of the trace irrelevant for deriving the empty constraint can be omitted. The tool terminates with a negative answer if it finds an invalid step in the proof. If all the antecedents are visited without any error, the checker returns *OK*.

c) Universal Winning Strategy Extraction: FerpCert extracts a certificate from the FERP proof in terms of a universal winning strategy. A winning strategy states how to set universal variables according to given values of the existential variables such that the formula evaluates to false. To this end, we developed an algorithm for obtaining symbolic encodings of winning strategies. It is inspired by the interactive approach by Beyersdorff et al. [2] which cannot serialize strategies and for which no implementation exists. The extracted certificate is represented as a Boolean circuit in Aiger format for And-Inverter-Graphs (AIG) [5]. To avoid redundancies in the AIG, we employ structural hashing.

d) Certificate Checking: To ensure that the generated certificate is indeed a winning strategy for the universal player, we perform an additional step of certificate validation. We use the CertCheck [15] tool to replace the universal variables of the original QBF by their winning strategy functions and to transform the resulting propositional formula into CNF. If this propositional formula is unsatisfiable, then the certificate indeed represents a winning strategy for the universal player.

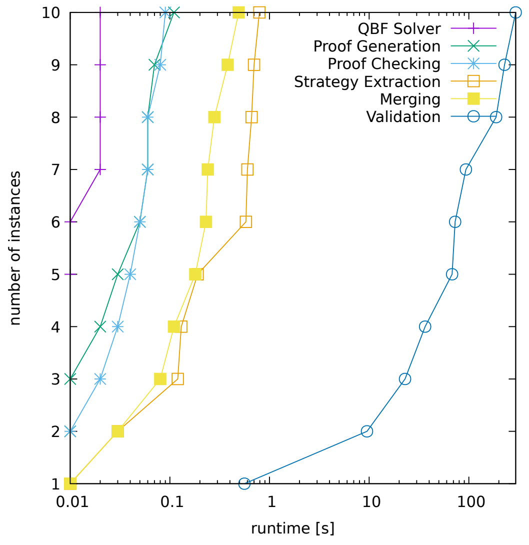
IV. Evaluation

We evaluate our certification framework FERPModels and compare it to two other certification frameworks: CaQE[1] [17] and QBFCert[2] [15]. QBFCert is based on Q-resolution proofs as produced by clause and cube learning approaches (QCDCL) (see [4]) and realizes a similar tool chain as FERPModels. In contrast, CaQE is based on clausal abstraction and produces winning strategies during the search. It does not generate proofs, so efficient result validation is possible.

We use the SAT solver Picosat[3] (version 965) to obtain the propositional resolution proof from the CNF generated by Ijtihad. For plugging the AIG certificates obtained from FerpCert into the original QBFs, we use the tool CertCheck of QBFCert. The obtained CNFs are checked by the SAT solver Kissat.[4] This SAT solver is also used to validate the winning strategies generated by the other two approaches. The QBF solver CaQE (version 2) directly generates AIG certificates during solving. The framework QBFCert uses the QBF solver DepQBF[5] [12] (version 4.01) for generating Q-resolution proofs.

a) Instances: As benchmarks we considered 10 formulas of the QParity$_n$ family [3] with parameter $n \in \{10, 20, \ldots, 100\}$ as well as 118 false formulas from the benchmark set of the PCNF track of QBFEval 20. We included those formulas that could be solved either by Ijtihad, CaQE, or DepQBF in their standard configuration within 15 minutes.

[1]http://https://www.react.uni-saarland.de/tools/caqe/
[2]http://fmv.jku.at/qbfcert
[3]http://fmv.jku.at/picosat
[4]http://fmv.jku.at/kissat
[5]https://lonsing.github.io/depqbf/



