Failed Literal Detection for QBF

Florian Lonsing and Armin Biere*

Institute for Formal Models and Verification
Johannes Kepler University, Linz, Austria
http://fmv.jku.at/

Abstract. Failed literal detection (FL) in SAT is a powerful approach for preprocessing. The basic idea is to assign a variable as assumption. If boolean constraint propagation (BCP) yields an empty clause then the negated assumption is necessary for satisfiability. Whereas FL is common in SAT, it cannot easily be applied to QBF due to universal quantification. We present two approaches for FL to preprocess prenex CNFs. The first one is based on abstraction where certain universal variables are treated as existentially quantified. Second we combine QBF-specific BCP (QBCP) in FL with Q-resolution to validate assignments learnt by FL. Finally we compare these two approaches to a third common approach based on SAT. It turns out that the three approaches are incomparable. Experimental evaluation demonstrates that FL for QBF can improve the performance of search- and elimination-based QBF solvers.

1 Introduction

The logic of quantified boolean formulae (QBF) is an extension of propositional logic (SAT) where variables are existentially or universally quantified. Whereas this often allows for more succinct encodings of problems, it also causes PSPACE-completeness of the decision problem of QBF.

A standard and common input format of QBF solvers is prenex conjunctive normal form (PCNF). The conversion of an arbitrary QBF encoding into PCNF might hide relevant structural properties like variable dependencies. This in turn can influence solver performance negatively. In order to overcome this drawback, several approaches for preprocessing PCNFs have been suggested. Some were ported from SAT to QBF such as binary clause reasoning, equivalence detection and variable elimination by resolution or expansion [2, 5, 6, 11, 21, 25, 26].

This work focuses on the detection of failed literals (FL) in QBF. The idea is to make an assumption, i.e. a trial assignment of a variable, and apply boolean constraint propagation with QBF-specific inference rules (QBCP, see also Section 2). If the empty clause is discovered then the negated assumption is a necessary assignment for satisfiability and can be added to the formula as a learnt unit clause. In contrast to SAT where application of FL is straightforward, complications arise in the context of QBF.

* The 2nd author is financially supported by the Austrian Science Foundation (FWF) NFN Grant S11408-N23 (RiSE).

Example 1. The satisfiable PCNF $\psi := \forall x \exists y.\ (x \vee \neg y) \wedge (\neg x \vee y)$ expresses equivalence of x and y. When assuming y in FL, then the first clause becomes empty due to universal reduction (see Section 2). But adding unit clause $(\neg y)$, which corresponds to the negated assumption, produces an unsatisfiable formula.

The problem pointed out in Example 1 is that y depends on x. That is, y must not be assigned as assumption without considering the value of x as was erroneously done by FL. This might destroy satisfiability of PCNFs due to violations of the quantifier ordering: x is outermost but y was assumed first. Hence that ordering must be respected in preprocessing just as in QBF solving in general.

The objective of our work is to apply *sound* variants of FL to PCNFs for preprocessing, i.e. where preprocessing by FL produces an equivalent formula. In the following we introduce and evaluate three approaches to detect necessary assignments of QBFs in PCNF. First we reconsider a known technique based on SAT solving (Section 3). A SAT solver is used to check if a unit clause is implied by the plain (i.e. ignoring quantifiers) CNF of some PCNF. Such unit clauses can then be added to the PCNF as well.

Second we introduce FL with respect to an abstraction of a PCNF (Section 4). A unit clause learnt by FL on the abstraction can also be added to the original PCNF. An abstraction is obtained separately for each assumption. Variables are treated as existentially quantified if they are smaller in the quantifier ordering than the current assumption.

Third we combine FL on the original PCNF with Q-resolution (Section 5). If an empty clause is discovered in FL then we try to derive the unit clause corresponding to the negated assumption by Q-resolution. The choice of Q-resolution candidates is heuristically guided by the current run of QBCP in FL.

A major contribution of our work is a comparison (Sections 6 and 7). We provide examples pointing out that the three aforementioned approaches are incomparable with respect to effectiveness. There are formulae where particular unit clauses can be learnt with one approach but not with another. This observations also raise questions whether conflict-driven clause learning (CDCL) in QBF solvers could be improved in this respect. Moreover, our abstraction-based approach can be regarded as a polynomial-time[1] alternative to the common SAT-based approach. Hence FL could be applied dynamically in search-based QBF solvers when interleaved with the search. This idea is similar to optimizations carried out in many SAT solvers when backtracking to the topmost decision level, e.g. after restarting or if a unit clause was learnt.

We implemented the three approaches for FL in our novel preprocessor QxBF and evaluated their effectiveness in combination with various search- and elimination-based QBF solvers (Section 7). Our experiments confirm observations regarding incomparability. Although the vast performance improvement is observed for elimination-based solvers, we conjecture that search-based solvers could benefit from dynamic applications of various approaches of FL.

[1] We interpret QBCP as the polynomial-time procedure defined in Section 2.

Related Work. FL for SAT originated in [9] and is an integral part of look-ahead SAT solvers [14]. A comprehensive treatment of FL for preprocessing is given in [1, 18], which also includes inferences of unit clauses based on complementary assumptions [13]. We consider that future work and refrain from discussion in this work.

FL was applied to QBF in [22] but with a special treatment of empty clauses and QBCP lacking universal reduction and pure literal detection. Thus the full propagation power of QBCP was not exploited. In contrast to this, our FL approaches presented in Sections 4 and 5 are an improvement.

A theoretical foundation of QBF preprocessing was given in [26] in terms of QBF models. Additionally binary clause reasoning for QBF was introduced in [25] where QBCP was used for detecting binary clause inferences. We combine QBCP in FL with Q-resolution to find derivations of unit clauses (Section 5).

In [21] a SAT solver was applied to detect necessary assignments of the CNF part of PCNFs. We consider the same approach mainly for reference (Section 3) but also show that it is incomparable to our FL approaches (Section 6).

