Blocked Literals are Universal*

Marijn J.H. Heule[1], Martina Seidl[2], and Armin Biere[2]

[1] Department of Computer Science, The University of Texas at Austin, USA
marijn@cs.utexas.edu
[2] Institute for Formal Models and Verification, JKU Linz, Austria
martina.seidl@jku.at biere@jku.at

Abstract. We recently introduced a new proof system for Quantified Boolean Formulas (QBF), called QRAT, that opened up a variety of new preprocessing techniques. This paper presents a concept that follows from the QRAT proof system: blocked literals. Blocked literals are redundant universal literals that can be removed or added to clauses. We show that blocked literal elimination (BLE) and blocked literal addition are not confluent. We implemented BLE in the state-of-the-art preprocessor bloqqer. Our experimental results illustrate that the BLE extension improves solver performance on the 2014 QBF evaluation benchmarks.

1 Introduction

Preprocessing a *quantified Boolean formula* (QBF) is crucial to effective QBF solving, but often tricky to implement. That motivated us to develop a new proof system for QBF [1], called QRAT, which facilitates expressing all state-of-the-art QBF preprocessing techniques in a uniform manner. By these means, the correctness of the output of a preprocessor can be checked efficiently. Moreover, the QRAT proof system opened up a variety of new preprocessing techniques. In this paper, we study two of such new preprocessing techniques.

Universal pure literal elimination [2] and *blocked clause elimination* (BCE) [3] are important QBF preprocessing techniques. The QRAT proof system revealed that universal pure literals can be generalized in a similar way as existential pure literal elimination, i.e, via BCE. We call this new generalized concept *blocked literals*. We study two new QBF preprocessing techniques: *blocked literal elimination* (BLE) and *blocked literal addition* (BLA). BLE is the dual of BCE. We show that neither BLE nor BLA are confluent, in contrast to BCE.

Additionally, this paper presents the first implementation and evaluation of a new preprocessing technique that originated from the QRAT proof system. The general rules in the QRAT proof system are very expensive to implement. However, by focusing on BLE, a restricted version of the one of the QRAT rules, we were able to extend the state-of-the-art preprocessor bloqqer [3] is such a way that its performance is clearly improved.

* This work was supported by the Austrian Science Fund (FWF) through the national research network RiSE (S11408-N23), Vienna Science and Technology Fund (WWTF) under grant ICT10-018, DARPA contract number N66001-10-2-4087, and the National Science Foundation under grant number CCF-1153558.

2 Preliminaries

The language of QBF extends the language of propositional logic by existential and universal quantifiers over the propositional variables. As usual, we assume a QBF to be in *prenex conjunctive normal form* (PCNF). Note that any QBF of arbitrary structure can be efficiently transformed to a satisfiability equivalent formula in PCNF [4]. A QBF in PCNF has the structure $\Pi.\psi$ where the prefix Π has the form $Q_1X_1Q_2X_2\ldots Q_nX_n$ with disjoint variable sets X_i and $Q_i \in \{\forall, \exists\}$. The matrix ψ is a propositional formula in conjunctive normal form, i.e., a conjunction of clauses. A clause is a disjunction of literals and a literal is either a variable x (positive literal) or a negated variable $\bar{x}$ (negative literal). The variable of a literal is denoted by $\mathsf{var}(l)$ where $\mathsf{var}(l) = x$ if $l = x$ or $l = \bar{x}$. The negation of a literal l is denoted by $\bar{l}$. The quantifier $\mathsf{Q}(\Pi, l)$ of a literal l is Q_i if $\mathsf{var}(l) \in X_i$. Let $\mathsf{Q}(\Pi, l) = Q_i$ and $\mathsf{Q}(\Pi, k) = Q_j$, then $l \leq_\Pi k$ iff $i \leq j$. We consider only closed QBFs, so ψ contains only variables which occur in the prefix. For a clause C, we denote by $\overline{C}$ the assignment that falsifies all literals in C, i.e., $\overline{C} = \{(\bar{l}) \mid l \in C\}$. By $\top$ and $\bot$ we denote the truth constants *true* and *false*. QBFs are interpreted as follows: a QBF $\forall x\Pi.\psi$ is false iff $\Pi.\psi[x/\top]$ or $\Pi.\psi[x/\bot]$ is false where $\Pi.\psi[x/t]$ is the QBF obtained by replacing all occurrences of variable x by t. Respectively, a QBF $\exists x\Pi.\psi$ is false iff both $\Pi.\psi[x/\top]$ and $\Pi.\psi[x/\bot]$ are false. If the matrix ψ of a QBF ϕ contains the empty clause after eliminating the truth constants, then ϕ is false as usual. Accordingly, if the matrix ψ of QBF ϕ is empty, then ϕ is true. Two QBFs ϕ_1 and ϕ_2 are *satisfiability equivalent* (written as $\phi_1 \sim \phi_2$) iff they have the same truth value.

