Handbook of Satisfiability
Armin Biere, Marijn Heule, Hans van Maaren and Toby Walsh
IOS Press, 2008

Chapter 2

Quantified Boolean Formulas

Olaf Beyersdorff, Mikoláš Janota, Florian Lonsing, Martina Seidl

2.1. Introduction

Motivated by the success achieved in practical SAT solving, *quantified Boolean formulas* (QBFs), the extension of propositional formulas with existential and universal quantifiers over the propositional variables, have become subject of intensive research (cf. Chapter **??** and Chapter **??**). Over the last two decades, many different solving approaches have been presented forming a wide, heterogeneous landscape of solving paradigms. While in SAT solving conflict-driven clause learning (CDCL) is clearly the predominant solving paradigm (cf. Chapter **??**), the picture for QBFs is less obvious. Two paradigms together with preprocessing have particularly impacted the progress in QBF research over the last years: search-based QBF solving and expansion-based solving.

From empirical evaluations [LSVG16, JJK$^+$14, PS19], it has become evident that search-based solvers (which have variants of Q-resolution as underlying proof systems) and expansion-based solvers (which are based on the $\forall$Exp+Res proof system) often show orthogonal behavior. This observation was concisely captured by the means of proof theory. In particular, separation and simulation results between the different variants of Q-resolution and $\forall$Exp+Res give a detailed picture of the relations between the different proof systems. For such proof-theoretical characterizations, however, it turned out that propositional techniques are insufficient for QBFs. To this end, novel separation techniques—for example, based on strategy extraction—have been introduced that provide an elegant tool for comparing proof systems.

In practical QBF solving proofs play another important role. Proofs that are polynomially checkable can be used to certify the correctness of the result of a QBF solver. In addition, they can be used to extract strategies in terms of so-called Skolem and Herbrand functions. For example, in the case of a synthesis problem [BKS14], such functions represent the required implementation or, in the case of planning, such functions represent the plan [EKLP17].

To obtain certification and strategy extraction also if preprocessing is used, the QRAT proof system has been introduced. It generalizes the RAT proof system

for SAT which is currently the formal foundation of the certificates produced by recent SAT solvers (see Chapter **??**).

Currently, we see a strong interaction between proof theory and solving. In this chapter, we survey this connection between theory and practice. In Section 2.3 QCDCL and Q-resolution are presented and in Section 2.4 we show how expansion-based solving and ∀Exp+Res are related. In Section 2.5 we discuss proofs for recent preprocessing techniques. An overview of strategy extraction from proofs is presented in Section 2.6. Section 2.7 compares different proof systems used in solving and introduces other powerful proof systems that are currently not implemented in any solver.

2.2. Preliminaries

A *literal* l is a Boolean variable x or its negation $\bar{x}$ and $\mathrm{var}(l) = x$. By $\bar{l}$, we denote the complementary literal of l, i.e., if $l = x$ then $\bar{l} = \bar{x}$ and if $l = \bar{x}$ then $\bar{l} = x$. A *clause* is a disjunction of literals (sometimes also interpreted as set of literals) and a *cube* is a conjunction of literals. The empty clause is denoted by $\bot$ or equivalently as the empty set $\emptyset$ of literals. A propositional formula in *conjunctive normal form* (CNF) is a conjunction of clauses. We sometimes write a formula as a set of clauses. By $\mathrm{var}(\psi)$ we denote the set of variables that occur in a formula ψ.

The *quantified Boolean formulas* (QBFs) considered in this chapter are in closed prenex conjunctive normal form (PCNF). A QBF $\Pi\psi$ in PCNF consists of a matrix ψ and a prefix $\Pi = \mathcal{Q}_1 X_1 \dots \mathcal{Q}_k X_k$ where $\mathcal{Q}_i \in \{\exists, \forall\}$, $\mathcal{Q}_i \neq \mathcal{Q}_{i+1}$, and X_i are pairwise disjoint sets of variables. Furthermore, $\mathrm{var}(\psi) \subseteq \mathrm{var}(\Pi)$ where $\mathrm{var}(\Pi)$ denotes the union of the sets X_i. If $x \in X_i$, we say that x is quantified at *level* i and write $\mathrm{lv}(x) = i$; we write $\mathrm{lv}(l)$ for $\mathrm{lv}(\mathrm{var}(l))$. If $\mathrm{lv}(l) = i$, $\mathsf{quant}(\Pi, l) = \mathcal{Q}_i$. We call $\mathcal{Q}_i X_i$ a *quantifier block*; $\mathcal{Q}_1 X_1$ is the *outermost* quantifier block and $\mathcal{Q}_k X_k$ is the *innermost* quantifier block. For two variables x, y, we have $x \leq_\Pi y$ iff $x \in X_i$ and $y \in X_j$ and $i \leq j$.

We use the standard recursive definition of the QBF semantics (we also write 1 for true and 0 for false). The QBF consisting only of the constant 0 is false and the QBF consisting only of the constant 1 is true. A QBF $\forall x \Pi\psi$ is true iff $\Pi\psi[0/x]$ and $\Pi\psi[1/x]$ are true, where $\Pi\psi[t/x]$ denotes the replacement of x by t. Further $\exists x \Pi\psi$ is true iff $\Pi\psi[0/x]$ or $\Pi\psi[1/x]$ is true. Often the evaluation of a QBF $\mathcal{Q}_1 X_1 \dots \mathcal{Q}_k X_k(\psi)$ is seen as a *game* between the *universal* and the *existential player*. In the i-th step of the game, the player $\mathcal{Q}_i$ assigns values to all the variables X_i. The existential player wins the game iff the matrix ψ evaluates to true under the assignment constructed in the game. The universal player wins iff the matrix ψ evaluates to false. For a universal variable u (an existential variable e), a *strategy* is a Boolean function over the existential (universal) variables that occur to the left of u (e) in the prefix. A QBF is true (false) iff there exists a *winning strategy* for the existential (universal) player, i.e. if the existential (universal) player has a strategy for all existential (universal) variables such that this player wins any game when assigning the variable according to this strategy. The Boolean functions forming a winning strategy of the existential (universal) player are often called *Skolem functions* (*Herbrand functions*). Note that given a

closed QBF, a winning strategy always exists for one and only one of the players.

A *proof system* for a language L over an alphabet Γ (usually binary) is a polynomial-time computable partial function $f : \Gamma^\star \to \Gamma^\star$ with $rng(f) = L$ [CR79]. If $f(x) = y$ then x is called an f-proof for y. The important features are soundness of the proof system (captured by $rng(f) \subseteq L$) and completeness of the system (captured by $L \subseteq rng(f)$). If L consists of all propositional tautologies, then f is called a *propositional proof system*, and proof systems for the language TQBF of true QBFs are called *QBF proof systems*. Equivalently, we can consider refutation proof systems where we start with the negation of the formula that we want to prove and derive a contradiction.

A proof system S for L *simulates* a proof system P for L if there exists a polynomial p such that for all P-proofs π of x there is an S-proof π' of x with $|\pi'| \leq p(|\pi|)$ where $|\pi|$ (resp., $|\pi'|$) denotes the length of π (resp., π'). If such a proof π' can even be computed from π in polynomial time we say that S *p-simulates* P. The system S is called *stronger* than P if S simulates P, but P does not simulate S. The latter is typically shown by exhibiting a sequence of formulas that require superpolynomial (often even exponential) size proofs in P, but have polynomial size proofs in S. The systems S and P are *incomparable* if neither S simulates P nor P simulates S.

For proof systems based on lines and inference rules (e.g., all the resolution-based proof systems below where lines are clauses), proofs can be classified as either dag-like or tree-like. In *tree-like proofs* every derived line can be used at most once in a further inference, implying that the proof graph is a tree. If derived lines can be reused, proofs are called *dag-like*. Often, as e.g., in resolution (both propositionally and in QBF), the dag-like proof system is stronger than the tree-like system [BEGJ00].

2.3. Proof Systems Based on Q-Resolution

Since the 1960s, the resolution principle [Rob65] has become a state-of-the art approach in various domains of automated reasoning, cf. [BG01]. In the field of propositional satisfiability testing (SAT), groundbreaking technology to leverage the power of resolution in CDCL SAT solvers was developed in the late 1990s [MSS99] and early 2000s [MMZ+01, ZMMM01], cf. Chapter **??**. Almost 20 years later, resolution still is the basic paradigm that underlies modern SAT solvers, which selectively integrate resolution with more powerful proof systems to potentially generate shorter proofs [WHJ14, HKB17].

The resolution principle has been lifted to the QBF setting [KBKF95], along with the CDCL approach called QCDCL [GNT06, Let02, ZM02a]. In the following, we present the *Q-resolution calculus* and variants of it as the formal foundation of QCDCL. Additionally, we point out how the workflow of QCDCL interacts with the rules of the Q-resolution calculus.

2.3.1. The Q-Resolution Calculus

Resolution is one of the best studied propositional proof systems (cf. Chapter **??**). It is a refutational proof system manipulating unsatisfiable CNFs as

$$\frac{C \cup \{l\}}{C} \quad \text{for all } x \in \mathsf{vars}(\Pi)\colon \{x, \bar{x}\} \not\subseteq (C \cup \{l\}),\ \mathsf{quant}(\Pi, l) = \forall, \text{ and } l' \leq_\Pi l \text{ for all } l' \in C \text{ with } \mathsf{quant}(\Pi, l') = \exists \qquad (\textit{red})$$

$$\frac{C_1 \cup \{p\} \qquad C_2 \cup \{\bar{p}\}}{C_1 \cup C_2} \quad \text{for all } x \in \mathsf{vars}(\Pi)\colon \{x, \bar{x}\} \not\subseteq (C_1 \cup C_2),\ \bar{p} \notin C_1,\ p \notin C_2, \text{ and } \mathsf{quant}(\Pi, p) = \exists \qquad (\textit{res})$$

$$\frac{}{C} \quad \text{for all } x \in \mathsf{vars}(\Pi)\colon \{x, \bar{x}\} \not\subseteq C \text{ and } C \in \psi \qquad (\textit{cl-init})$$

Figure 2.1. The rules of the Q-resolution calculus (Q-Res) [KBKF95] for PCNF $\phi = \Pi\psi$

sets of clauses. The only inference rule *propres* is defined by

$$\frac{C_1 \cup \{p\} \qquad C_2 \cup \{\bar{p}\}}{C_1 \cup C_2} \qquad (\textit{propres})$$

where C_1 and C_2 are clauses and p is the *pivot variable*. A *resolution refutation* derives the empty clause $\emptyset$. The generalization to QBF is shown in Figure 2.1.

We refer to a clause containing complementary literals l and $\bar{l}$ as being *tautological* or a *tautology*. Non-tautological clauses occurring in the CNF of the given QBF ϕ are selected by applications of the axiom rule *cl-init*. The resolution rule *res* is similar to the resolution rule *propres* in propositional logic. However, the pivot variable p is restricted to existential variables and derived clauses must not be tautological.

The main distinguishing feature between propositional resolution and Q-resolution (Q-Res) is the rule *red*, the *reduction* operation. *Universal reduction* eliminates *trailing* universal literals locally from a non-tautological clause C with respect to the linear quantifier ordering. A universal literal l is trailing in a clause C if all existential literals $l' \in C$ are smaller than l. Note that universal reduction is the only rule in the calculus that allows to eliminate universal literals from clauses. As such, it is crucial to obtain the empty clause in refutations of formulas in PCNF. In a more general setting, it was shown that a QBF proof system can be obtained [BBC16] by combining the universal reduction rule with propositional proof systems.

Example 2.3.1. Consider the PCNF $\phi = \exists x_1, x_3, x_4 \forall y_5 \exists x_2((\bar{x}_1 \vee x_2) \wedge (x_3 \vee y_5 \vee \bar{x}_2) \wedge (x_4 \vee \bar{y}_5 \vee \bar{x}_2) \wedge (\bar{x}_3 \vee \bar{x}_4))$. We resolve $(\bar{x}_3 \vee \bar{x}_4)$ and $(x_4 \vee \bar{y}_5 \vee \bar{x}_2)$ on the existential variable x_4 to obtain $C_0 = (\bar{x}_3 \vee \bar{y}_5 \vee \bar{x}_2)$. Note that the universal literal $\bar{y}_5$ cannot be reduced by universal reduction due to literal $\bar{x}_2$. We resolve C_0 and $(\bar{x}_1 \vee x_2)$ on x_2 and get $C_1 = (\bar{x}_1 \vee \bar{x}_3 \vee \bar{y}_5)$. Now literal $\bar{y}_5$ is trailing in C_1 and can be reduced, which produces the clause $C_2 = (\bar{x}_1 \vee \bar{x}_3)$.

The rules in Figure 2.1 define the common variant of Q-resolution, which is sound and refutational-complete for QBFs in PCNF [KBKF95]. The effect of imposing restrictions in the side conditions of the rules of the calculus reduces the number of possible proofs, often resulting in longer refutations. However,

$$\frac{C \cup \{l\}}{C} \quad l \in \{y, \bar{y}, y^*\},\ \mathsf{quant}(\Pi, l) = \forall, \text{ and } l' \leq_\Pi l \text{ for all } l' \in C \text{ with } \mathsf{quant}(\Pi, l') = \exists \qquad (\textit{ldqred})$$

$$\frac{C_1 \cup \{p\} \qquad C_2 \cup \{\bar{p}\}}{C_1 \cup C_2} \quad \bar{p} \notin C_1,\ p \notin C_2,\ \mathsf{quant}(\Pi, p) = \exists,\ \forall l_1 \in C_1, l_2 \in C_2\text{: if } \mathrm{var}(l_1) = \mathrm{var}(l_2) \text{ and } (l_1 \neq l_2 \text{ or } l_1 \text{ merged}) \text{ then } \mathsf{quant}(\Pi, l_1) = \forall \text{ and } \mathrm{lv}(p) \leq_\Pi \mathrm{lv}(l_1) \qquad (\textit{ldqres})$$

Figure 2.2. The long-distance Q-resolution calculus (LD-Q-Res) [ZM02a, BJ12] consists of the rules shown above in addition to the axiom rule *cl-init* from Figure 2.1. A literal y^* is called a merged literal and denotes the occurrence of a pair y and $\bar{y}$ of universal literals.

lifting these restrictions, either individually or in combined ways, results in more powerful variants of Q-resolution.

In the following, we discuss variants of Q-resolution that result from lifting restrictions or from defining rules that provide additional reasoning capabilities.

2.3.2. Extensions of Q-Resolution

Long-Distance Q-resolution. The CDCL paradigm for SAT solving was lifted to the QBF level soon after its inception [GNT06, Let02, ZM02a]. When deriving a learned clause C in QCDCL analogously to CDCL following the *first unique implication point (1-UIP)* principle, C might end up being a tautology caused by a pair of complementary universal literals [ZM02a]. Resolution steps producing tautologies are called *long-distance resolution* steps because the tautology is produced by resolving two clauses that contain two literals that appear positively in one clause and negatively in the other. In general, deriving and using tautological clauses as part of refutations is unsound, as the following example shows.

Example 2.3.2. Consider the PCNF $\phi = \forall x \exists y((x \vee \bar{y}) \wedge (\bar{x} \vee y))$ which is obviously satisfiable. In an attempt to produce a *wrong* refutation, we resolve the two clauses on variable y by neglecting the side condition of rule *res*, which produces the clause $(x \vee \bar{x})$. From this clause, we derive the empty clause by universal reduction, neglecting the side condition of rule *red*.

In contrast to the observation made in Example 2.3.2, tautologies derived during learning in QCDCL can be used in the same way as ordinary, non-tautological learned clauses to prune the search space. In [BJ12], long-distance resolution (LD-Q-Res) was formulated as a calculus shown in Figure 2.2. In the following, we use the common notation y^* to denote the occurrence of the pair $y \vee \bar{y}$ of complementary universal literals in a clause. This special literal y^* is called a *merged literal*.

Tautologies resulting from resolution steps according to rule *ldqres* are always caused by complementary universal literals and never by existential ones. Furthermore, for the soundness of the calculus, it is necessary to respect the quantification level restriction of the pivot variable and the merged universal literals. Under that restriction, the wrong refutation in Example 2.3.2 cannot be produced. Furthermore, the QCDCL framework by construction guarantees that

any tautologies derived during learning are derived under that restriction. Universal reduction by rule *ldqred* also allows the elimination of merged literals. The ability to produce certain tautological resolvents makes LD-Q-Res stronger than Q-Res [ELW13], cf. Section 2.7.

Example 2.3.3. Consider the PCNF $\phi = \exists x_1, x_3, x_4 \forall y_5 \exists x_2((\bar{x}_1 \vee x_2) \wedge (x_3 \vee y_5 \vee \bar{x}_2) \wedge (x_4 \vee \bar{y}_5 \vee \bar{x}_2) \wedge (\bar{x}_3 \vee \bar{x}_4) \wedge (x_3 \vee y_5 \vee \bar{x}_2))$ and the clause $C_0 = (\bar{x}_3 \vee \bar{y}_5 \vee \bar{x}_2)$ obtained by resolving $(\bar{x}_3 \vee \bar{x}_4)$ and $(x_4 \vee \bar{y}_5 \vee \bar{x}_2)$ on the existential variable x_4. We carry out a long-distance resolution step on C_0 using the clause $(x_3 \vee y_5 \vee \bar{x}_2)$ and the pivot variable x_3, which results in $C_1 = (y_5^* \vee \bar{x}_2)$. Note that the pivot variable x_3 has a smaller quantification level than the literals y_5 and $\bar{y}_5$ that result in the merged literal y_5^* according to rule *ldqres*.

Universal Pivots, Long-Distance Q-Resolution, and Combinations. The restriction of the quantifier type of pivot variables in Q-Res shown in Figure 2.1 can be relaxed to allow both existential and universal pivots in resolution steps. The resulting variant is called QU-resolution (QU-Res) and was shown to be stronger than Q-Res [VG12a]. Despite the possibility to eliminate universal literals from clauses by resolving on them when constructing a resolution refutation, the universal reduction rule is still crucial for the completeness of QU-Res.

Further variants of Q-resolution were obtained by combining the relaxation of the quantifier type restriction of pivot variables and the generation of certain tautological resolvents [BWJ14]. LQU-resolution extends long-distance Q-resolution by also admitting universal pivot variables like in QU-resolution. However, resolution steps on universal pivots must not produce tautological resolvents. This restriction is relaxed in LQU$^+$-resolution (LQU$^+$-Res), which is thus a generalization of LQU-resolution. For the soundness of LQU$^+$-Res, it is necessary to apply a stronger variant of the quantifier level restriction of pivot variables and merged literals than in long-distance resolution by rule *ldqres*.

Regarding proof complexity, both QU-Res and LD-Q-Res are stronger than Q-Res while being incomparable [BWJ14] to each other. LQU$^+$-Res is stronger than both QU-Res and LD-Q-Res (cf. Section 2.7).

Reductionless Q-resolution [BBM19, PSS19d] is a variant of LD-Q-Res which does not include the universal reduction rule *ldqred* (Figure 2.2). This variant was presented under the name *Q^ω-resolution* [BJK15] and implemented in the solver GhostQ [KSGC10]. A resolution refutation in reductionless Q-resolution does not end with the derivation of the empty clause but with a clause that contains only universal literals. Such clause could be reduced to the empty clause by universal reduction. In an early backtracking algorithm [CGS98] that precedes QCDCL, a similar rule was used to backtrack if a clause containing only universal literals was encountered during the search.

Enhanced Universal Reduction: Dependency Schemes. The linear ordering of the quantifier prefix of PCNFs has to be taken into account in practical QBF solving as well as in the theoretical context of QBF calculi, e.g., in the side condition of the universal reduction rule *red* in the Q-resolution calculus. The quantifier ordering imposes restrictions as it both limits the freedom of a QBF solver to assign variables and the potential number of universal literals

eliminated by universal reduction. As a way to overcome these restrictions, *dependency schemes* [Sam08, SS09] were introduced to relax the linear quantifier ordering.

