Solution extraction from long-distance resolution proofs *

Uwe Egly and Magdalena Widl

Institute of Information Systems, Vienna University of Technology, Austria
{uwe,widl}@kr.tuwien.ac.at

1 Introduction

Much effort has been devoted in the past decade to the development of decision procedures for quantified Boolean formulas (QBFs) in order to improve their performance. Besides solving the decision problem of whether a closed QBF is true and returning yes or no, modern QBF solvers can produce clause or cube proofs to certify their answer. These proofs can be used to derive solutions to problems formulated as QBFs. For instance, if the QBF describes a synthesis problem, the solution to this problem can be generated from the proof. Examples for representations of such solutions are control circuits (manifested in Herbrand or Skolem functions) or control strategies. We review two approaches for solution generation below. We focus on false QBFs and clause Q-refutations. For true QBFs, both approaches work in a symmetric fashion.

Balabanov and Jiang [1] propose to extract a Herbrand function for each universal ($\forall$) variable from a Q-refutation of a false QBF. These functions are constructed in such a way that replacing each $\forall$ variable by its Herbrand function in the matrix and removing the prefix yields an unsatisfiable Boolean formula. Unsatisfiability can be checked by a common SAT solver if required. Their algorithm traverses the refutation and constructs a formula for each $\forall$ variable from the nodes of the refutation where the variable is reduced by universal reduction. The run-time of their extraction algorithm is polynomial in *the size of the refutation*.

Goultiaeva et al. [3] employ a game-theoretic view to QBFs. They present an algorithm which executes a winning strategy for the universal player from the refutation. The existential ($\exists$) player chooses a move, i.e., an assignment of truth values to all $\exists$ variables in the outermost quantifier block. The algorithm modifies the Q-refutation according to this move and removes the current quantifier block from the prefix. From the resulting refutation, the $\forall$ player computes her move, i.e., an assignment of truth values to all $\forall$ variables in the outermost quantifier block which is now universal. Modifications take place as above and the $\exists$ player continues. The run-time of this algorithm is also polynomial in *the size of the refutation*.

For both methods, their runtime is directly related to the refutation size. It is therefore desirable to have short refutations. Since there are families of QBFs (like the one from Kleine Büning et al. [4] discussed below), for which any Q-refutation is exponential,

* This work was supported by the Austrian Science Foundation (FWF) under grant S11409-N23 and by the Vienna Science and Technology Fund (WWTF) through project ICT10-018.

the use of more powerful calculi to allow more succinct refutations is essential for fast extraction of Herbrand functions or strategies.

A way to strengthen the Q-resolution calculus is to add the additional inference rules presented in [1]. The resulting calculus is called *long-distance resolution* (LDQ-resolution) [9]. Its inference rules are reproduced in Appendix A as rules r_1 to r_4 and u_1. LDQ-resolution allows to resolve two clauses upon an $\exists$ pivot p resulting in a resolvent with complementary pairs of $\forall$ literals provided each pair's variable has a higher quantifier level than p. The derived clause is thus tautological.

This paper is intended to advertise the use of LDQ-resolution. First, LDQ-resolution may greatly simplify the clause learning component [2,6] in current QBF solvers. In this component, additional effort (in terms of Q-resolution steps) is required in order to avoid the generation of tautological clauses over $\forall$ literals. This additional effort often results in an increase of refutation size. Second, we show that LDQ-resolution has the potential to produce shorter refutations. More precisely, we show an exponential speed-up in refutation size compared to Q-resolution using the family $(\varphi_t)_{t\geq 1}$ of false QBFs introduced by Kleine Büning et al. in [4]. Third, we show that the strategy extraction algorithm in [3] can handle LDQ-refutations. Therefore, the performance of this algorithm directly benefits from exponentially shorter LDQ-refutations. Finally, we point out open questions with respect to the certificate extraction procedure in [1].

