The Complexity of Quantified Conflict Driven Clause Learning

Dissertation

zur Erlangung des akademischen Grades
doctor rerum naturalium (Dr. rer. nat.)

vorgelegt dem Rat der Fakultät für Mathematik und Informatik
der Friedrich-Schiller-Universität Jena

von M. Sc. Benjamin Böhm,
geboren am 18.11.1995 in Gera

Jena, 2025

Reviewers:

1. Prof. Dr. Olaf Beyersdorff, Friedrich Schiller University Jena
2. Prof. Dr. Meena Mahajan, Institute of Mathematical Sciences, Chennai
3. Dr. Friedrich Slivovsky, University of Liverpool

Date of the public thesis defence: 13.06.2025

Ehrenwörtliche Erklärung

Hiermit erkläre ich,

- dass mir die Promotionsordnung der Fakultät bekannt ist,
- dass ich die Dissertation selbst angefertigt habe, keine Textabschnitte oder Ergebnisse eines Dritten oder eigenen Prüfungsarbeiten ohne Kennzeichnung übernommen und alle von mir benutzten Hilfsmittel, persönliche Mitteilungen und Quellen in meiner Arbeit angegeben habe,
- dass ich die Hilfe eines Promotionsberaters nicht in Anspruch genommen habe und dass Dritte weder unmittelbar noch mittelbar geldwerte Leistungen von mir für Arbeiten erhalten haben, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen,
- dass ich die Dissertation noch nicht als Prüfungsarbeit für eine staatliche oder andere wissenschaftliche Prüfung eingereicht habe.

Bei der Auswahl und Auswertung des Materials sowie bei der Herstellung des Manuskripts haben mich folgende Personen unterstützt:
Olaf Beyersdorff, Tomáš Peitl, Meena Mahajan

Ich habe die gleiche, eine in wesentlichen Teilen ähnliche bzw. eine andere Abhandlung* bereits bei einer anderen Hochschule als Dissertation eingereicht: Ja/Nein*.

(*Zutreffendes unterstreichen)
Wenn Ja, Name der Hochschule: n. z.
Ergebnis: n. z.

Jena, den

Unterschrift

Acknowledgements

First and foremost, I am deeply grateful to my advisor Olaf Beyersdorff for the exceptionally pleasant collaboration and his invaluable guidance throughout my academic journey. As I have initially studied mathematics, specialized in Algebra, he convinced me to shift to Theoretical Computer Science and offered me a position in his group after an oral exam (as he often does). He always respected my academic background and helped me adjust to this new thematic direction. I am thankful to Olaf for not only always sharing his professional expertise on QBF, but also encouraging me to broaden my horizons through participation in conferences and workshops. Unfortunately, nearly the first two years of my doctoral studies were marked by the pandemic. Despite these challenging circumstances, which limited participation in conferences and hindered engaging with other researchers, our collaboration remained strong and productive — even in a remote setting. I am truly thankful for this. Lastly, I have always enjoyed our Skat games together. It was always a lot of fun, and I must admit, it was amusing to occasionally have the chance to teach you a few tips on the game for once.

I also thank Tomáš Peitl and Meena Mahajan for all of the interesting discussions about QBF and QCDCL, which resulted in a respectful number of papers. Our collaborations, both in-person and remote, were always very pleasant.

I would like to thank my colleagues Agnes, Marlene, Luc, Tim and Kaspar as well as my former colleagues Joshua, David, Tomáš and Steffen for the daily teamwork and the many interesting discussions, both within and beyond our research seminar. I am also grateful for the enjoyable moments we shared during conference trips — whether in airports, train stations, or exploring new cities together. On a lighter note, I suppose I should point out that I still owe a few of you money for some restaurant visits, and I'm thankful that these debts haven't been called in yet.

A special thank you goes to my former advisor during my mathematics degree, Burkhard Külshammer. Even though I ultimately took a somewhat different thematic direction, I have never forgotten my academic roots, and I remain deeply grateful for his guidance in helping me transition from studying to conducting research.

Last but not least, I would like to thank my family, and especially my parents Petra and Hartmut, for their unwavering support. They always had my back, both in terms of motivation and, during my studies, financially. I suspect that to this day, they still don't fully understand what exactly I do for a living — but that never stopped them from wholeheartedly supporting me in every endeavour.

Contents

1 Introduction **15**

2 SAT and QBF Proof Complexity **25**
2.1 Propositional Formulas 25
2.2 Propositional Resolution 31
2.3 Quantified Boolean Formulas 35
2.4 QBF Resolution 38

3 SAT and QBF Solving **47**
3.1 DPLL 47
3.2 CDCL and QCDCL 49

4 Formalizations of QCDCL **63**

5 Gauge Lower Bound **73**
5.1 Quasi Level-ordered Proofs and Primitivity 74
5.2 Formula Gauge 88
5.3 Applications 90

6 Analysing Standard QCDCL **97**
6.1 Lower Bounds for QCDCL via Primitivity 98
6.2 Plain QCDCL vs. Cube Learning and Pure Literal Elimination 104
6.3 Cube Learning vs. Pure Literal Elimination 112
6.4 Comparison with Q-Res 116

7 Comparing Decision Policies **121**
7.1 Learning Asserting Constraints 123
7.2 Separation on True Formulas 127
7.3 Separation on False Formulas 136

8 Comparing Reduction and Propagation Policies **141**

9 QCDCL Characterisations **151**

10 Runtime vs. Extracted Proof Size of QCDCL **167**
10.1 The Separation Technique 168
10.2 Separations for QCDCL Models 172
10.2.1 QCDCL Based on LD-Q-Res 172
10.2.2 QCDCL Based on Q-Res 175
10.2.3 QCDCL Based on QU-Res 178

11 Conclusion **179**

Abstract

Solving Quantified Boolean Formulas (QBFs) depicts an extension of the well-known SAT problem. As the canonical PSPACE-complete problem, one expects QBF to be even harder than the NP-complete SAT in terms of the hierarchy of complexity classes. Although we cannot assume there being a polynomially bounded algorithm for SAT solving, the main paradigm used in praxis, CDCL ("conflict driven clause learning"), which generates Resolution refutations for unsatisfiable propositional formulas, turns out to have short running times on many industrial instances. 2011 Pipatsrisawat and Darwiche proved that CDCL, formalized as a proof system, is able to simulate Resolution refutations, which results in the equivalence of CDCL and Resolution.

One of the main approaches in QBF solving is extending known SAT solving methods by lifting them to a quantified setting. In particular, CDCL can be lifted to so-called QCDCL. In this thesis, we analyse similarities and differences between CDCL and QCDCL. We identify weaknesses of QCDCL by proving exponential lower bounds and use them to compare and separate different variants of QCDCL. One of the main goals is to find a QCDCL variant that is able to simulate its underlying proof system the same way as CDCL simulates Resolution. By introducing several kinds of QCDCL modifications, we can improve its potential performance in relation to the standard version broadly used in practice.

Zusammenfassung

Das Lösen quantifizierter Boolescher Formeln stellt eine Erweiterung des SAT-Problems dar. Als kanonisches PSPACE-vollständiges Problem ist dieses sogenannte QBF-Problem in der Hierarchie der Komplexitätsklassen als noch schwieriger einzuschätzen als das bereits NP-vollständige SAT-Problem. Obwohl nicht von der Existenz von polynomial zeitbeschränkten Lösungsalgorithmen für SAT auszugehen ist, präsentiert sich das in der Praxis hauptsächlich angewandte Paradigma CDCL ("conflict driven clause learning"), welches Resolutionswiderlegungen von unerfüllbaren Formeln generiert, auf vielen industriellen Instanzen mit kurzen Laufzeiten. 2011 bewiesen Pipatsrisawat und Darwiche, dass das als Beweissystem formalisierte CDCL sogar jede vorgegebene Resolutionswiderlegung simulieren kann, woraus die komplexitätstheoretische Äquivalenz zum wohlbekannten Resolutionskalkül folgt.

Eines der wichtigsten Ansätze zum Lösen quantifizierter Boolescher Formeln ist die Erweiterung von CDCL auf das QBF-Problem, welche QCDCL genannt wird. In dieser Dissertation untersuchen wir, welche Eigenschaften sich von CDCL auf QCDCL übertragen lassen und worin Unterschiede liegen. Wir identifizieren Schwachstellen von QCDCL in Form von exponentiellen unteren Schranken und nutzen diese, um verschiedene Versionen von QCDCL miteinander zu vergleichen. Dabei ist eines der Ziele, eine Version von QCDCL zu finden, welche in der Lage ist ähnlich wie CDCL das zugrundeliegende Beweissystem zu simulieren. Durch die Einführung zahlreicher Modifikationen verbessern wir die potenzielle Leistungsfähigkeit von QCDCL im Vergleich zum derzeit in der Praxis genutzten Standardmodell.

Chapter 1

Introduction

Two-player games are among the oldest known forms of games in human history. One of the oldest known two-player board games is *senet* which was invented more than 5,000 years ago in ancient Egypt. Senet not only served as recreation, but also as a status symbol, occasionally holding religious significance. Roughly explained, senet is a racing game in which the aim is to transport characters from the starting point to the end point, with the possible moves being determined by rolling dice. Although the exact rules of the game have been lost over the years, some have been deciphered and reconstructed so that senet can still be played today — albeit not with the original rules (cf. [85]).

Other ancient board games likely followed a similar approach, such as the *Royal Game of Ur* — named after the Royal Cementery of Ur, where it was excavated in 1926 (cf. [63]) — or *Hounds and Jackals*. All of these games were, to some extent, based on luck rather than purely on players' skill. This distinguishes them from other board games that are still well-known and famous today, such as *chess*, *checkers* or even *tic-tac-toe*. These games are so-called *perfect information games*, in which there are neither any secrets between the two players nor any elements of randomness.

The strategic thinking that is required in games like chess or tic-tac-toe reflects the logical structures of today's computational problems. People often find problems easier to solve when framed as games. For example, the task of proving the infiniteness of natural numbers can be rephrased more intuitively as: "For each finite set of natural numbers, find a natural number that is not included.". Indirect proofs can be likened to two-player games: Player one claims that a given statement is true, while player two tries to lead player one's arguments to contradiction. It is therefore unsurprising that proofs by contradiction are often viewed as more intuitive than direct proofs. Hence, it is mathematically sensible to formulate problems as two-player games (with perfect information).