3 Universal Blocked Literals

This section presents the new concept of blocked literals; redundant universal literals that can be removed or added to clauses. Removing blocked literals from clauses is a generalization of *universal pure literal elimination*, which removes universal literals that are pure, i.e., occur either only positively or only negatively in the formula. In the popular game-based view of QBF [5][3], the optimal strategy for the universal player is to assign pure literals to false. Such a move will only shrink clauses and not satisfy clauses. The preprocessing techniques presented here will have the same property, but it is literal-based instead of variable-based. We explain the new concept of blocked literal using the previous concepts of blocked clauses, outer clauses, and outer formulas. These previous concepts are defined slightly differently (i.e., simplified) compared to earlier work [1] to make them easier to understand for readers that are less familiar with QBF.

[3] The evaluation of a QBF is described as a game between the existential player who owns the existential variables and the universal player who owns the universal variables of the formula. The existential player wants to satisfy the formula, while the universal player wants to falsify the formula.

Definition 1 (Outer Clause [1]): Let C be a clause occurring in QBF $\Pi.\psi$. The *outer clause* of C on literal $l \in C$, denoted by $\mathcal{OC}(\Pi, C, l)$, is given by the clause $\{k \mid k \in C, k \leq_\Pi l, k \neq l\}$.

Definition 2 (Outer Formula [6]): Let l be a literal occurring in QBF $\Pi.\psi$. The *outer formula* of $\Pi.\psi$ on l, denoted by $\mathcal{OF}(\Pi, \psi, l)$, is given by the unquantified formula $\{\mathcal{OC}(\Pi, C, l) \mid l \in C, C \in \psi\}$.

Definition 3 (Blocking Literal and Blocked Clause, see also [3]): Let C be a clause occurring in QBF $\Pi.\psi$. An existential literal $l \in C$ is called a *blocking literal* with respect to $\Pi.\psi$ if and only if $\overline{C}$ satisfies $\mathcal{OF}(\Pi, \psi, \bar{l})$. Clause C is called a *blocked clause* if and only if there exists a blocking literal $l \in C$.

Definition 4 (Blocked Literal, instance of [1]): Let C be a clause occurring in QBF $\Pi.\psi$. Universal literal l is called a *blocked literal* with respect to C and $\Pi.\psi$ if and only if $\overline{C}$ satisfies $\mathcal{OF}(\Pi, \psi, \bar{l})$.

Notice the subtle difference in the names of the concepts: the new *blocked* literals are always universal literals, while *blocking* literals are always existential literals. This naming convention is motivated as follows: blocked literals are redundant –like blocked clauses– while blocking literals [3] are *not* redundant, but the reason why the clauses, in which they occur, are blocked and thus redundant.

3.1 Blocked Literal Elimination

We refer to *blocked literal elimination* (BLE) as removing blocked literals in a formula until fixpoint. For the formal proof of the soundness of (a generalization of) BLE, we refer to Theorem 2 of our IJCAR'14 paper [1]. From that theorem it follows that BLE preserves satisfiability, but not logical equivalence. In the game-based view of QBF, blocked literals can be ignored by the universal player, so it makes sense to remove them to simplify the formula at hand.

