A Duality-Aware Calculus for Quantified Boolean Formulas

Katalin Fazekas, Martina Seidl, Armin Biere
Institute for Formal Models and Verification
Johannes Kepler University Linz, Austria

***Abstract*—Learning and backjumping are essential features in search-based decision procedures for Quantified Boolean Formulas (QBF). To obtain a better understanding of such procedures, we present a formal framework, which allows to simultaneously reason on prenex conjunctive and disjunctive normal form. It captures both satisfying and falsifying search states in a symmetric way. This symmetry simplifies the framework and offers potential for further variants.**

I. Introduction

Quantified Boolean Formulas (QBF) extend the language of propositional logic by quantifiers over the propositional variables. In consequence, the decision problem becomes PSPACE-complete making QBF solving interesting for many applications in verification, synthesis, artificial intelligence, etc. (see [1] for a survey). The combination of conflict-driven clause and solution-driven cube learning (QCDCL, see e.g., [2], [3]) is the most successful approach for search-based QBF solving [4], [5], lifting conflict-driven clause-learning (CDCL), originating in SAT [6], to QBF. However, if a QBF solver only gets a QBF in prenex conjunctive normal form (PCNF) as input, the search is biased towards conflicts. This asymmetry impedes the whole search process. To overcome this asymmetry, *duality-aware QCDCL solving* considers not only a representation of the input in PCNF but also in prenex disjunctive normal form (PDNF), treating solution and conflict states symmetrically [7]. In [8] it was shown that duality-aware reasoning can easily be added to PCNF-based QCDCL solvers. This paper gives a concise characterization of the behavior of such solvers exploiting the symmetry in the search for conflicts and solutions.

Related Work: The seminal paper of Nieuwenhuis, Oliveras, and Tinelli [9] introduced a rule-based calculus to concisely model the classical CDCL approach for SAT. This work forms the basis of many theoretical investigations on SAT and SMT solving, for instance the recent formalization of CDCL-SAT solving in Isabelle [10]. For QBF, however, we are not aware of any similar rule-based formulation of QCDCL. The abstract QBF solver presented rather informally in [11] lacks non-chronological backtracking (*backjumping*) and learning, i.e., the essential rules of QCDCL. The literature on QCDCL (e.g., [2], [3]) is missing a precise formalization too. In [12] the focus is on the relation of QCDCL with Q-resolution instead of capturing the search precisely.

Outline: We introduce a new calculus which formally captures the behavior of QCDCL solvers. To this end, we first provide generic rules defining a state transition system. Then we introduce a strategy that describes how to apply these rules to get a duality-aware QBF solver which is correct and terminates. Finally, we relate our calculus to standard PCNF QCDCL solvers.

II. Preliminaries

A propositional formula over variables $\mathcal{V}$ is in conjunctive normal form (CNF) if it is a conjunction of clauses. A *clause* is a disjunction of literals and a literal is a variable or its negation. A propositional formula is in disjunctive normal form (DNF) if it is a disjunction of cubes, where a *cube* is a conjunction of literals. If literal ℓ is v or $\neg v$, then $var(\ell) = v$. A *quantified Boolean formula* (QBF) $\mathcal{Q}(\varphi)$ consists of the propositional matrix φ over $\mathcal{V}$ and the quantifier prefix $\mathcal{Q} = Q_1V_1 \ldots Q_nV_n$ where V_i are disjoint sets of variables, $Q_i \in \{\exists, \forall\}$, and $Q_i \neq Q_{i+1}$. In this paper, we assume $\mathcal{V} = \bigcup V_i$, i.e., all variables of the matrix are quantified. Such QBFs are called closed. If φ of QBF $\mathcal{Q}(\varphi)$ is in CNF (DNF), then $\mathcal{Q}(\varphi)$ is in prenex CNF (DNF), denoted PCNF (PDNF). We also write $\ell_\forall$ and $\ell_\exists$ for a universal or existential literal. For variable $v \in V_i$ the quantification level $level(v)$ is i. In that way, the prefix $\mathcal{Q}$ defines a partial ordering relation $<_\mathcal{Q}$ between variables v_i and v_j such that $v_i <_\mathcal{Q} v_j$ if $level(v_i) < level(v_j)$. This ordering is extended to the literals over the variables, that is, $\ell_i <_\mathcal{Q} \ell_j$ if $var(\ell_i) <_\mathcal{Q} var(\ell_j)$. An assignment A is a consistent list of literals which defines a mapping from literals to truth values as follows. If $v \in A$ then v is true under A, if $\neg v \in A$ then v is false under A. An assignment A is called total assignment of $\mathcal{V}$ if every $v \in \mathcal{V}$ is assigned by A, otherwise it is called partial. Occasionally we interpret assignments as cubes. Given an assignment A and a CNF φ with a fixed quantifier prefix $\mathcal{Q}$, we denote by $\varphi[A]$ the CNF under assignment A, where all clauses containing ℓ are removed and all occurrences of $\neg\ell$ are deleted (and $\mathcal{Q}$ is unchanged). For DNF φ, $\varphi[A]$ is defined dually. The negation of a quantifier prefix $\mathcal{Q}$ (denoted with $\neg\mathcal{Q}$) is the simultaneous substitution of $\forall$ quantifiers for $\exists$ quantifiers and vice versa in $\mathcal{Q}$. We define tree models and tree refutations of QBFs as in [13]. We denote an empty clause or cube by $\emptyset$, an empty CNF (DNF) by $\top$ ($\bot$). A QBF $\forall x\mathcal{Q}(\varphi)$ is true iff $\mathcal{Q}(\varphi)[x]$ and $\mathcal{Q}(\varphi)[\neg x]$ are true. A