2 Short proofs for hard formulas

Let $(\varphi_t)_{t\geq 1}$ be a family of QBFs in PCNF. The formula φ_t has prefix

$$\exists d_0 d_1 e_1 \forall x_1 \exists d_2 e_2 \forall x_2 \exists d_3 e_3 \cdots \forall x_{t-1} \exists d_t e_t \forall x_t \exists f_1 \cdots f_t$$

and the following clauses in the matrix:

K_0	$\overline{d_0}$	K_1	$d_0 \vee \overline{d_1} \vee \overline{e_1}$	
K_{2j}	$d_j \vee \overline{x_j} \vee \overline{d_{j+1}} \vee \overline{e_{j+1}}$	K_{2j+1}	$e_j \vee x_j \vee \overline{d_{j+1}} \vee \overline{e_{j+1}}$	$j = 1, \ldots, t-1$
K_{2t}	$d_t \vee \overline{x_t} \vee \overline{f_1} \vee \cdots \vee \overline{f_t}$	K_{2t+1}	$e_t \vee x_t \vee \overline{f_1} \vee \cdots \vee \overline{f_t}$	
B_{2j-1}	$x_j \vee f_j$	B_{2j}	$\overline{x_j} \vee f_j$	$j = 1, \ldots, t$

Kleine Büning et al. prove in [4, Theorem 3.2] that any Q-refutation for φ_t is exponential in t. Van Gelder [8] shows that there exists a short Q-resolution refutation for formulas of this class if resolution over universal variables is allowed. We show that φ_t has polynomial size LDQ-refutations. They have the form of a directed acyclic graph (DAG). Following the notation in [1], we will use x^* as a shorthand for $x \vee \overline{x}$.

1. Derive $d_t \vee \overline{x_t} \vee \bigvee_{i=1}^{t-1} \overline{f_i}$ from B_{2t} and K_{2t}. Derive $e_t \vee x_t \vee \bigvee_{i=1}^{t-1} \overline{f_i}$ similarly.
2. Use both clauses from Step 1 together with $K_{2(t-1)}$ and derive the clause $d_{t-1} \vee \overline{x_{t-1}} \vee \bigvee_{i=1}^{t-1} \overline{f_i} \vee x_t^*$. Observe that the quantification level of d_t and e_t is less than the level of x_t. Use $B_{2(t-1)}$ to get $d_{t-1} \vee \overline{x_{t-1}} \vee \bigvee_{i=1}^{t-2} \overline{f_i} \vee x_t^*$. Derive the clause $e_{t-1} \vee x_{t-1} \vee \bigvee_{i=1}^{t-2} \overline{f_i} \vee x_t^*$ in a similar way.
3. Iterate the procedure to derive $d_2 \vee \overline{x_2} \vee \bigvee_{i=1}^{1} \overline{f_i} \vee \bigvee_{i=3}^{t} x_i^*$ as well as $e_2 \vee x_2 \vee \bigvee_{i=1}^{1} \overline{f_i} \vee \bigvee_{i=3}^{t} x_i^*$.

4. With K_2, derive $d_1 \vee \overline{x_1} \vee \overline{f_1} \vee \bigvee_{i=2}^{t} x_i^*$. Use B_2 to obtain $d_1 \vee \overline{x_1} \vee \bigvee_{i=2}^{t} x_i^*$. Derive $e_1 \vee x_1 \vee \bigvee_{i=2}^{t} x_i^*$ in a similar fashion.
5. Use the two derived clauses together with K_0 and K_1 to obtain $\bigvee_{i=1}^{t} x_i^*$, which can be reduced to $\square$.

The number of clauses in this refutation is in $O(t)$.

We have identified formulas which have short LDQ-refutations, but any Q-refutation of the same formula is exponential. Hence the processing of the short refutations by extraction algorithms can be much faster provided the algorithms can handle such a refutation without too much additional complication.

3 Strategy extraction from long-distance resolution

We show that the strategy extraction method from Q-refutations presented by Goultiaeva et al. in [3] can be applied in the same manner to LDQ-refutations.

As sketched in the introduction, the method is described as a game between a $\forall$ player and an $\exists$ player. The game proceeds through the quantifier prefix from the left to the right alternating the two players according to the quantifier blocks. The $\exists$ player arbitrarily chooses an assignment to the variables in the current block. The assignment for the $\forall$ player is then computed according to previous assignments as follows.

First, all leaf nodes of the refutation, the clauses of the input formula, are modified according to the existential assignment as defined below similarly to [3,7].