In the year 1913 Zermelo showed via set theory that such games always have determined game result as long as they are finite (cf. [102]). That means that one of the two players always has a winning strategy, provided that a possible draw is counted as a win for one of the two players. This result is particularly astonishing because propositional logic only began to be of interest to theory in the 1970s. While constructing

and memorizing such a strategy for tic-tac-toe is relatively easy and feasible, the same is obviously not true for chess. Although chess computers are already significantly superior to human chess players, calculating a complete winning strategy still remains unfeasible as of today.

The rules and winning conditions can be encoded as propositional formulas on boolean variables. For example, let us consider the formula

$$(u \vee \neg x) \wedge (\neg u \vee x).$$

It is easy to see that this formula is satisfied if and only if the variable x is assigned the same truth value as u. The well-known *SAT problem*, which involves deciding whether a given propositional formula is satisfiable, is not only *NP-complete* but also considered the canonical NP-complete problem [49].

As an NP-complete problem, SAT is solvable by non-deterministic Turing machines in polynomial time, and all other problems in NP can be efficiently reduced to it. In intuitive terms, verifying a potential solution (an assignment in this case) is feasible in polynomial time, but finding such a solution might not be.

The question of whether or not the SAT problem (or any NP-complete problem) can be solved in polynomial time by *deterministic* Turing machines is one of the major open problems in theoretical computer science as well as one of the seven Millennium Prize Problems[1] with a 1 million dollar prize for a correct proof. Although being an open problem, it is widely believed that this question can be answered with 'no', implying that there is no algorithm that can efficiently decide the satiability of given formulas.

Nevertheless, it turned out that many (mostly industrial) instances can be solved in short time in practice. While in theory there are several approaches for finding satisfying assignments (or proving the unsatisfiability) of propositional formulas, only one paradigm came out of the top and is nowadays pretty much the standard procedure for SAT solving: *Conflict Driven Clause Learning* (*CDCL*).

CDCL [79, 80, 103] can be viewed as an extension of the simpler *DPLL algorithm* (*Davis–Putnam–Logemann–Loveland Algorithm*, cf. [53]). DPLL systematically explores the truth table of a formula by assigning variables via decisions while taking shortcuts where appropriate. CDCL extends this by non-chronological backtracking and by adding new subformulas learned in previous iterations of the algorithm. These learned subformulas, represented as *clauses* (disjunctions of literals), assume that propositional formulas are in conjunctive normal form (CNF), i.e. a conjunction of clauses. CDCL continues to learn new clauses until a satisfying assignment is found, or the empty clause is learned (which proves the unsatisfiability of the original formula). For satisfiable formulas, CDCL returns a satisfying assignment, while for unsatisfiable formulas, it not only returns a negative answer, but also a proof of unsatisfiability consisting of the learned clauses [7] in the well known propositional proof system *Resolution*.

Although performing well in practice, one expects exponential lower bounds in terms of running time on several instances. Finding lower bounds, both for proof systems as

[1] claymath.org/millennium-problems

well as proof generating algorithms, represents one of the main disciples in the field of proof complexity. One naive approach for proving CDCL lower bounds is to search for formulas that need Resolution refutations of exponential size. Since CDCL generates a Resolution refutation, such a lower bound immediately translates to an exponential running time. As for the question whether or not there are more lower bounds that do not originate from Resolution hardness, there is no easy answer. In fact, in [86] it was shown that Resolution not only p-simulates CDCL as a proof system, but one can also extract a CDCL run out of given Resolution refutation in polynomial time. However, this result has to be treated with caution as the CDCL model used in that result is a rather non-deterministic version — meaning that given a Resolution refutation, one can reproduce the proof by making the 'correct' decisions and learning the 'correct' clauses. Obviously, this is not the way a practical solver would operate since decisions and the learning scheme both deeply depend on the heuristics that are applied. Even worse, in [100] it was proven that deterministic CDCL with commonly used heuristics is in fact *not* able to p-simulate Resolution.

One can conclude that choosing a suitable formalization for the solver greatly affects the theoretical results one might obtain. It is a rather philosophical question on which of the two frameworks one should focus. Because Resolution can be interpreted as a non-deterministic proof system as well, we will concentrate on the non-deterministic versions of solvers[2] whenever we compare them with theoretical proof systems like Resolution. This also means that an upper bound for an algorithm only refers to the existence of one single run in polynomial time — without guaranteeing that practical solvers will easily find such a short run.

Returning to perfect information games, we can now formalize their winning conditions using propositional formulas: One of the two players wins by satisfying the formula, while the other wins by falsifying it. Obviously, in order to obtain 'interesting' games, the formula should neither be a tautology nor unsatisfiable. It remains to determine the allowed turns of each player.

Each variable from the propositional formula gets quantified by either an existential or a universal quantifier (i.e., $\exists$ or $\forall$). The quantifiers are written in front of the propositional formula and also define the order in which the variables must be assigned. The player who aims at satisfying the formula controls the existentially quantified variables, while the other one controls the universal variables (we will use the notions *existential player* and *universal player*). In our example above, we could add the quantifier $\forall u \exists x$ and obtain the formula

$$\forall u \exists x \cdot (u \vee \neg x) \wedge (\neg u \vee x).$$

As a two-player game, the universal player has to assign the variable u first. It is easy to see that no matter how u gets assigned, the existential player can win the game by

[2]In the non-deterministic solver framework, certain aspects that are determined in practice using heuristics are not further specified. For example, the CDCL allows certain selection options in terms of decisions and learned clauses because it is not crucial for the solver's correctness to define decision heuristics or learning schemes beforehand.

setting x to the same truth value as u. This is only possible because x got quantified after u and therefore the assignment of x is allowed to depend on u, while u must not depend on anything.

A *Quantified Boolean Formula* (*QBF*) comprises a propositional formula (called the *matrix*) paired with quantifiers (the *prefix*) such that all variables are quantified, leaving no free variables. A QBF can either be true or false. The former is implied by the existence of a winning strategy for the existential player, while the ladder is implied by the existence of a winning strategy for the universal player.

The problem of finding the truth value of a QBF is simply called the QBF problem. As the canonical PSPACE-complete problem [98], it can be solved by Turing machines with polynomial space and arbitrary time. Since NP is contained in PSPACE, we can easily reduce the SAT problem to QBF by quantifying all variables of a propositional formula existentially. Because it is widely believed that NP $\neq$ PSPACE, one can assume that the QBF problem is even harder than SAT.

While some propositional lower bound techniques like *feasible interpolation* [74, 87] can be lifted to QBF [21], or approaches through so-called *Prover-Delayer games* are applicable both for SAT [27, 88] and QBF [24, 25, 26], other lower bounds like the *size-width lower bound* [8] fail for QBFs [20]. Instead, new techniques like *strategy extraction* [2, 3, 4, 18, 59] were established for finding QBF lower bounds [10, 16, 19].

Many SAT solving approaches can be extended to QBF, like QDPLL [56] and QCDCL [58, 104] as extensions of DPLL and CDCL. However, unlike in SAT, QCDCL is not the only paradigm established in QBF solving. Roughly speaking, QBF solvers can be divided into CDCL-based solvers and expansion-based solvers. Expansion is an approach in which a QBF is transformed into a propositional formula via variable expansion and passed to a SAT solver at some point. As naively expanding all variables might obviously result in an exponential blow-up, successful expansion-based solvers like *RAReQS* [65, 66], which are connected to the expansion-based QBF proof systems $\forall$Exp+Res [67, 68], or IR-calc and IRM-calc [18], in which resolution- and expansion-based approaches are conflated, perform more fine-tuned and therefore more complex operations.

However, we will concentrate on CDCL-based approaches in this thesis as we want to draw comparisons to the SAT case and check which results can or cannot be lifted to the QBF level. QCDCL fundamentally follows the same idea as its propositional ancestor: Assign variables via unit propagation or decisions and start clause learning as soon as a conflict occurs. An integral part of not only QCDCL but many QBF proof systems such as Q-Resolution is universal reduction, which allows the omission of universal variables from clauses where they are not blocked by existential variables to the right. For example, the clause

$$(x \vee u)$$

under the prefix

$$\exists x \forall u$$

can be reduced to the clause

$$(x),$$

because setting x to 0 would allow the universal player to win the game by setting u to 0. Hence, the existential player has to assume that u is already set to 0 and therefore should set x to 1, ignoring any other assignment.

Another difference between CDCL and its counterpart for QBF is the fact that QCDCL does not stop when a satisfying assignment was found, unlike CDCL. Note that the satisfiability of the matrix does not necessarily imply that the QBF is true. In fact, most QBFs consist of a satisfiable matrix — otherwise the universal player would trivially win the game no matter what. However, in order to ensure completeness and finiteness of the algorithm, QCDCL needs to learn constraints even when its generated assignment has satisfied the matrix. In this case, instead of clauses, the algorithm learns *cubes* (i.e., conjunctions of literals) that represent such a satisfying assignment to some extend. Consequently, QCDCL not only uses clauses but also cubes for unit propagation, whereas cubes behave dual to clauses.

As an example, consider the QBF

$$\forall u, v \exists x \cdot (u \vee v \vee \neg x) \wedge (\neg u \vee \neg v \vee x).$$

Assume that u was assigned to true and v was assigned to false by decision, then the matrix gets satisfied no matter how x is assigned. If x is set to true, then all variables are assigned without falsifying a clause. QCDCL can now learn the cube $(u \wedge \neg v \wedge x)$, which can then existentially reduced to $(u \wedge \neg v)$. This cube is added to the formula as follows:

$$\forall u, v \exists x \cdot [(u \vee v \vee \neg x) \wedge (\neg u \vee \neg v \vee x)] \vee (u \wedge \neg v)$$

This formula is now a so-called *augmented QBF*, which is a QBF whose matrix is a disjunction of a CNF and a *DNF* (i.e., disjunctive normal form). QCDCL works not only on quantified CNFs, but also on such augmented QBFs as it can use both clauses and cubes to generate new assignments via decisions and unit propagation. However, generated refutations will consist of only one type of constraint (depending on the truth value of the input formula). Analogously to false QBFs, the trueness of a QBF can be shown via so-called *Q-Consensus*, which can be described as Q-Resolution on cubes. Learning and deriving the empty cube then proves that the input QBF was true. Consequently, QCDCL stops as soon as either the empty clause or the empty cube is learned.