QBF $\exists x \mathcal{Q}(\varphi)$ is true iff $\mathcal{Q}(\varphi)[x]$ or $\mathcal{Q}(\varphi)[\neg x]$ is true. Two QBFs ψ_1 and ψ_2 are equivalent (written as $\psi_1 \equiv \psi_2$) if they have the same truth value.

Definition 1. *QBFs $\mathcal{Q}(\mathcal{C})$ in PCNF and $\mathcal{Q}(\mathcal{D})$ in PDNF have the* duality property *if $\mathcal{Q}(\mathcal{C}) \equiv \mathcal{Q}(\mathcal{D})$.*

The duality property holds under assignment A, if $\mathcal{Q}(\mathcal{C}[A]) \equiv \mathcal{Q}(\mathcal{D}[A])$. Based on the duality property, dual search-based solvers [7], [8] use both a CNF as well as a DNF of the input QBF (as explained e.g. in [7]) to treat conflicts and solutions symmetrically. Note that in that case the introduced Tseitin variables are existentially quantified in the CNF, and universally quantified in the DNF encoding. Therefore, the joint prefix $\mathcal{Q}$ of CNF and DNF has some variables that occur only in the CNF matrix (the existential Tseitin variables) and some that occur only in the DNF matrix (the universal Tseitin variables).

Definition 2. *Given QBF $\mathcal{Q}(\varphi)$ in PCNF and a clause C, we define $\varphi \vDash_{\mathcal{Q}} C$ to hold if $\mathcal{Q}(\varphi \wedge C) \equiv \mathcal{Q}(\varphi)$.*

Definition 3. *Given QBF $\mathcal{Q}(\varphi)$ in PDNF and a cube C, we define $\varphi \vDash^{\mathcal{Q}} C$ to hold if $\mathcal{Q}(\varphi \vee C) \equiv \mathcal{Q}(\varphi)$.*

Note that $\vDash_{\mathcal{Q}}$ and $\vDash^{\mathcal{Q}}$ define the entailment relation w.r.t. prefix $\mathcal{Q}$. Standard propositional entailment is denoted by $\vDash$. Immediately from the definitions we obtain:

Lemma 1. *Let $\mathcal{Q}(\varphi)$ be a closed QBF in PCNF and C a clause. Then $\varphi \vDash_{\mathcal{Q}} C$ iff $\neg\varphi \vDash^{\neg\mathcal{Q}} \neg C$.*

In the following we fix a quantifier prefix $\mathcal{Q}$ together with the ordering $<_{\mathcal{Q}}$ (on all variables and literals).

Definition 4. *The* universally tailing literals *$\mathcal{T}^{\mathcal{Q}}_{\forall}(C)$ of clause C are $\{\ell_\forall \in C \mid \ell_\exists <_{\mathcal{Q}} \ell_\forall$ for all $\ell_\exists$ in $C\}$.*

Definition 5. *The* universal reduction *of a clause C is defined as $\mathcal{R}^{\mathcal{Q}}_{\forall}(C) = C \setminus \mathcal{T}^{\mathcal{Q}}_{\forall}(C)$.*

Analogously, existential reduction $\mathcal{R}^{\mathcal{Q}}_{\exists}(C)$ removes the tailing existential literals $\mathcal{T}^{\mathcal{Q}}_{\exists}(C)$ from cube C. Universal (existential) reduction can be extended to a set of clauses (cubes) $\mathcal{C}$ as $\mathcal{R}^{\mathcal{Q}}_{\forall}(\mathcal{C}) = \{\mathcal{R}^{\mathcal{Q}}_{\forall}(C) \mid C \in \mathcal{C}\}$ ($\mathcal{R}^{\mathcal{Q}}_{\exists}(\mathcal{C}) = \{\mathcal{R}^{\mathcal{Q}}_{\exists}(C) \mid C \in \mathcal{C}\}$). W.l.o.g. we assume clauses (cubes) of the input QBFs to be non-tautological (non-contradictory) and $\forall$-reduced ($\exists$-reduced).

The literal $\ell_\exists$ is an *existential unit* in clause C iff $\mathcal{R}^{\mathcal{Q}}_{\forall}(C) = \{\ell_\exists\}$. Then C is called a *unit clause*. Unit cubes and universal units are defined analogously. We also simply call C a unit, if C is a unit clause or unit cube. We further extend these definitions to clauses C (and cubes) under an assignment A, by using $C[A]$ instead of C. For instance, $\ell_\exists$ is an existential unit in C under A iff $\ell_\exists$ is an existential unit in $C[A]$. In this case we assume that C is not satisfied by A, i.e., $C[A] \neq \top$. A literal is called pure in QBF $\mathcal{Q}(\varphi)$ if it occurs only in exactly one polarity.