Definition 1. *A (partial)* assignment *to a set V of variables of a QBF in PCNF is a set σ of literals of V such that if $l \in \sigma$ then $\bar{l} \notin \sigma$. A clause K under an assignment σ is denoted $K_{\lceil\sigma}$ and defined as follows: $K_{\lceil\sigma} = \top$ if $K \cap \sigma \neq \emptyset$ and $K_{\lceil\sigma} = K \setminus \{q \mid q \in K \wedge \overline{q} \in \sigma\}$ otherwise.*

Second, the inner nodes and $\square$ are derived by applying LDQ-resolution rules or, in cases where the pivot element has been removed by the assignment, by applying additional sound derivation rules (P-rules) presented by Goultiaeva et al. [3] and reproduced in Appendix A. Then the refutation, now containing rules outside the LDQ-resolution calculus, is transformed back into an LDQ-refutation. This transformation results in each $\forall$ variable of the next quantifier block to be reduced at most once. At the $\forall$ player's turn, the assignment to each of these variables is determined by either choosing the opposite polarity of the variable's literal removed in the remaining universal reduction or any polarity in case the variable is not reduced at all. It is shown in [3] that for a Q-refutation, after each restriction of either the $\exists$ or the $\forall$ player, $\square$ is derived.

We show that this method also works for LDQ-refutations. First, we show in a fashion similar to [3,7] that in a refutation applying LDQ-rules and P-rules, each node generated from the restricted leaves subsumes the restriction of the node by the partial assignment. The proof of the following lemma can be found in Appendix A.

Lemma 1. *(cf. Lemma 2.6 in [7]) Given a QBF in PCNF $\psi = \exists V \mathcal{P} \phi$ with V the set of all variables of the outermost quantifier block, $\mathcal{P}$ the prefix of ψ without $\exists V$, and ϕ the matrix of ψ, let K, K^l and K^r be clauses of ϕ, and σ an assignment to V. Then it holds that $\mathsf{res}(K^l_{\lceil\sigma}, K^r_{\lceil\sigma}, p) \subseteq \mathsf{res}(K^l, K^r, p)_{\lceil\sigma}$ and $\mathsf{red}(K_{\lceil\sigma}, x) \subseteq \mathsf{red}(K, x)_{\lceil\sigma}$.*

Using this lemma, we show with an argument similar to Goultiaeva et al. [3], that (1) by applying the restriction rules with respect to an assignment to the variables of an $\exists$ quantifier block to an LDQ-refutation, the resulting proof derives $\square$, and (2) that further restricting the LDQ-refutation by the computed assignment to the $\forall$ variables of the next turn also derives $\square$. (More details can be found in Appendix A.)

The key reason why this works for LDQ-refutations in the same fashion as for Q-refutations is the following. In a restricted refutation, the applications of rules deriving tautologies of some $\forall$ variables (LD-steps, r_2 to r_4 in Appendix A) are always removed by the partial assignment to $\exists$ variables of the previous quantifier blocks. This is the case because an LD-step can result in a tautology $x \vee \overline{x}$ of some $\forall$ variable x only if the pivot element p (an $\exists$ variable) has a lower quantifier level than x. Thus before the $\forall$ player's turn, the pivot element of each LD-step that results in tautologies involving $\forall$ variables of the respective quantifier block is contained in the partial assignment. Therefore, either of the parents of the LD-step is set to $\top$, and by applying the corresponding derivation rule, only one polarity of the $\forall$ variable is left in the derived clause.

4 Conclusion and future work

We have shown an exponential speed-up in refutation size if LDQ-resolution instead of Q-resolution is employed to any QBF φ_t belonging to the family introduced in [4]. Given the fact that state of the art methods for the extraction of Herbrand functions or strategies require traversing the refutation, it is desirable to retrieve LDQ-refutations instead of Q-refutations from QBF solvers in order to speed up such extraction methods. We have further shown that strategies can be retrieved from LDQ-refutations by applying the state of the art algorithm described in [3]. This implies an exponential speed-up for the strategy extraction from any QBF in the family $(\varphi_t)_{t\geq 1}$.

It remains an open question how Herbrand functions can be extracted efficiently from LDQ-refutations. It is possible to build Boolean formulas from truth tables generated by the strategy extraction method in [3]. However, since each possible assignment to the existential variables has to be considered, this naive method is exponential in the quantifier prefix.

Further, to the best of our knowledge, a workflow including output and verifiation of LDQ-resolution proofs is currently not supported by state of the art QBF solvers. (According to Van Gelder [8], QuBE-cert produces tautological clauses, but it is unclear whether the LDQ calculus is used.) In QBF solvers based on clause learning such as DepQBF [5], the generation of tautological clauses is avoided by additional resolution steps which remove existential variables with a higher quantifier level in order to enable a separate universal reduction for each problematic universal variable. By allowing long distance resolution steps, the learning component could be simplified and the obtained refutations could be shorter.

References

1. V. Balabanov and J.-H. R. Jiang. Unified QBF certification and its applications. *Formal Methods in System Design*, 41(1):45–65, Apr. 2012.