While, as mentioned above, it was proven that a non-deterministic version of CDCL is equivalent to its underlying system Resolution, the same cannot be said about QCDCL. In fact, the well-known QBF proof system Q-Resolution is not even able to simulate QCDCL runs because Clause Learning sometimes performs 'forbidden' resolution steps, which is why in 2002 Zhang and Malik [104] introduced Long-distance Q-Resolution as an extension of Q-Resolution which can simulate QCDCL runs, which was later formalized as a sound and complete proof system in [3].

Obviously, the purpose of QBFs is not only analysing board games and computing their game value (cf. [94] for a collection of QBF applications). However, no matter how technically complicated such industrial instances are, we can still always interpret them as two-player games with winning strategies for one of the two players.

The question whether QCDCL (as a non-deterministic proof system) can p-simulate Long-distance Q-Resolution was answered 2016 when in [64] it was shown that this is not even possible with Q-Resolution. Although QCDCL is conceptionally similar to CDCL, its behaviour from a complexity-theoretic perspective differs significantly from its predecessor. In this thesis, we want to consider the following questions:

On which QBF instances does QCDCL fail and why? It is obvious that QCDCL needs exponential running time on formulas that are hard for the underlying proof system. However, since classic QCDCL does not p-simulate its underlying proof system, there is expected to be another layer of hardness which can be considered a "genuine QCDCL hardness". We aim to identify such a QCDCL-specific hardness and, ideally, establish a corresponding lower bound technique.

What are already established modifications of QCDCL and how do they impact its performance? Cube Learning and Pure Literal Elimination [44] can be regarded as modifications to the plain QCDCL version. While Cube Learning is normally performed by default in order to be able to deal with true QBFs, Pure Literal Elimination is omitted in most QCDCL solvers. We want to formalize and compare different versions of QCDCL with or without Cube Learning and Pure Literal Elimination and check whether or not previously shown lower bounds still hold.

How can QCDCL be improved with further modifications not yet established? Based on the lower bounds results, we want to introduce further modifications that overcome the weaknesses we exploited in the separations. Again, we compare different variants of QCDCL and show separations and p-simulations in order to propose new and better QCDCL variants for practice.

Can we modify QCDCL such that it becomes equivalent to its underlying proof system? Using the modifications introduced earlier, we want to find out if and which of the QCDCL variants are able to p-simulate their underlying proof systems. At best, we would like to propose a QCDCL variant that is p-equivalent to Long-distance Q-Resolution similar to the p-simulation of Resolution by CDCL (cf. [86]).

Contributions

We start with introducing a formalization for the QCDCL algorithm in Chapter 4 that allows us to treat it as a separate proof system. We describe how we can summarize all information of a QCDCL iteration into a so-called *trail*, which is a representation of the assignment generated in each run (cf. Definition 4.1). We will explicitly define *Clause Learning* and *Cube Learning* (cf. Definition 4.4) as well as *Backtracking* (cf.

Definition 4.5). Each QCDCL run can then be written as a *QCDCL proof*, which we can analyse in terms of size and compare it with other already established proof systems like Q-Resolution. Using this framework, we can show that QCDCL-like proof systems are sound (cf. Theorem 4.9) and complete (cf. Theorem 4.13).

We first introduced the framework in the paper "Understanding the Relative Strength of QBF CDCL Solvers and QBF Resolution" at ITCS'21 [13] and as an extended version 2023 in LMCS [14] — and applied it in all follow-up papers.

In Chapter 5, we set up the most important lower bound technique for our results: The gauge lower bound. Our goal is finding exponential lower bounds for QCDCL. Obviously, as long-distance Q-Resolution simulates QCDCL, lower bounds for long-distance Q-Resolution immediately transfers to QCDCL as well. Beyond that, no lower bound techniques for QCDCL were known so far. We will define sufficient conditions for exponential QCDCL lower bounds which do not necessarily lead to exponential lower bounds for long-distance Q-Resolution. We will show that for formulas Φ of a particular class of QBFs, there exists a measure which we call gauge(Φ) such that Φ needs so-called fully reduced primitive Q-Resolution refutations of size $2^{\Omega(\text{gauge}(\Phi))}$. In order to apply this technique to QCDCL, we need to check if the QCDCL variant in question generates primitive derivations on Φ, which is easily provable for several hand-crafted formulas as we will show in the following chapters.

This chapter is based on the paper "Lower Bounds for QCDCL via Formula Gauge" published at SAT'21 [32] and 2023 in JAR as an extended version [33]. Some results and proofs were revised and do not rely on the algorithm from [32, 33] any more.

In Chapter 6, we apply the gauge lower bound technique to QCDCL variants that are more or less already established in practice. More precisely, we consider QCDCL variants in which we can turn Cube Learning on and off as well as variants with and without a modification called Pure Literal Elimination. We prove that the plain QCDCL version (i.e. without Cube Learning and Pure Literal Elimination) always generates primitive proofs on the class of formulas the gauge lower bound can be applied to (cf. Corollary 6.5). This implies that the gauge lower bound is exactly a lower bound technique for this plain QCDCL version.

Furthermore, we show that, at least on some particular formulas, the gauge lower bound can be applied to the other QCDCL variants as well. We prove that in order to prevent the generation of primitive proofs (and therefore preventing the application of the gauge lower bound), the corresponding trails need to be 'out of order' to some extend, which is only possible if Cube Learning or Pure Literal Elimination have a 'sufficiently large impact' on the shape of the trails (cf. Proposition 6.4).

Using this observation and our lower bound technique, we compare the four QCDCL variants with one another and with Q-Resolution as well as long-distance Q-Resolution. We show the following relations:

- Enabling Cube Learning exponentially strengthens the QCDCL variants (cf. Theorems 6.11 and 6.32).

- Each Variant with Pure Literal Elimination is incomparable to all variants without Pure Literal Elimination (cf. Theorems 6.21,6.26 and 6.29).
- All variants considered in this chapter are incomparable to Q-Resolution (cf. Theorem 6.33)
- All variants considered in this chapter are exponentially weaker than, but p-simulated by long-distance Q-Resolution (cf. Corollary 6.34).

This chapter was first published at IJCAI'22 under the title "QCDCL with Cube Learning or Pure Literal Elimination - What is Best?" [36] and has won the *Distinguished Paper Award*. An extended version was published 2024 in AIJ [38].

After having considered already known QCDCL variants, we introduce several modifications for QCDCL in Chapter 7. We refer to such modifications as *policies*, that can be divided into several categories. In this chapter, we first concentrate on *Decision Policies*. These policies determine which variables we are allowed to chose as decision variables during the creation of a trail. The gauge lower bound we used in previous chapters highly depended on level-ordered decisions, hence it is a natural idea to relax this property.

We will define the three new QCDCL variants determined by the corresponding decision policy:

- $\mathsf{QCDCL}^{\textsc{Any}}$: Decisions can be ordered arbitrarily.
- $\mathsf{QCDCL}^{\textsc{Uni-Any}}$: Existential decisions are level-ordered while universal decisions are ordered arbitrarily.
- $\mathsf{QCDCL}^{\textsc{Exi-Any}}$: Universal decisions are level-ordered while existential decisions are ordered arbitrarily.

First, we will show that all of these variants are still sound and complete (cf. 7.8). Furthermore, we will prove that $\mathsf{QCDCL}^{\textsc{Uni-Any}}$ is guaranteed to find learnable asserting clauses during Clause Learning, while $\mathsf{QCDCL}^{\textsc{Exi-Any}}$ can always detect learnable asserting cubes during Cube Learning (cf. Proposition 7.5). As for $\mathsf{QCDCL}^{\textsc{Any}}$, we can at least guarantee that there always exist learnable new constraints after conflicts (cf. Proposition 7.6).

Although the benefit of asserting constraints was not theoretically verified so far, it can be believed that $\mathsf{QCDCL}^{\textsc{Uni-Any}}$ works better on false formulas (for which Clause Learning is more relevant), while $\mathsf{QCDCL}^{\textsc{Exi-Any}}$ should work better on true formulas. To some extend, this assumption is somewhat witnessed by the following separations. We construct a family of true formulas, which needs exponential-sized proofs in the classic QCDCL, but has polynomial proofs in $\mathsf{QCDCL}^{\textsc{Exi-Any}}$ (cf. Corollary 7.20). To the best of our knowledge, this is the first lower bound of a true QBF for QCDCL. On the other hand, we show that classic QCDCL and $\mathsf{QCDCL}^{\textsc{Uni-Any}}$ can be exponentially separated on false formulas (cf. Corollary 7.28). That means that all three new QCDCL variants are,

in theory, exponentially stronger than the original version, meaning that these policies in fact improve the potential strength of the solver.

This chapter is based on the paper "Should Decisions in QCDCL Follow Prefix Order?" first published at SAT'22 [37] and 2023 in JAR as an extended version [39].

In Chapter 8, we introduce further types of QCDCL policies: Reduction and Propagation Policies. These policies determine how we do Unit Propagation while generating trails. Our new Reduction Policies allow us to reduce literals normally (ALL-RED), arbitrarily (ANY-RED) or turning them off completely (NO-RED). Propagation Policies determine whether we can use clauses for existential Unit Propagations only (EXI-PROP) or allow them to propagate universal literals as well (ALL-PROP). We will show that all policy combinations are sound and complete (cf. Proposition 8.3), although not all combinations make sense necessarily. We construct a family of formulas that only has short proofs in QCDCL variants in which we can reduce arbitrarily during Unit Propagation — as long as we only consider level-ordered decisions (cf. Theorem 8.8). We also prove that all QCDCL variants with the EXI-PROP policy (in particular all QCDCL variants from the previous chapters) are p-simulated by a potentially[3] weaker version of long-distance Q-Resolution, which we call modified long-distance Q-Resolution (or mLD-Q-Res for short). These observations will become important in the next chapter.