Informally, a dependency scheme is a binary relation $\mathcal{D}$ over the set of variables of a QBF that expresses variable *independence*. Given two variables x and y which are quantified differently, if $(x, y) \notin \mathcal{D}$ then a QBF solver may safely assume that the value of y can be chosen independently of the value of x. Otherwise, if $(x, y) \in \mathcal{D}$, then it has to assume that the value of one depends on the value of the other. Neglecting the dependency of the values of variables, e.g., bears the risk of unsound results in solving.

Dependency schemes can be regarded as an approximate representation of the *actual* independence of variables. For two variables, actual independence means that their values can be chosen independently from each other without affecting the result of a semantical evaluation of a QBF. Checking whether two variables in a QBF are independent is as hard as solving the QBF itself, i.e., PSPACE-complete. Therefore, dependency schemes represent sound approximations of the actual independence of variables in a QBF. These approximations result in the presence of *spurious dependency pairs*. A dependency scheme $\mathcal{D}$ may contain spurious dependency pairs (x, y) such that $(x, y) \in \mathcal{D}$ but still the values of x and y can be chosen independently. A dependency scheme $\mathcal{D}$ is considered to be more refined than some other scheme $\mathcal{D}'$ if $\mathcal{D}$ contains fewer spurious dependency pairs than $\mathcal{D}'$. The potential amount of spurious dependency pairs present in a dependency scheme is expressed by a hierarchy of schemes defined by different levels of refinement. For practical applications, only dependency schemes are of interest that can be computed in polynomial time such as the *standard dependency scheme* [SS09] or the *resolution path dependency scheme* [VG11, SS12], the latter being the most refined, practically relevant scheme currently known. In contrast to that, the *trivial dependency scheme* exactly represents the linear ordering of the quantifier prefix and thus is least refined.

With respect to Q-resolution calculi, dependency schemes are relevant in that they can be used to strengthen the universal reduction rule as follows.

Definition 1 (Dependency-Aware Reduction [LB10])**.** Let $\phi = \Pi\psi$ be a PCNF and $\mathcal{D}$ be a dependency relation represented by a dependency scheme.

$$\frac{C \cup \{l\}}{\mathrm{C}} \qquad \begin{array}{l} C \text{ is a clause, } \mathsf{quant}(\Pi, l) = \forall, \text{ and,} \\ (l, l') \notin \mathcal{D} \text{ for all } l' \in C \text{ with } \mathsf{quant}(\Pi, l') = \exists \end{array} \qquad (\textit{dep-red})$$

Dependency-aware universal reduction by rule *dep-red* generalizes the traditional universal reduction rule *red* of the Q-resolution calculus (Figure 2.1). Instead of the linear ordering of variables ($\leq_\Pi$) in the prefix of the PCNF ϕ, rule *dep-red* makes use of a dependency relation $\mathcal{D}$. A universal literal l can be eliminated from a clause C if all existential literals l' in C are independent of l.

Note that rule *dep-red* indeed generalizes traditional universal reduction by rule *red* since it can be instantiated by the trivial dependency scheme. With dependency-aware universal reduction, it is possible to potentially reduce universal literals that cannot be reduced by traditional universal reduction, depending on the actual dependency relation used in rule *dep-red*.

Q-resolution calculi can be equipped with the dependency-aware universal reduction rule based on tractable dependency schemes. Using the reflexive resolution path scheme, e.g., a variant of Q-resolution is obtained that is stronger than traditional Q-resolution [BB17] (cf. Section 2.7.5). Certain theoretical properties of dependency schemes were studied that are sufficient to prove the soundness of combining these schemes with universal reduction in Q-resolution calculi, including long-distance Q-resolution [BB16, PSS16, PSS19a, PSS19c, SS16].

The benefits of exploiting variable independence were also empirically observed in the context of solving. Early approaches are based on exploiting quantifier structure that is present in the parse tree of a given QBF [GNT07]. Alternatively, such structure can be (partially) reconstructed from a PCNF by shifting quantifiers inside the CNF part [AB02, Ben05], which is also called *mini-scoping*. For example, the expansion-based solver Quantor [Bie04] applies a limited form of mini-scoping to allow for a better scheduling of expansion steps. Dependency schemes can be regarded as a generalization of mini-scoping. Already the standard dependency scheme [SS09], which is less refined than the resolution path scheme [VG11, SS12], potentially allows to identify more independent variables than mini-scoping [LB09].

Since QCDCL solvers rely on Q-resolution, dependency-aware universal reduction can be applied to leverage variable independence in QCDCL. The QCDCL solver DepQBF [LE17] tightly integrates dependency-aware universal reduction via the standard dependency scheme in the QCDCL workflow [LB10]. Additionally, it employs variable independence to generate assignments in QCDCL on a more flexible basis than in traditional QCDCL, which operates with the linear quantifier order. The approach implemented in DepQBF was generalized in the QCDCL solver Qute [PSS19a, PSS19b], which starts with an empty dependency relation and learns dependency pairs on demand.

Q-Resolution with Symmetries. Another powerful extension of the basic Q-resolution calculus (cf. Figure 2.1) exploits symmetries [KS18b] of a given formula. A symmetry for a QBF $\Pi\psi$ is a bijective map $\sigma\colon L \to L$ where $L = \{x, \bar{x} \mid x \in \mathrm{var}(\Pi)\}$ such that for all $x \in \mathrm{var}(\Pi)$, $\overline{\sigma(x)} = \sigma(\bar{x})$ and x, $\sigma(x)$ belong to the same quantifier block. Further, if σ is applied to all literals in ψ, then ψ has to map to itself (up to the order of literals and clauses). Next we introduce the symmetry rule as an extension of Q-Res.

Definition 2 (Symmetry Rule [KS18a])**.** Let $\phi = \Pi\psi$ be a PCNF.

$$\frac{C}{\sigma(C)} \qquad C \text{ is a clause, } \sigma \text{ is a symmetry of } \Pi\psi \qquad (sym)$$

Formula families that are hard for Q-Res such as $QParity_n$ or $KBKF_n$ have polynomial-size proofs [KS18a] with the symmetry rule (cf. Section 2.7.5).

2.3.3. Solving with Q-Resolution: QCDCL

Conflict-driven clause learning, called QCDCL [GNT06, Let02, ZM02a, ZM02b], is a generalization of the CDCL approach for SAT solving, cf. Chapter **??**. A QCDCL solver derives new learned clauses during the search with the purpose to

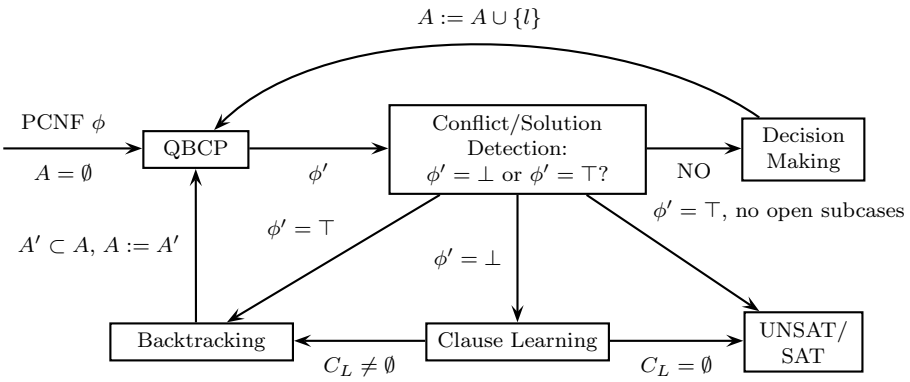

Figure 2.3. Flowchart of QCDCL. For simplicity of the presentation, cube learning is not shown. Instead, we assume that backtracking is carried out immediately after a satisfiable subcase has been detected ($\phi' = \top$).

prune the search space. Depending on the variable assignments that are successively generated, the solver may also produce *learned cubes*. Dual to clauses, a cube is a conjunction of literals. While learned clauses prune parts of the search space that contain falsifying assignments, learned cubes prune parts that contain satisfying assignments. Propositional resolution and Q-resolution [KBKF95] are the proof systems that underlie CDCL and QCDCL. For unsatisfiable and satisfiable QBFs, a QCDCL solver can produce clause and cube proofs, respectively. Cube proofs are formulated in a calculus that is dual to Q-resolution of clauses. Note that for a formula in PCNF, initially no cubes are given. Cubes are added to the formula only whenever an assignment satisfying the CNF is found.

Despite the similarity between CDCL and QCDCL on a theoretical level, clause learning in a QCDCL solver is more complex than in a CDCL solver. This is due to the interaction between the QCDCL workflow and the rules of the Q-resolution calculus. In the following, we point out this interaction on an abstract level. In particular, we do not give a comprehensive presentation of QCDCL and refer to Chapter **??** instead. Moreover, for simplicity we focus on clause learning and hence omit cube learning in our presentation of QCDCL.

We represent a variable assignment A as a set of literals such that $v \in A$ if variable v is assigned true, and $\neg v \in A$ if v is assigned false, and $v \notin A$ and $\neg v \notin A$ if v is unassigned. For a PCNF ϕ and an assignment A, $\phi[A]$ is the formula simplified under assignment A, i.e., the assigned variables are replaced by the respective truth constants which are then eliminated in the standard way. Figure 2.3 shows a high-level flowchart of the QCDCL workflow. The building blocks are similar to CDCL: propagation in stage QBCP (quantified Boolean constraint propagation), decision making, learning, and backtracking. Given a PCNF $\phi = \Pi\psi$, variable assignments are successively enumerated by means of QBCP and decision making. The formula is simplified under the current assignment A to obtain the formula ϕ'. If ϕ' is not reduced to a syntactic truth constant $\bot$ or $\top$ then the current assignment is further extended by decision making and

successive application of QBCP.

Otherwise, if $\phi' = \top$ then a satisfiable subcase has been determined and closed. In this case, if there are no more open subcases remaining to be explored, then QCDCL terminates with the result that the input formula ϕ is satisfiable. If there are open subcases remaining, then a certain part A' of the current assignment A is retracted during backtracking, and a new assignment A is generated. In implementations of QCDCL, determining whether there are open subcases left and backtracking after finding satisfiable subcases is driven by cube learning, which we omit in our presentation. Instead, we assume that backtracking is carried out immediately after a satisfiable subcase has been detected.

Case $\phi' = \bot$ designates an unsatisfiable subcase, which triggers clause learning. A new learned clause C_L is derived by Q-resolution using clauses depending on assignment A. If C_L is empty ($C_L = \emptyset$), then QCDCL terminates with the result that the input formula ϕ is unsatisfiable. In this case, a Q-resolution proof of ϕ can be obtained by considering the derivations of the learned clauses. Otherwise, if C_L is non-empty ($C_L \neq \emptyset$) then it is added to formula ϕ, backtracking is driven depending on C_L, and a new assignment A is generated.

In the following, we describe the interaction of universal reduction and Q-resolution with the workflow of QCDCL. The universal reduction rule *red* of the Q-resolution calculus (Figure 2.1) is crucial not only for the derivation of clauses, but also in the generation of assignments by propagation in QBCP. Similar to BCP in CDCL, QBCP in QCDCL consists of unit literal detection. In addition to BCP, QBCP also applies universal reduction to shorten clauses under assignment A that is currently being generated. The combination of unit literal detection and universal reduction potentially detects more assignments and shortens more clauses than unit literal detection alone.

Definition 3 (Unit Literal Detection [CGS98])**.**

Given a QBF ϕ, a clause $C \in \phi$ is *unit* iff $C = (l)$ and $q(l) = \exists$. The existential literal l in C is called a *unit literal*. *Unit literal detection (UL)* collects the assignment $\{l\}$ from the unit clause $C = (l)$. Unit literal detection applied to a QBF ϕ collects the respective assignments from all unit clauses in ϕ.

Learned clauses C_L generated in QCDCL have the property that they become unit after backtracking, similar to clause learning in CDCL. Such clauses are *asserting* clauses and are generated according to, e.g., the 1-UIP scheme. Note that the variables of unit literals in learned clauses are restricted to existential ones. This is necessary for the combination of unit literal detection and universal reduction in QBCP. In general, it is possible to equip QCDCL with sound variants of clause learning and certain backtracking schemes where non-asserting clauses are learned. Non-asserting clauses do not become unit after backtracking.

Definition 4 (QBCP)**.** Given a PCNF ϕ and the empty assignment $A = \{\}$, i.e. $\phi[A] = \phi$.

1. Apply universal reduction (UR) to $\phi[A]$ to obtain $\phi[A]'$.

2. Apply unit literal detection (UL) to $\phi[A]'$ to collect new assignments.

3. Add assignments found by UL to A, repeat steps 1 and 2 and stop if A does not change anymore or if $\phi[A] = \top$ or $\phi[A] = \bot$.

The result of QBCP applied to ϕ is an extended assignment A and a simplified formula $\phi[A]'$ obtained from $\phi[A]$ by UR.

Pure literal detection [CGS98] is another approach to generate assignments in QBCP in addition to unit literal detection. It increases the capability of QBCP to potentially generate assignments and shorten clauses. For simplicity of the presentation, however, we do not present pure literal detection.

Example 2.3.4. Let ϕ be a PCNF as follows:

$$\forall y_5 \exists x_1 \forall y_2 \exists x_3, x_4((\bar{y}_5 \vee x_4) \wedge (y_5 \vee \bar{x}_4) \wedge (x_1 \vee y_2 \vee \bar{x}_4) \wedge (\bar{x}_1 \vee x_3 \vee \bar{x}_4) \wedge (\bar{y}_2 \vee \bar{x}_3))$$

Initially, no simplifications of ϕ by QBCP are possible. We assign y_5 by decision making to obtain assignment $A = \{y_5\}$. The formula simplifies to

$$\phi[y_5] = \exists x_1 \forall y_2 \exists x_3, x_4((x_4) \wedge (x_1 \vee y_2 \vee \bar{x}_4) \wedge (\bar{x}_1 \vee x_3 \vee \bar{x}_4) \wedge (\bar{y}_2 \vee \bar{x}_3))$$

By UL and UR, we obtain the following assignments and simplified formulas.

- By UL: $\phi[y_5, x_4] = \exists x_1 \forall y_2 \exists x_3((x_1 \vee y_2) \wedge (\bar{x}_1 \vee x_3) \wedge (\bar{y}_2 \vee \bar{x}_3))$.
- By UR: $\phi[y_5, x_4] = \exists x_1 \forall y_2 \exists x_3((x_1) \wedge (\bar{x}_1 \vee x_3) \wedge (\bar{y}_2 \vee \bar{x}_3))$.
- By UL: $\phi[y_5, x_4, x_1] = \forall y_2 \exists x_3((x_3) \wedge (\bar{y}_2 \vee \bar{x}_3))$.
- By UL: $\phi[y_5, x_4, x_1, x_3] = \forall y_2((\bar{y}_2))$.
- By UR: $\phi[y_5, x_4, x_1, x_3] = \bot$.

A conflict (i.e., empty clause) is derived in the last step because the clause $C_\emptyset = (\bar{y}_2 \vee \bar{x}_3)$ is falsified under the assignment that was generated by QBCP. Hence, by the soundness of QBCP we have shown that $\phi[y_5]$ and $\phi[y_5, x_4, x_1, x_3]$ are satisfiability-equivalent, i.e., $\phi[y_5]$ is unsatisfiable. Since the decision variable y_5 is universal, also ϕ is unsatisfiable. Note that in this example, universal reduction is crucial to determine the unsatisfiability of ϕ because unit literal detection alone cannot derive a conflict.

For every assignment l collected by unit literal detection in QBCP the corresponding *antecedent clause* is recorded. The antecedent clause of assignment l is the clause in the original PCNF ϕ that contains l and became unit under the current assignment obtained by QBCP. Given a conflict as in Example 2.3.4, the falsified clause $C_\emptyset$, and the set of antecedent clauses of unit literals in the current assignment, new learned clauses can be derived in QCDCL in a similar but more complex way as in CDCL. We illustrate such a derivation in the following example.

Example 2.3.5. Let ϕ be the PCNF from Example 2.3.4:

$$\forall y_5 \exists x_1 \forall y_2 \exists x_3, x_4((\bar{y}_5 \vee x_4) \wedge (y_5 \vee \bar{x}_4) \wedge (x_1 \vee y_2 \vee \bar{x}_4) \wedge (\bar{x}_1 \vee x_3 \vee \bar{x}_4) \wedge (\bar{y}_2 \vee \bar{x}_3))$$

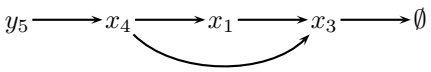

Figure 2.4. Implication graph related to the application of QBCP in Example 2.3.4. Variable y_5 is assigned as a decision.

The assignment generated by decision making and QBCP in Example 2.3.4 is $A = \{y_5, x_4, x_1, x_3\}$, with variables assigned in that ordering. The antecedent clauses of the assignments $\{x_4\}$, $\{x_1\}$, and $\{x_3\}$ are $C_{x_4} = (\bar{y}_5 \vee x_4)$, $C_{x_1} = (x_1 \vee y_2 \vee \bar{x}_4)$, and $C_{x_3} = (\bar{x}_1 \vee x_3 \vee \bar{x}_4)$, respectively. We resolve C_{x_1} and C_{x_4} and obtain $C_1 = (\bar{y}_5 \vee x_1)$ after universal reduction. Further, we resolve C_1 with C_{x_3} and obtain $C_2 = (\bar{y}_5 \vee x_3 \vee \bar{x}_4)$, which we resolve with C_{x_4} to get $C_3 = (\bar{y}_5 \vee x_3)$. Finally, by resolving C_3 and the clause $C_\emptyset = (\bar{y}_2 \vee \bar{x}_3)$ that was falsified during QBCP in Example 2.3.4, we obtain the empty clause by universal reduction.

In Example 2.3.5, we constructed a Q-resolution proof of the given PCNF by making what appears as an *ad-hoc* selection of clauses to be resolved. QCDCL solvers use information gathered in QBCP in a *systematic* way to carry out resolution steps to derive learned clauses. Pivot selection in QCDCL is crucial as it determines whether a Q-resolution or a long-distance Q-resolution derivation of the learned clause is obtained. In practical terms, a wrong pivot selection may result in failure to derive a learned clause.

In the following, we illustrate how the pivot selection influences the derivation of a learned clause in QCDCL. To this end, we informally introduce the concept of *implication graphs*.

Example 2.3.6. An implication graph related to the application of QBCP in Example 2.3.4 is shown in Figure 2.4. The implication graph contains a vertex for every decision variable and variable assigned by unit literal detection. It also contains a special vertex $\emptyset$ representing the falsified clause. We add an edge (x, y) if y was assigned by unit literal detection and $\bar{x}$ appears in the antecedent clause of y. For example, $C_{x_3} = (\bar{x}_1 \vee x_3 \vee \bar{x}_4)$ is the antecedent clause of assignment $\{x_3\}$ and the edges (x_1, x_3) and (x_4, x_3) indicate that $\bar{x}_1$ and $\bar{x}_4$ appear in C_{x_3}.

Implication graphs generated in QBCP are very similar to the ones generated by BCP in CDCL. However, universal literals eliminated by universal reduction are not explicitly represented by an edge. Hence, for example, the clause $C_\emptyset = (\bar{y}_2 \vee \bar{x}_3)$ that is falsified in Example 2.3.4 has two literals but in the implication graph in Figure 2.4 there is only one edge from x_3 to the special vertex $\emptyset$. We illustrate the selection of pivot variables and its importance for the derivation of learned clauses by the following example.

Example 2.3.7. Let ϕ be a PCNF as follows:

$$\exists x_1, x_3, x_4 \forall y_5 \exists x_2 ((\bar{x}_1 \vee x_2) \wedge (x_3 \vee y_5 \vee \bar{x}_2) \wedge (x_4 \vee \bar{y}_5 \vee \bar{x}_2) \wedge (\bar{x}_3 \vee \bar{x}_4))$$

Suppose that we assign x_1 to true as a decision variable. Then we obtain the assignment $A := \{x_1, x_2, x_3, x_4\}$, in that ordering, by QBCP. Note that for obtaining assignments x_3 and x_4 universal reduction of y_5 and $\bar{y}_5$ was applied.

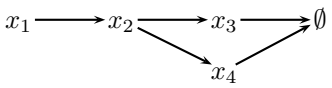

Figure 2.5. Implication graph related to the application of QBCP in Example 2.3.7. Variable x_1 is assigned as a decision.