2. E. Giunchiglia, M. Narizzano, and A. Tacchella. Clause/Term Resolution and Learning in the Evaluation of Quantified Boolean Formulas. *Journal of Artificial Intelligence Research (JAIR)*, 26:371–416, Sept. 2006.
3. A. Goultiaeva, A. Van Gelder, and F. Bacchus. A uniform approach for generating proofs and strategies for both true and false QBF formulas. In *International joint conference on Artificial Intelligence (IJCAI)*, IJCAI'11, pages 546–553. AAAI Press, 2011.
4. H. Kleine Büning, M. Karpinski, and A. Flögel. Resolution for Quantified Boolean Formulas. *Information and Computation*, 117(1):12–18, Feb. 1995.
5. F. Lonsing and A. Biere. DepQBF: A Dependency-Aware QBF Solver (System Description). *Journal on Satisfiability, Boolean Modeling and Computation (JSAT)*, 7:71–76, 2010.
6. F. Lonsing, U. Egly, and A. Van Gelder. Efficient clause learning for quantified boolean formulas via QBF pseudo unit propagation. In *Theory and Application of Satisfiability Testing (SAT)*, 2013.
7. A. Van Gelder. Input Distance and Lower Bounds for Propositional Resolution Proof Length. In *Theory and Application of Satisfiability Testing (SAT)*, 2005.
8. A. Van Gelder. Contributions to the Theory of Practical Quantified Boolean Formula Solving. In M. Milano, editor, *Principles and Practice of Constraint Programming (CP)*, Lecture Notes in Computer Science, pages 647–663. Springer Berlin Heidelberg, 2012.
9. L. Zhang and S. Malik. Towards a Symmetric Treatment of Satisfaction and Conflicts in Quantified Boolean Formula Evaluation. In P. Hentenryck, editor, *Principles and Practice of Constraint Programming (CP)*, volume 2470 of *Lecture Notes in Computer Science*, pages 200–215. Springer Berlin Heidelberg, 2006.

A Strategy extraction from LDQ-resolution proofs

We write e for an existential variable, x (with or without subscript) for a universal variable, x^* for $x \vee \overline{x}$, $\mathsf{var}(q')$ for the variable of a literal $q' \in \{q, \overline{q}\}$, $\mathsf{lev}(q')$ for the quantifier level of the variable $\mathsf{var}(q')$ counting from the left to the right of the prefix, K^l, K^r, and K for leaf clauses or derived clauses, p for the existential pivot variable, and $\square$ for the empty clause. By QBF we refer to a QBF in PCNF. An LDQ-refutation is an LDQ-resolution derivation of $\square$.

The following reproduces the rules of the LDQ-resolution calculus (LDQ-rules) presented by Balabanov and Jiang [1] with the differences that universal reduction is not integrated in the resolution rules but represented by a separate rule and that only one universal literal is reduced in one step.

$$[p]\ \frac{K^l \vee p \qquad K^r \vee \overline{p}}{K^l \vee K^r} \tag{r_1}$$

$$[p]\ \frac{K^l \vee p \vee x \qquad K^r \vee \overline{p} \vee \overline{x}}{K^l \vee K^r \vee x^*} \quad \mathsf{lev}(p) < \mathsf{lev}(x) \tag{r_2}$$

$$[p]\ \frac{K^l \vee p \vee x \qquad K^r \vee \overline{p} \vee x^*}{K^l \vee K^r \vee x^*} \quad \mathsf{lev}(p) < \mathsf{lev}(x) \tag{r_3}$$

$$[p]\ \frac{K^l \vee p \vee x^* \qquad K^r \vee \overline{p} \vee x^*}{K^l \vee K^r \vee x^*} \quad \mathsf{lev}(p) < \mathsf{lev}(x) \tag{r_4}$$

$$[x]\ \frac{K \vee x'}{K} \quad \begin{array}{l}\text{for } x' \in \{x, \overline{x}, x^*\} \text{ and} \\ \text{for all } e \in K \text{ it holds that } \mathsf{lev}(e) < \mathsf{lev}(x')\end{array} \tag{u_1}$$

As in [1], we extend universal reduction to delete also x^* and call x^* the merged variable in r_2, r_3, r_4, because $x, \overline{x}$ occur in both parent clauses, but they are merged into one occurrence of x^* in the resolvent. We require the restriction on the quantifier level in these rules to also apply to other merged variables in K^l and K^r. In r_1, K^l and K^r may contain $x, \overline{x}$ or x^* but *no merging is allowed*.