III. Abstract QCDCL Solving

Our abstract QCDCL solver is described in terms of a state transition system. It explicitly traverses the assignment tree of a QBF given in CNF and DNF to prove its (un)satisfiability. States S of the form $A \parallel \mathcal{D} \parallel \mathcal{C}$ consist of a CNF $\mathcal{C}$, a DNF $\mathcal{D}$, and of an assignment A (also called *trail* in solver implementations) over variables of the fixed quantifier prefix $\mathcal{Q}$. The literals in A are either decided or implied (see below). A decision literal is written as ℓ^d. The initial state of our abstract QCDCL solver is $\emptyset \parallel \mathcal{D} \parallel \mathcal{C}$, where $\mathcal{D}$ contains the input QBF in DNF, while $\mathcal{C}$ is the input QBF in CNF. Therefore the duality property holds, i.e., $\mathcal{Q}(\mathcal{C}) \equiv \mathcal{Q}(\mathcal{D})$. Next, we introduce the rules of our abstract QCDCL solver.

$$\text{Unit}_\exists\text{:}\ \frac{A \parallel \mathcal{D} \parallel \mathcal{C} \wedge C}{A\ \ell_\exists \parallel \mathcal{D} \parallel \mathcal{C} \wedge C} \qquad \ell_\exists \text{ existential unit in } C[A]$$

$$\text{Unit}_\forall\text{:}\ \frac{A \parallel \mathcal{D} \vee C \parallel \mathcal{C}}{A\ \neg\ell_\forall \parallel \mathcal{D} \vee C \parallel \mathcal{C}} \qquad \ell_\forall \text{ universal unit in } C[A]$$

Unit propagation extends the current assignment by unit literals with respect to their quantifier type.

$$\text{Pure}_\exists\text{:}\ \frac{A \parallel \mathcal{D} \parallel \mathcal{C}}{A\ \ell_\exists \parallel \mathcal{D} \parallel \mathcal{C}} \qquad \ell_\exists \in \mathcal{R}^{\mathcal{Q}}_{\exists}(\mathcal{D}[A]) \text{ is pure}$$

$$\text{Pure}_\forall\text{:}\ \frac{A \parallel \mathcal{D} \parallel \mathcal{C}}{A\ \neg\ell_\forall \parallel \mathcal{D} \parallel \mathcal{C}} \qquad \ell_\forall \in \mathcal{R}^{\mathcal{Q}}_{\forall}(\mathcal{C}[A]) \text{ is pure}$$

Due to the duality property, we can identify *pure literals* either from $\mathcal{C}$ or $\mathcal{D}$. Note that this is not possible in decision procedures without duality-awareness because there $\mathcal{D}$ is not a complete representation of the input QBF.

$$\text{Decide:}\ \frac{A \parallel \mathcal{D} \parallel \mathcal{C}}{A\ \ell^d \parallel \mathcal{D} \parallel \mathcal{C}} \qquad \ell \text{ is unassigned and all } \ell' \text{ with } \ell' <_{\mathcal{Q}} \ell \text{ are assigned in } A$$

Rule Decide adds the decision literals to the current assignment A. The quantifier prefix restricts the set of decision candidate variables. Further, each decision must preserve consistency with A.

$$\text{Learn}_{\text{CNF}}\text{:}\ \frac{A \parallel \mathcal{D} \parallel \mathcal{C}}{A \parallel \mathcal{D} \parallel \mathcal{C} \wedge C} \qquad \mathcal{C} \vDash_{\mathcal{Q}} C$$

$$\text{Learn}_{\text{DNF}}\text{:}\ \frac{A \parallel \mathcal{D} \parallel \mathcal{C}}{A \parallel \mathcal{D} \vee C \parallel \mathcal{C}} \qquad \mathcal{D} \vDash^{\mathcal{Q}} C$$

Learn$_{\text{CNF}}$ and Learn$_{\text{DNF}}$ describe the clause and cube learning of the solver. The only restriction on learned clauses, and dually for cubes, is that they are implied by the formula w.r.t. the prefix, which by definition requires $\mathcal{Q}(\mathcal{C}) \equiv \mathcal{Q}(\mathcal{C} \wedge C)$. Deciding equivalence of two QBFs is

PSPACE hard. In practice polynomial derivation techniques are used, e.g., some form of Q-resolution.

$$\boxed{\text{Undo}_\exists\text{: } \frac{A\,\ell^d_\exists A' \parallel \mathcal{D} \parallel \mathcal{C}}{A \parallel \mathcal{D} \parallel \mathcal{C}}} \qquad \boxed{\text{Undo}_\forall\text{: } \frac{A\,\ell^d_\forall A' \parallel \mathcal{D} \parallel \mathcal{C}}{A \parallel \mathcal{D} \parallel \mathcal{C}}}$$

The Undo-rules are responsible for backtracking. These steps have no side condition, but at least one decision literal in the current assignment is required. Backtracking is allowed only precisely before such a decision literal, because backtracking to other points of the trail could lead to unnecessary repetitions of steps.