A SAT solver was integrated into a search-based QBF solver in [24]. It was observed that the two solvers are capable of learning clauses the other one can not learn. We make similar observations and provide examples regarding the detection of unit clauses by SAT solving and by our FL approaches (Section 6).

2 Preliminaries

For a set of propositional variables V, a *literal* is either a variable $x \in V$ or its negation $\neg x$ where $v(x) = x$ and $v(\neg x) = x$ denotes the variable of a literal. A *clause* C_i is a disjunction over literals where $\{x, \neg x\} \not\subseteq C_i$ for all $x \in V$. For clauses C_i, a propositional formula $\phi := C_1 \wedge \ldots \wedge C_n$ is in *conjunctive normal form (CNF)*. For a CNF ϕ and a literal l, the set of *occurrences* of l is $O(l) := \{C \mid C \in \phi, l \in C\}$.

A quantified boolean formula (QBF) $\psi := Q_1 S_1 \ldots Q_n S_n . \phi$ in *prenex conjunctive normal form (PCNF)* consists of a CNF ϕ over a set of variables V and a *quantifier prefix* $Q_1 S_1 \ldots Q_n S_n$. The quantifier prefix is a linearly ordered set of *scopes* S_i forming a partition on V. A scope S_i is *existential* ($Q_i = \exists$) if it is associated with an existential quantifier and *universal* ($Q_i = \forall$) otherwise. For scopes S_i and S_{i+1}, $Q_i \neq Q_{i+1}$ for $1 \leq i < n$. For a literal x with $v(x) \in S_i$, $q(x) := Q_i$ is the *quantifier type* of (the variable of) x. For clause C and $Q \in \{\forall, \exists\}$, $L_Q(C) := \{l \in C \mid q(l) = Q\}$. For literals l, k with $v(l) \in S_i$ and $v(k) \in S_j$, $l \leq k$ if, and only if $i \leq j$ for $1 \leq i, j \leq n$.

Given CNF ϕ, an *assignment* is a mapping $A : V \to \{\mathit{true}, \mathit{false}\}$ from variables in ϕ to truth values. An assignment m is a *CNF-model* of ϕ, written as $m \models \phi$, if every clause in ϕ is satisfied under m.

We introduce QBF semantics based on tree-like models as in [26]. This allows for a simpler definition of necessary assignments (see below) and proofs. In this framework, a necessary assignment is a property of all models of a QBF. This cannot easily be expressed within standard QBF semantics based on recursive

evaluation like e.g. in [7]. Further, relying on tree-like models, our results can naturally be generalized to QBF solving using dependency schemes [23].

Given a PCNF $\psi := Q_1 S_1 \dots Q_n S_n.\, \phi$. An *assignment tree* T is a tree of assignments according to the following restrictions. Every node N in T except the root represents a truth assignment to a variable v in V. A node has at most (exactly) one sibling if it assigns a truth value to an existential (universal) variable. Two siblings altogether denote assignments *true* and *false*. Every path P from the root to a leaf of T corresponds to an assignment A for variables in CNF ϕ. A node N for variable v is an ancestor of another node N' for variable v' in P if and only if $v \leq v'$. That is, assignments along every path P respect the quantifier ordering. An assignment tree m is a *PCNF-model* of ψ, written as $m \models \psi$, if every path P in m is a CNF-model of ϕ.

A CNF is *satisfiable* if it has a CNF-model. Two CNFs ϕ and ϕ' are *model-equivalent*, written as $\phi \equiv_m \phi'$, if and only if for all assignments m, $m \models \phi$ if and only if $m \models \phi'$. Two CNFs ϕ and ϕ' are *satisfiability-equivalent*, written as $\phi \equiv_s \phi'$, if and only if ϕ is satisfiable then ϕ' is satisfiable and vice versa. A transformation of a CNF ϕ into a CNF ϕ' is *sound* if and only if $\phi \equiv_m \phi'$. Satisfiability, model-equivalence, satisfiability-equivalence and soundness with respect to PCNFs are defined accordingly. The following properties are well known.

Proposition 1 ([21, 26]). *Given PCNF $\psi := Q_1 S_1 \dots Q_n S_n.\, \phi$ and CNF ϕ' where $\phi \equiv_m \phi'$. Let $\psi' = Q_1 S_1 \dots Q_n S_n.\, \phi'$. Then $\psi \equiv_m \psi'$.*

Proposition 2 (e.g. [7]). *For CNF ϕ and literal x, $\phi \wedge \{\neg x\}$ is unsatisfiable iff $\phi \equiv_m \phi \wedge \{x\}$.*

Given PCNF ψ and $x_i \in V$. Assignment $x_i \mapsto t$, where $t \in \{\mathit{false}, \mathit{true}\}$, is *necessary* for satisfiability of ψ iff $x_i \mapsto t$ is part of every path in every PCNF-model of ψ.

Given PCNF $\psi := Q_1 S_1 \dots Q_n S_n.\, \phi$, the assignment of literal l with $v(l) \in S_i$ yields the formula $\psi[l] := Q_1 S_1 \dots Q_i S_i' \dots Q_n S_n.\, \phi'$ where $S_i' := S_i \setminus \{v(l)\}$ and clauses $O(l)$ and literals $\neg l$ in $O(\neg l)$ are deleted in ϕ'.

For clause C, *universal reduction* is denoted by $UR(C) := C \setminus \{l_u \in L_\forall(C) \mid \forall l_e \in L_\exists(C), l_e < l_u\}$. A clause $C \in \phi$ where $UR(C) = \{l\}$ is *unit* and $\psi \equiv_s \psi[l]$ [6, 8]. For a QBF $\psi := Q_1 S_1 \dots Q_n S_n.\, \phi$, a literal l where $O(l) \neq \emptyset$ and $O(\neg l) = \emptyset$ is *pure* [8]: if $q(l) = \exists$ then $\psi \equiv_s \psi[l]$, and if $q(l) = \forall$ then $\psi \equiv_s \psi[\neg l]$.