Theorem 1. *Blocked literal elimination is not confluent.*

Proof. Consider the true QBF $\Pi.\psi_{\text{SAT}} := \forall a, b \exists x.(a \vee b \vee x) \wedge (\bar{a} \vee \bar{b} \vee \bar{x})$. The outer formula $\mathcal{OF}(\Pi, \psi_{\text{SAT}}, \bar{a}) = (\bar{b})$ and $\mathcal{OF}(\Pi, \psi_{\text{SAT}}, b) = (a)$. BLE can remove literal a from $(a \vee b \vee x)$ because $\overline{(a \vee b \vee x)} = (\bar{a}) \wedge (\bar{b}) \wedge (\bar{x})$ satisfies $\mathcal{OF}(\Pi, \psi_{\text{SAT}}, \bar{a})$. Similarly, BLE can remove literal $\bar{b}$ from $(\bar{a} \vee \bar{b} \vee \bar{x})$ because $\overline{(\bar{a} \vee \bar{b} \vee \bar{x})} = (a) \wedge (b) \wedge (x)$ satisfies $\mathcal{OF}(\Pi, \psi_{\text{SAT}}, b)$. None of the other literals in ψ_{SAT} is blocked (although all x and $\bar{x}$ literals are blocking). BLE can remove either a from $(a \vee b \vee x)$ or $\bar{b}$ from $(\bar{a} \vee \bar{b} \vee \bar{x})$, but not both. Notice that removing both is also unsound as the resulting formula is unsatisfiable.

3.2 Blocked Literal Addition

Apart from eliminating blocked literals, one might also extend clauses by adding blocked literals in a similar fashion as adding hidden literals [7] or covered literals [8]. We will refer to *blocked literal addition* (BLA) as a procedure that adds blocked literals until fixpoint to a given formula.

We expect that BLA will be less useful in practice compared to BLE, because it weakens the formula. Weakening a formula is typically only useful when the formula is reduced in size. This is not the case for BLA. However, one could use BLA in combination with hidden literal and covered literal addition to check whether a clause is redundant.

Theorem 2. *Blocked literal addition is not confluent.*

Proof. Let $\Pi.\psi_{\text{UNSAT}} := \exists x, y\, \forall a\, \exists z.(x \vee a) \wedge (y \vee \bar{a}) \wedge (\bar{x} \vee z) \wedge (\bar{y} \vee \bar{z})$ be a false QBF. Further let ψ^C_{UNSAT} be ψ_{UNSAT} with $(\bar{x} \vee z)$ replaced by $C = (\bar{a} \vee \bar{x} \vee z)$, ψ^D_{UNSAT} be ψ_{UNSAT} with $(\bar{y} \vee \bar{z})$ replace by $D = (a \vee \bar{y} \vee \bar{z})$. Then $\mathcal{OF}(\Pi, \psi^C_{\text{UNSAT}}, a) = (x)$ and $\mathcal{OF}(\Pi, \psi^D_{\text{UNSAT}}, \bar{a}) = (y)$. Then $\overline{C} = (a) \wedge (x) \wedge (\bar{z})$ satisfies $\mathcal{OF}(\Pi, \psi^C_{\text{UNSAT}}, a)$. So BLA can add literal $\bar{a}$ to $(\bar{x} \vee z)$ in ψ_{UNSAT}. Further BLA can add literal a to $(\bar{y} \vee \bar{z})$, since $\overline{D} = (\bar{a}) \wedge (y) \wedge (\bar{z})$ satisfies $\mathcal{OF}(\Pi, \psi^D_{\text{UNSAT}}, \bar{a})$. Adding one of these blocked literals changes the other outer formula and *unblocks* the other literal.

Adding one literal does not only unblock the other, but adding both $\bar{a}$ to $(\bar{x} \vee z)$ and a to $(\bar{y} \vee \bar{z})$ is unsound as it results in a satisfiable formula.