$$\boxed{\text{Final}_{\text{CNF}}\text{: } \frac{A \parallel D \parallel \mathcal{C} \wedge \emptyset}{\bot}} \qquad \boxed{\text{Final}_{\text{DNF}}\text{: } \frac{A \parallel \mathcal{D} \vee \emptyset \parallel \mathcal{C}}{\top}}$$

If the empty clause (cube) is in $\mathcal{C}$ ($\mathcal{D}$), the formula simplifies to $\bot$ ($\top$). We also denote such a state by $\bot$ ($\top$). If the application of a rule causes a transition from a state $S = (A \parallel \mathcal{D} \parallel \mathcal{C})$ to a state $S' = (A' \parallel \mathcal{D}' \parallel \mathcal{C}')$, we denote this by $S \vdash S'$. For multiple rule applications we write $S \vdash^* S'$. Now we obtain the following lemmas from our definitions and, for instance, using facts from [14].

Lemma 2. *If $S \vdash S'$ using the* Learn *or* Final *rules, then $\mathcal{Q}(\mathcal{C}) \equiv \mathcal{Q}(\mathcal{C}')$ and $\mathcal{Q}(\mathcal{D}) \equiv \mathcal{Q}(\mathcal{D}')$. For all other rules, $\mathcal{C}' = \mathcal{C}$ and $\mathcal{D}' = \mathcal{D}$.*

Lemma 3. *If $S \vdash S'$ using rules* Unit *or* Pure*, then $\mathcal{Q}(\mathcal{C}[A]) \equiv \mathcal{Q}(\mathcal{C}[A'])$ and $\mathcal{Q}(\mathcal{D}[A]) \equiv \mathcal{Q}(\mathcal{D}[A'])$.*

Corollary 1. *Each rule preserves the duality property.*

To get closer to real QBF solvers and to enforce termination, we have to restrict the application of rules based on the actual state of the solver.

Definition 6 (Leaf Condition)**.** *State $A \parallel \mathcal{D} \parallel \mathcal{C}$ has*

$$\begin{array}{rl} \textit{(Conflict Condition) } L_\bot & \textit{iff } \emptyset \in \mathcal{R}^{\mathcal{Q}}_\forall(\mathcal{C}[A]) \\ \textit{(Satisfaction Condition) } L_\top & \textit{iff } \emptyset \in \mathcal{R}^{\mathcal{Q}}_\exists(\mathcal{D}[A]) \\ \textit{(Leaf Condition) } L & \textit{iff } L_\top \vee L_\bot \end{array}$$

The duality property guarantees $L_\bot$ and $L_\top$ to be mutually exclusive. Whenever the solver finds a satisfying or falsifying assignment, we say that it reached a leaf of the assignment tree. Then L is true in that state.

Definition 7. *A state has the* Propagation Condition*, denoted P, if one of the* Unit *or* Pure *rules can be applied.*

Definition 8 (Driving Condition)**.** *In $AA' \parallel \mathcal{D} \parallel \mathcal{C}$ assume A' contains a decision ℓ^d, where ℓ^d is existential if C is a clause in $\mathcal{C}$ and universal if C is a cube in $\mathcal{D}$.*

$$\begin{array}{rl} \textit{(Driving Clause) } D_\bot(C) & \textit{iff } C[AA'] = \emptyset,\ C[A] \textit{ unit} \\ \textit{(Driving Cube) } D_\top(C) & \textit{iff } C[AA'] = \emptyset,\ C[A] \textit{ unit} \\ \textit{(Driving Condition) } D(C) & \textit{iff } D_\bot(C) \vee D_\top(C) \end{array}$$

The driving condition holds (i.e., D is true) in a state $AA' \parallel \mathcal{D} \parallel \mathcal{C}$ iff there is a clause (cube) C in $\mathcal{C}$ ($\mathcal{D}$) s.t. $D(C)$ holds. Then C is a driving clause (cube) *and C* is driving *the existential (universal) unit literal $\ell' \in C[A]$.*

Table I
ADDITIONAL STRATEGY CONSTRAINTS.

Rule	**Pre-condition**	**Post-condition**
Unit	$\neg F \wedge \neg L \wedge \neg D$	
Pure	$\neg F \wedge \neg L \wedge \neg D$	
Decide	$\neg F \wedge \neg L \wedge \neg D \wedge \neg P$	
Learn	$\neg F \wedge \ L \wedge \neg D$	$D \vee F$
Undo	$\neg F \wedge \ L \wedge \ D$	$\neg D$
Final	F	

If a driving clause C is learned in state $AA' \parallel \mathcal{D} \parallel \mathcal{C}$, then the Undo rule can be applied to remove A' from the trail and then the Unit rule can be used to add $C[A]$ (the driven literal) to the current assignment. In that way, backjumping can be simulated with a sequence of small steps. This makes the whole process more transparent.