We assume that clauses in a PCNF are fully reduced by UR. For clauses C_1, C_2 with $v \in L_\exists(C_1), \neg v \in L_\exists(C_2)$, the *Q-resolvent* C, written as $(C_1, C_2) \vdash_v C$, is defined as follows [6]: let $C' := (C_1 \cup C_2) \setminus \{v, \neg v\}$. If $\{x, \neg x\} \subseteq C'$ for $x \in V$ then no Q-resolvent exists, otherwise $C := UR(C')$. We write $\psi \vdash^* C$ if clause C can be derived from QBF ψ by Q-resolution. Adding Q-resolvents to a PCNF yields a model-equivalent formula.

Lemma 1 ([26]). *Given PCNF ψ and clause C. Then $\psi \wedge C$[2] $\equiv_m \psi \wedge UR(C)$.*

[2] For PCNF $\psi := Q_1 S_1 \dots Q_n S_n.\, \phi$, let $\psi \wedge C$ denote $Q_1 S_1 \dots Q_n S_n.\, (\phi \wedge C)$.

Proof. Our definition of PCNF-models differs from the one in [26]. There, nodes N assigning existential variables do *not* have a sibling. The proof in [26] also applies to our semantical framework. The subtree rooted at a sibling of such node N can be deleted, and the resulting assignment tree is still a PCNF-model. □

Lemma 2. *Given PCNF ψ. If $\psi \vdash^* C$ then $\psi \equiv_m \psi \wedge C$.*

Proof. Q-resolution can be regarded as a combination of resolution for propositional logic and UR. Adding propositional resolvents yields a model-equivalent formula [7]. The claim then follows from Lemma 1 and Proposition 1. □

Given PCNF ψ and a literal x, $\psi' := QBCP(\psi, x)$ denotes a formula obtained from $\psi[x]$ by applying UR, unit clause and pure literal rule. Literal x is called an *assumption*. For clause C, $C \in QBCP(\psi, x)$ if $\psi' = QBCP(\psi, x)$ and $C \in \psi'$. We write $\emptyset \in QBCP(\psi, x)$ if the empty clause can be obtained.

3 SAT-based FL

First we review a well-known technique for inferring unit clauses from the CNF part of a PCNF based on propositional satisfiability testing. This has already been applied to QBF [21]. We include it here as a reference for our approaches introduced in Sections 4 and 5. We show that these approaches have different effectiveness in Sections 6 and 7.

A general approach for preprocessing PCNFs can be obtained from QBF semantics and model-equivalence of CNFs. When combining Propositions 1 and 2, a SAT solver can be used to check if a unit clause is implied by the CNF part of a PCNF for QBF preprocessing [21].

Whereas propositional satisfiability testing allows to exploit Proposition 2 to full extent, a subset of all necessary assignments of a CNF can be identified by failed literal detection (FL) for SAT [1, 9, 18]. FL is a common approach in SAT which can be carried out in *polynomial time* based on BCP[3] as follows.

Proposition 3. *Given a CNF ϕ and literal x. If $\emptyset \in BCP(\phi, x)$ then $\phi \equiv_m \phi \wedge \{\neg x\}$. Literal x is called a* failed literal.

If x is a failed literal then every CNF-model of ϕ must set x to *false*.

Definition 1. *Given a PCNF ψ and a literal x. If $\psi \equiv_m \psi \wedge \{x\}$ then $\neg x$ is called a failed literal. Assigning x to* true *is a necessary assignment for ψ.*

Example 1 pointed out that Proposition 3 is not directly applicable to QBF because satisfiability might be destroyed. The reason is that assumptions are made out of quantifier ordering. This can be regarded as modifying the quantifier prefix. Carrying out $QBCP(\psi, y)$ in Example 1 is similar to shifting y to the leftmost position in the prefix, yielding $\exists y \forall x$ prior to assigning it. This might allow applications of UR impossible based on the original prefix.

[3] "BCP" denotes QBCP without pure literal rule and universal reduction.

In the following we introduce two approaches to detect necessary assignments by FL for QBF similar to Proposition 3, i.e. based on QBCP. The goal is to have a polynomial-time alternative to satisfiability testing as in Propositions 1 and 2. However, as shown in Section 6, it turns out that the three approaches are actually incomparable. In the following section we apply FL with QBCP to an abstraction of the original PCNF.

4 Abstraction-Based FL

In an early approach of FL for QBF presented in [22], QBCP included neither UR nor pure literal rule. This results in limited propagation power as both UR and universal pure literals possibly trigger additional unit literals. We conjecture that such restrictions of QBCP combined with a special treatment of empty clauses like in [22] are sufficient to ensure soundness of FL.

In our approaches of FL for QBF we allow the *full* set of rules in QBCP as defined in Section 2. In the following, we analyze special cases where FL based on QBCP is sound. These results will then be used to prove soundness of FL when QBCP is applied to an abstraction of the original PCNF.

Lemma 3. *Given* $\psi := Q_1 S_1 \dots Q_n S_n.\ \phi$ *and literal* x *where* $v(x) \in S_1$*. If* $\emptyset \in QBCP(\psi, x)$ *then* $\psi \equiv_m \psi \wedge \{\neg x\}$*.*

Proof. Since $x \in S_1$ we assume that x is assigned first on every path in every model of ψ. By definition of *QBCP*, $\psi[x] \equiv_s \emptyset$, i.e. ψ does not have a model where x is assigned *true* on any path. Therefore, x must be assigned *false* on all paths in all models m (if any) of ψ. Hence $\psi \equiv_m \psi \wedge \{\neg x\}$. □

Due to Lemma 3, FL with QBCP and assumptions from the leftmost scope is sound. For universal assumptions, this can be generalized to arbitrary scopes.