Chapter 8 serves as a preparation for Chapter 9, in which we choose three particular variants of QCDCL and prove their p-equivalence to Q-Res, mLD-Q-Res and QU-Res, respectively. The simulation of Q-Resolution was already shown in [13] and [14][4], so that the other two simulations could be interpreted as generalizations. The results are largely very technical — therefore we provide a very high-level proof sketch of all shown characterisations (cf. Theorem 9.6). This chapter ends with an overview of the simulation order of selected QCDCL variants and their underlying proof systems (cf. Figure 9.1).

Both Chapter 8 and Chapter 9 are based on the SAT'23 paper "QCDCL vs QBF Resolution: Further Insights" [34] which has won the *Best Student Paper Award*. An extended version was published in JAIR [35].

We will finish with Chapter 10, in which we compare the size of proofs generated by and extracted from QCDCL runs with the runtime (i.e., the size of QCDCL proofs) itself. In the previous chapters, we mainly concentrated on comparing versions of QCDCL with their underlying proof system or comparing the versions with one another. In this chapter however, we will only consider proofs that can actually be generated by QCDCL. While for CDCL, it is known that the proof size in the underlying proof system Resolution matches the CDCL runtime up to a polynomial factor, we show that in QBF there is an

[3] It is still an open question whether LD-Q-Res is exponentially stronger than mLD-Q-Res or if they are both p-equivalent. So far, we only know that LD-Q-Res trivially p-simulates mLD-Q-Res.

[4] The simulation of Q-Resolution followed an approach similar to the simulation of Resolution by CDCL from [86] and could be interpreted as a generalization. Furthermore, our simulation from [13] improved the bound from [86] by a factor of n.

exponential gap between QCDCL runtime and the size of the extracted proofs in QBF resolution systems. Hence searching for a small refutation via QCDCL will provably cause an exponential overhead for some instances.

We will first prove our master theorem (cf. Theorem 10.2) which we will apply on three variants of QCDCL that generate Q-Res, LD-Q-Res and QU-Res proofs, respectively. By crafting suitable formulas, we separate each QCDCL variant not only from their underlying proof system, but also from their restriction to generatable formulas.

This chapter is based on the paper "Runtime vs. Extracted Proof Size: An Exponential Gap for CDCL on QBFs" [15], published at AAAI'24.

For an overview of all publications corresponding to the main chapters, see Table 1.1.

Chapter	Reference	published at	Co-authors
4	[13, 14][5]	ITCS'21, LMCS	Olaf Beyersdorff
5	[32, 33][6]	SAT'21, JAR	Olaf Beyersdorff
6	[36, 38][7]	IJCAI'22, AIJ	Tomáš Peitl, Olaf Beyersdorff
7	[37, 39]	SAT'22, JAR	Tomáš Peitl, Olaf Beyersdorff
8	[34, 35][8]	SAT'23, JAIR	Olaf Beyersdorff
9	[34, 35][8]	SAT'23, JAIR	Olaf Beyersdorff
10	[15]	AAAI'24	Olaf Beyersdorff, Meena Mahajan

Table 1.1: Overview of publications corresponding to the chapters.

[5] Most of the results were refined in later papers and therefore omitted. For this chapter, we only used the QCDCL framework which was first introduced in that paper.

[6] Originally, the main results of that paper were proven via algorithm. For this thesis, these results and their proofs were rewritten in a more formal way without relying on an algorithm. In particular, Definition 5.8, Lemma 5.9 and the proof of Theorem 5.10 did not appear in the original paper.

[7] *Distinguished Paper Award* at IJCAI'22.

[8] *Best Student Paper Award* at SAT'23. Definition 8.12 and Proposition 8.13 are new and did not appear in the original paper.

Chapter 2

SAT and QBF Proof Complexity

Most results and definitions of this chapter can be viewed as folklore. In terms of an introduction to SAT and QBF theory, we refer to [29]. However, we reserve the right to formalize some notions differently.

2.1 Propositional Formulas

The simplest type of logical formulas which we will consider are *propositional formulas* which consist of *variables* and *connectives*. To ensure that the set of all propositional formulas is countable, we have to assume that the entirety of all variables is countable as well.

We recall the most common connectives:

- Negation: $\neg$
- Conjunction: $\wedge$
- Disjunction: $\vee$
- Implication: $\rightarrow$
- Equivalence: $\leftrightarrow$

Furthermore, we will also consider the two symbols $\top$ (*verum*) and $\bot$ (*falsum*) as connectives that represent 'always true' and 'always false' formulas.

We can now define *propositional formulas* (or *Boolean formulas*) recursively.

Definition 2.1 (propositional formulas)**.** *It holds the following:*

- $\top$, $\bot$ *and each variable is a propositional formula.*
- *If* φ *and* ψ *are propositional formulas, then* $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$, $\varphi \rightarrow \psi$ *and* $\varphi \leftrightarrow \psi$ *are also propositional formulas.*

We denote the set of variables that appear in a propositional formula φ as $var(\varphi)$. The size of φ, denoted as $|\varphi|$, is defined as the number of variables appearing in φ whereby variables that occur multiple times are also counted multiple times.

Each propositional formula is not only a syntactical structure, but can also be evaluated semantically as a representation of a *Boolean function*. For this evaluation, we use the *truth values* 0 (false) and 1 (true). Note that we treat 0 and 1 not the same as $\bot$ and $\top$. In order to evaluate formulas, we first need to define *assignments* of variables.

Definition 2.2 (assignments)**.** *Let V be a set of variables. An* assignment of V *is a function $\sigma : V \longrightarrow \{0,1\}$ that maps each variable from V to the truth values 0 or 1. We denote the set of all assignments of V as $\langle V \rangle$.*

Obviously, if V is finite, say $|V| = n$, we have $|\langle V \rangle| = 2^n$.

Definition 2.3 (Boolean functions)**.** *A* Boolean function on V *is a function*

$$f : \langle V \rangle \longrightarrow \{0,1\}$$

that maps assignments to truth values.

Sometimes it makes sense to extend a Boolean function to a larger set of variables. For example, if W is another set of variables with $V \subseteq W$, we can define f on $\langle W \rangle$ as $f(\tau) := f(\tau|_V)$ for $\tau \in \langle W \rangle$, where $\tau|_V$ is the restriction of τ to V. In applications, there is normally no need to specify the domain of a Boolean function as long as the variables are obvious. Therefore, we will omit a new notation for extensions of f and instead consider f as a function in which we can input any assignments of sets of variables that contain at least all variables from V.

Propositional formulas can be interpreted as representations of Boolean functions. It is easy to see that representations of Boolean functions are not unique in general, but each propositional formula corresponds to a unique Boolean function.

Definition 2.4 (formulas as Boolean functions, equivalence)**.** *If φ is a propositional formula, then we denote the Boolean function on $var(\varphi)$ determined by φ as f_φ. Similar to the definition of propositional formulas, we can describe f_φ recursively:*

- $f_\bot(\sigma) := 0$,
- $f_\top(\sigma) := 1$,
- $f_v(\sigma) := \sigma(v)$ *if v is a variable,*
- $f_{\neg\varphi}(\sigma) := 1 - f_\varphi(\sigma)$,
- $f_{\varphi\wedge\psi}(\sigma) := \min(f_\varphi(\sigma), f_\psi(\sigma))$,
- $f_{\varphi\vee\psi}(\sigma) := \max(f_\varphi(\sigma), f_\psi(\sigma))$,
- $f_{\varphi\to\psi}(\sigma) := \max(1 - f_\varphi(\sigma), f_\psi(\sigma))$,

- $f_{\varphi \leftrightarrow \psi}(\sigma) := \max(f_\varphi(\sigma) \cdot f_\psi(\sigma), (1 - f_\varphi(\sigma)) \cdot (1 - f_\psi(\sigma)))$,

for all $\sigma \in \langle var(\varphi) \rangle$ and propositional formulas φ and ψ. We will say that two formulas φ and ψ are equivalent *if f_φ and f_ψ are equal on $\langle var(\varphi) \cup var(\psi) \rangle$. We will denote such an equivalence as $\varphi \equiv \psi$.*

In some situations, we do not necessarily want to assign all appearing variables, but instead apply an assignment to a propositional formula syntactically.

Definition 2.5 (applying assignments to propositional formulas)**.** *If σ is any assignment of a set of variables V, then we can* apply *σ to a propositional formula φ and denote the resulting formula as $\varphi|_\sigma$. We again define the application σ of an assignment on a formula recursively:*

- $\bot|_\sigma := \bot$
- $\top|_\sigma := \top$
- $v|_\sigma := \begin{cases} \bot & \text{if } v \in V,\ \sigma(v) = 0, \\ \top & \text{if } v \in V,\ \sigma(v) = 1, \\ v & \text{if } v \notin V, \end{cases}$
  if v is a variable
- $(\neg\varphi)|_\sigma := \neg\varphi|_\sigma$
- $(\varphi \wedge \psi)|_\sigma := \varphi|_\sigma \wedge \psi|_\sigma$
- $(\varphi \vee \psi)|_\sigma := \varphi|_\sigma \vee \psi|_\sigma$
- $(\varphi \rightarrow \psi)|_\sigma := \varphi|_\sigma \rightarrow \psi|_\sigma$
- $(\varphi \leftrightarrow \psi)|_\sigma := \varphi|_\sigma \leftrightarrow \psi|_\sigma$

for all propositional formulas φ and ψ.

The SAT problem deals with the question of whether a given propositional formula φ can be evaluated to 1 under at least one assignment $\sigma \in \langle \mathrm{var}(\varphi) \rangle$. Hence, we call a formula φ *satisfiable* if there exists some $\sigma \in \langle \mathrm{var}(\varphi) \rangle$ such that $f_\varphi(\sigma) = 1$.

Alternatively, we can also define satisfiability in a more syntactic way. While the main task in SAT is showing satisfiability, we often desire to find an actual assignment that satisfies the given formula. In many cases, such a satisfying assignment does not necessarily need to assign all variables of a formula in order to satisfy it. For example, the assignment σ that only maps x to 1 already satisfies the propositional formula $x \vee \neg y$, although σ itself can technically not be inserted into the Boolean function $f_{x \vee \neg y}$ as σ does not assign y.