Definition 9 (Final Condition)**.** *In state $A \parallel \mathcal{D} \parallel \mathcal{C}$ we define the following final conditions:*

$$\begin{array}{rl} \textit{(Inconsistency Condition) } F_\bot & \textit{iff } \emptyset \in \mathcal{C} \\ \textit{(Tautology Condition) } F_\top & \textit{iff } \emptyset \in \mathcal{D} \\ \textit{(Final Condition) } F & \textit{iff } F_\top \vee F_\bot \end{array}$$

Lemma 4. *If in $A \parallel \mathcal{D} \parallel \mathcal{C}$, F or D holds, then also L.*

Now, to guarantee termination, a strategy for applying our rules is enforced by further restricting their side conditions (see Table I). Our strategy requires to stop propagation as soon as a leaf is reached (which in turn is required for learning to be applied). In SAT it has been considered to lift this requirement, which however requires additional care during learning [15]. It is unclear at this point whether this also applies to our calculus or QBF. Decisions can be made (rule Decide) only when no other rule is applicable but the Final Condition does not hold yet. The strongest constraint is introduced for the Learn rules. It ensures that learning prunes the search space. Hence, the learned clause (or cube) is either empty or driving. The Undo-rules force the solver to backtrack exactly where the driving clause (cube) leads. Below we assume the constraints of Table I.

Lemma 5. *If in $A \parallel \mathcal{D} \parallel \mathcal{C}$ there is* ***no*** *existential (universal) decision literal in A and $\neg F \wedge \neg D$ and $L_\bot$ ($L_\top$) hold, then $\mathcal{C} \vDash_{\mathcal{Q}} \emptyset$ ($\mathcal{D} \vDash^{\mathcal{Q}} \emptyset$).*

Proof: By semantics of QBF and Lemma 3. ■

Lemma 6. *If in $A \parallel \mathcal{D} \parallel \mathcal{C}$ there is an existential (universal) decision literal in A and $\neg F \wedge \neg D$ and $L_\bot$ ($L_\top$) hold, then there exists a driving clause (cube) C s.t. $\mathcal{C} \vDash_{\mathcal{Q}} C$ ($\mathcal{D} \vDash^{\mathcal{Q}} C$).*

Proof: Given $A \parallel \mathcal{D} \parallel \mathcal{C}$ where $\neg F \wedge L_\bot \wedge \neg D$ holds, i.e., $\emptyset \notin \mathcal{C}$, but $\emptyset \in \mathcal{R}^{\mathcal{Q}}_{\forall}(\mathcal{C}[A])$ and there is no driving clause in $\mathcal{C}$. Then A has the form $L_0 \ell^d_1 L_1 ... \ell^d_n L_n$ for some $n > 0$, where $\ell^d_1, ..., \ell^d_n$ are the existential decision literals of A and $L_0, ..., L_n$ contain the implied literals (from Unit *and* Pure *rules) and the universal decision literals. Let C' be the clause $(\neg\ell^d_1 \vee ... \vee \neg\ell^d_n)$. Then by construction $C'[A] = \emptyset$ and $\neg\ell^d_n$ is an existential unit in C' under the assignment $A' = \{L_0\ell^d_1 L_1 ... \ell^d_{n-1} L_{n-1}\}$ (which is a prefix of A), thus C' is driving ℓ^d_n. We further have to show that $\mathcal{C} \vDash_{\mathcal{Q}} C'$, i.e. $\mathcal{Q}(\mathcal{C}) \equiv \mathcal{Q}(\mathcal{C} \wedge C')$. Due to Lemma 3 and QBF semantics, if $\mathcal{Q}(\mathcal{C})$ is true, then there is no tree model with a branch that contains $\ell^d_1, ..., \ell^d_n$. Therefore, conjoining $(\neg\ell^d_1 \vee ... \vee \neg\ell^d_n)$ to $\mathcal{Q}(\mathcal{C})$ is satisfiability preserving. The case $L_\top$ is analogous.* ■

In practice the learned clause or cube is not built as in the proof above. Conflict and solution analysis of QCDCL solvers rely on some form of Q-resolution, where the derived clause or cube can be safely added to the formula by construction. We further obtain the following facts, without complete proofs, due to space constraints.

Lemma 7. *There are no infinite derivations of the form $(\emptyset \parallel \mathcal{D} \parallel \mathcal{C}) \vdash S_1 \vdash S_2 \vdash \cdots \vdash S_i \vdash \cdots$*

Lemma 8. *If $(\emptyset \parallel \mathcal{D} \parallel \mathcal{C}) \vdash^* S$ and no rule applies to S then S is either $\bot$ or $\top$.*

Lemma 9. *If $(\emptyset \parallel \mathcal{D} \parallel \mathcal{C}) \vdash^* S \in \{\bot, \top\}$, $Q(\mathcal{C}) \equiv S$.*

Theorem 1. *Our abstract QCDCL calculus is sound and complete. Applying the additional strategy constraints always produces terminating derivations.*

IV. Extensions

In our framework termination depends on learning of the empty clause or cube. Thus, in its basic form, it can not simulate pure search-based solvers without learning. Further, as memory is limited in practice, it is necessary to forget learned clauses and cubes which became irrelevant. Moreover, in practice, it can be beneficial to stop the current search and start over again with an empty trail. Extending the framework with further rules, we can easily capture also these aspects of practical solvers.