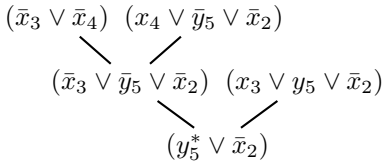

Figure 2.6. Derivation related to Example 2.3.7

Figure 2.5 shows the corresponding implication graph. The antecedent clauses of assignments $\{x_2\}$, $\{x_3\}$, and $\{x_4\}$ are $C_{x_2} = (\bar{x}_1 \vee x_2)$, $C_{x_3} = (x_3 \vee y_5 \vee \bar{x}_2)$, and $C_{x_4} = (x_4 \vee \bar{y}_5 \vee \bar{x}_2)$, respectively, and $C_\emptyset = (\bar{x}_3 \vee \bar{x}_4)$ is the clause that is falsified in QBCP.

Traditional QCDCL solvers derive learned clauses using Q-resolution (Figure 2.1). To this end, pivot variables are selected in reverse assignment ordering and we start by resolving the falsified clause $C_\emptyset$ with an antecedent clause. This approach is similar to the pivot selection in CDCL SAT solvers. In our example, we first resolve the falsified clause $C_\emptyset$ with C_{x_4}, since x_4 was assigned last, and the resulting clause with C_{x_3}. However, this results in the tautology $(y_5^* \vee \bar{x}_2)$, as shown in Figure 2.6.

To obtain a legal Q-resolution derivation of a learned clause, tautologies must be avoided. The reason for the tautology produced in Example 2.3.7 is that complementary universal literals appearing in relevant antecedent clauses were eliminated by universal reduction in QBCP. To avoid tautologies, such universal literals must be eliminated by universal reduction during the derivation of the learned clause. This can be achieved by selecting existential literals as pivots that block the universal literals from being eliminated. To this end, pivots must be selected in a more flexible way, i.e., not in strict reverse assignment ordering.

Example 2.3.8. Referring to Example 2.3.7 and Figure 2.6, the tautology can be avoided by first resolving the intermediate clause $(\bar{x}_3 \vee \bar{y}_5 \vee \bar{x}_2)$ with C_{x_2} on variable x_2 to obtain $(\bar{x}_1 \vee \bar{x}_3)$ after eliminating $\bar{y}_5$ by universal reduction. Note that the pivot variable x_2 is larger than the variable of the universal literal $\bar{y}_5$ in the quantifier ordering. The clause $(\bar{x}_1 \vee \bar{x}_3)$ is further resolved with C_{x_3}, resulting in $(\bar{x}_1 \vee y_5 \vee \bar{x}_2)$. A final resolution step with C_{x_2} on variable x_2 produces the learned clause $(\bar{x}_1)$. The complete Q-resolution derivation is shown in Figure 2.7.

The approach illustrated in Example 2.3.8 guarantees that tautologies in the derivations of learned clauses are always avoided [GNT06]. The resulting derivations are *linear* (cf. [CL73]). Informally, linear derivations consist of a single chain of resolution steps where every clause that appears in the derivation is either

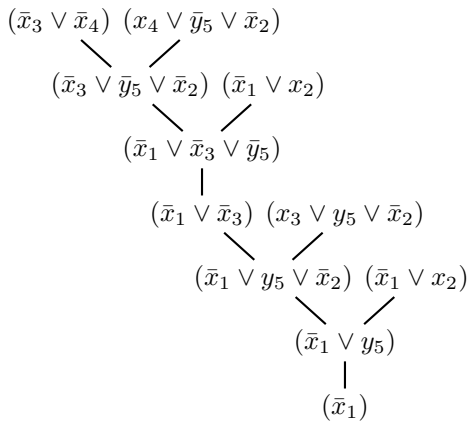

Figure 2.7. Derivation related to Example 2.3.8. Nodes having one parent represent the result of universal reduction steps.

a given clause or the result of a resolution step that uses the immediate previous resolvent in the chain. Note that in Example 2.3.8 pivot variables were selected in the ordering x_4, x_2, x_3, x_2, which is in contrast to the reverse assignment ordering x_4, x_3, x_2. However, this approach has an exponential worst case [VG12a]. The size of the derivation of a single learned clause may be exponential in the size of the implication graph because universal literals eliminated in parts of the derivation may later be reintroduced. In the chain of resolution steps in Figure 2.7, the universal literal $\bar{y}_5$ was first eliminated but later literal y_5 was introduced. Hence we had to select variable x_2 as pivot a second time.

As a remedy to the exponential worst case, *long-distance Q-resolution derivations* of learned clauses can be produced. With this approach, pivots can always be selected in reverse assignment ordering [ELW13], which is similar to the pivot variables selection in CDCL SAT solvers. Moreover, the size of the long-distance Q-resolution derivation of a learned clause is linear in the size of the respective implication graph. Quaffle [ZM02a] was among the first solvers that applied long-distance Q-resolution for clause learning. Figure 2.8 shows a long-distance Q-resolution derivation of the learned clause $(\bar{x}_1)$ related to Examples 2.3.7 and 2.3.8.

As an alternative to long-distance Q-resolution derivations, traditional Q-resolution derivations of learned clauses can be generated based on QPUP (QBF pseudo unit propagation) clauses [LEVG13]. In this approach, resolution steps are carried out starting from certain points in the implication graph and the resolution process proceeds towards the falsified clause. This is in contrast to traditional QCDCL and CDCL SAT solvers, where the resolution process starts at the falsified clause and proceeds towards a unique implication point. In Example 2.3.5, we actually constructed a refutation following the QPUP approach, see also the related implication graph in Figure 2.4. The sizes of the Q-resolution derivations of learned clauses produced by QPUP-based learning are always linear in the size of the implication graph. In contrast to Q-resolution derivations

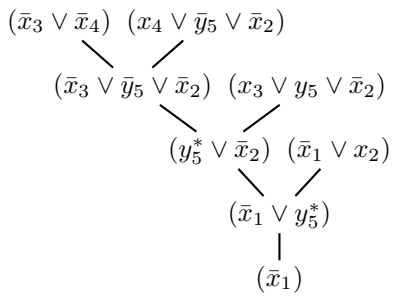

Figure 2.8. Long-distance Q-resolution derivation of the learned clause $(\bar{x}_1)$ related to Examples 2.3.7, and 2.3.8. The derivation continues the one shown in Figure 2.6.

obtained by systematically avoiding tautologies (Example 2.3.8 and Figure 2.7), which are linear (i.e., single chains of resolution steps) but may have exponential size, QPUP derivations in general are non-linear and have the form of a directed acyclic graph (DAG).

We presented several ways to enhance the traditional Q-resolution calculus (Figure 2.1). These include stronger Q-resolution variants, i.e., variants of rule *res*, and stronger variants of universal reduction (rule *red*) based on dependency schemes. The Q-resolution calculus can also be strengthened by a more general variant of the axiom rule *cl-init*. This general variant employs assignments generated in QCDCL and calls to a QBF oracle.

Definition 5 (Generalized Axiom [LBB+15, LES16])**.** Let $\phi = \Pi\psi$ be a PCNF. The *generalized axiom* is defined as follows.

$$\frac{\begin{array}{l}A \text{ is a (partial) assignment generated in QCDCL,}\\ \phi[A] \text{ is unsatisfiable,}\\ \text{and } C = (\bigvee_{l \in A} \bar{l}) \text{ is a clause}\end{array}}{\mathrm{C}} \quad (\textit{gen-cl-init})$$

The generalized axiom *gen-cl-init* can be added to the Q-resolution calculus in addition to the traditional axiom *cl-init* in Figure 2.1. Furthermore, it can be combined with any variants of the Q-resolution and universal reduction rules. Applications of the axiom rule require to check the satisfiability of $\phi[A]$. Since this problem is PSPACE-complete in general, for practical applications these satisfiability checks must be bounded and hence are incomplete. If $\phi[A]$ is found unsatisfiable, then the clause C can be derived and used as an ordinary learned clause in QCDCL. Note that learning C is sound only if A is an assignment generated in QCDCL.

Any QBF decision procedure can be used as an oracle to carry out the satisfiability test in the side condition of axiom *gen-cl-init*. Due to this property, the general axiom acts as an interface to integrate arbitrary decision procedures, and hence proof systems, in QCDCL. This way, it is possible to benefit from the combination of proof systems such as Q-resolution and, e.g., bounded expansion by means of clause learning. The resulting variant of QCDCL has the potential to produce proofs that are exponentially shorter than the proofs produced by traditional QCDCL.

Similar to the clause-based variant in Definition 5, a dual, cube-based variant can be formulated [LBB+15] for enhanced cube learning.

2.4. Expansion-Based Proof Systems

Taking a different approach than QCDCL, a number of solvers reduce QBF to SAT. Most commonly, they rely on iterative elimination of quantified variables until only one type of quantifier is left. Once the formula contains a single type of quantifier, an off-the-shelf SAT solver is invoked.

One of the first solvers taking this approach is the solver QUBOS [AB02], presented in 2002, which expands one kind of quantified variables inside-out. The kind of quantified variables to be expanded is chosen according to some heuristics. For universal variables expansion is performed as follows. If a QBF contains a subformula $\forall x\phi$ such that ϕ contains no further universal quantifiers, then $\forall x\phi$ is replaced by the conjunction $(\phi[0/x] \wedge \phi[1/x])$. The approach works analogously for existential quantification. Several simplifications are applied to restrict the space consumption of the solver.

The solver Quantor [Bie04] also expands the innermost universal variables, but in contrast to QUBOS, it is designed to work on PCNF only. Consider the QBF

$$\Pi\exists x((C_1 \vee x) \wedge \ldots \wedge (C_m \vee x) \wedge (D_1 \vee \bar{x}) \wedge \ldots (D_n \vee \bar{x}) \wedge E_1 \wedge \ldots E_l)$$

where C_i, D_j, E_k are clauses neither containing x nor $\bar{x}$. The expansion of the innermost existentially quantified variable x would result in the non-PCNF QBF

$$\Pi(((C_1 \wedge \ldots \wedge C_m) \vee (D_1 \wedge \ldots \wedge D_n)) \wedge E_1 \wedge \ldots \wedge E_l).$$

For restoring the PCNF structure of this formula the Cartesian product of clauses C_i and D_j has to be taken. To keep the PCNF structure and to avoid the extra transformation step, Quantor removes innermost existential variables in the style of the Davis-Putnam decision procedure (also known as variable elimination, see Chapter **??**). To eliminate the variable x of the formula above, all resolvents of $(C_i \vee x)$ and $(D_j \vee \bar{x})$ are added to the formula. Then the clauses containing x or $\bar{x}$ are removed. Hence, variable elimination directly returns the same formula in PCNF that would have been obtained by expanding x and performing the additional normal form transformations.

Later, the solver Nenofex [LB08] implemented the direct expansion of existential variables by giving up the requirement that the formula has to be in PCNF. The solver QMRES [PV04] follows a similar approach but tries to mitigate the size of the expansion by representing sets of clauses as ZBDDs. The solver AIGSolve [PS09] operates on And-Inverter Graphs (AIG) and solves QBFs by successively eliminating the innermost quantified variables.

A different take on elimination is adopted in the solver sKizzo [Ben04], which operates on a PCNF input and eliminates only universal variables. This, however, requires the introduction of fresh existential variables. The reason for such new variables is that if the variable x is eliminated from a formula $\forall x\exists y(\phi)$, the result is no longer prenex: $\exists y(\phi[0/x]) \wedge \exists y(\phi[1/x])$. To get back to prenex form, fresh

variables are introduced: $\exists y' y''(\phi[0/x, y'/y] \wedge \phi[1/x, y''/y])$. In sKizzo the process of introducing fresh variables is referred to as *propositional Skolemization* because effectively the fresh variables correspond to the individual points of a Skolem function for the original existential variable.

Solvers relying on full expansion of quantifiers inherently suffer from large memory consumption. Some solvers counter this issue by expanding quantifiers gradually but at the cost of multiple SAT calls. This approach was pioneered by the solver RAReQS [JMS11, JKMSC12, JKMC16] and followed by QELL [THJ15] and Ijtihad [BBH+18]. While the former two QBF solvers rely on one SAT solver instance per quantifier block, the latter one uses only two SAT solvers: one for dealing with the existentially quantified variables and one for dealing with the universally quantified variables.

At the proof level, solving by expansion corresponds to the proof system ∀Exp+Res [JMS13, JM15] discussed in Section 2.4.1. Sections 2.4.2 and 2.4.4 describe the algorithms AReQS and RAReQS, where AReQS accepts only 2QBF inputs and RAReQS extends AReQS by recursion to an arbitrary number of levels. Sections 2.4.3 and 2.4.5 connect the algorithms to the ∀Exp+Res system.

2.4.1. The Calculus ∀Exp+Res

The ∀Exp+Res calculus provides a transformation that enables refuting QBF by classical propositional resolution [JMS13, JM15]. This transformation stems from the idea that the existential player needs to respond to any assignment to the universal variables, but the player may respond differently to different assignments. To represent the different choices of the existential player, ∀Exp+Res annotates existential variables by the relevant assignments to the universals.

The calculus is defined in Figure 2.9. It comprises two rules. The *Axiom rule* introduces new annotated clauses into the proof and the *Res* rule enables resolving them. An *annotated clause* is obtained by choosing a clause C from the matrix and an assignment τ to all the universal variables. The clause C is then transformed by applying τ to it and annotating each existential variable by $[\tau]$. The operation $x^{[\tau]}$ annotates x with the assignment τ restricted to the universal variables preceding x in the quantifier prefix. Note that in contrast to the calculi presented in the previous section, no special treatment of tautologies is necessary. The rest of the proof is carried out by resolving the annotated clauses. A proof in ∀Exp+Res that derives the empty clause, is called a *refutation*.

Note that the Axiom rule of the calculus potentially enables introducing an exponential number of annotated clauses. However, if the rule is applied to some assignment τ and clause C such that τ assigns some of the literals of C to true, the clause is reduced to true, denoted as $\top$; such clause is useless for the rest of the proof. Observe also that since τ always assigns all universal variables, there are no universal literals in the annotated clauses. At the end of this chapter, Section 2.7 discusses more general proof systems that enable assigning only some universal variables.

Example 2.4.1. Figure 2.10 shows a possible ∀Exp+Res refutation of the formula

$$\exists e_1 \forall u_1 u_2 \exists e_2 \left((e_1 \vee u_1 \vee e_2) \wedge (\bar{e}_1 \vee \bar{u}_1 \vee e_2) \wedge (u_2 \vee \bar{e}_2)\right)$$

$$\frac{}{\{l^{[\tau]} \mid l \in C, l \text{ is existential}\} \cup \{\tau(l) \mid l \in C, l \text{ is universal}\}} \text{ (Axiom)}$$

C is a clause from the matrix and τ is an assignment to all universal variables.

$$\frac{C_1 \cup \{x^\tau\} \qquad C_2 \cup \{\bar{x}^\tau\}}{C_1 \cup C_2} \text{ (Res)}$$

Figure 2.9. The rules of ∀Exp+Res (adapted from [JM15]). The annotation $x^{[\tau]}$ considers only the universal variables that precede x in the prefix.

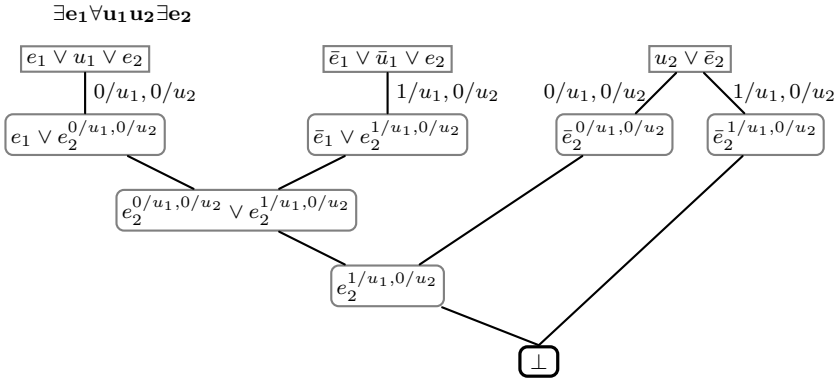

Figure 2.10. An example refutation in ∀Exp+Res

The matrix clauses are depicted as rectangles and the annotated clauses as rectangles with rounded corners. The example demonstrates some distinctive features of the calculus. The clause $u_2 \vee \bar{e}_2$ is expanded in two different ways. The annotated clause $e_2^{0/u_1,0/u_2} \vee e_2^{1/u_1,0/u_2}$ contains two versions of the variable e_2. Intuitively, this clause tells us that e_2 should be true in one of the two places, when both u_1 and u_2 are false or if u_1 is true and u_2 is false.

2.4.2. AReQS: Expansion-based solving for 2-level QBF

This section presents the algorithm AReQS [JMS11], which decides closed formulas with two levels of quantification, i.e., $\forall X \exists Y \phi$ or $\exists X \forall Y \phi$. The algorithm enables natively solving formulas with a matrix that is not necessarily in CNF and we present it as such. However, when connecting the algorithm to the proof system ∀Exp+Res, we consider only PCNF since that is what is supported by the proof system.

To simplify the presentation we will only be looking at formulas of the form $\forall X \exists Y \phi$. The case $\exists X \forall Y$ is symmetric since $\forall X \exists Y \phi$ is true if and only if $\exists X \forall Y \neg\phi$ is false. In some scenarios, we need to call a SAT solver with non-CNF input. While SAT solvers in practice often require CNF input, this represents

only a minor obstacle as CNF can be achieved by the addition of fresh variables, cf. [Tse68, PG86].

The algorithm AReQS is presented as a search for the winning move for the given formula anchored in the game perspective (see Section 2.2). Finding a winning move for a prefix of the form $\forall\exists$ shows the formula to be false, conversely, demonstrating the absence of such move shows the formula to be true.

Definition 6 (winning move)**.** An assignment τ to variables X is a *winning move* for a QBF $\forall X\Phi$ if $\Phi[\tau]$ is false and it is a *winning move* for $\exists X\Phi$ if $\Phi[\tau]$ is true.

Calculating a winning move is conceptually simple when given a formula with a single quantifier block. Such formulas represent a 1-move game, where only one of the players is allowed to make a single move upon which the game terminates. Deciding such game translates naturally to propositional satisfiability. In particular, there exists a winning move for the formula $\exists X\phi$ if and only if ϕ is satisfiable. Analogously, there exists a winning move for the $\forall$-player for the formula $\forall X\phi$ if and only if $\neg\phi$ is satisfiable. Further, the satisfying assignments of the respective propositional formula are winning moves of the original formula.

This observation motivates the following approach to solving QBF. Start eliminating quantifiers until only one type of quantifier is left, at which point invoke a SAT solver. The question is how to eliminate quantifiers. The approach taken here is by *expansion* into Boolean connectives. This is enabled by the observation that the formula $\forall X\exists y\Phi$ has the same set of winning moves as the formula $\forall X(\Phi[0/y] \vee \Phi[1/y])$.

A key insight is that a full expansion is not necessarily needed and that a *partial expansion* may decide the formula. This is possible because assigning a variable to a specific value *reduces* the set of winning moves, cf. [JKMC16, Sec. 3].

Example 2.4.2. Let $\phi = (u \vee e_1) \wedge (\bar{u} \vee e_2) \wedge (\bar{e}_1 \vee \bar{e}_2)$ and consider $\forall u\exists e_1 e_2\phi$. The formula is true and therefore there is no winning move for the $\forall$ player. One could expand e_1 and e_2 with all the 4 possible assignments to e_1 and e_2. However, considering only two is sufficient. Expanding by the assignments $\{1/e_1, 0/e_2\}$ and $\{0/e_1, 1/e_2\}$ yields $\forall u(\phi[1/e_1, 0/e_2] \vee \phi[0/e_1, 1/e_2]) = \forall u(\bar{u} \vee u) = \forall u(1)$. Since $\forall u(1)$ has no winning moves, the original formula also does not have any and therefore it is true.

The idea of partial expansions is formalized in the following definition as an ω-abstraction.