Lemma 4. *Given PCNF* $\psi := Q_1 S_1 \dots Q_n S_n.\ \phi$ *and literal* x *where* $v(x) \in S_i$ *and* $Q_i = \forall$*. If* $\emptyset \in QBCP(\psi, x)$ *then* ψ *is unsatisfiable.*

Proof. The PCNF where QBCP is carried out can be regarded as $\psi' := \forall x Q_1 S_1 \dots Q_{i-1} S_{i-1} \forall (S_i \setminus \{x\}) \dots Q_n S_n.\ \phi$. Note the change in the prefix by moving $\forall x$ to the front. If $\emptyset \in QBCP(\psi', x)$ then ψ' is unsatisfiable due to Lemma 3. Then ψ is unsatisfiable as well because if a PCNF with prefix pattern $\forall x \dots \exists y$ (pattern of ψ') is unsatisfiable then also with $\exists y \dots \forall x$ (pattern of ψ). □

We introduce an approach where FL is carried out on an abstraction of the original PCNF. The abstraction affects the quantifier prefix.

Definition 2 (Quantifier Abstraction). *For* $\psi := Q_1 S_1 \dots Q_{i-1} S_{i-1} Q_i S_i \dots \dots Q_n S_n.\ \phi$*, the quantifier abstraction of* ψ *with respect to* S_i *is* $Abs(\psi, i) := \exists (S_1 \cup \dots \cup S_{i-1}) Q_i S_i \dots Q_n S_n.\ \phi$*.*

By Definition 2 variables from scopes smaller than S_i are treated as existentially quantified. This gives an overapproximation of ψ with respect to PCNF-models, following from the definition in Section 2.

Corollary 1. *For PCNF ψ and PCNF-model m: if $m \models \psi$ then $m \models Abs(\psi, i)$.*

In practice, FL with assumptions from S_i and QBCP is carried out on $Abs(\psi, i)$ instead of ψ. This is called *abstraction-based FL*. Unit clauses learnt by FL on $Abs(\psi, i)$ can then be added to the original PCNF ψ (see Example 2 below). First we prove soundness of abstraction-based FL with respect to $Abs(\psi, i)$.

Lemma 5. *Given PCNF $\psi := Q_1 S_1 \dots Q_n S_n.\ \phi$ and literal x where $v(x) \in S_i$. If $\emptyset \in QBCP(Abs(\psi, i), x)$ then $Abs(\psi, i) \equiv_m Abs(\psi, i) \wedge \{\neg x\}$.*

Proof. We distinguish cases by the quantifier type of x.

If $q(x) = Q_i = \exists$ then $Abs(\psi, i) = \exists(S_1 \cup \dots \cup S_{i-1} \cup S_i) \dots Q_n S_n.\ \phi$. Due to Lemma 3, $Abs(\psi, i) \equiv_m Abs(\psi, i) \wedge \{\neg x\}$.

If $q(x) = Q_i = \forall$ then $Abs(\psi, i) = \exists(S_1 \cup \dots \cup S_{i-1}) \forall S_i \dots Q_n S_n.\ \phi$. If $\emptyset \in QBCP(Abs(\psi, i), x)$ then $Abs(\psi, i)$ is unsatisfiable due to Lemma 4, and so is $Abs(\psi, i) \wedge \{\neg x\}$. □

If QBCP on $Abs(\psi, i)$ with assumption x where $v(x) \in S_i$ yields the empty clause then x is a failed literal and clause $\{\neg x\}$ can be added to $Abs(\psi, i)$. Due to Corollary 1 and the second case of the proof of Lemma 5, $Abs(\psi, i)$ preserves an intuitive property of universal quantification in ψ: if one branch of a universal variable cannot lead to a solution then ψ is unsatisfiable. Relying on Corollary 1 and Lemma 5, we prove that failed literals obtained on $Abs(\psi, i)$ are also sound with respect to original PCNF ψ.

Theorem 1. *Given PCNF $\psi := Q_1 S_1 \dots Q_n S_n.\ \phi$ and literal x where $v(x) \in S_i$. If $\emptyset \in QBCP(Abs(\psi, i), x)$ then $\psi \equiv_m \psi \wedge \{\neg x\}$.*

Proof. We show both directions of $\psi \equiv_m \psi \wedge \{\neg x\}$. If $m \models \psi \wedge \{\neg x\}$ then also $m \models \psi$ since ψ is less constrained than $\psi \wedge \{\neg x\}$.

For the other direction assume $\emptyset \in QBCP(Abs(\psi, i), x)$. Let m be a PCNF-model of ψ, i.e. $m \models \psi$ (if no such m exists then the claim follows immediately). By Corollary 1, also $m \models Abs(\psi, i)$. By Lemma 5, also $m \models Abs(\psi, i) \wedge \{\neg x\}$. Clause $\{\neg x\}$ is satisfied under m. Therefore, $m \models \psi \wedge \{\neg x\}$. □

Example 2. Given PCNF[4] $\psi := \forall a_1 \exists e_2, e_3 \forall a_4 \exists e_5.\ \{a_1, e_2\}, \{\neg a_1, e_3\}, \{e_3, \neg e_5\}, \{a_1, e_2, \neg e_3\}, \{\neg e_2, a_4, e_5\}$. We have $\emptyset \in QBCP(Abs(\psi, 2), \neg e_3)$ since the assumption will make clauses $\{\neg a_1, e_3\}$ and $\{e_3, \neg e_5\}$ unit because a_1 is treated as existential. Clause $\{\neg e_2, a_4, e_5\}$ becomes unit due to UR. Finally clause $\{a_1, e_2\}$ is empty and unit clause $\{e_3\}$ is learnt.

Although abstraction-based FL uses all QBCP rules in contrast to [22], in general this does not result in full propagation power of QBCP on $Abs(\psi, i)$. Depending on i, i.e. the quantifier level of the current assumption, $Abs(\psi, i)$ typically has fewer universal variables than ψ. This influences detection of unit literals by UR and pure literal rule. Hence we expect more powerful QBCP for assumptions from S_1 than from S_n. For assumptions from different scopes, our approach is more dynamic than restricting the set of QBCP rules in advance as in [22].