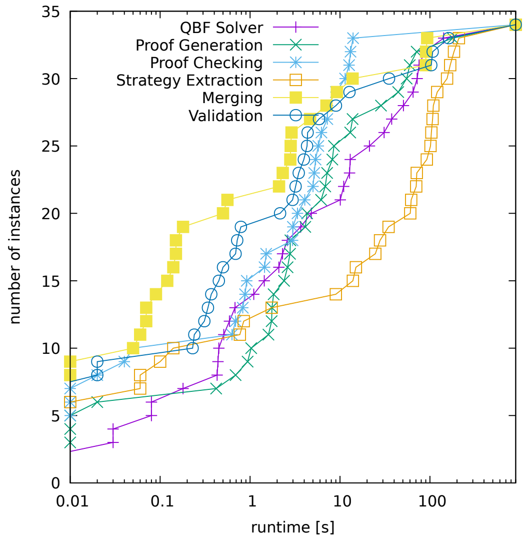


Figure 3: Runtime of FERPModels on QParity$_n$ (top) QBFEval (bottom) formulas.

The experiments were run on Intel Xeon E5-2620v4 CPUs with a time limit of 15 minutes and a memory limit of 50GB.

b) Results and Analysis: Table I compares the stages of the three frameworks w.r.t. number of solved instances and runtime. While CaQE and DepQBF could only solve one ($n = 10$) resp. two ($n \in \{10, 20\}$) of the QParity$_n$ formulas, Ijtihad could solve all of the 10 formulas. A detailed analysis of the runtimes of the individual components of FERPModels is shown in the first plot of Figure 3. The most time-consuming part of our tool chain is the final SAT check validating the correctness of the AIG certificates. This experiment indicates that our certification approach can fully exploit the power of expansion-based solving.

Out of the 118 formulas from QBFEval 20, Ijtihad solved 92 formulas. About 90% of the proofs could be checked by FERPModels within the given time limit. In contrast, CaQE solved 61 formulas. At this point, a comparison was not possible

Table I: Comparison of Certification Frameworks.

	Instances					**Average Runtime* [s]**					
stage	sv	pg	ch	ex	va	sv	pg	ch	ex	mg	va
Parity (10 formulas)											
FERPModels	10	10	10	10	10	0.0	0.0	0.0	0.4	0.2	101.4
CaQE	1	–	–	–	1	0.10	–	–	–	0.01	0.48
`QBFCert`	2	–	2	2	1	0.01	–	0.02	0.01	0.00	0.03
QBFEval (118 formulas)											
FERPModels	92	86	82	46	38	34.9 (19.8)	81.4 (17.1)	33.0 (3.7)	135.6 (53.0)	54.6 (9.8)	14.1 (14.1)
CaQE	61	–	–	–	60	93.4 (81.8)	–	–	–	0.6 (0.6)	0.7 (0.7)
`QBFCert`	49	–	48	47	46	52.3 (21.2)	–	13.4 (4.8)	1.8 (0.9)	0.7 (0.7)	9.0 (9.0)

* avg runtime of instances that completed the stage (avg runtime of all validated instances)
– . . . not applicable

sv . . . solved formulas	pg . . . proof generation	ch . . . proof checking
ex . . . strategy extraction	mg . . . merging	va . . . validation

as CaQE does not produce proofs. The solver DepQBF solved 49 formulas. The results of 48 formulas could be checked. Hence, we could conisderably increase the number of formulas with certified results. For almost all formulas solved by CaQE and DepQBF winning strategies in terms of AIGs could be generated and validated. With FERPModels winning strategies could be obtained for 46 formulas. Of those 38 could be checked by a SAT solver within the time limit. The smaller amount of extracted winning strategies can be explained by the fact that the pipeline of `FerpCert` contains more stages. The runtimes of proof generation (pg) in Table I and the second plot of Figure 3 indicate that a considerable amount of time is need for getting the resolution proof by an extra SAT solver call. Further, the extraction algorithm of `FerpCert` is more expensive than the extraction algorithm of `QBFCert` which is linear in the proof size [1]. From a proof-theoretical point of view this is not surprising [4].

V. Conclusion

We presented the first certification framework for expansion-based solving. The produced certificates in terms of proofs and universal winning strategies not only increase the trust in the solving result of an expansion-based solver, but also provide concrete solutions to application problems. Our experiments indicated that our framework works very well for formulas well suited for expansion-based solving. In the future, we also want to obtain existential strategies. Theoretically, this works dually, in practice, there are some technical challenges because the expansion-based solver generates a DNF formula that is then converted to a CNF formula introducing additional Tseitin variables. Another interesting line of research is the integration of preprocessing techniques to FERPModels, which are crucial for modern QBF solvers.