Definition 7 (ω-abstraction)**.** Let X, Y be finite sets of variables and ω be a set of assignments to Y. The *ω-abstraction* of a closed QBF $\forall X\exists Y\phi$ is $\forall X \bigvee_{\mu\in\omega} \phi[\mu]$.

The algorithm AReQS starts by an empty abstraction and strengthens it gradually using the *Counterexample Guided Abstraction Refinement* paradigm (CEGAR) [CGJ+03]. The ω-abstraction is an abstraction in the sense that its set of winning moves is a superset of the winning moves of the original formula. Adding more assignments to the set ω restricts the set of winning moves, i.e. it *refines* the abstraction.

Algorithm 1 shows the AReQS ideas in pseudocode. It consists of a loop, which in each iteration first finds a winning move for the current ω-abstraction

Algorithm 1: CEGAR Algorithm for 2QBF

input : closed $\forall X \exists Y(\phi)$ with ϕ propositional
output : τ if there exists a winning move τ for $\forall X$, $\bot$ otherwise

```
1 ω ← ∅
2 while true do
3     α ← ⋁_{μ∈ω} φ[μ]                          // build abstraction
4     τ ← SAT(¬α)                               // find a candidate
5     if τ = ⊥ then return ⊥                    // no winning move, formula true
6     μ ← SAT(φ[τ])                             // find a countermove
7     if μ = ⊥ then return τ                    // no countermove, formula false
8     ω ← ω ∪ {μ}                               // refine
```

and then it tests whether that move is also a winning move of the formula being solved. If it is, we are done. If, however, the winning move for the abstraction is *not* a winning move of the original formula, the abstraction is refined.

A winning move of the abstraction is referred to as a *candidate*. A candidate τ is a winning move of the original $\forall X \exists Y \phi$ if and only if $\exists Y \phi[\tau]$ is false. Equivalently, it is a winning move if $\phi[\tau]$ is unsatisfiable. Consequently, if τ is *not* a winning move, there exists an assignment μ satisfying $\phi[\tau]$; the assignment μ is called a *countermove* to the candidate move τ.

The algorithm maintains a set of countermoves ω, initialized to the empty set (line 1). In each iteration of the loop it first constructs an abstraction according to Definition 7 (line 3). Note that upon initialization, the set ω is empty and therefore $\alpha = 0$ when first entering the loop. Subsequently, the algorithm tries to find a winning move for the abstraction (line 4). If no candidate is found, it means that there is no winning move for the given formula and the algorithm terminates. If a candidate τ is found, a SAT solver is used to find a countermove to it (line 6). If there is no countermove, then τ is indeed a winning move and the algorithm terminates. If there is a countermove μ, this countermove is added to the set ω (line 8).

Example 2.4.3. Consider the prefix $\forall u_1 u_2 u_3 \exists e_1 e_2 e_3$ and the following matrix

$$\begin{array}{l} u_1 \vee u_2 \vee e_1 \vee e_2 \\ u_1 \vee u_3 \vee \bar{e}_1 \vee e_2 \\ \bar{u}_1 \vee u_2 \vee e_3 \\ \bar{u}_2 \vee u_3 \vee \bar{e}_1 \vee e_3 \end{array}$$

The following is a possible run of the algorithm. Let $\tau_1 = \{0/u_1, 0/u_2, 0/u_3\}$ be the first candidate with $\phi[\tau_1] = \{e_1 \vee e_2, \bar{e}_1 \vee e_2\}$. The assignment $\mu_1 = \{0/e_1, 1/e_2, 0/e_3\}$ is a countermove to τ_1. The ω-abstraction refined by τ_1 is $\forall u_1 u_2 u_3 (\bar{u}_1 \vee u_2)$. The assignment $\tau_2 = \{1/u_1, 0/u_2, 0/u_3\}$ is a winning move for the abstraction and is chosen as the next candidate. We have $\phi[\tau_2] = \{e_3\}$, satisfiable by $\mu_2 = \{0/e_1, 0/e_2, 1/e_3\}$ chosen as the countermove.

Refining by μ_2 yields the ω-abstraction $\forall u_1 u_2 u_3 ((\bar{u}_1 \vee u_2) \vee (u_1 \vee u_2))$ which has no winning moves as its matrix is a tautology. This means that the original formula also has no winning moves and therefore it is true.

Example 2.4.3 uncovers an interesting property of AReQS in CNF. The assignments to the universal variables effectively select or deselect clauses for the existential quantifier to satisfy. The solvers QESTO [JMS15] and CAQE [RT15] develop this idea for arbitrary number of levels. A uniform proof system for the different CEGAR-based approaches was presented in [Ten17]. The solvers QuAbS [Ten16] and CQESTO [Jan18a] develop the ideas even further to non-CNF.

Since the winning moves of the ω-abstraction over-approximate the winning moves of the original formula, and the algorithm deems the formula true only if the abstraction has no winning moves, the original formula in such case must also be true. Termination follows from the observation that no candidates may repeat. Indeed, if μ is a countermove to τ, after the refinement the abstraction is of the form $\forall X(\alpha' \vee \phi[\mu])$. Consequently, any subsequent candidate τ' must falsify $\phi[\mu]$. This means that τ' cannot be τ as τ together with μ satisfy ϕ. By a similar argument we can show that countermoves also do not repeat.

2.4.3. Connecting AReQS to ∀Exp+Res

In order to relate AReQS to ∀Exp+Res, we focus only on the refutation of formulas of the form $\forall X \exists Y \phi$ where ϕ is in CNF. If $\forall X \exists Y \phi$ is false, there must be a winning move for the universal player. This means that AReQS finds an assignment τ to the first block such that there is no countermove to it. More specifically, $\phi[\tau]$ is unsatisfiable. Since $\phi[\tau]$ is unsatisfiable in purely propositional sense, there also exists a propositional resolution refutation π for it. The ∀Exp+Res refutation for the original formula $\forall X \exists Y \phi$ is constructed by following the structure of π.

Each leaf of the resolution refutation π is a clause $C' \in \phi[\tau]$ obtained from some clause $C \in \phi$. Construct a clause C^τ by invoking the Axiom rule on C using the assignment τ. The Axiom rule removes all universal variables from C and annotates each existential variable with τ. The rest of the proof is constructed by applying the same resolution steps as in π: whenever there is a resolution step on x in π, perform a resolution step on x^τ.

Applying the Axiom rule parameterized by the substitution τ to the clauses of the matrix, yields annotated clauses corresponding to the clauses in $\phi[\tau]$, where each existential variable is annotated by τ.

Example 2.4.4. Consider the clauses $C_1 = u_1 \vee e_1$, $C_2 = u_2 \vee \bar{e}_1$, and $C_3 = \bar{u}_1 \vee e_2$ under the prefix $\forall u_1 u_2 \exists e_1 e_2$. The matrix becomes unsatisfiable under the assignment $\tau = \{0/u_1, 0/u_2\}$. Substituting τ into the matrix gives $C'_1 = e_1$, $C'_2 = \bar{e}_1$, and $C'_3 = \top$, where the empty clause is obtained by resolving C'_1 and C'_2. The corresponding ∀Exp+Res proof applies the Axiom rule on C_1 and C_2 yielding $C_1^\tau = e_1^\tau$, $C_2^\tau = \bar{e}_1^\tau$, which are then resolved into $\bot$.

2.4.4. RAReQS: Expansion-based QBF Solving

This section presents the algorithm RAReQS [JKMSC12, JKMC16], which generalizes AReQS to an arbitrary number of quantification levels by recursion. The algorithm tries to find a winning move for the given formula but rather than calling a SAT solver, it uses recursive calls as a subroutine. The base case, with one quantifier block left, is handled by a SAT solver.

Algorithm 2: Basic recursive CEGAR for QBF (existential player)

```
1  Function solve(∃XΦ)
   input  : ∃XΦ is a closed QBF in prenex form with no adjacent blocks with the
            same quantifier
   output : a winning move for ∃XΦ if there is one, ⊥ otherwise
2  begin
3      if Φ has no quantifiers then
4          return SAT(Φ)
5      ω ← ∅
6      while true do
7          α ← ⋀_{μ∈ω} Φ[μ]
8          τ′ ← solve(prenex(∃Xα))                    // find a candidate solution
9          if τ′ = ⊥ then return ⊥                    // no winning move, formula false
10         τ ← {c/x | τ′(x) = c and x ∈ X}            // filter a move for X
11         μ ← solve(Φ[τ])                            // find a countermove
12         if μ = ⊥ then return τ                     // found winning move, formula true
13         ω ← ω ∪ {μ}                                // refine
```

Analogously to AReQS quantifiers are expanded into a propositional operator (disjunction of conjunction) and a partial expansion lets us approximate a set of winning moves. An abstraction of a QBF with at least two quantifier blocks is obtained by partially expanding the second quantifier block. Just as before, a *candidate* refers to a winning move of an abstraction.

Applying expansion to arbitrary quantifiers requires additional care. Consider for instance $\exists x \forall y \exists z \phi$. Expanding the middle quantifier $\forall y$ into a conjunction yields a QBF in non-prenex form: $\exists x(\exists z \phi[0/y] \wedge \exists z \phi[1/y])$. Therefore, the abstraction is calculated by additionally prenexing the expanded formula. We write $\texttt{prenex}(\Phi)$ for the prenexed form of a QBF Φ. Prenexing may require fresh variables. For instance, $\texttt{prenex}(\exists x(\exists z \phi[0/y] \wedge \exists z \phi[1/y]))$ gives $\exists x z^0 z^1(\phi[0/y, z^0/z] \wedge \phi[1/y, z^1/z]))$. We will see later that these fresh variables correspond to the annotations in ∀Exp+Res.

Definition 8 (ω-abstraction)**.** Let X, Y be finite sets of variables and ω be a set of assignments to Y.
The ω-abstraction of a closed QBF $\forall X \exists Y \Phi$ is the formula $\texttt{prenex}(\forall X \bigvee_{\mu \in \omega} \Phi[\mu])$.
The ω-abstraction of a closed QBF $\exists X \forall Y \Phi$ is the formula $\texttt{prenex}(\exists X \bigwedge_{\mu \in \omega} \Phi[\mu])$.

The algorithm is presented as a recursive function `solve`. Its implementation for the existential quantifier is presented in Algorithm 2. The implementation for the universal quantifier is analogous. The function accepts a closed QBF $QX\Phi$ where $Q \in \{\forall, \exists\}$; it returns a winning move for QX, if such exists and it returns $\bot$ otherwise.

The base case of the recursion is when Φ does not contain any quantifiers, in which case a SAT solver is used to find a winning move. In the general case, an ω-abstraction is gradually strengthened by countermoves in a similar vein as in AReQS. The prenexing step causes a small technical difficulty. A winning move for the abstraction potentially contains values for variables that correspond to

copies of variables originally appearing in inner quantifiers. These need to be filtered out as they do not exist in the original formula (line 10).

Observe that in each iteration there are two recursive calls. The first call obtains a winning move for the abstraction—a candidate (line 8). The second call checks whether the candidate is a winning move or not (line 11). In the first call the number of quantifier levels drops by two because the second quantifier is eliminated by expansion and the third quantifier is merged with the first one during prenexing. In the second call, the number of quantifier levels drops by one.

Example 2.4.5. Consider the QBF $\exists vw\Phi$, where

$$\Phi = \forall u \exists xy((v \vee w \vee x) \wedge (\bar{v} \vee y) \wedge (\bar{w} \vee y) \wedge (u \vee \bar{x}) \wedge (\bar{u} \vee \bar{y})),$$

The candidates $\{0/v, 0/w\}$ and $\{1/v, 1/w\}$ are countered by $\{0/u\}$ and $\{1/u\}$, respectively. Refinement expands into $\exists vw(\Phi[0/u] \wedge \Phi[1/u])$, with prenex form

$$\begin{aligned}\exists vwx'y'x''y''((v \vee w \vee x') \wedge (\bar{v} \vee y') \wedge (\bar{w} \vee y') \wedge (\bar{x}') \\ \wedge (v \vee w \vee x'') \wedge (\bar{v} \vee y'') \wedge (\bar{w} \vee y'') \wedge (\bar{y}''))\end{aligned}$$

This purely propositional abstraction is unsatisfiable. Hence, there is no winning move for $\exists$ and the function terminates and returns $\bot$, which means that the formula is false.

The arguments for correctness and termination are analogous to AReQS. Candidate moves or counterexamples cannot repeat. If in any of the iterations of the loop a candidate τ was countered by some μ, the move τ will never be drawn from the abstraction because the refinement guarantees that τ is losing in the strengthened abstraction.

RAReQS, compared to the presentation here, implements a modification that mitigates the growth of the abstraction for larger number of qualification levels (starting at 4). Rather than prenexing the whole prefix, only the outermost quantifier is prenexed and the rest is maintained in non-prenex form. For instance, $\forall X \exists Y \forall Z \exists W \phi$ may be expanded as $\forall X Z' Z''((\exists W \phi') \vee (\exists W \phi''))$. Any winning move for $\forall X Z' Z''$ must be a winning move for each of the individual disjuncts. Hence, candidate tests and refinements are carried out separately for each disjunct. From the game-theoretical point of view, each disjunct can be seen as a separate game that the quantifier must win. Therefore, the approach is referred to as *multi-games* [JKMC16, Sec 4.1]. The existential case is again analogous.

2.4.5. Connecting RAReQS to ∀Exp+Res

We show how to produce ∀Exp+Res refutations (Figure 2.9) from Algorithm 2. Consider the following naming scheme for fresh variables used in prenexing. Part of the name of each existential variable is an annotation corresponding to a substitution to some universal variables. In the input formula these annotations are empty. An annotation is extended whenever a fresh copy of the variable is added to an abstraction. As the recursion proceeds, the annotations are getting longer and at the same time the universal variables disappear from the formula.

Consider the prefix $\exists E_1 \forall U_1 \exists E_2 \forall U_2 \ldots \forall U_{n-1} \exists E_n \phi$. If the given formula happens to start with the universal quantifier, add a dummy $\exists E_1$. Expanding by some assignments μ_1 and μ_2 to the variables U_1 gives the following transformations.

$$\exists E_1((\exists E_2 \forall U_2 \ldots \forall U_{n-1} \exists E_n \phi[\mu_1]) \wedge (\exists E_2 \forall U_2 \ldots \forall U_{n-1} \exists E_n \phi[\mu_2])) \tag{2.1}$$

$$\begin{aligned} \exists E_1 E_2^{\mu_1} E_2^{\mu_2} \forall U_2 \ldots \forall U_{n-1} \exists E_n^{\mu_1} \exists E_n^{\mu_2} (\ & \phi[\mu_1, E_2^{\mu_1}/E_2, \ldots, E_n^{\mu_1}/E_n] \wedge \\ & \phi[\mu_2, E_2^{\mu_2}/E_2, \ldots, E_n^{\mu_2}/E_n] \ ) \end{aligned} \tag{2.2}$$

The original formula is expanded in (2.1) and prenexed in (2.2). The notation X^μ represents the set of variables obtained by appending μ to the annotation of each variable in X. Since ϕ is in CNF, the matrix of (2.2) is also in CNF. At the same time, each clause observes the invariant that if it comes from an expansion by μ_i, then it has μ_i in its annotations.

The multi-game expansion approach (Section 2.4.4) enables the subsequent expansions to be more surgical; in the example above, only one of the conjuncts can be expanded when expanding U_2.

The base case of the recursion of Algorithm 2 takes place when there are no universal variables left in the matrix. This means that in such a case the matrix contains annotated clauses containing only existential variables, as required by the expansion rule in the calculus. These are handed to a SAT solver. If the expansion is false, the SAT solver deems the expansion unsatisfiable, which is then certified by propositional resolution refutation. This refutation is readily translated to a $\forall$Exp+Res refutation as each propositional resolution step corresponds to a $\forall$Exp+Res resolution step.

Example 2.4.6. Consider $\exists e_1 \forall u_1 \exists e_2 \forall u_2 \exists e_3((e_1 \vee u_1 \vee e_2 \vee u_2 \vee e_3) \wedge \bar{e}_1 \wedge \bar{e}_2 \wedge \bar{e}_3)$ first expanded by $u_1 = 0$ yielding $\exists e_1 \exists e_2^{0/u_1} \forall u_2 \exists e_3^{0/u_1}(e_1 \vee e_2^{0/u_1} \vee u_2 \vee e_3^{0/u_1}) \wedge \bar{e}_1 \wedge \bar{e}_2^{0/u_1} \wedge \bar{e}_3^{0/u_1}$ which is further expanded by $u_2 = 0$, yielding thus $\exists e_1 \exists e_2^{0/u_1} \exists e_3^{0/u_1,0/u_2}(e_1 \vee e_2^{0/u_1} \vee e_3^{0/u_1,0/u_2}) \wedge \bar{e}_1 \wedge \bar{e}_2^{0/u_1} \wedge \bar{e}_3^{0/u_1,0/u_2}$. Propositional refutation of the last formula gives a $\forall$Exp+Res refutation.

2.5. Preprocessing

Over the years, many preprocessing techniques have been developed for QBFs (cf. [BLS11, BK07, GMN10, LE18, SDB06, VGWL12, WRMB17]). Preprocessing a QBF before it is passed to a complete solver is beneficial in many cases [LSVG16]. It is particularly helpful when the solver does not try to reconstruct information that was lost by the transformation to prenex conjunctive normal form (cf., for example, [JKMC16, GB13, HPSS18, Ten16, LB08, ESW09, AB02] for solvers that do not require the input formula to be in prenex conjunctive normal form). Preprocessors like sQueezBF [GMN10], Bloqqer [BLS11], HQSpre [WRMB17], or QRATPre+ [LE18, LE19] apply rewriting techniques that preserve (satisfiability) equivalence and that modify formulas in such a way that information relevant for the solver becomes easier to access and such that irrelevant information is eliminated.

2.5.1. Background

There are three types of preprocessing rules: (1) clause elimination rules, (2) clause addition rules, and (3) clause modification rules. The application of these rules might also require a modification of the prefix, because new variables could be introduced while some variables might vanish.

Clause elimination rules [HJL$^+$15] remove clauses while preserving unsatisfiability. Examples of such rules are tautology elimination, subsumption, the existential pure literal rule, and blocked clause elimination. Tautology elimination removes clauses containing a positive and negative occurrence of a variable. Subsumption as implemented in [Bie04] removes clauses that are supersets of other clauses. The existential pure literal rule [CGS98] removes all clauses with an existential literal that occurs only positively or only negatively in the formula (there is also a universal pure literal rule, but as being a clause modification rule, it is discussed below). Blocked clause elimination [BLS11] removes a clause C from a QBF $\Pi\psi$ if C contains an existential literal l such that for all $D \in \psi$ with $\bar{l} \in D$, there exists a literal k with $k \leq_\Pi l$ with $k, \bar{k} \in C \cup D \backslash \{\bar{l}\}$, i.e., the clause $C \cup D \backslash \{\bar{l}\}$ is a tautology.

Example 2.5.1. Consider the QBF ϕ given by

$$\forall x_1 x_2 \exists y_1 y_2((x_1 \vee y_1) \wedge (y_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{y}_1) \wedge (\bar{y}_2 \vee y_1) \wedge (x_1 \vee y_1 \vee x_2) \wedge (\bar{x}_1 \vee x_1 \vee y_2)).$$

The last clause is a tautology, hence it can be safely removed. After the removal of the last clause, the existential variable y_2 occurs only in one polarity, therefore the existential pure literal elimination rule is applicable and clause $(\bar{y}_2 \vee y_1)$ can be removed. Also x_2 occurs only in one polarity, but since it is universally quantified, we are not allowed to remove clauses $(y_1 \vee x_2)$ and $(x_1 \vee y_1 \vee x_2)$ with the existential pure literal rule. Nevertheless, we can remove the latter clause, because it is subsumed by $(x_1 \vee y_1)$. Next, we eliminate clause $(x_1 \vee y_1)$, because it is blocked on y_1. Finally, we obtain the simplified QBF $\forall x_1 x_2 \exists y_1((y_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{y}_1))$ which is satisfiability equivalent to ϕ.