3.3 Universal Expansion

In general, we expect that BLE is more effective than BLA. However, in the context of *universal expansion*, an effective preprocessing technique [9], the opposite is true. Universal expansion eliminates the innermost universal variable by duplicating some clauses containing innermost existential literals. More formally, universal expansion applies the following rule [1,9]:

$$\frac{\Pi \forall x \exists Y.\psi, C_1 \vee \bar{x}, \ldots, C_n \vee \bar{x}, D_1 \vee x, \ldots, D_m \vee x, E_1, \ldots, E_p}{\Pi \exists Y Y'.\psi, C_1, \ldots, C_n, E_1, \ldots, E_p, D'_1, \ldots, D'_m, E'_1, \ldots, E'_p}$$

A copy Y' of the set of innermost existential variables Y is introduced. In the primed clauses, variables from Y are replaced by variables from Y'. For applying universal expansion on variable x, the clauses of the formula are partitioned into four groups: (i) those that contain literal x; (ii) those that contain literal $\bar{x}$; (iii) clauses that contain at least one innermost existential literal (i.e., from Y), but not x nor $\bar{x}$; and (iv) the others. Notice that only the clauses in the third group are duplicated, so one would like to have that group as small as possible. Yet BLE may remove x (or $\bar{x}$) literals from clauses, thereby moving them from the first group (or second group) to the third group. BLA on the other hand will add literals x (or $\bar{x}$), thereby moving clauses from the third group to the first (or second group). Therefore, applying BLA to add x and $\bar{x}$ literals to clauses is a useful pre-step before universal expansion.

4 Evaluation

We extended our state-of-the-art preprocessor bloqqer v35[4] with blocked literal elimination such that BLE complements bloqqer's preprocessing techniques.

[4] `http://fmv.jku.at/bloqqer`

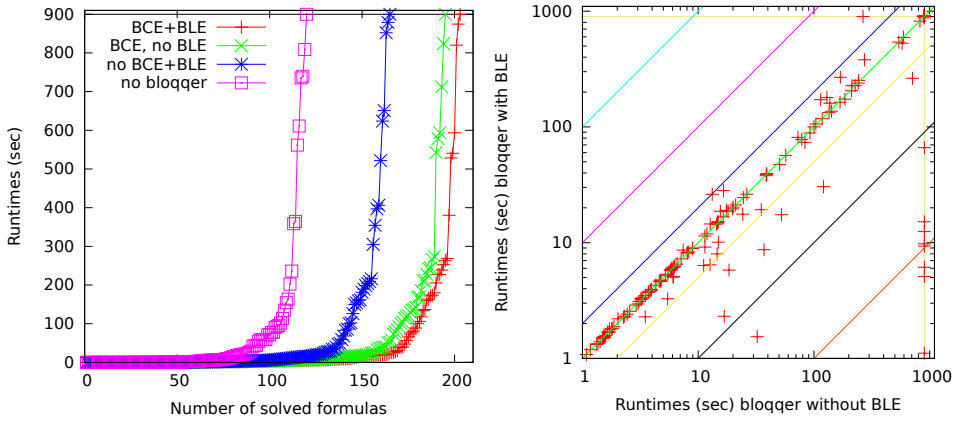

Fig. 1. Runtimes on Formulas from QBFLib Benchmark Set 2014.

We evaluated the impact of BLE on the QBFLib 2014 Benchmarks[5] which were used in the QBF Gallery 2014, the competition of the QBF solving community. This benchmark set consists of 345 formulas stemming from various problem families mainly encoding verification and planning problems. The various problem families differ strongly in their formula structure, in particular in the number of quantifier alternations. Our experiments were run on a cluster of 31 nodes with Intel Q9550 CPUs and 8 GB of memory. The memory limit was never reached. We set an overall timeout of 900 seconds for preprocessing and solving, and we limited the memory consumption to 7 GB. The formulas, which could not be solved directly by bloqqer, were handed over to DepQBF 3.04[6], one of the most successful solvers of the QBF Gallery 2014.