References

[1] Valeriy Balabanov and Jie-Hong R. Jiang. Unified QBF certification and its applications. *Formal Methods Syst. Des.*, 41(1):45–65, 2012.

[2] Olaf Beyersdorff, Leroy Chew, and Mikolas Janota. On unification of QBF resolution-based calculi. In *MFCS (2)*, volume 8635 of *Lecture Notes in Computer Science*, pages 81–93. Springer, 2014.

[3] Olaf Beyersdorff, Leroy Chew, and Mikolás Janota. Proof complexity of resolution-based QBF calculi. In *STACS*, volume 30 of *LIPIcs*, pages 76–89. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015.

[4] Olaf Beyersdorff, Janota Mikolás, Florian Lonsing, and Martina Seidl. Quantified Boolean Formulas. In *Handbook of Satisfiability*, volume 336, pages 1177 – 1221. IOS Press, 2021.

[5] Armin Biere, Keijo Heljanko, and Siert Wieringa. AIGER 1.9 and beyond. Technical Report 11/2, Institute for Formal Models and Verification, Johannes Kepler University, Altenbergerstr. 69, 4040 Linz, Austria, 2011.

[6] Roderick Bloem, Nicolas Braud-Santoni, Vedad Hadzic, Uwe Egly, Florian Lonsing, and Martina Seidl. Expansion-based QBF solving without recursion. In *FMCAD*, pages 1–10. IEEE, 2018.

[7] Roderick Bloem, Uwe Egly, Patrick Klampfl, Robert Könighofer, and Florian Lonsing. Sat-based methods for circuit synthesis. In *FMCAD*, pages 31–34. IEEE, 2014.

[8] Roderick Bloem, Robert Könighofer, and Martina Seidl. SAT-based synthesis methods for safety specs. In *VMCAI*, volume 8318 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2014.

[9] Uwe Egly, Martin Kronegger, Florian Lonsing, and Andreas Pfandler. Conformant planning as a case study of incremental QBF solving. *Ann. Math. Artif. Intell.*, 80(1):21–45, 2017.

[10] Mikolás Janota, William Klieber, João Marques-Silva, and Edmund M. Clarke. Solving QBF with counterexample guided refinement. In *SAT*, volume 7317 of *Lecture Notes in Computer Science*, pages 114–128. Springer, 2012.

[11] Mikoláš Janota and Joao Marques-Silva. Expansion-based QBF solving versus Q-resolution. *Theoretical Computer Science*, 577:25–42, 2015.

[12] Florian Lonsing and Armin Biere. DepQBF: A dependency-aware QBF solver. *JSAT*, 7(2-3):71–76, 2010.

[13] Florian Lonsing, Martina Seidl, and Allen Van Gelder. The QBF gallery: Behind the scenes. *Artif. Intell.*, 237:92–114, 2016.

[14] Christian Miller, Christoph Scholl, and Bernd Becker. Proving QBF-hardness in bounded model checking for incomplete designs. In *MTV*, pages 23–28. IEEE Computer Society, 2013.

[15] Aina Niemetz, Mathias Preiner, Florian Lonsing, Martina Seidl, and Armin Biere. Resolution-based certificate extraction for QBF - (tool presentation). In *SAT*, volume 7317 of *Lecture Notes in Computer Science*, pages 430–435. Springer, 2012.

[16] Luca Pulina and Martina Seidl. The 2016 and 2017 QBF solvers evaluations (QBFEVAL'16 and QBFEVAL'17). *Artif. Intell.*, 274:224–248, 2019.

[17] Markus N. Rabe and Leander Tentrup. CAQE: A certifying QBF solver. In *FMCAD*, pages 136–143. IEEE, 2015.

[18] Ankit Shukla, Armin Biere, Luca Pulina, and Martina Seidl. A survey on applications of quantified boolean formulas. In *ICTAI*, pages 78–84. IEEE, 2019.