[4] Unless stated otherwise, all PCNFs in examples provided in the paper are satisfiable. For brevity, pure literal detection is ignored and CNFs (clauses) are presented as sets of clauses (literals).

5 QBCP-Guided Q-Resolution

Abstraction-based FL from the previous section allows to apply QBF-specific inference rules like UR and universal pure literals to a larger extent than previous approaches where QBCP rules were restricted. However, it still lacks full propagation power as could be obtained on the original PCNF.

In this section we present an approach for FL which operates on the original PCNF, thus taking full benefits from QBF-specific QBCP rules. Because this might in general destroy satisfiability as pointed out in Example 1, we apply Q-resolution to validate failed literals detected by QBCP. This approach is inspired by CDCL in search-based QBF solvers [12, 29].

An assignment A is generated by making an assumption x and carrying out QBCP with the full set of inference rules on the original PCNF. If $\emptyset \in QBCP(\psi, x)$ then candidate clauses for Q-resolution are selected from ψ entirely with respect to assignment A, i.e. its implication graph like in CDCL for SAT solving [27]. If the unit clause $\{\neg x\}$ corresponding to the negated assumption can be deduced in that way then x is a valid failed literal. This is called *QBCP-guided Q-resolution*. Soundness follows right from Lemma 2. The following example shows a nontrivial application.

Example 3. Given PCNF $\psi := \exists e_1,e_2 \forall a_3 \exists e_4,e_5.\ \{a_3, e_5\}, \{\neg e_2, e_4\}, \{\neg e_1, e_4\}, \{e_1, e_2, \neg e_5\}$. With assumption $\neg e_4$ we get $\emptyset \in QBCP(\psi, \neg e_4)$ since $\{\neg e_1\}$, $\{\neg e_2\}$ and $\{\neg e_5\}$ become unit. Finally $\{a_3, e_5\}$ is empty by UR. The negated assumption $\{e_4\}$ is then derived by resolving clauses in reverse-chronological order as they were affected by assignments: $(\{a_3, e_5\}, \{e_1, e_2, \neg e_5\}) \vdash \{e_1, e_2\}$, $(\{e_1, e_2\}, \{\neg e_2, e_4\}) \vdash \{e_1, e_4\}$, $(\{e_1, e_4\}, \{\neg e_1, e_4\}) \vdash \{e_4\}$.

Note that selecting Q-resolution candidates based on the current assignment generated by QBCP is only a heuristic. That is, even when it fails the negated assumption can possibly be deduced by selecting arbitrary clauses for Q-resolution.

Example 4. Given PCNF $\psi := \forall a_1 \exists e_2 \forall a_3 \exists e_4.\ \{a_1, e_2\}, \{e_2, a_3, e_4\}, \{e_2, a_3, \neg e_4\}$. We have $\emptyset \in QBCP(\psi, \neg e_2)$ immediately by UR in the first clause, but e_2 obviously cannot be produced by Q-resolution from that single clause affected by the assignment. However, we have $(\{e_2, a_3, e_4\}, \{e_2, a_3, \neg e_4\}) \vdash^* \{e_2\}$.

Due to Lemma 4 unsatisfiability can be concluded *without* Q-resolution if $\emptyset \in QBCP(\psi, x)$ for $x \in S_i$ where $Q_i = \forall$. This property is similar to abstraction-based FL but we expect more QBF-specific inferences in QBCP with this approach. Further the empty clause might be derived when validating a failed literal by Q-resolution. In this case, unsatisfiability follows immediately.

Example 5. Given PCNF $\psi := \forall a_1 \exists e_2,e_3,e_4,e_5.\ \{e_2, \neg e_5\}, \{\neg e_2, e_5\}, \{a_1, e_2\}, \{\neg a_1, e_3\}, \{\neg e_3, e_4\}, \{\neg e_3, \neg e_4\}$. We have $\emptyset \in QBCP(\psi, e_2)$ because $\{a_1, e_2\}$ is satisfied which makes $\{a_1\}$ pure and $\{e_3\}$ unit in $\{\neg a_1, e_3\}$. Finally $\{e_4\}$ is unit and $\{\neg e_3, \neg e_4\}$ becomes empty. Q-resolution as in Example 3 produces the empty clause: $(\{\neg e_3, \neg e_4\}, \{\neg e_3, e_4\}) \vdash \{\neg e_3\}$, $(\{\neg e_3\}, \{\neg a_1, e_3\}) \vdash \emptyset$. Note that $\emptyset \notin QBCP(Abs(\psi, 2), e_2)$ because $\{e_3\}$ does not become unit since the universal pure literal rule is not applicable as before.

6 Comparing FL Approaches

We presented one approach for FL based on SAT testing and two based on QBCP to find necessary assignments in PCNFs: SAT-based FL according to Proposition 2 (Section 3), abstraction-based FL (Section 4) and QBCP-guided Q-resolution (Section 5). As argued above, the last two benefit from QBF-specific inferences in QBCP increasingly in that order. This is due to larger numbers of universal variables in the formulae where FL is applied to.

In the following we compare the three approaches according to their effectiveness. We name examples which demonstrate that they are incomparable: one approach is able to detect necessary assignment the other one cannot.

Proposition 4. *Abstraction-based FL and QBCP-guided Q-resolution are incomparable with respect to detecting necessary assignments.*

Example 6. For the PCNF from Example 2 unit clause $\{e_3\}$ *cannot* be derived by Q-resolution, i.e. neither by QBCP-guided Q-resolution nor when allowing arbitrary candidate clauses. This is in contrast to abstraction-based FL. Note that assigning e_3 to *true* is necessary as can be seen from a semantical evaluation. Every path in every PCNF-model has to assign e_3 to *true*.