Clause addition rules enrich a formula with new clauses besides modifying and removing clauses. Variable elimination [Bie04] replaces the clauses in which a certain existential variable occurs by all non-tautological resolvents on that variable. Therefore, the *res*-Rule as introduced in Section 2.3.1 is applied on the existential variables of the innermost quantifier block given that a certain limit of additional clauses is not reached.

Partial universal expansion [Bie04, BK07] is a restricted form of universal expansion as discussed in Section 2.4. It eliminates a universal variable x of the innermost universal quantifier block by copying the matrix of the QBF. Then x is set to true in one copy and to false in the other copy. For combing these two CNFs, in one copy new names must be assigned to the variables of the innermost existential quantifier block.

Example 2.5.2. Consider the QBF $\forall x_1 x_2 \exists y_1((y_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{y}_1))$ from the previous example. Eliminating y_1 results in QBF $\forall x_1 x_2((x_2 \vee \bar{x}_1))$. The universal expansion of the universal variables derives a formula containing the empty clause.

If we expand x_1 in $\forall x_1 x_2 \exists y_1((y_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{y}_1))$, we obtain $\forall x_2 \exists y_1 y_1'((y_1 \vee x_2) \wedge (\bar{y}_1) \wedge (y_1' \vee x_2))$. Note that y_1 was renamed to y_1' in the case that x_1 was set to false, for the case that x_1 was set to true, we keep y_1. Eliminating variable x_2 finally results in $\exists y_1 y_1' y_1'' y_1'''((\bar{y}_1) \wedge (y_1'') \wedge (\bar{y}_1'') \wedge (y_1'''))$.

Clause modification rules add, remove, and rename literals. One important rule is universal reduction, the *red*-Rule, as introduced above in Section 2.3.1. Strengthening [Bie04] removes a literal l from a clause C if there is another clause $D \vee \bar{l}$ in the formula such that D subsumes C. The unit literal elimination rule [CGS98] is applicable if a QBF $\Pi\psi$ contains a clause C with exactly one existential literal l and for all universal literals $k \in C$, $l \leq_\Pi k$. Then all clauses containing l are removed and so are all occurrences of $\bar{l}$. Universal pure literal [CGS98] elimination removes all occurrences of a universal literal l if it occurs only in one polarity in the considered QBF. Covered literal addition adds literals l_c to a clause C that occur in all non-tautological resolvents of C with clauses C' on an existential pivot l, where $l \in C$ and $\bar{l} \in C'$. Additionally, $l_c \leq_\Pi l$ must hold for all covered literals l_c. The equivalence replacement rule [Bie04] can be applied in QBF $\Pi\psi$ when two literals l, k with l is existentially quantified and $k \leq_\Pi l$ are found to be equivalent by detecting the clauses $(l \vee \bar{k})$ and $(\bar{l} \vee k)$ in ψ. Then l may be replaced by k. Finally, the blocked literal elimination rule [HSB15] can remove a universal literal l from a clause C if all resolvents on l contain two opposite literals k and $\bar{k}$ with $k \leq_\Pi l$. Blocked literal elimination generalizes the universal pure literal rule.

Example 2.5.3. Consider the QBF

$$\forall x_1 x_2 \exists y_1 y_2((y_1 \vee x_2 \vee \bar{y}_2) \wedge (x_1 \vee \bar{y}_2) \wedge (\bar{x}_1 \vee y_2) \wedge (\bar{y}_1 \vee \bar{x}_1 \vee \bar{y}_2)).$$

By strengthening, the clause $D = (\bar{y}_1 \vee \bar{x}_1 \vee \bar{y}_2)$ can substituted by $(\bar{y}_1 \vee \bar{x}_1)$, which is obtained by resolving D on $\bar{y}_2$ with clause $(\bar{x}_1 \vee y_2)$, because $(\bar{x}_1)$ subsumes D. By equivalence replacement, we may substitute y_2 by x_1 and under omission of the two tautological clauses $(x_1 \vee \bar{x}_1)$, we get $\forall x_1 x_2 \exists y_1((y_1 \vee x_2 \vee \bar{x}_1) \wedge (\bar{y}_1 \vee \bar{x}_1))$. Now x_1 and x_2 are pure literals and can therefore be omitted. The application of unit literal elimination results in a conflict.

For almost all of these preprocessing rules, Q-resolution proofs (see Section 2.3) can be generated to witness their correct application [JGM13]. However, this approach does not work for universal expansion which is a very crucial technique in preprocessing. In order to capture all recent preprocessing techniques including universal expansion, the QRAT proof system was developed.

2.5.2. The QRAT Proof System

The QRAT proof system [HSB17] lifts the RAT proof system (see Chapter **??**) from SAT to QBF. The RAT proof system introduces rules for adding and removing clauses such that the truth value of a formula is preserved. Besides the QBF variants of clause addition and clause deletion rules that have to take the variable ordering imposed by the prefix into account, the QRAT proof system provides additional rules for clause modification. Before we introduce the rules of QRAT,

we first define the notions of *outer resolvent* and *implication via unit propagation*.

Definition 2.5.1. The *outer resolvent* of clauses $C \vee l$ and $D \vee \bar{l}$ on literal l w.r.t. prefix Π is defined as $\mathsf{OR}(\Pi, C \vee l, D \vee \bar{l}, l) = C \cup \{k \mid k \in D, k \leq_\Pi l, k \neq l\}$.

In contrast to the classical resolvent $C \cup D$, the outer resolvent contains only those literals of D that occur to the left of the pivot literal l in the quantifier prefix.

Example 2.5.4. Given prefix $\Pi = \forall x_1 \exists y_1 \forall x_2 \exists y_2 y_3$ and clauses $C = (y_1 \vee \bar{x}_2 \vee y_3)$ and $D = (x_1 \vee \bar{y}_1 \vee y_3)$ the outer resolvent $\mathsf{OR}(\Pi, C, D, y_1) = (x_1 \vee \bar{x}_2 \vee y_3)$. The outer resolvent $\mathsf{OR}(\Pi, D, C, \bar{y}_1) = (x_1 \vee y_3)$.

Proof systems like resolution and Q-resolution derive clauses that—when added to the formula under consideration—preserve logical equivalence. The QRAT proof system is an interference-based proof system, i.e., the addition of newly derived clauses or the elimination of a redundant clause only preserves satisfiability equivalence meaning that the set of models/counter-models might change. As in RAT, the concept of reverse unit propagation is used to justify clause addition or deletion.

Definition 2.5.2. A propositional formula ψ *implies a clause* $C = (l_1 \vee \ldots \vee l_n)$ *via unit propagation* (denoted by $\psi \vdash_1 C$) iff unit propagation derives the empty clause $\bot$ on $\psi \wedge \bar{l}_1 \wedge \ldots \wedge \bar{l}_n$ where unit propagation refers to the application of the unit literal elimination rule until fixpoint.

A clause that is implied by unit propagation is sometimes also called an *asymmetric tautology* [HJL+15] or a clause that has the redundancy property of *reverse unit propagation* (RUP) [VG12b].

Example 2.5.5. Consider the propositional formula $(x \vee y) \wedge y$. The clause $(x \vee y)$ is an asymmetric tautology, because for $(y \wedge \bar{x} \wedge \bar{y})$ unit propagation derives a conflict.

Based on outer resolvents, we introduce the *quantified resolution asymmetric tautology* property (QRAT) next.

Definition 2.5.3. A clause C has QRAT on a literal $l \in C$ w.r.t. QBF $\Pi\psi$ iff $\psi \vdash_1 \mathsf{OR}(\Pi, C, D, l)$ for all $D \in \psi$ with $\bar{l} \in D$.

If a clause C has QRAT on literal l w.r.t. $\Pi\psi$, we call l a QRAT-literal of C for $\Pi\psi$. The following properties of QRAT-literals proven in [HSB17] finally allow us to formulate the rules of the QRAT proof system (see Figure 2.11). If a clause C is an asymmetric tautology in QBF $\Pi\psi$ or if C contains an existential QRAT-literal w.r.t. $\Pi\psi$ then C is redundant in $\Pi\psi$, i.e., $\Pi\psi$ and $\Pi(\psi \cup \{C\})$ are satisfiability equivalent. If a clause C contains a universal QRAT-literal for QBF $\Pi\psi$, then l is redundant in C, i.e., $\Pi(\psi \cup \{C\})$ and $\Pi(\psi \cup \{C \backslash \{l\}\})$ are satisfiability equivalent. Note that the clause and literal addition rules can introduce new variables that have to be included in the prefix. The position in the prefix depends on the usage of the new variable in later steps of the QRAT proof.

Rule	Rule Application	Side Conditions
Clause Elimination	$\Pi(\psi \cup \{C\}) \xrightarrow{\mathsf{QRATE}(C,l)} \Pi\psi$	• l is existential or $l = \bot^*$, $l \in C$ • l is a QRAT-literal of C for $\Pi\psi$
Clause Addition	$\Pi\psi \xrightarrow{\mathsf{QRATA}(C,l)} \Pi'(\psi \cup \{C\})$	• l is existential or $l = \bot^*$, $l \in C$ • l is a QRAT-literal of C for $\Pi'(\psi)$ • Π' is obtained from Π by adding the variables only occurring in C *if $l = \bot$ then C is an asymmetric tautology
Literal Elimination	$\Pi(\psi \cup \{C\}) \xrightarrow{\mathsf{QRATE}(C,l)} \Pi(\psi \cup \{C'\})$	• l is universal, $l \in C$, $C' = C \backslash \{l\}$ • l is a QRAT-literal of C for $\Pi\psi$
Literal Addition	$\Pi(\psi \cup \{C\}) \xrightarrow{\mathsf{QRATA}(C,l)} \Pi'(\psi \cup \{C'\})$	• l is universal, $C' = C \cup \{l\}$, $l \notin C$ • l is a QRAT-literal of C for $\Pi'\psi$ • Π' is obtained from Π by adding the variable of l if it is new
Extended Universal Reduction	$\Pi(\psi \cup \{C\}) \xrightarrow{\mathsf{EUR}(C,l)} \Pi(\psi \cup \{C'\})$	• l is universal, $l \in C$, $C' = C \backslash \{l\}$ • for all $k \in C$ with $k >_\Pi l$, l and k are independent w.r.t. the resolution-path dependency scheme

Figure 2.11. Rules of the QRAT Proof System

Besides the clause addition, clause elimination, literal addition, and literal elimination rules based on the QRAT property, there is a generalized version of universal reduction included that is based on resolution-path dependency schemes (see also Section 2.3.2).

Given a QBF ϕ_0, a derivation in the QRAT proof system is a sequence $M_1, \ldots, M_n$ where M_i is one of the rules shown in Figure 2.11 applied on QBF ϕ_{i-1} to obtain QBF ϕ_i. Hence, starting from a given QBF $\phi = \phi_0$, a QRAT derivation $M_1, \ldots, M_n$ produces new, equi-satisfiable formulas $\phi_1, \ldots, \phi_n$. If ϕ_n contains the empty clause, then the derivation is a *proof of unsatisfiability*. If the matrix of ϕ_n is empty, then the derivation is a *proof of satisfiability*.

Example 2.5.6. Consider the following true QBF $\Pi\psi$ such that $\Pi = \forall x \exists y_1 y_2$ and $\psi = ((x \vee y_1) \wedge (\bar{x} \vee y_2) \wedge (\bar{y}_1 \vee \bar{y}_2))$. It has the following QRAT proof of satisfiability:

$\Pi\psi$	$\xrightarrow{\mathsf{QRATA}((\bar{y}_1 \vee \bar{x}), \bar{y}_1)}$	$\Pi(\psi \cup \{(\bar{y}_1 \vee \bar{x})\})$	$(\bar{y}_1 \vee \bar{x})$ is blocked on $\bar{y}_1$
	$\xrightarrow{\mathsf{QRATE}((y_2 \vee \bar{x}), y_2)}$	$\Pi((x \vee y_1) \wedge (\bar{y}_1 \vee \bar{y}_2) \wedge (\bar{y}_1 \vee \bar{x}))$	$\mathsf{OR}(\Pi, (y_2 \vee \bar{x}), (\bar{y}_1 \vee \bar{y}_2), y_2)$ has RUP
	$\xrightarrow{\mathsf{QRATE}((\bar{y}_2 \vee \bar{y}_1), \bar{y}_2)}$	$\Pi((x \vee y_1) \wedge (\bar{y}_1 \vee \bar{x}))$	$(\bar{y}_2 \vee \bar{y}_1)$ is blocked on $\bar{y}_2$
	$\xrightarrow{\mathsf{QRATE}((\bar{y}_1 \vee \bar{x}), \bar{y}_1)}$	$\Pi((x \vee y_1))$	$(\bar{y}_1 \vee \bar{x})$ is blocked on $\bar{y}_1$
	$\xrightarrow{\mathsf{QRATE}((y_1 \vee x), y_1)}$	$\Pi(\top)$	$(y_1 \vee x)$ is blocked on y_1

Note that for a blocked clause with blocking literal l, l is also a QRAT-literal.

Example 2.5.7. Consider the following false QBF $\Pi\psi$ with $\Pi = \forall x \exists y_1 y_2$ and $\psi = ((x \vee y_1) \wedge (x \vee y_2) \wedge (\bar{y}_1 \vee \bar{y}_2))$. It has the following QRAT proof:

$\Pi\psi$	$\xrightarrow{\mathsf{QRATA}((\bar{y}_1), \bar{y}_1)}$	$\Pi(\psi \cup \{(\bar{y}_1)\})$	$\mathsf{OR}(\Pi, \bar{y}_1, (x \vee y_1), \bar{y}_1)$ has RUP
	$\xrightarrow{\mathsf{QRATE}((x \vee y_1), x)}$	$\Pi((y_1) \wedge (x \vee y_2) \vee (\bar{y}_1) \wedge (\bar{y}_1 \vee \bar{y}_2))$	$(x \vee y_1)$ is blocked on x
	$\xrightarrow{\mathsf{QRATE}(\bot, \bot)}$	$\Pi(\bot \wedge \ldots)$	$(\bot)$ has RUP

2.5.3. QRAT and Its Extension QRAT$^+$ for Preprocessing

Heule et al. [HSB17] showed how to express preprocessing techniques in terms of QRAT rules, implemented QRAT proof generation in the preprocessor Bloqqer and provided a proof checking tool. In many cases, the proof generation is straightforward. For example, a clause C that contains a pure literal l obviously has QRAT on l. Hence, if l is existential, QRAT witnesses that C may be safely removed. If l is universal then QRAT witnesses that l may be safely removed from C. The argument generalizes to blocked clauses and blocked literals. Also tautologies and subsumed clauses have the QRAT property. For other rules the translation is more involved, because multiple different QRAT rule applications are necessary. In the following, we show an example of a QRAT proof for the restricted form universal expansion as used in preprocessing, i.e., the expanded universal variable occurs in the innermost universal quantifier block. For the details we refer to [HSB17].

Example 2.5.8. In Example 2.5.2 we have seen that for QBF $\Pi\psi$ with $\Pi = \forall x_1 x_2 \exists y_1$ and $\psi = ((y_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{y}_1))$ universally expanding x_1 results in the formula $\forall x_2 \exists y_1 y_1'((y_1 \vee x_2) \wedge (\bar{y}_1) \wedge (y_1' \vee x_2))$. In the following, we certify this rewriting of $\Pi\psi$ with the following QRAT proof.

$\Pi\psi$	$\xrightarrow{\mathsf{QRATA}((y_1 \vee x_2 \vee \bar{x}_1), \bot)}$	$\Pi((y_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{y}_1) \wedge (y_1 \vee x_2 \vee \bar{x}_1))$	(1)
	$\xrightarrow{\mathsf{QRATA}((\bar{y}_1' \vee x_1 \vee y_1), \bar{y}_1')}$	$\Pi'((y_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{y}_1) \wedge (y_1 \vee x_2 \vee \bar{x}_1) \wedge (\bar{y}_1' \vee x_1 \vee y_1))$	(2)
	$\xrightarrow{\mathsf{QRATA}((y_1' \vee x_1 \vee \bar{y}_1), y_1')}$	$\Pi'((y_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{y}_1) \wedge (y_1 \vee x_2 \vee \bar{x}_1) \wedge (\bar{y}_1' \vee x_1 \vee y_1) \wedge (y_1' \vee x_1 \vee \bar{y}_1))$	(3)
	$\xrightarrow{\mathsf{QRATA}((y_1' \vee x_2 \vee x_1), y_1')}$	$\Pi'((y_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{y}_1) \wedge (y_1 \vee x_2 \vee \bar{x}_1) \wedge (\bar{y}_1' \vee x_1 \vee y_1) \wedge (y_1' \vee x_1 \vee \bar{y}_1) \wedge (y_1' \vee x_2 \vee x_1))$	(4)
	$\xrightarrow{\mathsf{QRATE}((y_1 \vee x_2), \bot)}$	$\Pi'((\bar{x}_1 \vee \bar{y}_1) \wedge (y_1 \vee x_2 \vee \bar{x}_1) \wedge (\bar{y}_1' \vee x_1 \vee y_1) \wedge (y_1' \vee x_1 \vee \bar{y}_1) \wedge (y_1' \vee x_2 \vee x_1))$	(5)
	$\xrightarrow{\mathsf{QRATE}((\bar{y}_1' \vee x_1 \vee y_1), y_1)}$	$\Pi'((\bar{x}_1 \vee \bar{y}_1) \wedge (y_1 \vee x_2 \vee \bar{x}_1) \wedge (y_1' \vee x_1 \vee \bar{y}_1) \wedge (y_1' \vee x_2 \vee x_1))$	(6)
	$\xrightarrow{\mathsf{QRATE}((y_1' \vee x_1 \vee \bar{y}_1), \bar{y}_1)}$	$\Pi'((\bar{x}_1 \vee \bar{y}_1) \wedge (y_1 \vee x_2 \vee \bar{x}_1) \wedge (y_1' \vee x_2 \vee x_1))$	(7)
	$\xrightarrow{\mathsf{EUR}^*}$	$\forall x_2 \exists y_1 y_1'((\bar{y}_1) \wedge (y_1 \vee x_2) \wedge (y_1' \vee x_2))$	(8)

In Step (1), the subsumed clause $(y_1 \vee x_2 \vee \bar{x}_1)$ is introduced. This is possible, because it has RUP. In Step (2) the clause $(\bar{y}'_1 \vee x_1 \vee y_1)$ with the fresh variable y'_1 variable is introduced. Obviously, $\bar{y}'_1$ is a QRAT literal for prefix $\Pi' = \Pi\exists y'_1$. The clause introduced in Step (3) is needed for renaming occurrences of y_1 to y'_1 as done in Step (4). In fact, the clauses introduced in Step (2) and Step (3) form a conditional equivalence between y_1 and y'_1. In Step (6) and Step (7) the clauses of the conditional equivalence are removed again, because no further renaming is necessary. The removal is possible, because y_1 and $\bar{y}_1$ are QRAT literals. Finally, all occurrences of x_1 are removed by the multiple applications of EUR (indicated by EUR*) and we obtain indeed the formula $\forall x_2 \exists y_1 y'_1((y_1 \vee x_2) \wedge (\bar{y}_1) \wedge (y'_1 \vee x_2))$.

Originally, QRAT was developed to provide a uniform way to certify the result of a preprocessor. As the QRAT proof system provides sound rules to add, remove and modify clauses, the QRAT rules themselves can be used within a preprocessor to modify a formula. For this purpose, even a generalization of QRAT called QRAT$^+$ was developed [LE18]. The version of QRAT as introduced above uses propositional unit propagation for testing the QRAT property (see Def 2.5.3). Here the quantification of the variables is not taken into account. QRAT$^+$ uses a QBF-specific variant of unit propagation which also includes universal reduction (i.e, QBCP by Definition 4). So certain clauses can be shortened during unit propagation, resulting in a more powerful redundancy property.

For the purpose of constructing refutations, it was shown that the QRAT and QRAT$^+$ proof systems are equivalent, i.e., the systems mutually p-simulate each other [CC19]. For preprocessing, however, QRAT$^+$ has a practical advantage over QRAT [LE19]. Due to the more powerful redundancy property in QRAT$^+$, for example, it can remove particular redundant clauses C in a *single* application of a rule. The same clause C can also be removed by QRAT, but only by applying a *sequence* of rules. Hence, in practice the search for redundant clauses in preprocessing based on QRAT$^+$ is simpler because it is not necessary to search for a sequence of rules that justify the redundancy of a clause.