Instead of using the Boolean function of a given formula φ, we can try to find an assignment σ such that applying σ to φ leads to $\top$. However, the way we have defined the application of assignments on formulas so far, it is not guaranteed that we always obtain precisely $\top$ after applying a satisfying assignment. In the example above, after applying the assignment σ that only maps x to 1 on $x \vee \neg y$, we get $(x \vee \neg y)|_\sigma = \top \vee \neg y$. Therefore, we recall some trivial shortening rules:

- $\varphi \wedge \top = \varphi$
- $\varphi \wedge \bot = \bot$
- $\varphi \vee \top = \top$
- $\varphi \vee \bot = \varphi$
- $\varphi \to \top = \top$
- $\varphi \to \bot = \neg\varphi$
- $\top \to \varphi = \varphi$
- $\bot \to \varphi = \top$
- $\varphi \leftrightarrow \top = \top \leftrightarrow \varphi = \varphi$
- $\varphi \leftrightarrow \bot = \bot \leftrightarrow \varphi = \neg\varphi$

for any propositional formula φ.

Note that the above equalities are technically only equivalences. However, for the sake of readability, we will already call two formulas equal if they are equal after applying the above shortening rules from now on. If we want to point out that two formulas are equal even without applying shortening rules, we will refer to them as being *syntactically equal*. For operators like $\mathrm{var}(\varphi)$ or $|\varphi|$, we will assume that φ is already shortened using the shortening rules.

Example 2.6. *The formula $\top \vee \neg y$ is*

- *equal and equivalent, but not syntactically equal to $\top$,*
- *equal, syntactically equal and equivalent to itself,*
- *equivalent, but not equal or syntactically equal to $z \vee \neg z$.*

We have $var(\top \vee \neg y) = var(\top) = \emptyset$, but $var(z \vee \neg z) = \{z\}$.

Now we can give an alternative definition of satisfiability:

A propositional formula φ is called *satisfiable*, if there exists an assignment σ of some variables such that $\varphi|_\sigma = \top$. In this case, we call σ a *satisfying assignment* of φ. Propositional formulas that are not satisfiable are called *unsatisfiable*. This means that φ is unsatisfiable if $f_\varphi(\sigma) = 0$ for all $\sigma \in \langle \mathrm{var}(\varphi) \rangle$, or alternatively if $\varphi \equiv \bot$. Propositional formulas φ are called *tautologies* or *tautological* if $f_\varphi(\sigma) = 1$ for all $\sigma \in \langle \mathrm{var}(\varphi) \rangle$, or alternatively if $\varphi \equiv \top$ or if $\neg\varphi$ is unsatisfiable.

For SAT solving, it is important to transform a given formula into a normal form with a clear structure that is easier to handle for algorithms. We refer to variables and negated variables as *literals*. We will write negated variables as $\bar{v}$ instead of $\neg v$ for the sake of readability. We will sometimes also use the notations $v^1 := v$ and $v^0 := \bar{v}$ for a literal v. We use $\mathrm{var}(v) = \mathrm{var}(\bar{v}) := v$ to denote the variable of a literal.

A propositional formula is a *CNF* (or *in CNF*; *conjunctive normal form*) if it is a conjunction of disjunctions of literals. Analogously, it is a *DNF* (*disjunctive normal form*) if it is a disjunction of conjunctions of literals.

For example, the formula

$$(x \vee \bar{y}) \wedge \bar{x} \wedge (a \vee b \vee c) \wedge z$$

is a CNF, while the formula

$$(x \wedge \bar{y}) \vee \bar{x} \vee (a \vee b \wedge c) \vee z$$

is a DNF. The formulas

$$a \vee b \vee \bar{c} \vee d$$

and

$$a \wedge b \wedge \bar{c} \wedge d$$

are both in CNF and DNF.

We will also refer to disjunctions of literals as *clauses* and conjunctions of literals as *cubes*. That means that CNFs are conjunctions of clauses and DNFs are disjunctions of cubes.

We will sometimes use set-theoretical operators to indicate properties of CNFs, DNFs, clauses and cubes: For a literal x and a clause or cube C we will write $x \in C$ to indicate that x is one of the literals of the disjunction or conjunction in C. If C is a clause and φ is a CNF, we write $C \in \varphi$ to indicate that C is one of the clauses in the conjunction in φ. The same holds if C is a cube and φ is a DNF. For two clauses C, D we will write $C \subseteq D$ to indicate that $\mathrm{var}(C) \subseteq \mathrm{var}(D)$. In this case, we will call C a *subclause* of D. Analogously, we will use the notion *subcube*.

It is common to write assignments as cubes that indicate which literal is set to 1. For example, an assignment $\sigma \in \langle V \rangle$ can be expressed as $\sigma = \bigwedge_{x \in V} x^{\sigma(x)}$. In this way, we will often interpret assignments as the set of literals that is set to 1 by this assignment. We denote the negation of a set of literals L as $\bar{L}$, regardless of whether L is just a set, a clause, cube or assignment. In particular, the negation of an assignment is again an assignment.

It is easy to transform a propositional formula φ into a CNF or DNF: Let $A_0 := \{\sigma \in \langle \mathrm{var}(\varphi) \rangle |\ f_\varphi(\sigma) = 0\}$ and $A_1 := \{\sigma \in \langle \mathrm{var}(\varphi) \rangle |\ f_\varphi(\sigma) = 1\}$, then we can construct an equivalent CNF

$$\varphi \equiv \bigwedge_{\sigma \in A_0} \bigvee_{v \in \mathrm{var}(\varphi)} v^{1-\sigma(v)}$$

and a DNF

$$\varphi \equiv \bigvee_{\sigma \in A_1} \bigwedge_{v \in \mathrm{var}(\varphi)} v^{\sigma(v)}.$$

Intuitively, this means that the clauses of the CNF correspond to the 0-rows in the truth table of φ (after negation), while the cubes of the DNF correspond to the 1-rows.

However, such a transformation is not polynomially bounded, in general. For example, the formula $(y \to y) \wedge (x_1 \vee x_2 \vee \ldots \vee x_n)$ is a tautology and therefore has 2^{n+1} satisfying assignments in its variables. This means, in the transformation above, we would obtain a DNF consisting of 2^{n+1} cubes of length $n + 1$.

We could work around this problem by introducing auxiliary variables and constructing an "almost" equivalent CNF of size polynomial in the size of the original formula. In order to do that, we need to define the notion of *equisatisfiability* first.

Definition 2.7 (equisatisfiability)**.** *We call two propositional formulas φ and ψ* equi satisfiable*, if φ is satisfiable if and only if ψ is satisfiable.*

Obviously, this notion is mainly used on formulas for which satisfiability is not known yet. In our case, we want to transform a given formula into an equisatisfiable CNF (before having decided satisfiability). This transformation, which is called *Tseitin transformation*, divides a given formula into its subformulas and introduces *Tseitin variables* for each of these subformulas.

Before formalizing this transformation, we need to define *subformulas*. This can be done recursively:

Definition 2.8 (subformulas)**.** *If φ is a propositional formula, then*

- *φ is a subformula of φ itself,*
- *if $\varphi = \neg\psi$, then ψ is a subformula of φ,*
- *if $\varphi = \psi \star \chi$ such that $\star$ is one of the connectives $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, then ψ and χ are subformulas of φ, and*
- *for each subformula ψ of φ, all subformulas of ψ are also subformulas of φ.*

Note that we have to treat the connectives $\wedge$ and $\vee$ as binary connectives. We will assume that clauses and cubes with more than two variables are enclosed in brackets from left to right. For example, a clause $a \vee b \vee c \vee d$ will be treated as $((a \vee b) \vee c) \vee d$.

The recursive definition allows us to order subformulas of a formula φ from the "outside" to the "inside" (i.e., starting with φ itself and ending with the variables). More precisely, by following this order, we can denote subformulas of φ as $\varphi_0, \varphi_1, \ldots, \varphi_m$ for some $m \in \mathbb{N}$ such that $\varphi_0 = \varphi$ and each subformula of some φ_i appears as φ_k with $k > i$ (if we allow subformulas to appear more than once).

The first step of the Tseitin transformation is to write down the ordered subformulas $\varphi_0, \ldots, \varphi_m$, where we will omit subformulas that are variables, and introduce the Tseitin variables $x_0, \ldots, x_m$ such that $\{x_0, \ldots, x_m\} \cap \mathrm{var}(\varphi) = \emptyset$. Note that m is polynomial in the number of connectives used in φ (and therefore polynomial in the formula size).

Each x_i should represent the subformula φ_i. Every φ_i can be expressed as $\neg\psi$, $\psi \wedge \chi$, $\psi \vee \chi$, $\psi \rightarrow \chi$ or $\psi \leftrightarrow \chi$ with $\psi, \chi \in \{\varphi_{i+1}, \ldots, \varphi_m\} \cup \mathrm{var}(\varphi)$. Let φ_i' be the formula φ_i, in which each φ_j for $j > i$ is replaced with the corresponding x_j.

Then for each $i \in \{0, \ldots, m\}$, we can define the formula

$$\lambda_i := x_i \leftrightarrow \varphi_i'.$$

Note that φ_i' is a propositional formula that consists of at most two variables. Therefore, each λ_i can be transformed into an equivalent CNF μ_i of constant size using the naive CNF transformation (i.e., via truth tables).

Now we have obtained CNFs $\mu_0, \ldots, \mu_m$ such that $\mu_i \equiv x_i \leftrightarrow \varphi_i'$. We combine these CNFs into the larger CNF

$$\varphi' := x_0 \wedge \bigwedge_{i=0}^{m} \mu_i.$$

We claim that φ and φ' are equisatisfiable. Obviously, we have

$$\varphi' \equiv x_0 \wedge \bigwedge_{i=0}^{m} (x_i \leftrightarrow \varphi_i').$$

Let $\sigma \in \langle \mathrm{var}(\varphi) \rangle$ be an assignment that satisfies φ. We will extend σ to an assignment $\tau \in \langle \mathrm{var}(\varphi') \rangle$ by setting $\tau(v) := \sigma(v)$ if $v \in \mathrm{var}(\varphi)$, and $\tau(x_i) := f_{\varphi_i}(\sigma)$.

Obviously, since τ sets each x_i to the truth value we obtain by evaluating the subformula φ_i under σ, the formulas φ_i and φ_i' have the same truth value under τ. We conclude that τ satisfies each λ_i and therefore also each μ_i. Because, as an extension of σ, the assignment τ satisfies φ, it also satisfies x_0. Therefore, τ satisfies φ'.