We define a (partial) assignment and a clause under a (partial) assignment similarly to [3,7]:

Definition 1. *A (partial)* assignment *to a set V of variables of a QBF in PCNF is a set σ of literals of V such that if $l \in \sigma$ then $\overline{l} \notin \sigma$. A clause K under an assignment σ is denoted $K_{\lceil\sigma}$ and defined as follows: $K_{\lceil\sigma} = \top$ if $K \cap \sigma \neq \emptyset$ and $K_{\lceil\sigma} = K \setminus \{q \mid q \in K \wedge \overline{q} \in \sigma\}$ otherwise.*

The algorithms `play` and `restrict` describe the algorithm presented by Goultiaeva et al. [3]. `play` implements the alternating turns of the universal ($\forall$) and the existential ($\exists$) player moving from the left to the right in the quantifier prefix. Each player chooses an assignment to the variables in the current quantifier block. The proof is modified after each assignment and results in an LDQ-refutation of the QBF under

the partial assignment. The $\exists$ player chooses an arbitrary assignment and the $\forall$ player chooses an assignment that depends on the current modified proof.

The modification of the LDQ-resolution consists of two steps represented by the procedures `restrict` and `transform`. First, `restrict` restricts the proof according to the chosen assignment. It modifies the leaves of the proof by applying the assignment according to Definition 1 and then modifies the successor nodes by either applying one of the LDQ-resolution rules described above or, if the pivot variable has been removed from at least one of the parents, by applying one of the additional rules presented in [3,7]. The additional rules (P-rules) are reproduced in the following (symmetric rules are omitted):

$$[p]\ \frac{K^l \vee p \qquad \top}{\top} \tag{r_5}$$

$$[p]\ \frac{K^l \vee p \qquad K^r}{K^r} \quad \overline{p} \notin K^r \tag{r_6}$$

$$[p]\ \frac{K^l \qquad K^r}{\mathsf{narrower}(K^l,K^r)} \quad \overline{p}, p \notin K^l \text{and } \overline{p}, p \notin K^r \tag{r_7}$$

$$[x]\ \frac{K}{K} \quad x \notin K \tag{u_2}$$

$$[x]\ \frac{\top}{\top} \tag{u_3}$$

where $\mathsf{narrower}(K^l, K^r)$ returns the clause containing less literals. If K^l and K^r contain the same number of literals, K^l is returned. Observe that $\square$ is the narrowest clause and $\top$ contains all literals.

Note that, even though each P-rule represents a sound derivation step, applying P-rules results in a proof outside the LDQ-calculus. Starting at the leaves of the proof, the function `transform` (Π^p), where Π^p is a proof containing clauses derived using LDQ-rules and P-rules, removes parts of the proof that have become redundant due to the application of P-rules. Thereby also all applications of P-rules are removed. More specifically, for node $K = \top$ derived by rule r_5, its parent $K^l \vee p$ is removed with all its ancestors, and K is merged with its other parent $\top$. A similar procedure is applied to nodes derived by rules r_6 and r_7. Each node derived by rules u_1 to u_3 is merged with its parent. When $\square$ is derived, the procedure stops and all nodes that are not ancestors of $\square$ are removed. $\top$-clauses are eliminated by eventually applying rule r_6 or by removing nodes when $\square$ is found. The resulting refutation is an LDQ-refutation.

After restricting the LDQ-refutation by a computed assignment $\sigma_\forall$ to $\forall$ variables, only one redundant clause, namely $\square$, has to be removed because after the previous restriction by $\sigma_\exists$ and the following repair, the refutation contains at most one universal reduction for any universal variable x of the current quantifier block. This is the case because the condition for x to be reduced from any clause K is that K does not contain any $\exists$ variables with a quantifier level higher than x. Any $\exists$ variable in K is therefore in $\sigma_\exists$, and after restricting the refutation, K is either (1) set to $\top$ or (2) all $\exists$ variables are removed. The topologically first clause (furthest to $\square$) where (2) happens derives $\square$ after reducing each of its $\forall$ variables. An assignment $\sigma_\forall$ to a $\forall$ variable x as computed in Line 8 of Algorithm 1 chooses the opposite polarity of x when it is reduced in node K. This way, according to Definition 1, K becomes $\square$ and only the $\square$ derived from K has to be removed.