Example 7. For the PCNF from Example 3 we have $\emptyset \notin QBCP(Abs(\psi, 3), \neg e_4)$ which is in contrast to QBCP-guided Q-resolution. Due to abstraction UR is not applicable to make $\{a_3, e_5\}$ empty. Similarly, clause $\{e_4\}$ cannot be inferred when UR is excluded from QBCP rules as in [22].

Note that Proposition 4 severely affects QBF solvers relying on Q-resolution for conflict-driven clause learning (CDCL). For certain PCNFs *no* such solver will ever be able to deduce all necessary assignments.

The following result was also obtained in the more general context of clause learning when SAT solving was combined with search-based QBF solving [24].

Proposition 5. *SAT-based FL and QBCP-guided Q-resolution are incomparable with respect to detecting necessary assignments.*

Example 8. Given PCNF $\psi := \exists e_1 \forall a_2 \exists e_3.\, \{e_1, a_2, e_3\}, \{e_1, a_2, \neg e_3\}, \{e_1, \neg a_2, e_3\}, \{e_1, \neg a_2, \neg e_3\}$. A SAT solver will find out that the CNF with assumption $\neg e_1$ is unsatisfiable, hence $\{e_1\}$ can be learnt. This is possible neither by QBCP-guided Q-resolution nor by abstraction-based FL.

Example 9. For the PCNF from Example 3, SAT-based FL cannot learn $\{e_4\}$ because the CNF has a model with assumption $\neg e_4$.

Proposition 6. *SAT-based FL and abstraction-based FL are incomparable with respect to detecting necessary assignments.*

Example 10. For the PCNF from Example 2, SAT-based FL cannot detect $\{e_3\}$ because the CNF has a model with assumption $\neg e_3$.

Example 11. See Example 8.

7 Experiments

We implemented the three approaches of FL for QBF presented in Sections 3 to 5 in our novel preprocessor QxBF[5]. The idea is to profit from all approaches based on the observations from the previous section. The tool operates in rounds with three phases. First, QBCP with the full set of QBF-specific rules is carried out on the *original* formula, including any unit clauses learnt in earlier rounds. The second phase consists of either abstraction-based FL or QBCP-guided Q-resolution. Finally, SAT-based FL is applied in the third phase because it turned out to be most effective in practice (see below). Rounds are run in cyclic fashion until completion unless a time limit is reached.

The SAT solver PicoSAT [3] is used for SAT-based FL by Proposition 2. This allows for incremental SAT solving and optimizations based on CNF-models to reduce the number of SAT solver calls like in [21]. Additionally, unit clauses learnt by PicoSAT are propagated using QBF-specific QBCP rules within QxBF.

Table 1 compares the impact of different FL approaches on the performance of QBF solvers based on search (DepQBF [17] and QuBE7.1 [10]) and variable elimination (Quantor [2], squolem [15] and Nenofex [16]) using all benchmarks from QBFEVAL'10 [20]. For QuBE7.1 internal preprocessing was disabled (QuBE7.1-np). We used latest publicly available versions of solvers except internal versions of Nenofex and DepQBF, *all* without proprietary preprocessing[6]. Results using sQueezeBF [11], a state-of-the-art QBF preprocessor, are reported for reference. We cannot expect FL to be competitive with sQueezeBF as the latter applies a larger pool of inference rules (details are given below).

We combined abstraction-based FL and QBCP-guided Q-resolution with SAT-based FL (lines "ABS+SAT" and "QRES+SAT") in rounds and phases as described above. At most 40 seconds were assigned to each approach, totalling a maximum of 80 seconds for entire preprocessing. Additionally, heuristic limits were imposed on numbers of propagations in QBCP and decisions in PicoSAT. Performance of elimination-based solvers increases considerably both in terms of solved formulae and run time (lower part of table). Results are less impressive for search-based solvers. Further, they only differ slightly with respect to individual FL approaches when applied to DepQBF (middle part of table), with a limit of 80 seconds each. We combined only "ABS+SAT" with elimination-based solvers as the performance with "QRES+SAT" is likely similar. We did not apply "SAT" alone due to incomparability observed in Section 6. DepQBF performs best with SAT-based FL but also preprocessing times are larger. We argue that tuning the run time of abstraction-based FL and QBCP-guided Q-resolution while maintaining effectiveness is easier in practice. Run time of QBCP is close to linear with respect to formula size in contrast to SAT solving time in SAT-based FL.

Our FL approaches are not competitive compared to sQueezeBF. In the first line of Table 1, we allowed 900 seconds altogether for the combination

[5] Project web page: `http://fmv.jku.at/qxbf/`

[6] Setup: Ubuntu Linux 9.04, Intel®Q9550 2.83 GHz with 900 seconds / 3GB total time and memory limit. Exceeding the memory limit is counted as a time out.

Table 1. Solver performance with(out) time-limited failed literal preprocessing. Times are average total run times *including* preprocessing and time outs, with average preprocessing times in parentheses. The leftmost column indicates FL approaches: no preprocessing ("None"), SAT-based FL ("SAT"), abstraction-based FL ("ABS") and QBCP-guided Q-resolution ("QRES").

QBFEVAL'10: 568 formulae					
Preprocessing	*Solver*	*Solved*	*Time (Preproc.)*	*SAT*	*UNSAT*
sQueezeBF	DepQBF	435	233.28 (36.94)	201	234
sQueezeBF+(ABS+SAT)		434	239.84 (42.79)	201	233
SAT	DepQBF	379	322.31 (7.17)	167	212
QRES+SAT		378	322.83 (6.22)	167	211
ABS+SAT		378	323.19 (7.21)	167	211
ABS		375	327.64 (3.33)	168	207
QRES		374	327.63 (1.83)	167	207
None		372	334.60 (—)	166	206
ABS+SAT	QuBE7.1-np	320	432.22 (7.21)	143	177
None		318	434.69 (—)	143	175
ABS+SAT	Quantor	229	553.65 (7.21)	112	117
	Nenofex	224	553.37 (7.21)	104	120
None		211	573.65 (—)	103	108
	Quantor	203	590.15 (—)	99	104
ABS+SAT	squolem	154	658.28 (7.21)	63	91
None		124	708.80 (—)	53	71