For the other direction, let τ be a satisfying assignment of φ'. Then τ satisfies x_0 and each λ_i. That means that for each i, the formula φ_i' has the same truth value as x_i under τ. By definition of the φ_i', each φ_i also has the same truth value as the corresponding x_i under τ. In particular, $\varphi_0 = \varphi$ and x_0 have the same truth value under τ. Because τ satisfies x_0, it also satisfies $\varphi_0 = \varphi$. By restricting τ to $\mathrm{var}(\varphi)$, we obtain a satisfying assignment for φ.

2.2 Propositional Resolution

While for satisfiable formulas it suffices to find a satisfying assignment to prove satisfiability, the task is much more complex for unsatisfiable formulas. Checking all assignments for a given formula for which unsatisfiabilty is suspected is often infeasible. Instead, it makes sense to prove the correctness of certain implications. In our case, showing that a formula φ implies $\bot$ is the same as proving that $\varphi \to \bot$ (which is equivalent to $\neg\varphi$) is a tautology. Furthermore, in many cases it makes sense to show the implication of another formula as an intermediate step for implying $\bot$. More precisely, if we can show that φ implies ψ, such that $\varphi \wedge \psi$ again implies $\bot$, then we have effectively shown that φ was unsatisfiable from the start. Of course, this process can be generalized to arbitrary lengths, say we prove that $\varphi \to \psi_1$, $(\varphi \wedge \psi_1) \to \psi_2, \ldots, (\varphi \wedge \psi_1 \wedge \ldots \wedge \psi_{m-1}) \to \psi_m$ and finally $(\varphi \wedge \psi_1 \wedge \ldots \wedge \psi_m) \to \bot$ are tautologies, then we have proven the unsatisfiability of φ, as well.

Procedures like the one described above are called *proof systems*. Although proof systems can be defined for pretty much any structure, we will concentrate on *propositional proof systems* for now. For a formal way of describing proof systems, we refer to the Cook-Reckhow definition [50]:

Definition 2.9 (proof systems by Cook-Reckhow)**.** *Let L be a language. A polynomial-time function S is called a proof system, if*

- *For each $\varphi \in L$ there exists a word π such that $\mathsf{S}(\pi) = \varphi$ (*completeness*).*
- *For each word π we have $\mathsf{S}(\pi) \in L$ (*soundness*).*

For $\mathsf{S}(\pi) = \varphi$, we say that π is an S proof for φ.

Due to its abstractness, the Cook-Reckhow definition is almost never used as a working definition and concrete proof systems are often defined much more grounded via inference rules as we will see later.

In our case, we will define the language L as the set of unsatisfiable formulas in CNF. Generally, we will focus on so-called *refutational proof systems* (i.e., proof systems for sets of unsatisfiable or false[1] formulas) in this thesis. One of the most well-known propositional proof systems is *Resolution* [90], which consists of its eponymous inference rule only.

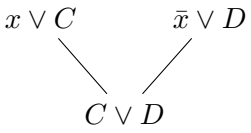
Definition 2.10 (Resolution)**.** *Let C and D be two clauses. Then for a literal x the clauses $x \vee C$ and $\bar{x} \vee D$ are* resolvable*, and we define the resolvent of $x \vee C$ and $\bar{x} \vee D$ over x, which we will be denoted by $(x \vee C) \overset{x}{\bowtie} (\bar{x} \vee D)$, as*

$$(x \vee C) \overset{x}{\bowtie} (\bar{x} \vee D) := C \vee D.$$

We call this inference rule the resolution rule *or* resolution step*.*

A Resolution proof *π for a CNF φ of a clause E is a sequence of clauses $\pi = (C_1, \ldots, C_m)$, such that $C_m = E$ and for each i we have $C_i \in \varphi$ or $C_i = C_j \overset{x}{\bowtie} C_k$ for some literal $x \in C_j$ with $\bar{x} \in C_k$ and $j, k < i$. The clauses C_i with $C_i \in \varphi$ are called* axioms*. If $E = \bot$, we call π a* Resolution refutation *of φ. We write $C \in \pi$ to indicate that C is one of the clauses from π. We define the length of π as $|\pi| := m$.*

Although we treat Resolution proofs as sequences of clauses, it is common to represent them as DAGs (directed acyclic graphs). More precisely, a resolution step can be depicted as:

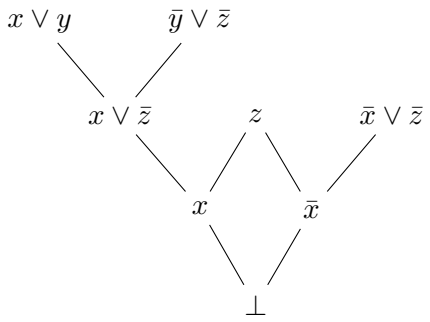


Example 2.11. *Let $\varphi := (x \vee y) \wedge (\bar{y} \vee \bar{z}) \wedge z \wedge (\bar{x} \vee z) \wedge (\bar{x} \vee \bar{z})$. Then*

$$\pi := (x \vee y, \bar{y} \vee \bar{z}, x \vee \bar{z}, z, x, \bar{x} \vee \bar{z}, \bar{x}, \bot)$$

is a Resolution *refutation of φ with the DAG representation*

[1] The notion "false" will be defined for quantified Boolean formulas later.



Note that $\bar{x} \vee z$ was not needed as an axiom clause.

Similar to formulas, we can also apply assignments to proofs. More precisely, if $\pi = (C_1, \ldots, C_m)$ is a Resolution proof for a formula φ, then we can apply an assignment σ to all C_i and obtain a Resolution proof for $\varphi|_\sigma$. Because applying σ to all clauses in a proof may break some of the resolution steps, we have to remove some clauses that cannot be derived any more.

Formally, we define $I := \{i \in \{1, \ldots, m\} |\ C_i|_\sigma \neq \top\}$ as the set of indices i such that σ does not satisfy C_i. We can write these indices explicitly as $I = \{j_1, \ldots, j_k\}$ with $j_a < j_{a+1}$ for each $a \in \{1, \ldots k-1\}$. Let $\pi|_\sigma := \{D_{j_1}, \ldots, D_{j_k}\}$ be a sequence of clauses. We will construct these clauses from left to right, such that it holds $D_{j_b} \subseteq C_{j_b}|_\sigma$ for all $b \in \{1, \ldots, k\}$.

For each $b \in \{1, \ldots, k\}$, we set $D_{j_b} := C_{j_b}|_\sigma$ if C_{j_b} is an axiom[2]. If C_{j_b} was derived by a resolution step, say $C_{j_b} = C_g \overset{x}{\bowtie} C_h$ for some $g, h < j_b$ and some variable x with $x \in C_g$ and $\bar{x} \in C_h$, we distinguish two cases:

Case 1. σ does neither satisfy C_g nor C_h. That means that $g = j_c$ and $h = j_d$ for some $c, d < b$ and $C_{j_b} = C_{j_c} \overset{x}{\bowtie} C_{j_d}$. If $x \in D_{j_c}$ and $\bar{x} \in D_{j_d}$, then these two subclauses are resolvable as well and we set $D_{j_b} := D_{j_c} \overset{x}{\bowtie} D_{j_d}$ and get $D_{j_b} \subseteq C_{j_b}|_\sigma$. If $x \notin D_{j_c}$, we set $D_{j_b} := D_{j_c}$ and also get $D_{j_b} \subseteq C_{j_b}|_\sigma$. In the remaining case where $\bar{x} \notin D_{j_d}$, we can analogously set $D_{j_b} := D_{j_d}$.

Case 2. σ does satisfy C_g or C_h. Then σ cannot satisfy both, otherwise C_{j_b} would be satisfied by σ. W.l.o.g. let us assume that σ does not satisfy C_g. That means that $g = j_c$ for some $c < b$. Because all literals from C_h except $\bar{x}$ also appear in C_{j_b}, we conclude that $\sigma(x) = 0$. Hence $C_{j_c}|_\sigma \subseteq C_{j_b}|_\sigma$. By setting $D_{j_b} := D_{j_c}$, we get $D_{j_b} \subseteq C_{j_b}|_\sigma$.

The sequence of clauses $\pi|_\sigma$ now consists of clauses that are axioms from $\varphi|_\sigma$ or derived via resolution steps inside the proof. We conclude that $\pi|_\sigma$ is a valid Resolution proof of $\varphi|_\sigma$. Furthermore, if π was a Resolution refutation, then $\pi|_\sigma$ is also a Resolution refutation of size at most $|\pi|$.

Next we define a special case for Resolution proofs, called *treelike* Resolution.

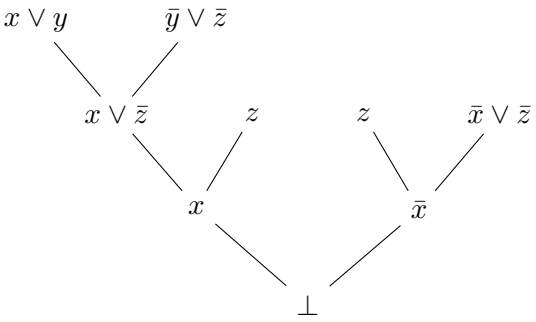
Definition 2.12 (tree-like Resolution)**.** *A* Resolution *proof is called* treelike*, if the corresponding DAG is a tree. More precisely, each clause in the proof can only be used for at most one resolution step (but the same clause can be derived more than once).*

[2]The first clause of a Resolution proof is always an axiom.

Example 2.13. *Let* $\varphi := (x \vee y) \wedge (\bar{y} \vee \bar{z}) \wedge z \wedge (\bar{x} \vee z) \wedge (\bar{x} \vee \bar{z})$ *be the CNF from Example 2.11. Then*

$$\pi := (x \vee y, \bar{y} \vee \bar{z}, x \vee \bar{z}, z, x, z, \bar{x} \vee \bar{z}, \bar{x}, \bot)$$

is a treelike Resolution *refutation of* φ *with the tree representation*



Note that copying clauses which were used more than once (in our case z) is the naive approach for transforming a non-treelike proof into a treelike one.

In order to show that Resolution is in fact a proof system, we need to check for soundness and completeness.