In the following we present a proof that `restrict`, when called in Line 4 or Line 9 of `play`, returns a derivation of $\square$ using LDQ-rules and P-rules. Proposition 1 shows that this holds for an arbitrary assignment to all $\exists$ variables in the outermost quantifier block, and Proposition 2 shows the same for the computed assignment to $\forall$ variables.

We write $\mathsf{res}(K^l,K^r,p)$ for a resolution step over pivot element p according to rules r_1 to r_7, and $\mathsf{red}(K,x)$ for a reduction step reducing variable x according to rules u_1 to u_3.

We start by showing that any node generated by applying an LDQ-rule or a P-rule from the restricted leaves subsumes the restriction of the node by the partial assignment.

Lemma 1. *(cf. Lemma 2.6 in [7]) Given a QBF in PCNF $\psi = \exists V \mathcal{P}\phi$ with V the set of all variables of the outermost quantifier block, $\mathcal{P}$ the prefix of ψ without $\exists V$, and ϕ the matrix of ψ, let K, K^l and K^r be clauses of ϕ, and σ an assignment to V. Then it holds that $\mathsf{res}(K^l_{\lceil\sigma}, K^r_{\lceil\sigma}, p) \subseteq \mathsf{res}(K^l, K^r, p)_{\lceil\sigma}$ and $\mathsf{red}(K_{\lceil\sigma}, x) \subseteq \mathsf{red}(K, x)_{\lceil\sigma}$.*

Proof. We distinguish between each case of K containing or not containing literals of σ.

$C_0(\sigma, K)$: $\exists q \in \sigma$ such that $q \in K \wedge \overline{q} \in K$. This case never happens because no tautologies over $\exists$ variables are allowed.
$C_1(\sigma, K)$: $\forall q \in \sigma$ it holds that $q \notin K$ and $\overline{q} \notin K$. Then $K_{\lceil\sigma} = K$.
$C_2(\sigma, K)$: $\exists q \in \sigma$ such that $q \in K$. Then $K_{\lceil\sigma} = \top$.
$C_3(\sigma, K)$: $\forall q \in \sigma$ it holds that $q \notin K$ and $\exists q \in \sigma$ such that $\overline{q} \in K$. Then $K_{\lceil\sigma} = K \setminus \{\overline{q} \mid q \in \sigma\}$.

For resolution steps, either of the following cases applies (symmetric cases are omitted):
(1) $C_1(\sigma, K^l)$ and $C_1(\sigma, K^r)$: $K^l_{\lceil\sigma} = K^l$, $K^r_{\lceil\sigma} = K^r$, and $\mathsf{res}(K^l, K^r, p)_{\lceil\sigma} = \mathsf{res}(K^l, K^r, p)$. By applying rule r_1, r_2, r_3, or r_4, we obtain $\mathsf{res}(K^l_{\lceil\sigma}, K^r_{\lceil\sigma}, p) = \mathsf{res}(K^l, K^r, p)$. Therefore the subset relation holds.
(2) $C_2(\sigma, K^l)$ and $C_1(\sigma, K^r)$: $K^l_{\lceil\sigma} = \top$, $K^r_{\lceil\sigma} = K^r$, and $\mathsf{res}(K^l, K^r, p)_{\lceil\sigma} = \top$. By applying rule r_5, we obtain $\mathsf{res}(K^l_{\lceil\sigma}, K^r_{\lceil\sigma}, p) = \top$. Therefore the subset relation holds.
(3) $C_3(\sigma, K^l)$ and $C_1(\sigma, K^r)$: Since $C_1(\sigma, K^r)$, $\forall q \in \sigma$ it holds that $p \neq q$ and $p \neq \overline{q}$. Then $K^l_{\lceil\sigma} = K^l \setminus \{\overline{q} \mid q \in \sigma\}$, $K^r_{\lceil\sigma} = K^r$, and $\mathsf{res}(K^l, K^r, p)_{\lceil\sigma} = K^l \cup K^r \setminus \{\overline{q} \mid q \in \sigma\}$. By applying rule r_1, r_2, r_3, or r_4, we obtain $\mathsf{res}(K^l_{\lceil\sigma}, K^r_{\lceil\sigma}, p) = K^l \cup K^r \{\overline{q} \mid q \in \sigma\}$. Therefore the subset relation holds.
(4) $C_2(\sigma, K^l)$ and $C_2(\sigma, K^r)$: $K^l_{\lceil\sigma} = \top$, $K^r_{\lceil\sigma} = \top$, and $\mathsf{res}(K^l, K^r, p)_{\lceil\sigma} = \top$. By applying rule r_5, we obtain $\mathsf{res}(K^l_{\lceil\sigma}, K^r_{\lceil\sigma}, p) = \top$. Therefore the subset relation holds.
(5) $C_2(\sigma, K^l)$ and $C_3(\sigma, K^r)$: Then $K^l_{\lceil\sigma} = \top$, and $K^r_{\lceil\sigma} = K^r \setminus \{\overline{q} \mid q \in \sigma\}$. We have to distinguish two cases:
(a) $\exists q \in \sigma$ such that $\mathsf{var}(q) = p$. Then $\mathsf{res}(K^l, K^r, p)_{\lceil\sigma} = (K^l \cup K^r) \setminus \{\overline{q} \mid q \in \sigma\}$. By applying rule r_6, we obtain $\mathsf{res}(K^l_{\lceil\sigma}, K^r_{\lceil\sigma}, p) = K^r \setminus \{\overline{q} \mid \overline{q} \in \sigma\}$. Therefore the subset relation holds.
(b) Otherwise (σ does not contain the pivot p). Then $\mathsf{res}(K^l, K^r, p)_{\lceil\sigma} = \top$. By applying rule r_5, we obtain $\mathsf{res}(K^l_{\lceil\sigma}, K^r_{\lceil\sigma}, p) = \top$. Therefore the subset relation holds.
(6) $C_3(\sigma, K^l)$ and $C_3(\sigma, K^r)$ $K^l_{\lceil\sigma} = K^l \setminus \{\overline{q} \mid q \in \sigma\}$, $K^r_{\lceil\sigma} = K^r \setminus \{\overline{q} \mid q \in \sigma\}$, and $\mathsf{res}(K^l, K^r, p)_{\lceil\sigma} = K^l \cup K^r \setminus \{\overline{q} \mid q \in \sigma\}$. By applying rule r_1, r_2, r_3, or r_4, we obtain $\mathsf{res}(K^l_{\lceil\sigma}, K^r_{\lceil\sigma}, p) = K^l \cup K^r \setminus \{\overline{q} \mid q \in \sigma\}$. Therefore the subset relation holds.