of sQueezeBF and DepQBF because the former does not support setting resource limits. In this experiment, sQueezeBF alone solved 39 formulae and timed out on 15. Overall performance decreases slightly (second line) if "ABS+SAT" with 80 seconds time limit as before is applied additionally to formulae which were simplified but not solved by sQueezeBF. However, now 64 formulae were solved solely by preprocessing. This indicates that "ABS+SAT" is incomparable to sQueezeBF at least from a practical perspective. From the 514 formulae simplified but not solved by sQueezeBF, 489 were not solved by "ABS+SAT" alone. Among them, still 147 assignments were fixed on average (median 20) by "ABS+SAT". Further, the total number of remaining (i.e. neither eliminated nor assigned) variables in all 489 formulae was reduced by 2% due to "ABS+SAT".

Table 2 compares the effectiveness of the three FL approaches in more detail. We considered formulae from QBFEVAL'10 where *all* approaches ran *until completion* within 900 seconds but, differently from Table 1, *neither* with propagation *nor* decision limits. In general FL fixes substantially more assignments than QBCP on the original formula (column "None"). Abstraction-based and SAT-based FL are best according to average and median numbers of fixed assignments, but the latter is more costly in terms of run time. The large difference between average and median values is due to few benchmarks from (blackbox) model checking (instances "`biu*`" and "`*BMC*`", see also below) where SAT-based FL fixed fewer assignments. Further, QBCP-guided Q-resolution performs fewer propagations per assumptions in QBCP than abstraction-based FL. The reason is that the former typically detects spurious empty clauses earlier due to universal reduction. This also results in smaller run times.

Table 2. Average and median run times, fixed assignments, and propagations per assumption for FL approaches. "None" is QBCP on original formula only.

QBFEVAL'10: 524 formulae completed by all				
Preprocessing	None	ABS	QRES	SAT
Avg. Fixed	607.17	730.31	724.10	715.77
Med. Fixed	103.5	137.00	135.00	181.50
Avg. Time	0.02	3.19	0.76	10.80
Med. Time	0.00	0.16	0.02	0.20
Avg. Props/As	—	118.80	51.08	—
Med. Props/As	—	40.01	6.68	—

Motivated from incomparability observed in Section 6, we compared the sets of assignments that were fixed by different FL approaches in Table 3. We considered all three pairwise combinations. Like in Table 2, we focused on formulae where *both* FL approaches of a pair ran *until completion* within 900 seconds. For each pairwise combination, only formulae where sets of fixed assignments (FA) were different were taken into account (first line). We then separated formulae by larger numbers of unique FAs (second line), i.e. FAs detected by one approach but not by the other. For example in section "ABS vs. QRES", on 121 formulae abstraction-based FL found more unique FAs than QBCP-guided Q-resolution. Equal numbers of unique FAs do not show up in that statistics. The third, fourth and fifth line report total, average and median numbers of unique FAs over formulae with different FAs. For example in section "ABS vs. QRES", abstraction-based FL detected 3752 (average 28.86, median 1) unique FAs compared to 58 (average 0.44, median 0) by QBCP-guided Q-resolution. The last two lines show average and median differences between unique FAs detected by left and right approaches in each section. Larger values indicate that the left approach is better than the right one. For example in section "ABS vs. QRES", for each of the 130 considered formulae we subtracted the number of unique FAs detected by abstraction-based FL from the one of QBCP-guided Q-resolution. On average abstraction-based FL detected 28.41 more unique FAs than QBCP-guided Q-resolution whereas the median is 1.

In general average values suggest that abstraction-based FL is better than QBCP-guided Q-resolution and SAT-based FL (sections "ABS vs. QRES" and "ABS vs. SAT") and that QBCP-guided Q-resolution is better than SAT-based FL (section "QRES vs. SAT"). But median values indicate the opposite tendency. As in Table 2, few benchmarks account for skew statistics in Table 3. For example in section "ABS vs. QRES", abstraction-based FL found 1603 unique FAs on instance `lognBWLARGEB1-shuffled` compared to 0 by QBCP-guided Q-resolution. In section "ABS vs. SAT", abstraction-based FL found between 1000 and 7000 unique FAs on some "`biu*`" instances from bounded model checking compared to 0 by SAT-based FL. In contrast to this, SAT-based FL found 2668 on instance `c3_BMC_p1_k256-shuffled` compared to 0 by abstraction-based FL. Similar observations can be made for section "QRES vs. SAT", hence Table 3 confirms incomparability results from Section 6.

Table 3. Pairwise comparison of FL approaches (complete runs as in Table 2).

QBFEVAL'10: formulae with different fixed assignments (FAs)						
	ABS vs. QRES		ABS vs. SAT		QRES vs. SAT	
Formulae with Diff. FAs	130		183		220	
Formulae wrt. Unique FAs	121	9	57	126	36	180
Total Unique FAs	3752	58	24268	16648	24237	19874
Avg. Unique FAs	28.86	0.44	132.61	90.97	110.16	90.33
Med. Unique FAs	1	0	0	13	0	5
Avg. Diff. in Unique FAs	28.41		41.63		19.83	
Med. Diff. in Unique FAs	1		-14		-4.5	

8 Conclusion

We studied failed literal detection (FL) for QBF to infer necessary assignments. Whereas a common approach based on SAT solving turned out to be effective, it suffers from exponential run time and requires careful tuning in practice. We presented two alternatives based on abstraction and Q-resolution which rely on QBCP. The three approaches are incomparable: there are QBFs where a necessary assignment can be detected by one approach but not by the other. Experiments with our implementation in QxBF confirmed that observations. Moreover, abstraction-based FL is a polynomial-time alternative to SAT-based FL. This enables efficient dynamic applications in search-based QBF solvers. Incomparability suggests that FL could benefit from combinations of all three approaches in portfolio-style preprocessors. Combinations of FL with other preprocessing techniques for QBF are future work. Further, it is unclear if clause learning in QBF solvers could be improved. The common implementations based on Q-resolution are not optimal due to incomparability and results from [24].