Theorem 2.14. Resolution *is sound and complete.*

Proof. For soundness, we have to prove that whenever some CNF φ has a Resolution refutation π, then φ is unsatisfiable. It suffices to show that each resolution step is sound. More precisely, if $\sigma \in \langle \mathrm{var}(\varphi) \rangle$ is an assignment that satisfies both $x \vee C$ and $\bar{x} \vee D$, it also satisfies $C \vee D$. This is because σ has to set x to 0 or 1, which means that σ has to satisfy C (if $x \mapsto 0$) or D (if $x \mapsto 1$) in order to satisfy both $x \vee C$ and $\bar{x} \vee D$.

Hence, each assignment that satisfies φ also satisfies all clauses from π. But since $\bot$ is one of the clauses in π, such an assignment cannot exist and φ must therefore be unsatisfiable.

For completeness, we have to show that each unsatisfiable CNF φ has a Resolution refutation of arbitrary size. Since φ is unsatisfiable, for each $\sigma \in \langle \mathrm{var}(\varphi) \rangle$ we can find a clause $C_\sigma \in \varphi$ such that σ falsifies C_σ (otherwise φ would be satisfied by some σ). Let us now consider two assignments σ_0 and σ_1 that only differ in one variable, say $\sigma_0(x) = 0$ and $\sigma_1(x) = 1$. Then the corresponding clauses C_{σ_0} and C_{σ_1} can also only differ in at most one variable. We then define a new clause $C_{\{\sigma_0,\sigma_1\}}$ as follows:

$$C_{\{\sigma_0,\sigma_1\}} := \begin{cases} C_{\sigma_0} \overset{x}{\bowtie} C_{\sigma_1} & \text{if } x \in C_{\sigma_0} \text{ and } \bar{x} \in C_{\sigma_1} \\ C_{\sigma_0} & \text{if } x \notin C_{\sigma_0} \\ C_{\sigma_1} & \text{if } x \in C_{\sigma_0} \text{ and } \bar{x} \notin C_{\sigma_1} \end{cases}$$

Intuitively, we resolve C_{σ_0} and C_{σ_1} if they are resolvable, or choose one of the clauses that does not contain x in any polarity. In any case, we obtain a clause $C_{\{\sigma_0,\sigma_1\}}$ that

is falsified by both σ_0 and σ_1. We generalize this approach by defining the set $A_x := \{\{\sigma_0, \sigma_1\} \subseteq \langle \mathrm{var}(\varphi)\rangle | \, \sigma_0(x) \neq \sigma_1(x) \text{ and } \sigma_0|_{var(\varphi)\setminus\{x\}} = \sigma_1|_{var(\varphi)\setminus\{x\}}\}$ of unordered pairs of assignments that differ exactly on x. Then we define the new CNF

$$\varphi' := \bigwedge_{\{\sigma_0,\sigma_1\}\in A_x} C_{\{\sigma_0,\sigma_1\}}.$$

Obviously, φ implies φ' because all clauses from φ' are clauses from φ or clauses derived via resolution steps from φ. Also, because $\bigcup A_x = \langle \mathrm{var}(\varphi)\rangle$, and each assignment from $\bigcup A_x$ falsifies at least one clause from φ', we can conclude that φ' is unsatisfiable. It is also easy to see that the variable x got eliminated from all clauses, meaning that $\mathrm{var}(\varphi') = \mathrm{var}(\varphi)\setminus\{x\}$. By repeating this whole procedure for each remaining variable of φ', we will obtain $\bot$ at some point. Since all clauses were derived from φ via Resolution, the same is true for $\bot$ and therefore we have obtained a Resolution refutation of φ. □

In proof complexity, typically we do not consider proofs of one particular formula, but of families of formulas[3]. In our case, we will often consider Resolution proofs π_n of a family of unsatisfiable CNFs φ_n. We will always assume that $|\mathrm{var}(\varphi_n)|$ and $|\varphi_n|$ are polynomial in n. A formula (resp. family of formulas) φ_n is called *hard* for a proof system S, if for each S proof π_n of φ_n we have $|\pi_n| \in 2^{\Omega(n)}$ (i.e., π_n has *exponential* size). The proof π_n is considered to be *easy* for S, if there exists some S proof π_n for φ_n such that $|\pi_n| \in \mathrm{poly}(n)$ (i.e., π_n has *polynomial* size).

We can compare different proof systems with each other via *p-simulations* and *separations*. A proof system S_1 *p-simulates* another proof system S_2 for the same language, if each S_2 proof π' for some instance φ can be polynomially transformed into a S_2 proof π of φ with $|\pi| \in \mathrm{poly}(|\pi|')$.

If two proof systems S_1 and S_2 p-simulate each other, we say that S_1 and S_2 are *p-equivalent* and write $\mathsf{S}_1 \equiv \mathsf{S}_2$

A proof system S_1 is *(exponentially) separated* from another proof system S_2, if there exists an instance φ_n such that φ_n is easy for S_1 and hard for S_2 or vice versa. If S_1 and S_2 are separated and S_1 p-simulates S_2, then we say that S_1 is *exponentially stronger than* S_2 and S_2 is *exponentially weaker than* S_1.

If two proof systems S_1 and S_2 have exponential separations in both directions, we say that S_1 and S_2 are *incomparable*.

2.3 Quantified Boolean Formulas

Quantified Boolean formulas (or *QBFs* for short) can be treated as a generalization of propositional formulas. Besides a propositional formula, a QBF also consists of *quantifiers*.

Definition 2.15 (quantified Boolean formulas)**.** *A* quantified Boolean formula *is a formula* $\Phi = \mathcal{Q} \cdot \varphi$*, such that* φ *is a propositional formula over the variables* $var(\Phi) :=$

[3]We will often simply refer to "families of formulas" as "formulas".

$var(\varphi) = \{x_1, \ldots, x_k\}$ and $\mathcal{Q} = Q_1 x_1 Q_2 x_2 \ldots Q_k x_k$ such that $Q_i \in \{\exists, \forall\}$ for each $i \in \{1, \ldots, k\}$. We will call $\mathcal{Q}$ the prefix *and φ the* matrix *of Φ.*

Variables with an existential quantifier (i.e. $\exists$) are called existential variables*, while variables with a universal quantifier (i.e. $\forall$) are called* universal variables*. We use the notation $var_\exists(\Phi)$ for the set of all existential variables in Φ and $var_\forall(\Phi)$ for the set of all universal variables in Φ. Typically, we merge adjacent quantifiers of the same type and write $\mathcal{Q} = Q'_1 V_1 Q'_2 V_2 \ldots Q'_m V_m$ with pairwise disjoint sets of variables V_i such that $\biguplus_{i=1}^m V_i = var(\Phi)$ as well as $Q'_i \in \{\exists, \forall\}$ for each $i \in \{1, \ldots, m\}$ such that $Q'_j \neq Q'_{j+1}$ for each $j \in \{1, \ldots, m-1\}$. We will call Q'_i a* quantifier block*. A literal ℓ has* quantification level *i (with respect to Φ), denoted as $lv_\Phi(\ell)$, if and only if $var(\ell) \in V_i$. In case the formula is obvious, we simply write $lv(\ell) = i$. A QBF with the prefix $Q'_1 V_1 Q'_2 V_2 \ldots Q'_m V_m$ is called a Σ^b_m QBF if $Q'_1 = \exists$, and Π^b_m QBF if $Q'_1 = \forall$.*

If x and y are two variables with $x \in V_i$ and $y \in V_j$ with $i < j$, we say that x is quantified left of *y and y is* quantified right of *x. In this case, we will use the notation $x <_\Phi y$. If x and y are two literals, we write $x <_\Phi y$ to indicate $var(x) <_\Phi var(y)$.*

If φ is a CNF, we will call Φ a QCNF. If φ is a DNF, we will call Φ a QDNF.

Similar to the SAT problem, the *QBF problem* deals with the question if a given QBF evaluates to 1. However, since QBFs (in our setting) have no free variables, we cannot simply apply assignments to a QBF with the aim to satisfy it. Instead, we distinguish *true* and *false* QBFs.

Definition 2.16 (truth value of a QBF)**.** *Let $\Phi = \mathcal{Q} R x \cdot \varphi$ be a QBF such that $\mathcal{Q}$ contains all quantifiers for the variables from φ except x and $R \in \{\exists, \forall\}$. Then Φ can be expanded to*

$$\mathcal{Q} \cdot \varphi|_{\sigma_{x \mapsto 0}} \vee \varphi|_{\sigma_{x \mapsto 1}} \text{ if } R = \exists,$$

or

$$\mathcal{Q} \cdot \varphi|_{\sigma_{x \mapsto 0}} \wedge \varphi|_{\sigma_{x \mapsto 1}} \text{ if } R = \forall,$$

where $\sigma_{x \mapsto 0}$ is the assignment that only maps x to 0 and $\sigma_{x \mapsto 1}$ is the assignment that only maps x to 1.

After having expanded over all variables, we obtain a QBF with an empty prefix and a matrix that is either $\top$ or $\bot$. If this matrix is $\top$, then we call Φ true*, otherwise we call it* false*.*

Two QBFs Φ and Ψ with the same truth value are called equivalent *and we write $\Phi \equiv \Psi$. Therefore, for each true QBF Φ we have $\Phi \equiv \top$ and for each false QBF Φ we have $\Phi \equiv \bot$.*

Note that it does not matter in which order dissolve quantifiers that belong to the same quantifier block since the connectives $\vee$ and $\wedge$ are commutative.

Because expanding a QBF might lead to an exponential blow-up (especially for large QBFs or families of QBFs), it makes sense to approach the QBF problem from a different angle via *winning strategies*.