For reduction steps, either of the following cases applies:
(7) $C_1(\sigma, K)$: $K_{\lceil\sigma} = K$ and $\mathsf{red}(K, x)_{\lceil\sigma} = \mathsf{red}(K, x)$. Therefore the subset relation holds.
(8) $C_2(\sigma, K)$: $K_{\lceil\sigma} = \top$ and $\mathsf{red}(K, x)_{\lceil\sigma} = \top$. By applying rule u_3 we obtain $\mathsf{red}(K_{\lceil\sigma}, x) = \top$. Therefore the subset relation holds.
(9) $C_3(\sigma, K)$: $K_{\lceil\sigma} = K \setminus \{\overline{q} \mid q \in \sigma\}$ and $\mathsf{red}(K, x)_{\lceil\sigma} = (K \setminus \{x\}) \setminus \{\overline{q} \mid q \in \sigma\}$. By applying rule u_1, we obtain $\mathsf{red}(K_{\lceil\sigma}, x) = (K \setminus \{x\}) \setminus \{\overline{q} \mid q \in \sigma\}$. Therefore the subset relation holds. □

With respect to the application of rules r_2 to r_4 (LD-steps), we observe the following from the `play` algorithm and cases (1) to (6) in the above lemma: `play` iterates over the quantifier prefix from the left to the right. The condition for a clause to be derived by an LD-step is that the quantifier level of the pivot variable p is lower than the level of the merging variable x^*. If the algorithm is executing the existential quantifier level ℓ, any of the cases (1) to (6) except (5a), i.e. all cases where the pivot remains unassigned, can only happen on an LD-step if $\mathsf{lev}(x^*) - \ell > 1$. That is, $\mathsf{var}(x)$ is not in the next quantifier block. Otherwise (if x^* is on the next quantifier level) p must be assigned in the current turn and case (5a) applies. Case (5a), where p is assigned, can happen on any LD-step. In this case, the LD-step is modified by rule r_6, which removes the merged variable. It is thus always the case that after restricting the refutation according to an existential quantifier block on the ℓ-th level, the refutation does not contain any LD-step generating a variable x^* with $\mathsf{lev}(x^*) = \ell + 1$. Therefore, in the following $\forall$ player's turn, none of the respective $\forall$ variables occurs as tautology.