2.6. Extraction of Winning Strategies from Proofs

Only few solvers explicitly generate winning strategies while evaluating a formula. For example the incremental determinization approach [RTRS18] generalizes the ideas behind CDCL (see Chapter **??**) from searching for satisfying assignments to searching for Skolem functions. The solver QFun [Jan18b] uses strategies instead of assignments for generating expanded formulas (see also Section 2.4). Most solvers, however, generate winning strategies only implicitly. For the solving approaches discussed before, proofs serve not only as witnesses for satisfiability/unsatisfiability, but they provide the necessary information for extracting a winning strategy, i.e., a model for a true QBF or a countermodel for a false QBF. We distinguish between static function-extraction approaches (offline approaches) and dynamic round-based approaches (online approaches). Both are shortly discussed in the following.

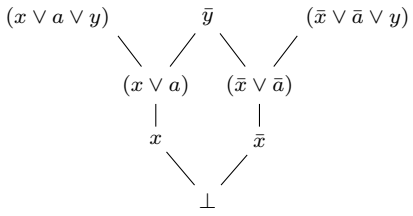

Figure 2.12. Q-Resolution refutation of QBF $\exists x \forall a \exists y((x \vee a \vee y) \wedge (\bar{x} \vee \bar{a} \vee y) \wedge \bar{y})$

2.6.1. Function-Extraction Approaches

Balabanov and Jiang [BJ11, BJ12] presented an approach that allows for the extraction of Herbrand functions from clause Q-resolution proofs, and dually for the extraction of Skolem functions from cube Q-resolution proofs. Given a proof, the computation of such a function is linear in the size of the proof, allowing to use the simple representation format of decision lists.

Definition 2.6.1. A decision list $D = (t_1, v_1), \ldots, (t_n, v_n), (\top, 0)$ is a finite sequence of pairs where t_i is a conjunction of literals and $v_i \in \{0, 1\}$. Let σ be an assignment over the variables occurring in a decision list D. Then D evaluates to v_i under σ if i is the least index such that t_i is true under σ.

Now let ϕ be a false QBF refuted by a Q-resolution proof π as described above. As ϕ is false, there is a winning strategy for the universal player. By traversing the proof in topological order, for each universal variable u, a decision list D_u is constructed that encodes such a winning strategy. Let C_i be a clause occurring in π that has been derived by removing the universal literal l from the previously derived clause C_k by the application of the universal reduction rule. If $l = u$, then the pair $(\bar{C}_i, 0)$ is appended to D_u. If $l = \bar{u}$, then the pair $(\bar{C}_i, 1)$ is appended to D_u. After the full proof has been traversed, all decision lists are closed with a pair $(\top, 0)$.

Example 2.6.1. Consider the false QBF $\exists x \forall a \exists y((x \vee a \vee y) \wedge (\bar{x} \vee \bar{a} \vee y) \wedge \bar{y})$. A possible resolution proof is shown in Figure 2.12. First, clause $(x \vee a)$ is derived by resolving the first and the third clause of the formula. Applying universal reduction results in the clause x. Further, $(\bar{x} \wedge \bar{a})$ is then derived by resolving the second and the third clause. Applying universal reduction results in the clause $\bar{x}$. Another application of the resolution rule on x and $\bar{x}$ derives the empty clause.

The proof contains two applications of the universal reduction rule from which the decision list $(\bar{x}, 0), (x, 1), (\top, 0)$ can be extracted. This decision list represents the Herbrand function $f_a(x) = x$ which is indeed a counter-model of the given formula.

This form of function extraction is also possible for long-distance Q-resolution proofs [BJJW15]. Also from QRAT proofs of satisfiability Skolem functions can be extracted by traversing the proof for constructing conditional expressions [HSB14, HSB17]. In contrast to decision lists, the conditions are not only conjunctions of

literals, but more complicated formulas involving all outer resolvents of the QRAT literal of each clause that is deleted. This approach allows also the extraction of incomplete Skolem functions in case the preprocessor cannot solve a given formula. If a complete solver can solve the preprocessed formula, the Skolem functions of the preprocessed formula can be combined with the incomplete Skolem functions for obtaining the Skolem functions of the original formula [FHSB17]. For QRAT proofs of unsatisfiability, it is only known for a restricted version of the proof system how to extract Herbrand functions [CC19].

2.6.2. Round-Based Approaches

An alternative approach directly implements the game semantics of QBFs. The quantifier prefix is processed from left to right. First, an arbitrary assignment is chosen by the existential player for the outermost existential variables. Based on this assignment, the proof is rewritten to a smaller refutation proof which also provides the information how the universal player has to assign the variables of the next universal quantifier block in order to win the game. Then it is again the existential player's turn to assign the variables of the next existential quantifier block. This assignment is again used to rewrite the proof, yielding the values of the next universal variables and so on. This approach works very similar for Q-resolution proofs and ∀Exp+Res proofs. The main difference is in finding the values for the universal variables.

Q-Resolution. Given an assignment of the outermost existential variables, Goultiaeva et al. [GVGB11] presented a set of rules that reduces a Q-resolution proof under this assignment such that the result is again a Q-resolution proof. The values that the outermost universal variables have in a winning strategy follow from the clause C in the reduced proof consisting only of universal literals—from such a clause the empty clause is derived by universal reduction. The polarity of the literal occurrences in C determines the value in the winning strategy. If a variable does not occur in C, or the conflict is derived by the application of the resolution rule, the value may be chosen freely. The approach works also for long-distance Q-resolution proofs [ELW13] and for proofs [PSS16, PSS19c] that involve universal reduction steps based on the reflexive resolution path dependency scheme.

Example 2.6.2. Consider the QBF $\exists x \forall a \exists y((x \vee a \vee y) \wedge (\bar{x} \vee \bar{a} \vee y) \wedge \bar{y})$. If x is set to true, the approach presented in [GVGB11] simplifies the Q-resolution proof of the original formula to a Q-resolution proof for the formula $\forall a \exists y((\bar{a} \vee y) \wedge \bar{y})$ that first resolves the two clauses and obtains the clause $\bar{a}$ from which the empty clause is derived with universal reduction. Clause $\bar{a}$ contains only universal literals, therefore, it is used to determine the value of a. Setting a to true results in the empty clause. This is the only choice for a winning move for the universal player under the assignment that sets x to true. If x had been set to false, the approach would force a to be set to false as well. The rewritings of the proofs are shown in Figure 2.13.

∀Exp+Res. After the outermost existential variables have been assigned a truth value, the proof reduced under this assignment contains only annotations of the

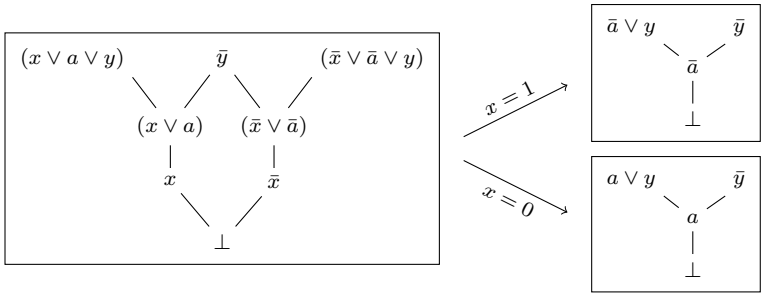

Figure 2.13. Setting the value of x in Q-resolution refutation of QBF $\exists x \forall a \exists y((x \vee a \vee y) \wedge (\bar{x} \vee \bar{a} \vee y) \wedge \bar{y})$

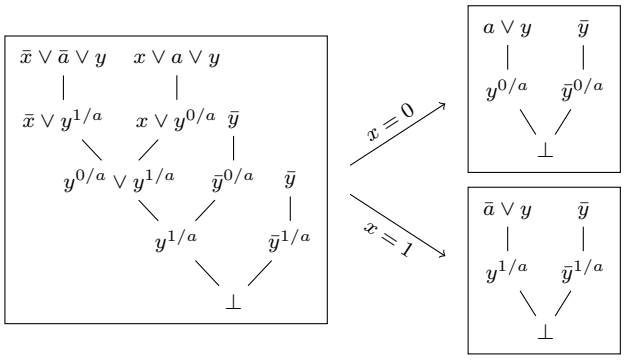

Figure 2.14. Setting the value of x in $\forall$Exp+Res refutation of QBF $\exists x \forall a \exists y((x \vee a \vee y) \wedge (\bar{x} \vee \bar{a} \vee y) \wedge \bar{y})$

variables occurring in the outermost universal quantifier block in one polarity, i.e., those annotations are unopposed [BCJ14]. The polarities indicate how the respective universal variables have to be set in a winning strategy. Then the annotations of the outermost universal variables are removed, and the variables in the next existential block are assigned a value.

Example 2.6.3. Consider the QBF $\exists x \forall a \exists y((x \vee a \vee y) \wedge (\bar{x} \vee \bar{a} \vee y) \wedge \bar{y})$. Instantiation results in the propositional formula $(x \vee y^{0/a}) \wedge (\bar{x} \vee y^{1/a}) \wedge \bar{y}^{0/a} \wedge \bar{y}^{1/a})$ that is refuted in three steps, e.g., by resolving the first and the second clause resulting in the clause $(y^{0/a} \vee y^{1/a})$. Two further applications of the resolution rule result in the derivation of the empty clause. If x is set to true, the instantiated formula simplifies to $(y^{1/a}) \wedge \bar{y}^{0/a} \wedge \bar{y}^{1/a}$. The resolution proof simplifies respectively, and contains only unopposed annotations with a. Hence, a has to be set to true. The approach works similar for the case that x is set to false. The rewritings of the proofs are shown in Figure 2.14.

$$\frac{}{\left\{l^{[\tau]} \mid l \in C, l \text{ is an existential literal}\right\}} \text{ (Axiom)}$$

C is a non-tautological clause from the original matrix. τ is a partial 0/1 assignment to universal variables, which has all universal variables from C as its domain and falsifies all universal literals in C, i.e., $\tau = \{0/u \mid u \text{ is universal in } C\}$, where the notation $0/u$ for literals u is shorthand for $0/x$ if $u = x$ and $1/x$ if $u = \bar{x}$.

$$\frac{\{x^\tau\} \cup C_1 \qquad \{\bar{x}^\tau\} \cup C_2}{C_1 \cup C_2} \text{ (Resolution)} \qquad \frac{C}{\mathsf{inst}(\tau, C)} \text{ (Instantiation)}$$

C_1, C_2, and C are clauses of annotated variables.
τ is a partial 0/1 assignment to universal variables with $\mathsf{rng}(\tau) \subseteq \{0, 1\}$.

Figure 2.15. The rules of IR-calc [BCJ14].

2.7. Connections Between Proof Systems

In this section we turn towards a proof-complexity analysis of the QBF proof systems defined earlier. Proof complexity offers two main threads towards this goal: a relative comparison of proof calculi via simulations and absolute (mostly exponential) lower bounds for the proof size of specific families of formulas in the systems. We survey and sketch some results for both these threads. Both approaches provide relevant information for solving as proof size implies lower bounds on solving time in the corresponding solvers.

2.7.1. Enhanced Expansion Calculi

Before we start the comparison between the different calculi, we introduce two more proof systems that combine the CDCL and expansion calculi from Sections 2.3 and 2.4. Both these systems use the idea of expanding universal variables as in ∀Exp+Res and keep track of the expansions in annotations, but incorporate additional features of CDCL-systems as in Section 2.3.

In ∀Exp+Res each axiom clause is immediately annotated with a *fixed, complete* assignment to the universal variables. The proof then proceeds exactly as a propositional resolution proof, with clauses in fully annotated variables. In short, ∀Exp+Res is propositional resolution on the conjuncts of a PCNF's complete expansion.

IR-calc, defined in [BCJ14], improves on this approach by working instead with *partial* assignments. In addition to resolution, the calculus is equipped with an *instantiation* rule by which partial annotations are grown throughout the course of the proof. To facilitate instantiation, the $\circ$ operator describes how two partial assignments τ and σ are combined: $\tau \circ \sigma := \tau \cup \{l \in \sigma \mid \bar{l} \notin \tau\}$.

The rules of IR-calc are given in Figure 2.15. We provide a description of each of the IR-calc rules and illustrate them with some simple examples.

Axiom clauses are introduced into the proof, or *downloaded*, by selecting a clause C from the matrix and applying the *download assignment* to the existential literals. By design, the download assignment σ for C is the smallest partial

Axiom and instantiation rules as in IR-calc in Figure 2.15.

$$\frac{\{x^{\tau\cup\xi}\}\cup C_1 \qquad \{\bar{x}^{\tau\cup\sigma}\}\cup C_2}{\mathsf{inst}(\sigma, C_1)\cup\mathsf{inst}(\xi, C_2)}\ \text{(Resolution)}$$

C_1 and C_2 are clauses of annotated variables. τ is a partial assignment to universal variables (i.e., $\mathsf{rng}(\tau) = \{0, 1\}$) and ξ, σ are extended partial assignments that assign universal variables to $\{0, 1, *\}$. $\mathsf{dom}(\tau)$, $\mathsf{dom}(\xi)$, and $\mathsf{dom}(\sigma)$ are mutually disjoint.

$$\frac{C\cup\{l^{\mu}\}\cup\{l^{\sigma}\}}{C\cup\{l^{\xi}\}}\ \text{(Merging)}$$

C is a clause of annotated variables. l is a literal and μ and σ are extended assignments with $\mathsf{dom}(\mu) = \mathsf{dom}(\sigma)$. $\xi = \{c/u \mid c/u \in \mu, c/u \in \sigma\}\cup\{*/u \mid c/u \in \mu, d/u \in \sigma, c \neq d\}$

Figure 2.16. The rules of IRM-calc [BCJ14]

assignment that falsifies every universal literal in C. Represented as a set of literals, then, $\sigma = \{\bar{l} \mid l \text{ is universal in } C\}$. When applying the download assignment, existentials are annotated only with universals *to their left* (i.e., those on which they depend). Consider the PCNF with prefix $\forall u\exists x\forall v\exists y$ and matrix clauses $\{u, \bar{x}, \bar{v}, y\}$ and $\{\bar{u}, x, \bar{y}\}$. Downloading the two matrix clauses gives rise to axioms $\{\bar{x}^{\bar{u}}, y^{\bar{u}v}\}$ and $\{x^{u}, \bar{y}^{u}\}$.

Instantiation applies a single partial assignment τ to all the annotations in a clause. As in the axiom rule, universals to the right of a variable are omitted from its annotation. Formally, for a partial assignment τ and an annotated clause C, the function $\mathsf{inst}(\tau, C)$ returns the annotated clause $\{l^{[\sigma\circ\tau]} \mid l^{\sigma} \in C\}$. For example, given the prefix $\forall u_1u_2\exists x\forall u_3u_4\exists y$ and a clause $\{x^{u_1}, y^{\bar{u}_4}\}$, instantiation by $\tau = \bar{u}_2u_3u_4$ derives $\{x^{u_1\bar{u}_2}, y^{\bar{u}_2u_3\bar{u}_4}\}$. Note that u_3 and u_4, which are right of x, do not appear in that variable's annotation after the instantiation. Also note that u_4 does not appear in the annotation to y, which is already annotated with the negated literal $\bar{u}_4$ before the instantiation takes place (see the earlier definition of $\circ$).

Resolution in IR-calc is identical to propositional resolution. We emphasize that annotations are labelling distinct variables (e.g., x^u and x^v are different variables), so that a resolution step is only valid if the annotations of the pivot literals match.

The calculus IRM-calc additionally incorporates long-distance steps in the spirit of LD-Q-Res. The rules of the calculus are depicted in Figure 2.16.

2.7.2. The Simulation Order of QBF Resolution Calculi

We now provide an overview of the simulation order of the resolution systems defined earlier. A pictorial representation of the simulation order is given in Figure 2.17.

In Figure 2.17, solid lines denote simulations (cf. Section 2.2) where the simulating system is placed at the higher position, e.g. ∀Exp+Res simulates tree-like Q-Res by the simulation labeled with 1. Furthermore, all simulations are strict, e.g. tree-like Q-Res does not simulate ∀Exp+Res as witnessed by a formula family

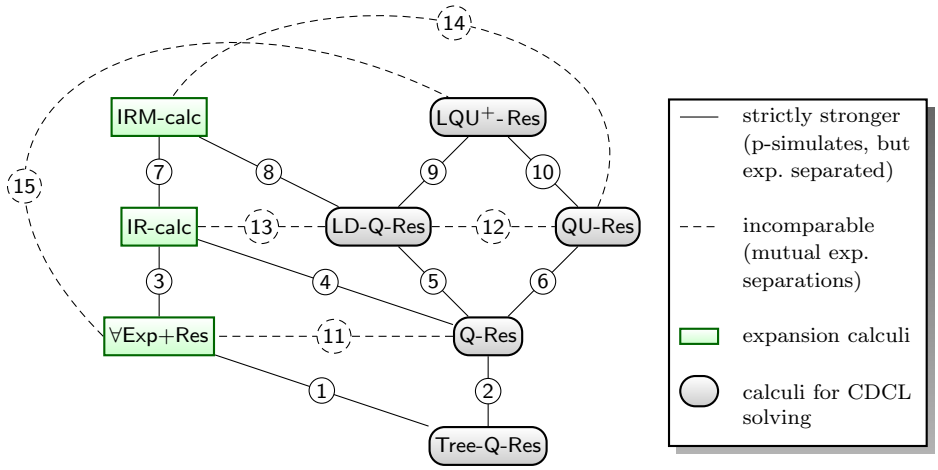

Figure 2.17. The simulation order of QBF resolution systems

with short proofs in ∀Exp+Res, but requiring exponential-size proofs in tree-like Q-Res. Dashed lines indicate incomparability results, e.g. ∀Exp+Res and Q-Res are incomparable as per line 11. This means that there are formulas with polynomial-size proofs in ∀Exp+Res, but requiring exponential-size proofs in Q-Res, and vice versa.

We highlight that our information with respect to the simulation order of the eight depicted systems is complete, i.e., we either have a simulation result or an incomparability result. Missing lines follow by transitivity. We also point out that all the separations (either by strict simulations or incomparability) are exponential separations. References to the original results as well as pointers to the separating formulas are placed in Table 2.1 accompanying Figure 2.17.

Most of the simulation results follow straightforwardly from the definitions. This applies to the two simulations 3 and 7 between the expansion calculi on the left as well as to all simulations between the CDCL-type calculi on the right. Nontrivial simulations arise between conceptually different proof systems, which applies to simulations 1, 4 and 8, details of which can be found in [BCJ14].

2.7.3. Lower Bounds and Separations

We now turn to the separations of the proof systems in Figure 2.17. This involves constructing specific formula families that provide the separations, e.g. for the separation of LD-Q-Res from Q-Res we need formulas ϕ_n that have polynomial-size proofs in LD-Q-Res, but all Q-Res proofs for ϕ_n are of exponential size. For this separation we have multiple canonical choices for ϕ_n as either the equality formulas EQ_n from [BBH19], the formulas $KBKF_n$ of Kleine Büning et al. [KBKF95], or the parity formulas [BCJ15].

In most instances, giving the upper bounds, i.e., constructing short proofs in the stronger system, turns out to be fairly easy, whereas establishing the lower

Table 2.1. Formulas and references establishing simulations and separations. Numbers in the first column refer to the labels in Figure 2.17.