$$\mathrm{Forget}_{\mathrm{CNF}}\colon \frac{A \parallel \mathcal{D} \parallel \mathcal{C} \wedge C}{A \parallel \mathcal{D} \parallel \mathcal{C}} \qquad \mathcal{C} \vDash_{\mathcal{Q}} C$$

$$\mathrm{Forget}_{\mathrm{DNF}}\colon \frac{A \parallel \mathcal{D} \vee C \parallel \mathcal{C}}{A \parallel \mathcal{D} \parallel \mathcal{C}} \qquad \mathcal{D} \vDash^{\mathcal{Q}} C$$

The side conditions of the Forget-rules guarantee that only redundant information is discarded. With these additional rules we can also simulate solvers without learning, if the strategy enforces to apply Forget right after backtracking (Undo) and propagation (Unit & Pure). Obviously, as soon as an empty clause or cube is learned, Final termination rules have to be applied immediately. Restart of the search has no side condition, but on the strategy level additional care is necessary in order to maintain termination.

$$\mathrm{Restart}\colon \frac{A \parallel \mathcal{D} \parallel \mathcal{C}}{\emptyset \parallel \mathcal{D} \parallel \mathcal{C}}$$

$$\mathrm{Learn}'_{\mathrm{DNF}}\colon \frac{A \parallel \mathcal{D} \parallel \mathcal{C}}{A \parallel \mathcal{D} \vee C \parallel \mathcal{C}} \qquad \mathcal{D} \vDash^{\mathcal{Q}} C \text{ or } C \vDash \mathcal{C}$$

Usually the input of a QBF solver is only available in PCNF, therefore we can not assume the duality property as invariant. However, the following weaker invariant over the clauses and cubes serves a similar purpose: $\mathcal{Q}(\mathcal{D}) \Rightarrow \mathcal{Q}(\mathcal{C})$. This invariant ensures that whenever the DNF is satisfied, the CNF is satisfied as well. To adapt our framework to this new invariant, some modifications are necessary. For instance, $\mathrm{Pure}_\exists$ has to search for the existential pure literals in the clause set (instead of the cube set). Moreover, since in that case the DNF is incomplete, $\mathrm{Forget}_{\mathrm{DNF}}$ can be applied without side condition, and the constraints of learning new cubes has to be weakened (see $\mathrm{Learn}'_{\mathrm{DNF}}$). There are now two possible solution scenarios (as in [3]). First, one of the cubes in the database is satisfied. In that case we learn as we did before and the driving cube construction remains the same. Second, all the clauses in the clause database are satisfied but no satisfied cube exists. Note that the Satisfaction Condition in Def. 6 has to be updated.

Definition 10 (Satisfaction Condition)**.** *$A \parallel \mathcal{D} \parallel \mathcal{C}$ has*

(DNF Satisfaction)	S_{DNF}	*iff* $\emptyset \in \mathcal{R}^{\mathcal{Q}}_{\exists}(\mathcal{D}[A])$
(CNF Satisfaction)	S_{CNF}	*iff* $\mathcal{C}[A] = \top$
(Satisfaction Cond.)	$L_\top$	*iff* $S_{\mathrm{CNF}} \vee S_{\mathrm{DNF}}$

Since we can learn cubes which are weakening the DNF, Lemma 2 ceases to hold. Instead we obtain:

Lemma 10. *If $S \vdash S'$ using the* Learn *or* Final *rules, then $\mathcal{Q}(\mathcal{C}) \equiv \mathcal{Q}(\mathcal{C}')$ and $\mathcal{Q}(\mathcal{D}') \Rightarrow \mathcal{Q}(\mathcal{C}')$.*

V. Conclusion and Future Work

We presented a formal framework for concisely capturing search-based QBF solving. Such a framework is useful for better understanding of various types of QCDCL solvers.

We plan to use our framework to close the gap between QCDCL solving and other approaches, including expansion-based techniques [16], [17]. Currently, it is not clear how these approaches relate to one another w.r.t. solving strength. While there is some work on relating proof systems (e.g., [18]), the actual search strategies have not been compared yet. Especially when different techniques are integrated as currently proposed in [19], a better understanding of the individual solving techniques is indispensable.

VI. Acknowledgment

The authors would like to thank the anonymous reviewers for their helpful comments. This research has been supported by the Austrian Science Fund (FWF) under projects W1255-N23 and S11408-N23.

References

[1] M. Benedetti and H. Mangassarian, "QBF-Based Formal Verification: Experience and Perspectives," *JSAT*, 2008.

[2] E. Giunchiglia, M. Narizzano, and A. Tacchella, "Clause/term resolution and learning in the evaluation of quantified Boolean formulas," *JAIR*, 2006.

[3] L. Zhang and S. Malik, "Towards a symmetric treatment of satisfaction and conflicts in quantified Boolean formula evaluation," in *CP*, 2002.

[4] F. Lonsing and A. Biere, "DepQBF: A Dependency-Aware QBF Solver," *JSAT*, vol. 7, 2010.

[5] E. Giunchiglia, P. Marin, and M. Narizzano, "Reasoning with Quantified Boolean Formulas," in *Handbook of Satisfiability*, 2009.

[6] J. P. M. Silva, I. Lynce, and S. Malik, "Conflict-Driven Clause Learning SAT Solvers," in *Handbook of Satisfiability*, 2009.