Lemma 2. *Given a QBF in PCNF $\psi = \exists V \mathcal{P} \phi$ with V the set of all variables of the outermost quantifier block, $\mathcal{P}$ the prefix of ψ without $\exists V$, and ϕ the matrix of ψ, an LDQ-resolution Π deriving a clause K from ψ, and an assignment $\sigma_\exists$ to V, it holds that $\Pi' = \texttt{restrict}(\Pi, \sigma_\exists)$ derives a clause K' from $\mathcal{P}\phi_{\lceil\sigma_\exists}$ such that $K' \subseteq K_{\lceil\sigma_\exists}$.*

Proof. By induction on the structure of Π using Lemma 1. □

Proposition 1. *Given a QBF in PCNF $\psi = \exists V \mathcal{P} \phi$ with V the set of all variables of the outermost quantifier block, $\mathcal{P}$ the prefix of ψ without $\exists V$, and ϕ the matrix of ψ, an LDQ-resolution Π deriving $\square$ from ψ, and an assignment $\sigma_\exists$ to V, it holds that $\Pi^p = \texttt{restrict}(\Pi, \sigma_\exists)$ derives $\square$ from $\mathcal{P}\phi_{\lceil\sigma_\exists}$.*

Proof. By Lemma 2, for any clause K derived in Π it holds that Π' derives a clause K' such that $K' \subseteq K_{\lceil\sigma_\exists}$. Therefore, if $K = \square$, then Π' must derive a clause $K' = \square$. □

Proposition 2. *Given a QBF in PCNF $\psi = \forall V \mathcal{P} \phi$ with V the set of all variables of the outermost quantifier block, $\mathcal{P}$ the prefix of ψ without $\forall V$, and ϕ the matrix of ψ, an LDQ-resolution Π deriving $\square$ from ψ, and an assignment $\sigma_\forall$ to V as computed in Line 8 of Algorithm 1, $\Pi^p = \texttt{restrict}(\Pi, \sigma_\forall)$ derives $\square$ from $\mathcal{P}\phi_{\lceil\sigma_\forall}$.*

Proof. For any $q \in \sigma_\forall$ it holds that q is either not reduced at all, or reduced exactly once in Π. If q is not reduced at all, then it is not involved in Π and therefore its assignment does not alter the proof. Let $R \subseteq \sigma_\forall$ be the set of dual literals that are reduced exactly once in the proof. Then there is a set $\mathcal{K}$ with $|\mathcal{K}| = |R|$ of nodes such that the nodes

in $\mathcal{K}$ are directly following one another, each reducing exactly one literal r in R. The last reduced node of $\mathcal{K}$ results in $\square$. This is the case because all literals of R are in the outermost quantifier block. `restrict` (Π,$\sigma_\forall$) then applies rule u_2 to each clause K, setting each K in $\mathcal{K}$ to $\square$.

$\square$

Algorithm 1: `play`

Input : QBF $\mathcal{P}.\psi$, LDQ-refutation Π

foreach *Quantifier block Q in $\mathcal{P}$ from left to right* **do**
  if *Q is existential* **then**
    $\sigma_\exists \leftarrow$ any assignment to variables in Q
    $\Pi^p \leftarrow$ `restrict` (Π,$\sigma_\exists$) *(Π^p is not an LDQ-resolution)*
    $\Pi \leftarrow$ `transform` (Π^p) *(Π is an LDQ-resolution)*
  else Q is universal
    $K \leftarrow$ topologically first node with no existential literals
    $\sigma_\forall \leftarrow \{\overline{x} \mid x \in K \wedge \mathsf{var}(x) \in Q\} \cup \{x \mid x \notin K \wedge \overline{x} \notin K \wedge \mathsf{var}(x) \in Q\}$
    $\Pi^p \leftarrow$ `restrict` (Π,$\sigma_\forall$)
    $\Pi \leftarrow$ `transform` (Π^p)

Algorithm 2: `restrict`

Input : LDQ-refutation Π, assignment σ for variables in outermost block
Output: Restricted refutation containing LD-rules and P-rules

```
1  foreach leaf node K in Π do
2      K ← K_⌈σ;
3  foreach inner node K topologically in Π do
4      if K is a resolution node then
5          K^l,K^r ← parents of K
6          p ← pivot of K
7          K ← res(K^l,K^r,p)
8      else K is a reduction node
9          K^c ← parent of K
10         x ← variable reduced from K^c
11         K ← red(K^c,p)
12
13 return Π
```