We want to thank Aina Niemetz and Mathias Preiner for implementing parts of QxBF, Martina Seidl for discussions, and the reviewers for valuable comments.

References

1. D. Le Berre. Exploiting the Real Power of Unit Propagation Lookahead. *Electronic Notes in Discrete Mathematics*, 9:59–80, 2001.
2. A. Biere. Resolve and Expand. In H. H. Hoos and D. G. Mitchell, editors, *SAT (Selected Papers)*, volume 3542 of *LNCS*, pages 59–70. Springer, 2004.
3. A. Biere. PicoSAT Essentials. *JSAT*, 4(2-4):75–97, 2008.
4. A. Biere, M. Heule, H. van Maaren, and T. Walsh, editors. *Handbook of Satisfiability*, volume 185 of *Frontiers in AI and Applications*. IOS Press, 2009.
5. U. Bubeck and H. Kleine Büning. Bounded Universal Expansion for Preprocessing QBF. In Marques-Silva and Sakallah [19], pages 244–257.
6. H. Kleine Büning, M. Karpinski, and A. Flögel. Resolution for Quantified Boolean Formulas. *Inf. Comput.*, 117(1):12–18, 1995.
7. H. Kleine Büning and T. Lettmann. *Propositional Logic: Deduction and Algorithms*. Cambridge University Press, New York, NY, USA, 1999.
8. M. Cadoli, M. Schaerf, A. Giovanardi, and M. Giovanardi. An Algorithm to Evaluate Quantified Boolean Formulae and Its Experimental Evaluation. *J. Autom. Reasoning*, 28(2):101–142, 2002.

9. Jon W. Freeman. *Improvements To Propositional Satisfiability Search Algorithms*. PhD thesis, University of Pennsylvania, 1995.
10. E. Giunchiglia, P. Marin, and M. Narizzano. QuBE7.0 (System Description). *JSAT*, 7:83–88, 2010.
11. E. Giunchiglia, P. Marin, and M. Narizzano. sQueezeBF: An Effective Preprocessor for QBFs Based on Equivalence Reasoning. In Strichman and Szeider [28].
12. E. Giunchiglia, M. Narizzano, and A. Tacchella. Clause/Term Resolution and Learning in the Evaluation of Quantified Boolean Formulas. *J. Artif. Intell. Res. (JAIR)*, 26:371–416, 2006.
13. J. F. Groote and J. P. Warners. The Propositional Formula Checker HeerHugo. *J. Autom. Reasoning*, 24(1/2):101–125, 2000.
14. M. Heule and H. van Maaren. Look-Ahead Based SAT Solvers. In Biere et al. [4].
15. T. Jussila, A. Biere, C. Sinz, D. Kröning, and C. M. Wintersteiger. A First Step Towards a Unified Proof Checker for QBF. In Marques-Silva and Sakallah [19].
16. F. Lonsing and A. Biere. Nenofex: Expanding NNF for QBF Solving. In H. Kleine Büning and X. Zhao, editors, *SAT*, volume 4996 of *LNCS*. Springer, 2008.
17. F. Lonsing and A. Biere. DepQBF: A Dependency-Aware QBF Solver (System Description). *JSAT*, 7:71–76, 2010.
18. I. Lynce and J. P. Marques Silva. Probing-Based Preprocessing Techniques for Propositional Satisfiability. In *ICTAI*, pages 105–. IEEE Computer Society, 2003.
19. J. Marques-Silva and K. A. Sakallah, editors. *Proceedings SAT'07*, volume 4501 of *LNCS*. Springer, 2007.
20. C. Peschiera, L. Pulina, A. Tacchella, U. Bubeck, O. Kullmann, and I. Lynce. The Seventh QBF Solvers Evaluation (QBFEVAL'10). In Strichman and Szeider [28].
21. F. Pigorsch and C. Scholl. An AIG-Based QBF-Solver Using SAT for Preprocessing. In S. S. Sapatnekar, editor, *DAC*, pages 170–175. ACM, 2010.
22. J. Rintanen. Improvements to the Evaluation of Quantified Boolean Formulae. In T. Dean, editor, *IJCAI*, pages 1192–1197. Morgan Kaufmann, 1999.
23. M. Samer. Variable Dependencies of Quantified CSPs. In I. Cervesato, H. Veith, and A. Voronkov, editors, *LPAR*, volume 5330 of *LNCS*. Springer, 2008.
24. H. Samulowitz and F. Bacchus. Using SAT in QBF. In P. van Beek, editor, *CP*, volume 3709 of *LNCS*, pages 578–592. Springer, 2005.
25. H. Samulowitz and F. Bacchus. Binary Clause Reasoning in QBF. In A. Biere and C. P. Gomes, editors, *SAT*, volume 4121 of *LNCS*, pages 353–367. Springer, 2006.
26. H. Samulowitz, J. Davies, and F. Bacchus. Preprocessing QBF. In F. Benhamou, editor, *CP*, volume 4204 of *LNCS*, pages 514–529. Springer, 2006.
27. J. P. Marques Silva, I. Lynce, and S. Malik. Conflict-Driven Clause Learning SAT Solvers. In Biere et al. [4], pages 131–153.
28. O. Strichman and S. Szeider, editors. *Theory and Applications of Satisfiability Testing - SAT 2010, 13th International Conference, SAT 2010, Edinburgh, UK, July 11-14, 2010. Proceedings*, volume 6175 of *LNCS*. Springer, 2010.
29. L. Zhang and S. Malik. Towards a Symmetric Treatment of Satisfaction and Conflicts in Quantified Boolean Formula Evaluation. In P. Van Hentenryck, editor, *CP*, volume 2470 of *LNCS*, pages 200–215. Springer, 2002.