[7] L. Zhang, "Solving QBF with combined conjunctive and disjunctive normal form," in *AAAI*, 2006.

[8] A. Goultiaeva, M. Seidl, and A. Biere, "Bridging the gap between dual propagation and CNF-based QBF solving," in *DATE*, 2013.

[9] R. Nieuwenhuis, A. Oliveras, and C. Tinelli, "Solving SAT and SAT Modulo Theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL (T)," *JACM*, 2006.

[10] J. C. Blanchette, M. Fleury, and C. Weidenbach, "A Verified SAT Solver Framework with Learn, Forget, Restart, and Incrementality," in *IJCAR*, 2016.

[11] R. Brochenin and M. Maratea, "Abstract Solvers for Quantified Boolean Formulas and their Applications," in *AI*IA*, 2015.

[12] M. Janota, "On Q-resolution and CDCL QBF solving," in *SAT*, 2016.

[13] A. Van Gelder, "Contributions to the theory of practical quantified Boolean formula solving," in *CP*, 2012.

[14] M. Cadoli, M. Schaerf, A. Giovanardi, and M. Giovanardi, "An algorithm to evaluate quantified Boolean formulae and its experimental evaluation," *JAR*, 2002.

[15] A. Goultiaeva and F. Bacchus, "Off the Trail: Re-examining the CDCL Algorithm," in *SAT*, 2012.

[16] M. Heule, M. Järvisalo, F. Lonsing, M. Seidl, and A. Biere, "Clause elimination for SAT and QSAT," *JAIR*, 2015.

[17] M. Janota, W. Klieber, J. Marques-Silva, and E. M. Clarke, "Solving QBF with counterexample guided refinement," *Artif. Intell.*, 2016.

[18] M. Janota and J. Marques-Silva, "Expansion-based QBF solving versus Q-resolution," *Theor. Comput. Sci.*, vol. 577, 2015.

[19] F. Lonsing, U. Egly, and M. Seidl, "Q-Resolution with Generalized Axioms," in *SAT*, 2016.

Appendix

Example 1. *Consider the QBF $\psi = \exists x \forall y.\ x \Leftrightarrow y$. It can be transformed to CNF as $\exists x \forall y \exists p.\ p \wedge (\neg p \vee \neg x \vee y) \wedge (\neg p \vee x \vee \neg y)$, and to DNF as $\exists x \forall y q.\ q \vee (\neg q \wedge \neg x \wedge \neg y) \vee (\neg q \wedge x \wedge y)$, where p and q are newly introduced Tseitin-variables. Given these two representations of the input formula ψ, the initial state of the abstract solver is $\emptyset \parallel \mathcal{D} \parallel \mathcal{C}$, where $\mathcal{Q} = \exists x \forall y \exists p \forall q$, $\mathcal{D} = q \vee (\neg q \wedge \neg x \wedge \neg y) \vee (\neg q \wedge x \wedge y)$ and $\mathcal{C} = p \wedge (\neg p \vee \neg x \vee y) \wedge (\neg p \vee x \vee \neg y)$. Note that p and q are interchangeable in the prefix, thus $\exists x \forall y q \exists p$ is another possible prefix for that example. A possible derivation from that state would be as follows.*

$$\begin{array}{llll r}
\emptyset & \parallel \mathcal{D} & \parallel \mathcal{C} & \vdash_{\text{Unit}_\exists} & (1) \\
p & \parallel \mathcal{D} & \parallel \mathcal{C} & \vdash_{\text{Unit}_\exists} & (2) \\
p\,x & \parallel \mathcal{D} & \parallel \mathcal{C} & \vdash_{\text{Learn}_{\text{CNF}}} & (3) \\
p\,x & \parallel \mathcal{D} & \parallel \mathcal{C} \wedge \emptyset & \vdash_{\text{Final}_{\text{CNF}}} & (4) \\
 & & \bot & & (5)
\end{array}$$

In the initial state there is an existential unit (p) in the CNF and there is a universal unit (q) in the DNF formula. Assume that the solver first propagates p, that yields state (2). In that state, there is still q as universal unit. Further, the second and third clauses of the CNF formula are the existential units $\neg x$ and x respectively, since under the assignment p the universal literals y and $\neg y$ are reduced.

Consider the case that the solver propagates x as a next step, which yields state (3). Here the second clause of the CNF formula is falsified, i.e. $\emptyset \in \mathcal{R}^{\mathcal{Q}}_{\forall}(\mathcal{C}[p, x])$, so the Conflict Condition ($L_\bot$) holds. Since there is no decision literal on the trail, no driving clause can be constructed, but the empty clause can be learned. After that step, in state (4), $F_\bot$ holds, therefore the only rule that can be applied is $\text{Final}_{\text{CNF}}$, that yields the state $\bot$.

It is not hard to see, that if the initial formula would have been $\forall x \exists y.\ x \Leftrightarrow y$ (instead of $\exists x \forall y.\ x \Leftrightarrow y$), then the derivation of state $\top$ could have been the dual of the above steps (i.e. apply rules $\text{Unit}_\forall, \text{Learn}_{\text{DNF}}, \text{Final}_{\text{DNF}}$ instead of the rules $\text{Unit}_\exists, \text{Learn}_{\text{CNF}}, \text{Final}_{\text{CNF}}$ respectively).