	Simulation	Separation
1	[JM15, BCMS18a]	[JM15], $QParity_n$ [BCCM19, BCJ15]
2	by def.	propositional formulas [BEGJ00]
3	[BCJ14]	[JM15, BCJ15]
4	[BCJ14]	$QParity_n$ [BCJ15]
5	by def.	$KBKF_n$ [ELW13], EQ_n [BBM19, BBH19], $QParity_n$ [BCJ15, Che17b]
6	by def.	$KBKF_n$ [VG12a]
7	by def.	$KBKF_n$ [BCJ15, BB18], EQ_n [BBM19, BB18]
8	[BCJ14]	version of $QParity_n$ [BCJ15]
9	by def.	version of $KBKF_n$ [BWJ14]
10	by def.	version of $KBKF_n$ [BWJ14], EQ_n [BBH19]
	Incomparable	
11	[JM15], $QParity_n$ [BCJ15]	
12	versions of $KBKF_n$ [BWJ14]	
13	version of $QParity_n$ [BCJ15], $KBKF_n$ [BCJ15, BB18, ELW13]	
14	$QParity_n$ [BCJ15], version of $KBKF_n$ [BCJ19]	
15	version of $QParity_n$ [BCJ15], $KBKF_n$ [BCJ15]	

bounds, i.e., showing the absence of short proofs in the weaker system, presents the main technical challenge. This is in line with the general picture in computational complexity, where the task of showing lower bounds turns out to be the most intricate.

Arguably, what is even more important than the actual lower bounds, is to devise generally applicable *lower-bound techniques*. For propositional resolution we have a number of such techniques (cf. [Seg07, Bus12]) and it is illuminating to review their applicability in the QBF context.

The (in)effectiveness of propositional techniques in QBF. The most widely used classical technique is the seminal *size-width technique* of [BSW01], which shows lower bounds for size via lower bounds for the width in resolution. However, this technique drastically fails even in the base system Q-Res [BCMS18a] and also does not seem applicable more widely [CB18].

Feasible interpolation is another widely-used propositional technique that connects circuit complexity to proof complexity. Feasible interpolation applies to a number of classical proof systems [Kra97, Pud97], and in [BCMS17] it was demonstrated that this technique lifts to all QBF resolution systems depicted in Figure 2.17. However, the applicability of feasible interpolation is restricted by the fact that it imports lower bounds for monotone Boolean circuits (of which we only have relatively few [Juk12]).

A further general approach is through *Prover-Delayer games*. While this is effective in both propositional resolution [PI00, BGL13] as well as in QBF resolution [BCS19] (and can even yield optimal bounds [BGL13, BGL10, BCS19]), it only applies to the weak tree-like versions of these systems, where proofs are always in form of a tree, i.e., if derived formulas are needed multiple times in the proof they need to be rederived.

There is another short-coming of using propositional ideas in QBF. On exis-

tentially quantified formulas, all the QBF resolution systems coincide with classical proposition resolution. As such, all formulas hard for propositional resolution will be hard for any of the QBF resolution systems shown in Figure 2.17, e.g. the existentially quantified pigeonhole formulas are hard for all the QBF resolution calculi. Neither is this the phenomenon one wants to study in QBF proof complexity (cf. [Che17a, BHP17] for discussions), nor can such formulas provide any of the separations in Figure 2.17 (except for Q-Res from tree-like Q-Res). The same applies to the lifted interpolation technique: while it applies to new classes of QBFs (not just existentially quantified formulas) it also does not provide any separations, as lower bounds via this technique will hold for all QBF resolution calculi simultaneously.

Summarising the current impact of propositional techniques in QBF it is fair to say that, at present, *ideas from the propositional world have limited impact to the QBF framework*, and *genuinely new ideas are needed* to cope with the more complex setting of quantifiers.

2.7.4. Genuine QBF Lower Bound Techniques

We will now turn to ideas for lower bounds specific to QBF and along the way explain some of the separations in Figure 2.17.

The principal approach to lower bounds in QBF is via *strategy extraction*, which originates from the game semantics of QBF. We recall the *two-player game* between an existential and a universal player, who in turn choose 0/1 values for the variables in the QBF in the order of the quantifier prefix, starting from the leftmost quantifier. The universal player wins as soon as one of the clauses in the QBF matrix gets falsified, otherwise the existential player wins. A fully quantified QBF is false if and only if the universal player has a winning strategy; likewise it is true if and only if the existential player has a winning strategy.

Strategy extraction computes a strategy from a proof of the formula. In particular, from a refutation of a false QBF in a QBF resolution system, a winning strategy for the universal player can be efficiently extracted [BCJ14, BJJW15]. This is practically important, as strategies witness the answers of QBF solvers [BJ12], but it can also be exploited for lower-bound techniques [BCJ15, BBC16, BBH19].

The basic idea of this method is both conceptually simple and elegant: If we know that a family φ_n of false QBFs requires 'complex' winning strategies, then proofs of φ_n must be large in all proof systems with 'feasible' strategy extraction.

Lower bounds from circuit complexity. The qualification 'complex' can be given different specific definitions. The first is to consider the computational complexity required to compute the witnessing functions. For Q-Res and QU-Res we know that winning strategies can be computed by decision lists (cf. Section 2.6). It is easy to verify that decision lists can be turned into AC^0 circuits (cf. [BCJ15]). Hence, winning strategies can be computed from Q-Res refutations in AC^0.

To turn this into a lower bound, we need to construct false QBFs where all winning strategies are hard for bounded-depth circuits. As an example, consider

the parity formulas $QParity_n$ from [BCJ15]

$$\exists x_1, \ldots, x_n \, \forall z \, \exists t_2, \ldots, t_n \; \mathrm{xor}(x_1, x_2, t_2) \cup \bigcup\nolimits_{i=3}^{n} \mathrm{xor}(t_{i-1}, x_i, t_i) \cup \{z \vee t_n, \bar{z} \vee \bar{t}_n\}$$

where $\mathrm{xor}(o_1, o_2, o)$ defines o to be $o_1 \oplus o_2$ through the following set of clauses

$$\{\bar{o}_1 \vee \bar{o}_2 \vee \bar{o},\, o_1 \vee o_2 \vee \bar{o},\, \bar{o}_1 \vee o_2 \vee o,\, o_1 \vee \bar{o}_2 \vee o\}.$$

Informally, $QParity_n$ reasons about $x_1 \oplus \cdots \oplus x_n$, where the variables t_i encode the prefix sums $x_1 \oplus \cdots \oplus x_i$. Hence, t_n encodes $x_1 \oplus \cdots \oplus x_n$. Through the only universal variable z the formula expresses the obvious contradiction that $x_1 \oplus \cdots \oplus x_n$ is neither 0 nor 1. Crucially, the only strategy of the universal player to win on this false QBF is to play $z = x_1 \oplus \cdots \oplus x_n$.

A seminal result of [FSS84, Hås87] states that every non-uniform family of bounded-depth circuits computing the parity function is of exponential size. Combined with the fact that strategy extraction for Q-Res and also QU-Res is possible in AC^0, this results in an exponential lower bound for $QParity_n$ in Q-Res and QU-Res, first shown in [BCJ15].

In contrast, it turns out that the $QParity_n$ formulas admit short proofs both in ∀Exp+Res [BCJ15] and in LD-Q-Res [Che17b]. We sketch the construction of the linear-size refutations of $QParity_n$ in ∀Exp+Res. We first instantiate all clauses in both polarities of z, generating the clauses

$$\mathrm{xor}(x_1, x_2, t_2^{c/z}) \cup \bigcup_{i=3}^{n} \mathrm{xor}(t_{i-1}^{c/z}, x_i, t_i^{c/z}) \cup \{t_n^{c/z}\}$$

for $c = 0, 1$.

Inductively, for $i = 2, \ldots, n$ we now derive clauses representing $t_i^{0/z} \leftrightarrow t_i^{1/z}$. This yields a contradiction using the clauses $t_n^{0/z}$ and $\bar{t}_n^{1/z}$.

Therefore the *QParity* formulas give one part of the separation between ∀Exp+Res and Q-Res: formulas easy for ∀Exp+Res, but hard for Q-Res. Formulas easy in Q-Res, but hard in ∀Exp+Res are contained in [JM15]. These formulas use n universal variables, which all need to be expanded in both polarities 0/1 in order to obtain an unsatisfiable set of clauses. Hence, the expansion phase in all ∀Exp+Res refutations of the formula is of size 2^n. It is not difficult to construct such formulas that are also easy for Q-Res.

Lower bounds through cost. We now explore a second way of how to interpret the qualification of 'complex' in winning strategies. Following the semantic approach of [BBH19], we consider false QBFs where all winning strategies require many moves of the universal player. We will see that such formulas are hard for QU-Res and further QBF systems.

We measure the size of winning strategies (for a single block) by the cost of a formula.

Definition 2.7.1 ([BBH19])**.** The *cost* of a false QBF is the minimum, over all winning strategies, of the largest number of responses for a single universal block.

Strategies that require many responses of the universal player (in one block) are costly. This measure provides an absolute lower bound on the number of clauses in QU-Res refutations.

Theorem 2.7.1 ([BBH19])**.** *Let π be a QU-Res refutation of a QBF Φ. Then $|\pi| \geq \mathrm{cost}(\Phi)$.*

We consider the *equality formulas* from [BBH19] as an example. The QBF EQ_n is defined as

$$\exists x_1 \cdots x_n \forall u_1 \cdots u_n \exists t_1 \cdots t_n \left(\bigwedge_{i=1}^{n} (x_i \vee u_i \vee \bar{t}_i) \wedge (\bar{x}_i \vee \bar{u}_i \vee \bar{t}_i) \right) \wedge \left(\bigvee_{i=1}^{n} t_i \right).$$

The only winning strategy for these formulas is to play $u_i = x_i$ for $i = 1, \ldots, n$. There is only one universal block. Hence, the cost of EQ_n is 2^n and all QU-Res (and Q-Res) proofs of EQ_n are of exponential size.

In contrast, EQ_n are easily verified to have linear-size proofs in LD-Q-Res [BBM19], hence the formulas provide the separation between Q-Res and and LD-Q-Res (line 5 in Figure 2.17).

Another such separation is provided by the $KBKF_n$ formulas of Kleine Büning et al. [KBKF95], which appear widely in the QBF literature (for examples, see [Egl16, BCJ15, BWJ14, LES16]). Again these formulas can be shown to be hard for Q-Res via a cost argument [BBH19], while they are easy for QU-Res [VG12a] and LD-Q-Res [ELW13], thus in particular providing the separation between QU-Res and Q-Res. Modified versions of the $KBKF_n$ formulas also provide the separations of LQU$^+$-Res from LD-Q-Res and QU-Res [BWJ14].

A related notion of cost is also effective as a lower bound measure for expansion calculi, whereby the $KBKF_n$ formulas can be shown to be hard for IR-calc [BB18].

2.7.5. Relations to Further Calculi

We briefly mention some facts on the proof complexity of further approaches defined in earlier subsections: dependency schemes, symmetries, and QRAT. However, in comparison to the systems in Figure 2.17 for which we know their precise relations, we only have partial results on their relative complexity.

For *dependency schemes* (cf. Section 2.3.2), the EQ_n and $KBKF_n$ formulas give theoretical justification for their use: both formulas are hard for Q-Res, but become easy when using the resolution path dependency scheme [BB17]. In fact, even the standard dependency scheme cannot identify any useful dependencies for the EQ_n and $KBKF_n$ formulas, i.e., the formulas exponentially separate Q-Res with the resolution path dependency scheme from Q-Res with the standard dependency scheme.

In addition, the formulas $KBKF_n$ and $QParity_n$ have short proofs if Q-Res is extended by the *symmetry rule* (see Section 2.3.2). It is, however, easy to destroy the symmetries by introducing universal pure literals into those formulas such that no symmetries can be exploited. Hence, LQU$^+$-Res and Q-Res with the symmetry rule are incomparable.

For the QRAT proof system (see Section 2.5) it is known that it is strictly stronger than LD-Q-Res and QU-Res [KHS17]. Recently, it has been shown that QRAT polynomially simulates ∀Exp+Res. Nothing is known about the relationships to LQU$^+$-Res or to stronger expansion calculi.

2.7.6. Stronger QBF Proof Systems

We now turn briefly to the wider landscape of stronger QBF calculi beyond resolution. We have seen previously that propositional resolution can be turned into the QBF system QU-Res by simply adding the universal reduction rule. This procedure can be generalised [BBC16]: every line-based propositional proof system P (fulfilling some natural conditions [BBH19]) can be lifted to a system P+∀red, which is sound and complete for QBFs.

In this way, we obtain natural QBF versions of the geometric system *cutting planes* [BCMS18b], working with linear inequalities instead of clauses, of the algebraic system *polynomial calculus*, working with polynomials, and of the logical *Frege systems*, working with arbitrary finite sets of axioms and a set of propositional rules [BBC16].

We note that the two lower bound techniques via strategy extraction from Section 2.7.4 remain applicable to these much stronger QBF systems. However, when using circuit complexity as in the first approach, we need to import stronger (conditional) circuit lower bounds, i.e., for NC^1 in case of Frege and for P/poly in case of extended Frege [BBC16].

For the QBF Frege systems, we can even *characterise* their reasons for hardness: lower bounds either arise through a lower bound for propositional Frege or through a circuit lower bound (the precise condition is NC$^1 \not\subseteq$ PSPACE in case of Frege) [BP16]. Hence, QBF Frege systems naturally unite the hardest problem in propositional proof complexity (lower bounds for Frege) with the hardest task in circuit complexity (unconditional circuit lower bounds).

The second approach via cost is also effective for strong systems. However, here we need to include another measure: the *capacity* of proof lines. Informally, the capacity of a line in a proof counts how many responses can be at most extracted from it; and the capacity of a proof is defined as the maximum of the capacities of its lines. As an example, clauses have capacity one as the universal player has a unique strategy to falsify the clause. Hence, resolution proofs always have capacity one.

We recall that cost measures how many responses need to extracted from the proof and that strategy extraction is efficient by the round-based algorithm in Section 2.6. This leads to the *size-cost-capacity theorem* [BBH19], a general result for QBF proof systems of the form P+∀red, stating that for each QBF Φ and each P+∀red proof π of Φ

$$|\pi| \geq \frac{cost(\Phi)}{capacity(\pi)} \ .$$

By this theorem, the equality formulas are hard not only for QU-Res, but also in QBF cutting planes and polynomial calculus [BBH19]. Via this approach, we can also show the hardness of a large class of random QBFs in these calculi.

In contrast, QBF Frege admits short proofs for equality and these random formulas, as the capacity of Frege lines can become exponentially large.

Finally, we also mention *sequent calculi* for QBF [KP90, CM05]. In contrast to all the systems considered before they can also manipulate the quantifier prefix and thus are very strong systems, even stronger than QBF Frege [BP16]. However, there are also various restricted versions of intermediate strength [Egl12].

QBF sequent calculi and Frege systems enjoy a close connection to first-order theories of bounded arithmetic [KP90, CM05, BP16]. Via different translations many of the QBF resolution systems considered here are also closely related to first-order logic [Egl16, SLB12].

Acknowledgements

We thank Uwe Egly and Christoph Scholl for careful reading and helpful comments to prepare the final version of this chapter.

Bibliography

[AB02] Abdelwaheb Ayari and David A. Basin. QUBOS: Deciding Quantified Boolean Logic Using Propositional Satisfiability Solvers. In *FMCAD*, volume 2517 of *LNCS*, pages 187–201. Springer, 2002.

[BB16] Olaf Beyersdorff and Joshua Blinkhorn. Dependency Schemes in QBF Calculi: Semantics and Soundness. In *CP*, volume 9892 of *LNCS*, pages 96–112. Springer, 2016.

[BB17] Joshua Blinkhorn and Olaf Beyersdorff. Shortening QBF Proofs with Dependency Schemes. In *SAT*, volume 10491 of *LNCS*, pages 263–280. Springer, 2017.

[BB18] Olaf Beyersdorff and Joshua Blinkhorn. Genuine lower bounds for QBF expansion. In *STACS*, volume 96 of *LIPIcs*, pages 12:1–12:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.

[BBC16] Olaf Beyersdorff, Ilario Bonacina, and Leroy Chew. Lower Bounds: From Circuits to QBF Proof Systems. In *ITCS*, pages 249–260. ACM, 2016.

[BBH+18] 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.

[BBH19] Olaf Beyersdorff, Joshua Blinkhorn, and Luke Hinde. Size, cost, and capacity: A semantic technique for hard random QBFs. *Logical Methods in Computer Science*, 15(1), 2019.

[BBM19] Olaf Beyersdorff, Joshua Blinkhorn, and Meena Mahajan. Building strategies into QBF proofs. In *STACS*, LIPIcs, pages 14:1–14:18. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2019.

[BCCM19] Olaf Beyersdorff, Leroy Chew, Judith Clymo, and Meena Mahajan. Short proofs in QBF expansion. In *SAT*, volume 11628 of *LNCS*, pages 19–35, 2019.

[BCJ14] Olaf Beyersdorff, Leroy Chew, and Mikoláš Janota. On unification of QBF resolution-based calculi. In *MFCS, II*, pages 81–93, 2014.

[BCJ15] Olaf Beyersdorff, Leroy Chew, and Mikoláš Janota. Proof complexity of resolution-based QBF calculi. In *STACS*, volume 30 of *LIPIcs*, pages 76–89. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.

[BCJ19] Olaf Beyersdorff, Leroy Chew, and Mikolás Janota. New resolution-based QBF calculi and their proof complexity. *ACM Transactions on Computation Theory*, 11(4):26:1–26:42, 2019.

[BCMS17] Olaf Beyersdorff, Leroy Chew, Meena Mahajan, and Anil Shukla. Feasible interpolation for QBF resolution calculi. *Logical Methods in Computer Science*, 13, 2017.

[BCMS18a] Olaf Beyersdorff, Leroy Chew, Meena Mahajan, and Anil Shukla. Are short proofs narrow? QBF resolution is not so simple. *ACM Transactions on Computational Logic*, 19, 2018.

[BCMS18b] Olaf Beyersdorff, Leroy Chew, Meena Mahajan, and Anil Shukla. Understanding cutting planes for QBFs. *Inf. Comput.*, 262:141–161, 2018.

[BCS19] Olaf Beyersdorff, Leroy Chew, and Karteek Sreenivasaiah. A game characterisation of tree-like Q-Resolution size. *J. Comput. Syst. Sci.*, 104:82–101, 2019.

[BEGJ00] Maria Luisa Bonet, Juan Luis Esteban, Nicola Galesi, and Jan Johannsen. On the relative complexity of resolution refinements and cutting planes proof systems. *SIAM J. Comput.*, 30(5):1462–1484, 2000.

[Ben04] Marco Benedetti. Evaluating QBFs via symbolic skolemization. In *LPAR*, volume 3452 of *LNCS*, pages 285–300. Springer, 2004.

[Ben05] Marco Benedetti. Quantifier Trees for QBFs. In *SAT*, volume 3569 of *LNCS*, pages 378–385. Springer, 2005.

[BG01] Leo Bachmair and Harald Ganzinger. Resolution theorem proving. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning (in 2 volumes)*, pages 19–99. Elsevier and MIT Press, 2001.

[BGL10] Olaf Beyersdorff, Nicola Galesi, and Massimo Lauria. A lower bound for the pigeonhole principle in tree-like resolution by asymmetric prover-delayer games. *Inf. Process. Lett.*, 110(23):1074–1077, 2010.

[BGL13] Olaf Beyersdorff, Nicola Galesi, and Massimo Lauria. A characterization of tree-like resolution size. *Inf. Process. Lett.*, 113(18):666–671, 2013.

[BHP17] Olaf Beyersdorff, Luke Hinde, and Ján Pich. Reasons for hardness in QBF proof systems. In *FSTTCS*, pages 14:1–14:15, 2017.

[Bie04] Armin Biere. Resolve and expand. In *SAT (Selected Papers*, volume 3542 of *LNCS*, pages 59–70. Springer, 2004.

[BJ11] Valeriy Balabanov and Jie-Hong R. Jiang. Resolution Proofs and Skolem Functions in QBF Evaluation and Applications. In *CAV*, volume 6806 of *LNCS*, pages 149–164. Springer, 2011.

[BJ12] Valeriy Balabanov and Jie-Hong R. Jiang. Unified QBF Certification and its Applications. *Formal Methods in System Design*, 41(1):45–65, 2012.

[BJJW15] Valeriy Balabanov, Jie-Hong Roland Jiang, Mikoláš Janota, and Magdalena Widl. Efficient Extraction of QBF (Counter)models from Long-Distance Resolution Proofs. In *AAAI*, pages 3694–3701. AAAI Press, 2015.