We considered different configurations of bloqqer to evaluate if and how BLE influences the solving runtime. We ran DepQBF alone and with the following configurations of bloqqer: (i) all options enabled, (ii) BLE disabled, and (iii) BCE and BLE disabled. The results of our experiments are summarized in the left diagram of Fig. 1. Combining BLE and BCE leads to the best results, i.e., solving 202 formulas (102 true, 100 false). When BLE is disabled, 194 formulas are solved (98 true, 96 false). A detailed comparison is given by the scatter plot of Fig. 1. Whereas the majority of the formulas remains unaffected by BLE, for some formulas the runtime improves noticeably. Only one formula (test4_quant_squaring2) could be solved with BLE disabled, but not without. The average preprocessing time is 42 seconds without BLE and BCE, and 44 seconds if they are enabled. Table 1 shows statistics on formulas which can only be solved if BLE is turned on. With BLE, bloqqer itself (i.e., without DepQBF) solves 78 formulas (37 true, 41 false), compared to 68 formula (33 true, 35 false) without BLE. The truth value of these formulas is certified by our checking tool [1].

[5] http://qbf.satisfiability.org/gallery/
[6] http://lonsing.github.io/depqbf/

Table 1. Formulas solved exclusively due to BLE. The columns show the number of variables (#vars), clauses (#cl), quantifier alternations (#Q), blocked literals (#bl).

	original formula			preprocessing					solving	
formula	#vars	#cl	#Q	#bl	#vars	#cl	#Q	time	time	val
adder-6-sat	1727	1259	4	1278	2157	5401	2	0.74	0.36	T
C88020_0_0_inp	1046	2644	21	3	1306	3466	15	0.2	874.32	F
cache-coh-2-fixp-5	9604	28198	2	3599	–	–	–	9.32	–	F*
ethernet-fixpoint-3	12514	33884	2	3879	–	–	–	9.76	–	F*
k_branch_n-14	7068	33865	33	389	–	–	–	5.09	–	T*
k_branch_n-20	13821	78949	44	1397	–	–	–	12.45	–	T*
k_branch_p-15	8035	39595	34	239	–	–	–	6.12	–	F*
k_branch_p-21	15161	88627	46	1532	–	–	–	15.12	–	F*
s820_d7_s	24757	26960	3	5365	25115	12869	3	54.7	11.44	T

* solved directly by bloqqer

5 Conclusion

We showed that blocked literal elimination—a special case of a rule in the QRAT proof system—can be applied as preprocessing technique for QBFs. We integrated BLE in our preprocessor bloqqer. Experiments showed the impact of this technique. We further proposed and motivated a technique called blocked literal addition where blocked literals are introduced to the formulas. However, the implementation of BLA is more involved because its application bears the danger of annihilating the effects of other preprocessing techniques. Further, we did not investigate the impact of the asymmetric variant of BLE yet nor the application of the general QRAT rules. Both will be subject to future work.

References

1. Heule, M.J.H., Seidl, M., Biere, A.: A Unified Proof System for QBF Preprocessing. In: IJCAR 2014. Volume 8562 of LNCS., Springer (2014) 91–106
2. Cadoli, M., Schaerf, M., Giovanardi, M., Giovanardi, M.: An algorithm to evaluate quantified boolean formulae and its experimental evaluation. In: Journal of Automated Reasoning, AAAI Press (1999) 262–267
3. Biere, A., Lonsing, F., Seidl, M.: Blocked clause elimination for QBF. In: CADE 2011. Volume 6803 of LNCS., Springer (2011) 101–115
4. Tseitin, G.S.: On the complexity of derivation in propositional calculus. In: Automation of Reasoning 2. Springer (1983) 466–483
5. Ansótegui, C., Gomes, C.P., Selman, B.: The Achilles' Heel of QBF. In: AAAI 2005, AAAI Press / The MIT Press (2005) 275–281
6. Heule, M.J.H., Seidl, M., Biere, A.: Efficient Extraction of Skolem Functions from QRAT Proofs. In: FMCAD 2014, IEEE (2014) 107–114
7. Heule, M.J.H., Järvisalo, M., Biere, A.: Clause elimination procedures for CNF formulas. In: LPAR-17. Volume 6397 of LNCS., Springer (2010) 357–371
8. Heule, M.J.H., Järvisalo, M., Biere, A.: Covered clause elimination. In: LPAR-17-short. Volume 13 of EPiC Series., EasyChair (2013) 41–46
9. Biere, A.: Resolve and expand. In: SAT 2004. Volume 3542 of LNCS. (2004) 59–70