Example 2. *Consider the initial state of the abstract solver as $\emptyset \parallel \mathcal{D} \parallel \mathcal{C}$, where $\mathcal{Q} = \forall x \exists y \forall z$, $\mathcal{C} = (x \vee y) \wedge (\neg x \vee \neg y)$ and $\mathcal{D} = (x \wedge \neg y \wedge z) \vee (x \wedge \neg y \wedge \neg z) \vee (\neg x \wedge y \wedge z) \vee (\neg x \wedge y \wedge \neg z)$. A possible derivation from that state would be as follows.*

$$\begin{array}{llll r}
\emptyset & \parallel \mathcal{D} & \parallel \mathcal{C} & \vdash_{\text{Decide}} & (1) \\
x^d & \parallel \mathcal{D} & \parallel \mathcal{C} & \vdash_{\text{Pure}_\exists} & (2) \\
x^d\,\neg y & \parallel \mathcal{D} & \parallel \mathcal{C} & \vdash_{\text{Unit}_\forall} & (3) \\
x^d\,\neg y\,\neg z & \parallel \mathcal{D} & \parallel \mathcal{C} & \vdash_{\text{Learn}_{\text{DNF}}} & (4) \\
x^d\,\neg y\,\neg z & \parallel \mathcal{D} \vee x & \parallel \mathcal{C} & \vdash_{\text{Undo}} & (5) \\
\emptyset & \parallel \mathcal{D} \vee x & \parallel \mathcal{C} & \vdash_{\text{Unit}_\forall} & (6) \\
\neg x & \parallel \mathcal{D} \vee x & \parallel \mathcal{C} & \vdash_{\text{Pure}_\exists} & (7) \\
\neg x\,y & \parallel \mathcal{D} \vee x & \parallel \mathcal{C} & \vdash_{\text{Unit}_\forall} & (8) \\
\neg x\,y\,\neg z & \parallel \mathcal{D} \vee x & \parallel \mathcal{C} & \vdash_{\text{Learn}_{\text{DNF}}} & (9) \\
\neg x\,y\,\neg z & \parallel \mathcal{D} \vee x \vee \emptyset & \parallel \mathcal{C} & \vdash_{\text{Final}_{\text{DNF}}} & (10) \\
 & & \top & & (11)
\end{array}$$

Initially, the only possible step is to apply rule Decide*, since $\neg P \wedge \neg F \wedge \neg L \wedge \neg D$ holds. The only variable satisfying the* *side condition of rule* Decide *is x. Assume it is decided to be true. Then in state (2) there is $\neg y$ as existential unit in $\mathcal{R}_{\forall}^{\mathcal{Q}}(\mathcal{C}[x])$ and at the same time it is pure in $\mathcal{R}_{\exists}^{\mathcal{Q}}(\mathcal{D}[x])$. Assume that pure literal propagation has higher priority and the solver propagates then $\neg y$ using rule* $\mathrm{Pure}_{\exists}$.

In state (3), although all clauses are satisfied in $\mathcal{C}$ by the current assignment, there is neither an empty clause nor empty cube in $\mathcal{C}$ or $\mathcal{D}$, so the Leaf Condition does not hold. But, there are z and $\neg z$ as universal units in cubes $(x \wedge \neg y \wedge z)$ and $(x \wedge \neg y \wedge \neg z)$ respectively. Assume that the solver decides to use the first cube for propagation, which extends the current assignment with $\neg z$. With this step (in state (4)) the second cube becomes empty. Therefore $L_{\top}$ (and thus L) holds. Since there is no driving cube in the DNF (otherwise in the very first step unit propagation instead of decision would have been possible), the only rule that is applicable is $\mathrm{Learn}_{\mathrm{DNF}}$. *The solver learns the driving cube x, which yields state (5). Now there is a driving cube, so rule* Undo *is applicable to backtrack. In state (6) the recently learned cube becomes universal unit, therefore* $\mathrm{Unit}_{\forall}$ *is a valid step and extends the current assignment with $\neg x$. Then, there is y as existential unit in $\mathcal{R}_{\forall}^{\mathcal{Q}}(\mathcal{C}[\neg x])$ and as pure in $\mathcal{R}_{\exists}^{\mathcal{Q}}(\mathcal{D}[\neg x])$. Application of rule* $\mathrm{Pure}_{\exists}$ *yields state (8). Just like in state (2), z and $\neg z$ are universal units, but this time in the cubes $(\neg x \wedge y \wedge z)$ and $(\neg x \wedge y \wedge \neg z)$ respectively. After unit propagation, there is an empty cube in $\mathcal{R}_{\exists}^{\mathcal{Q}}(\mathcal{D}[\neg x, y, \neg z])$, so $L_{\top}$ holds. This time there is no universal decision literal on the trail, thus the only cube to learn is $\emptyset$. Then $F_{\top}$ (and so F) holds, thus, finally, the only allowed step is* $\mathrm{Final}_{\mathrm{DNF}}$, *which terminates the derivation.*