[BJK15] Nikolaj Bjørner, Mikoláš Janota, and William Klieber. On Conflicts and Strategies in QBF. In *LPAR-Short Presentations*, volume 35 of *EPiC Series in Computing*, pages 28–41. EasyChair, 2015.

[BK07] Uwe Bubeck and Hans Kleine Büning. Bounded Universal Expansion for Preprocessing QBF. In *SAT*, volume 4501 of *LNCS*, pages 244–257. Springer, 2007.

[BKS14] Roderick Bloem, Robert Könighofer, and Martina Seidl. SAT-based synthesis methods for safety specs. In *VMCAI*, volume 8318 of *LNCS*, pages 1–20. Springer, 2014.

[BLS11] Armin Biere, Florian Lonsing, and Martina Seidl. Blocked Clause Elimination for QBF. In *CADE*, volume 6803 of *LNCS*, pages 101–115. Springer, 2011.

[BP16] Olaf Beyersdorff and Ján Pich. Understanding Gentzen and Frege systems for QBF. In *LICS*, pages 146–155, 2016.

[BSW01] Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow - resolution made simple. *J. of the ACM*, 48(2):149–169, 2001.

[Bus12] Samuel R. Buss. Towards NP-P via proof complexity and search. *Ann. Pure Appl. Logic*, 163(7):906–917, 2012.

[BWJ14] Valeriy Balabanov, Magdalena Widl, and Jie-Hong R. Jiang. QBF Resolution Systems and Their Proof Complexities. In *SAT*, volume 8561 of *LNCS*, pages 154–169. Springer, 2014.

[CB18] Judith Clymo and Olaf Beyersdorff. Relating size and width in variants of Q-resolution. *Inf. Process. Lett.*, 138:1–6, 2018.

[CC19] Leroy Chew and Judith Clymo. The Equivalences of Refutational QRAT. In *SAT*, volume 11628 of *LNCS*, pages 100–116. Springer, 2019.

[CGJ+03] Edmund M. Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. Counterexample-guided abstraction refinement for symbolic model checking. *J. ACM*, 50(5):752–794, 2003.

[CGS98] Marco Cadoli, Andrea Giovanardi, and Marco Schaerf. An Algorithm to Evaluate Quantified Boolean Formulae. In *AAAI/IAAI*, pages 262–267. AAAI Press / The MIT Press, 1998.

[Che17a] Hubie Chen. Proof complexity modulo the polynomial hierarchy: Understanding alternation as a source of hardness. *TOCT*, 9(3):15:1–15:20, 2017.

[Che17b] Leroy Chew. *QBF proof complexity*. PhD thesis, University of Leeds, 2017.

[CL73] Chin-Liang Chang and Richard C. T. Lee. *Symbolic logic and mechanical theorem proving*. Computer science classics. Academic Press, 1973.

[CM05] Stephen A. Cook and Tsuyoshi Morioka. Quantified propositional calculus and a second-order theory for NC^1. *Arch. Math. Log.*, 44(6):711–749, 2005.

[CR79] Stephen A. Cook and Robert A. Reckhow. The relative efficiency of propositional proof systems. *J. of Symbolic Logic*, 44(1):36–50, 1979.

[Egl12] Uwe Egly. On sequent systems and resolution for QBFs. In *SAT*, volume 7317 of *LNCS*, pages 100–113. Springer, 2012.

[Egl16] Uwe Egly. On stronger calculi for QBFs. In *SAT*, volume 9710 of *LNCS*, pages 419–434. Springer, 2016.

[EKLP17] Uwe Egly, Martin Kronegger, Florian Lonsing, and Andreas Pfan-

dler. Conformant planning as a case study of incremental QBF solving. *Ann. Math. Artif. Intell.*, 80(1):21–45, 2017.

[ELW13] Uwe Egly, Florian Lonsing, and Magdalena Widl. Long-Distance Resolution: Proof Generation and Strategy Extraction in Search-Based QBF Solving. In *LPAR*, volume 8312 of *LNCS*, pages 291–308. Springer, 2013.

[ESW09] Uwe Egly, Martina Seidl, and Stefan Woltran. A solver for qbfs in negation normal form. *Constraints*, 14(1):38–79, 2009.

[FHSB17] Katalin Fazekas, Marijn Heule, Martina Seidl, and Armin Biere. Skolem Function Continuation for Quantified Boolean Formulas. In *TAP*, volume 10375 of *LNCS*, pages 129–138. Springer, 2017.

[FSS84] Merrick L. Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984.

[GB13] Alexandra Goultiaeva and Fahiem Bacchus. Recovering and utilizing partial duality in QBF. In *SAT*, volume 7962 of *LNCS*, pages 83–99. Springer, 2013.

[GMN10] Enrico Giunchiglia, Paolo Marin, and Massimo Narizzano. sQueezeBF: An Effective Preprocessor for QBFs Based on Equivalence Reasoning. In *SAT*, volume 6175 of *LNCS*, pages 85–98. Springer, 2010.

[GNT06] Enrico Giunchiglia, Massimo Narizzano, and Armando Tacchella. Clause/Term Resolution and Learning in the Evaluation of Quantified Boolean Formulas. *JAIR*, 26:371–416, 2006.

[GNT07] E. Giunchiglia, M. Narizzano, and A. Tacchella. Quantifier Structure in Search-Based Procedures for QBFs. *TCAD*, 26(3):497–507, 2007.

[GVGB11] Alexandra Goultiaeva, Allen Van Gelder, and Fahiem Bacchus. A Uniform Approach for Generating Proofs and Strategies for Both True and False QBF Formulas. In *IJCAI*, pages 546–553. IJCAI/AAAI, 2011.

[Hås87] J. Håstad. *Computational Limitations of Small Depth Circuits*. MIT Press, Cambridge, 1987.

[HJL+15] Marijn Heule, Matti Järvisalo, Florian Lonsing, Martina Seidl, and Armin Biere. Clause Elimination for SAT and QSAT. *J. Artif. Intell. Res.*, 53:127–168, 2015.

[HKB17] Marijn J. H. Heule, Benjamin Kiesl, and Armin Biere. Short Proofs Without New Variables. In *CADE*, volume 10395 of *LNCS*, pages 130–147. Springer, 2017.

[HPSS18] Holger H. Hoos, Tomás Peitl, Friedrich Slivovsky, and Stefan Szeider. Portfolio-based algorithm selection for circuit qbfs. In *CP*, volume 11008 of *Lecture Notes in Computer Science*, pages 195–209. Springer, 2018.

[HSB14] Marijn Heule, Martina Seidl, and Armin Biere. Efficient extraction of Skolem functions from QRAT proofs. In *FMCAD*, pages 107–114. IEEE, 2014.

[HSB15] Marijn Heule, Martina Seidl, and Armin Biere. Blocked Literals Are Universal. In *NFM 2015*, volume 9058 of *LNCS*, pages 436–442.

Springer, 2015.

[HSB17] Marijn J. H. Heule, Martina Seidl, and Armin Biere. Solution validation and extraction for QBF preprocessing. *J. Autom. Reasoning*, 58(1):97–125, 2017.

[Jan18a] Mikoláš Janota. Circuit-based search space pruning in QBF. In *SAT*, volume 10929 of *LNCS*, pages 187–198. Springer, 2018.

[Jan18b] Mikoláš Janota. Towards generalization in QBF solving via machine learning. In *AAAI*, pages 6607–6614. AAAI Press, 2018.

[JGM13] Mikoláš Janota, Radu Grigore, and João Marques-Silva. On QBF Proofs and Preprocessing. In *LPAR*, volume 8312 of *LNCS*, pages 473–489. Springer, 2013.

[JJK+14] Mikoláš Janota, Charles Jordan, Will Klieber, Florian Lonsing, Martina Seidl, and Allen Van Gelder. The QBFGallery 2014: The QBF competition at the FLoC Olympic games. *JSAT*, 9:187–206, 2014.

[JKMC16] Mikoláš Janota, William Klieber, João Marques-Silva, and Edmund M. Clarke. Solving QBF with counterexample guided refinement. *Artif. Intell.*, 234:1–25, 2016.

[JKMSC12] Mikoláš Janota, William Klieber, João Marques-Silva, and Edmund Clarke. Solving QBF with counterexample guided refinement. In *SAT*, volume 7317 of *LNCS*, pages 114–128. Springer, 2012.

[JM15] Mikoláš Janota and Joao Marques-Silva. Expansion-based QBF solving versus Q-resolution. *Theor. Comput. Sci.*, 577:25–42, 2015.

[JMS11] Mikoláš Janota and Joao Marques-Silva. Abstraction-based algorithm for 2QBF. In *SAT*, volume 6695 of *LNCS*, pages 230–244. Springer, 2011.

[JMS13] Mikoláš Janota and Joao Marques-Silva. On propositional QBF expansions and Q-resolution. In *SAT*, volume 7962 of *LNCS*, pages 67–82. Springer, 2013.

[JMS15] Mikoláš Janota and Joao Marques-Silva. Solving QBF by clause selection. In *International Conference on Artificial Intelligence (IJCAI)*, pages 325–331. AAAI Press, 2015.

[Juk12] Stasys Jukna. *Boolean Function Complexity - Advances and Frontiers*, volume 27 of *Algorithms and combinatorics*. Springer, 2012.

[KBKF95] Hans Kleine Büning, Marek Karpinski, and Andreas Flögel. Resolution for Quantified Boolean Formulas. *Inf. Comput.*, 117(1):12–18, 1995.

[KHS17] Benjamin Kiesl, Marijn J. H. Heule, and Martina Seidl. A little blocked literal goes a long way. In *SAT*, volume 10491 of *LNCS*, pages 281–297. Springer, 2017.

[KP90] Jan Krajíček and Pavel Pudlák. Quantified propositional calculi and fragments of bounded arithmetic. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 36:29–46, 1990.

[Kra97] Jan Krajíček. Interpolation theorems, lower bounds for proof systems and independence results for bounded arithmetic. *J. of Symbolic Logic*, 62(2):457–486, 1997.

[KS18a] Manuel Kauers and Martina Seidl. Short proofs for some symmetric quantified boolean formulas. *Inf. Process. Lett.*, 140:4–7, 2018.

[KS18b] Manuel Kauers and Martina Seidl. Symmetries of quantified boolean formulas. In *SAT*, volume 10929 of *LNCS*, pages 199–216. Springer, 2018.

[KSGC10] William Klieber, Samir Sapra, Sicun Gao, and Edmund M. Clarke. A Non-prenex, Non-clausal QBF Solver with Game-State Learning. In *SAT*, volume 6175 of *LNCS*, pages 128–142. Springer, 2010.

[LB08] Florian Lonsing and Armin Biere. Nenofex: Expanding NNF for QBF solving. In *SAT*, volume 4996 of *LNCS*, pages 196–210. Springer, 2008.

[LB09] Florian Lonsing and Armin Biere. A Compact Representation for Syntactic Dependencies in QBFs. In *SAT*, volume 5584 of *LNCS*, pages 398–411. Springer, 2009.

[LB10] Florian Lonsing and Armin Biere. Integrating Dependency Schemes in Search-Based QBF Solvers. In *SAT*, volume 6175 of *LNCS*, pages 158–171. Springer, 2010.

[LBB+15] Florian Lonsing, Fahiem Bacchus, Armin Biere, Uwe Egly, and Martina Seidl. Enhancing Search-Based QBF Solving by Dynamic Blocked Clause Elimination. In *LPAR*, volume 9450 of *LNCS*, pages 418–433. Springer, 2015.

[LE17] Florian Lonsing and Uwe Egly. DepQBF 6.0: A Search-Based QBF Solver Beyond Traditional QCDCL. In *CADE*, volume 10395 of *LNCS*, pages 371–384. Springer, 2017.

[LE18] Florian Lonsing and Uwe Egly. QRAT+: Generalizing QRAT by a More Powerful QBF Redundancy Property. In *IJCAR*, volume 10900 of *LNCS*, pages 161–177. Springer, 2018.

[LE19] Florian Lonsing and Uwe Egly. QRATPre+: Effective QBF Preprocessing via Strong Redundancy Properties. In *SAT*, volume 11628 of *LNCS*, pages 203–210. Springer, 2019.

[LES16] Florian Lonsing, Uwe Egly, and Martina Seidl. Q-Resolution with Generalized Axioms. In *SAT*, volume 9710 of *LNCS*, pages 435–452. Springer, 2016.

[Let02] Reinhold Letz. Lemma and Model Caching in Decision Procedures for Quantified Boolean Formulas. In *TABLEAUX*, volume 2381 of *LNCS*, pages 160–175. Springer, 2002.

[LEVG13] Florian Lonsing, Uwe Egly, and Allen Van Gelder. Efficient clause learning for quantified boolean formulas via QBF pseudo unit propagation. In *SAT*, volume 7962 of *LNCS*, pages 100–115. Springer, 2013.

[LSVG16] Florian Lonsing, Martina Seidl, and Allen Van Gelder. The QBF Gallery: Behind the Scenes. *Artif. Intell.*, 237:92–114, 2016.

[MMZ+01] Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an Efficient SAT Solver. In *DAC*, pages 530–535. ACM, 2001.

[MSS99] João P. Marques Silva and Karem A. Sakallah. GRASP: A Search Algorithm for Propositional Satisfiability. *IEEE Trans. Computers*, 48(5):506–521, 1999.

[PG86] David A. Plaisted and Steven Greenbaum. A structure-preserving

clause form translation. *J. Symb. Comput.*, 2(3):293–304, 1986.

[PI00] Pavel Pudlák and Russell Impagliazzo. A lower bound for DLL algorithms for SAT. In *SODA*, pages 128–136, 2000.

[PS09] Florian Pigorsch and Christoph Scholl. Exploiting structure in an AIG based QBF solver. In *DATE*, pages 1596–1601. IEEE, 2009.

[PS19] Luca Pulina and Martina Seidl. The 2016 and 2017 QBF solvers evaluations (QBFEVAL'16 and QBFEVAL'17). *Artif. Intell.*, 274:224–248, 2019.

[PSS16] Tomáš Peitl, Friedrich Slivovsky, and Stefan Szeider. Long Distance Q-Resolution with Dependency Schemes. In *SAT*, volume 9710 of *LNCS*, pages 500–518. Springer, 2016.

[PSS19a] Tomáš Peitl, Friedrich Slivovsky, and Stefan Szeider. Combining Resolution-Path Dependencies with Dependency Learning. In *SAT*, volume 11628 of *LNCS*, pages 306–318. Springer, 2019.

[PSS19b] Tomáš Peitl, Friedrich Slivovsky, and Stefan Szeider. Dependency Learning for QBF. *J. Artif. Intell. Res.*, 65:180–208, 2019.

[PSS19c] Tomáš Peitl, Friedrich Slivovsky, and Stefan Szeider. Long-Distance Q-Resolution with Dependency Schemes. *J. Autom. Reasoning*, 63(1):127–155, 2019.

[PSS19d] Tomáš Peitl, Friedrich Slivovsky, and Stefan Szeider. Proof Complexity of Fragments of Long-Distance Q-Resolution. In *SAT*, volume 11628 of *LNCS*, pages 319–335. Springer, 2019.

[Pud97] Pavel Pudlák. Lower bounds for resolution and cutting planes proofs and monotone computations. *J. of Symbolic Logic*, 62(3):981–998, 1997.

[PV04] Guoqiang Pan and Moshe Y. Vardi. Symbolic decision procedures for QBF. In *CP*, volume 3258 of *LNCS*, pages 453–467. Springer, 2004.

[Rob65] John Alan Robinson. A Machine-Oriented Logic Based on the Resolution Principle. *J. ACM*, 12(1):23–41, 1965.

[RT15] Markus N. Rabe and Leander Tentrup. CAQE: A certifying QBF solver. In *FMCAD*, pages 136–143. IEEE, 2015.

[RTRS18] Markus N. Rabe, Leander Tentrup, Cameron Rasmussen, and Sanjit A. Seshia. Understanding and extending incremental determinization for 2QBF. In *CAV (2)*, volume 10982 of *LNCS*, pages 256–274. Springer, 2018.

[Sam08] Marko Samer. Variable Dependencies of Quantified CSPs. In *LPAR*, volume 5330 of *LNCS*, pages 512–527. Springer, 2008.

[SDB06] Horst Samulowitz, Jessica Davies, and Fahiem Bacchus. Preprocessing QBF. In *CP*, volume 4204 of *LNCS*, pages 514–529. Springer, 2006.

[Seg07] Nathan Segerlind. The complexity of propositional proofs. *Bulletin of Symbolic Logic*, 13(4):417–481, 2007.

[SLB12] Martina Seidl, Florian Lonsing, and Armin Biere. qbf2epr: A tool for generating EPR formulas from QBF. In *PAAR-2012*, pages 139–148, 2012.

[SS09] Marko Samer and Stefan Szeider. Backdoor Sets of Quantified

Boolean Formulas. *J. of Aut. Reas.*, 42(1):77–97, 2009.

[SS12] Friedrich Slivovsky and Stefan Szeider. Computing Resolution-Path Dependencies in Linear Time. In *SAT*, volume 7317 of *LNCS*, pages 58–71. Springer, 2012.

[SS16] Friedrich Slivovsky and Stefan Szeider. Soundness of Q-resolution with dependency schemes. *Theor. Comput. Sci.*, 612:83–101, 2016.

[Ten16] Leander Tentrup. Non-prenex QBF solving using abstraction. In *SAT*, volume 9710 of *LNCS*, pages 393–401. Springer, 2016.

[Ten17] Leander Tentrup. On expansion and resolution in CEGAR based QBF solving. In *CAV (2)*, volume 10427 of *Lecture Notes in Computer Science*, pages 475–494. Springer, 2017.

[THJ15] Kuan-Hua Tu, Tzu-Chien Hsu, and Jie-Hong R. Jiang. QELL: QBF reasoning with extended clause learning and levelized SAT solving. In *SAT*, volume 9340 of *LNCS*, pages 343–359. Springer, 2015.

[Tse68] G. S. Tseitin. On the complexity of derivations in the propositional calculus. *Studies in Constructive Mathematics and Mathematical Logic*, 1968.

[VG11] Allen Van Gelder. Variable Independence and Resolution Paths for Quantified Boolean Formulas. In *CP*, volume 6876 of *LNCS*, pages 789–803. Springer, 2011.

[VG12a] Allen Van Gelder. Contributions to the Theory of Practical Quantified Boolean Formula Solving. In *CP*, volume 7514 of *LNCS*, pages 647–663. Springer, 2012.

[VG12b] Allen Van Gelder. Producing and verifying extremely large propositional refutations - have your cake and eat it too. *Ann. Math. Artif. Intell.*, 65(4):329–372, 2012.

[VGWL12] Allen Van Gelder, Samuel B. Wood, and Florian Lonsing. Extended Failed-Literal Preprocessing for Quantified Boolean Formulas. In *SAT*, volume 7317 of *LNCS*, pages 86–99. Springer, 2012.

[WHJ14] Nathan Wetzler, Marijn Heule, and Warren A. Hunt Jr. DRAT-trim: Efficient Checking and Trimming Using Expressive Clausal Proofs. In *SAT*, volume 8561 of *LNCS*, pages 422–429. Springer, 2014.

[WRMB17] Ralf Wimmer, Sven Reimer, Paolo Marin, and Bernd Becker. HQSpre - An Effective Preprocessor for QBF and DQBF. In *TACAS*, volume 10205 of *LNCS*, pages 373–390, 2017.

[ZM02a] Lintao Zhang and Sharad Malik. Conflict Driven Learning in a Quantified Boolean Satisfiability Solver. In *ICCAD*, pages 442–449. ACM / IEEE Computer Society, 2002.

[ZM02b] Lintao Zhang and Sharad Malik. Towards a Symmetric Treatment of Satisfaction and Conflicts in Quantified Boolean Formula Evaluation. In *CP*, volume 2470 of *LNCS*, pages 200–215. Springer, 2002.

[ZMMM01] Lintao Zhang, Conor F. Madigan, Matthew W. Moskewicz, and Sharad Malik. Efficient Conflict Driven Learning in Boolean Satisfiability Solver. In *ICCAD*, pages 279–285. IEEE, 2001.