Definition 2.17 (winning strategies)**.** *For each QBF Φ and each variable $y \in var(\Phi)$, we define*

$$L(y) := \{x \in var_\forall(\Phi) |\, x <_\Phi y\} \text{ if } y \text{ is existential,}$$

and

$$L(y) := \{x \in var_\exists(\Phi) |\, x <_\Phi y\} \text{ if } y \text{ is universal.}$$

An existential winning strategy *for a QBF $\Phi = \mathcal{Q} \cdot \varphi$ is a function $s : \langle var_\forall(\Phi)\rangle \longrightarrow \langle var_\exists(\Phi)\rangle$ such that the following holds:*

- *For each $\sigma \in \langle var_\forall(\Phi)\rangle$ we have that $\sigma \cup s(\sigma)$ satisfies φ.*
- *For each $y \in var_\exists(\Phi)$ and for all $\sigma, \tau \in \langle var_\forall(\Phi)\rangle$ with $\sigma|_{L(y)} = \tau|_{L(y)}$ we have $(s(\sigma))(y) = (s(\tau))(y)$.*

Intuitively, this means that an existential winning strategy has to extend a given assignment of the universal variables to a satisfying assignment of the matrix, such that the assigned truth value of each existential variable y only depends on the universal variables left of y (i.e., $L(y)$). Analogously, we can define universal winning strategies*:*

A universal winning strategy *for a QBF $\Phi = \mathcal{Q} \cdot \varphi$ is a function $h : \langle var_\exists(\Phi)\rangle \longrightarrow \langle var_\forall(\Phi)\rangle$ such that the following holds:*

- *For each $\sigma \in \langle var_\exists(\Phi)\rangle$ we have that $\sigma \cup h(\sigma)$ falsifies φ.*
- *For each $y \in var_\forall(\Phi)$ and for all $\sigma, \tau \in \langle var_\exists(\Phi)\rangle$ with $\sigma|_{L(y)} = \tau|_{L(y)}$ we have $(h(\sigma))(y) = (h(\tau))(y)$.*

With the above definition, we can define truth values of QBFs in an alternative way: A QBF is called *true*, if it has an existential winning strategy. It is called *false*, if it has a universal winning strategy.

Example 2.18. *The formula $\Phi := \exists x, y \forall u, v \exists t \cdot (x \vee u \vee t) \wedge (\bar{x} \vee \bar{u} \vee t) \wedge (y \vee \bar{v} \vee \bar{t}) \wedge (\bar{y} \wedge v \vee \bar{t})$ is a QCNF. We define a universal winning strategy h as follows:*

$$(h(\sigma))(u) := \sigma(x),$$
$$(h(\sigma))(v) := 1 - \sigma(y).$$

We set u to the same truth value as x, and v to the value opposite to y. After applying the assignment of the variables x, y, u, v, we obtain the matrix $t \wedge \bar{t}$, which is unsatisfiable. That means that $\sigma \cup h(\sigma)$ falsifies the matrix of Φ. Note that the assignments of u and v only depend on variables left of u and v (i.e., $L_u = L_v = \{x, y\}$).

It is common to express a winning strategy as a formula itself for better readability. In our case, we can write h as

$$h(u) := x,$$
$$h(v) := \bar{y}.$$

Technically, we would need to map u and v to the corresponding Boolean function f_x and $f_{\bar{y}}$, but we omit that notation in this context.

The above connection with the existence of winning strategies provides the more intuitive way of interpreting QBFs as two-player games. One player, called the existential player, controls the existential variables while the other one, the universal player, controls the universal variables. The two players need to assign the variables in the correct order along the quantifiers from left to right with the existential player trying to satisfy the matrix, while the universal player aims to falsify it. Obviously, winning strategies of such a game directly correspond to the winning strategies we have defined above.

Besides propositional formulas, we can also transform QBFs into QCNFs. However, as the notion of equisatisfiability does not make much sense for QBFs, we will simply claim that each QBF can be transformed into an equivalent QCNF in polynomial time.

The transformation is similar to the propositional case: Let $\Phi = \mathcal{Q} \cdot \varphi$ be a QBF. We then transform φ into an equisatisfiable CNF φ' in polynomial time. Because we have added some Tseitin variables, we need to change the prefix as well. In the propositional case, satisfying assignments of φ needed to be expanded on these Tseitin variables in order to obtain a satisfying assignment for φ'. More precisely, for each assignment $\sigma \in \langle \mathrm{var}(\varphi) \rangle$ there exists an assignment $\tau_\sigma \in \langle \mathrm{var}(\varphi') \backslash \mathrm{var}(\varphi) \rangle$ such that $\sigma \cup \tau_\sigma$ satisfies φ'. Conversely, by omitting the Tseitin variables, we can transform a satisfying assignment of φ' into a satisfying assignment of φ simply by restriction.

This observation implies that these Tseitin variables need to be quantified existentially at the end of the prefix. Taken all together, if $\Phi = \mathcal{Q} \cdot \varphi$ is a QBF and φ' is a CNF obtained from φ via Tseitin transformation with the set of Tseitin variables T, then $\Phi' := \mathcal{Q} \exists T \cdot \varphi'$ is a QCNF with $\Phi \equiv \Phi'$.

2.4 QBF Resolution

Many propositional proof systems can be lifted to the QBF level, including Resolution. We call this extension Q-Resolution or Q-Res for short. As Resolution, Q-Res is a refutational proof system in which we can derive $\bot$ to prove that a given QBF was false. At least on existential variables, Q-Res behaves similar to the propositional case: Clauses can be resolved over existential variables to derive a resolvent that is semantically implied by the conjunction of its parental clauses. However, this resolution rule is not enough for QBFs (even if we would extend this rule to universal variables) as the following example shows: The QBF $\forall u \cdot u$ is obviously a false QBF since the universal player can set $u \mapsto 0$ to win the game. However, no resolution step is possible as we only have one clause with only one literal. In other words, we need a method for eliminating universal variables.

This is where the second inference rule comes into play: Suppose we have a clause $C \vee u$ with a universal literal u such that for all existential literals $x \in C$ we have $x <_\Phi u$, then we can derive C from $C \vee u$ via *(universal) reduction*. Intuitively, one may want to consider such a situation from the existential player's perspective. The existential player aims to satisfy the matrix, and in particular $C \vee u$. Since the universal player has no control over u or over any variable from C that is quantified after u, they have to satisfy $C \vee u$ before u gets assigned, otherwise the universal player can simply set $u \mapsto 0$ and

win the game. That means that the existential player effectively not only has to satisfy $C \vee u$, but C itself. Hence, we can derive C from $C \vee u$.

Unfortunately, the reduction rule forbids to derive at least some kinds of universal tautologies (i.e., $u \vee \bar{u}$) since the above argument only works for clauses without such subclauses. From now on, we will also assume that no clause of a given CNF is tautological.

Definition 2.19 (Q-Resolution)**.** *As in the propositional case, we write*

$$(x \vee C) \overset{x}{\bowtie} (\bar{x} \vee D) := C \vee D.$$

If C and U are two clauses such that U contains no existential literals, and we have $c <_\Phi u$ for all variables $c \in C$ and $u \in U$, and for each $v \in var_\forall(C)$ there exists a $y \in var_\exists(C)$ such that $v <_\Phi y$ (in other words, U contains all reducible universal literals of $C \vee U$), then we define the universal reduction step[4] as

$$red(C \vee U) := C.$$

A Q-Res *proof π for a QCNF $\Phi = \mathcal{Q} \cdot \varphi$ of a clause E is a sequence of clauses $\pi = (C_1, \ldots, C_m)$ such that:*

- *$C_m = E$.*
- *For each i and each $u \in var_\forall(\Phi)$ we have $\{u, \bar{u}\} \not\subseteq C_i$.*
- *For each i we have $C_i \in \varphi$ or $C_i = C_j \overset{x}{\bowtie} C_k$ for an existential variable $x \in C_j$ with $\bar{x} \in C_k$ and $j, k < i$, or $C_i = red(C_j)$ for some $j < i$.*

If $E = \bot$, then π is called a Q-Res *refutation of Φ. Again, we use the notation $C \in \pi$ to indicate that C is contained in the sequence π and we define the length as $|\pi| := m$.*

We will omit a formal proof of soundness and completeness for Q-Res and refer to [72]. Simply put, the two players will play the game on the Q-Res refutation instead of the QBF itself. While we still always obtain a refutation after applying existential assignments to a Q-Res refutation, the universal player only has to take care to assign their universal variables without destroying the path to the empty clause $\bot$ in the proof. This can be done by assigning each universal variable according to the reduction step. Because such a reduction step will only occur in the first quantifier block (after having applied all assignments of existential variables left of this universal block), this reduction step is only performed in one polarity and therefore unique.

[4] We define the classic universal reduction as a maximal reduction, meaning we reduce all literals that are reducible and not just one. In resolution-like proofs, there is no benefit in postponing a reduction. However, in later chapters we will observe that in the context of QBF solving, it makes sense to do partial reductions. We will introduce a different notation for partial reduction later.

Conversely, we can transform a universal winning strategy into a Q-Res refutation by representing the strategy as an assignment tree[5].

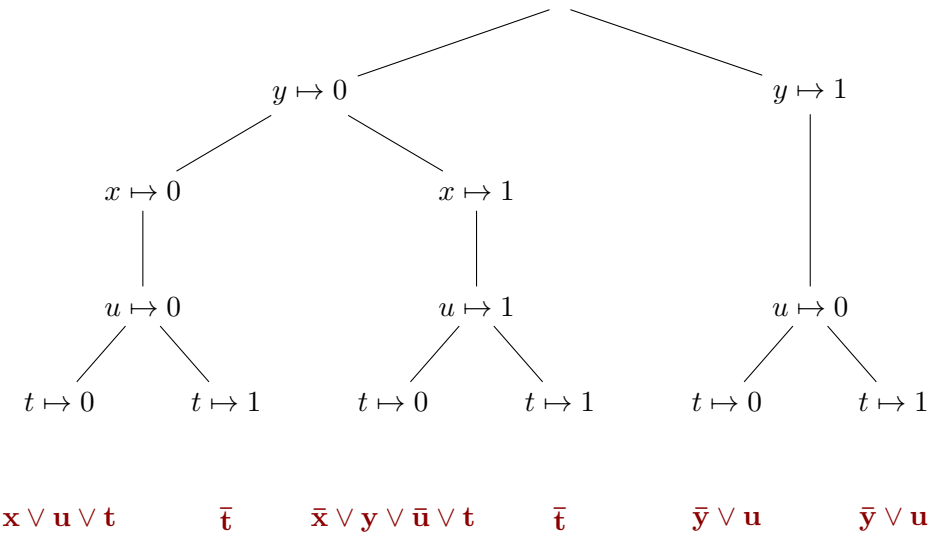
Example 2.20. *Let $\Phi := \mathcal{Q} \cdot \varphi$ be a QCNF with the prefix*

$$\mathcal{Q} := \exists x, y \forall u \exists t$$

and matrix

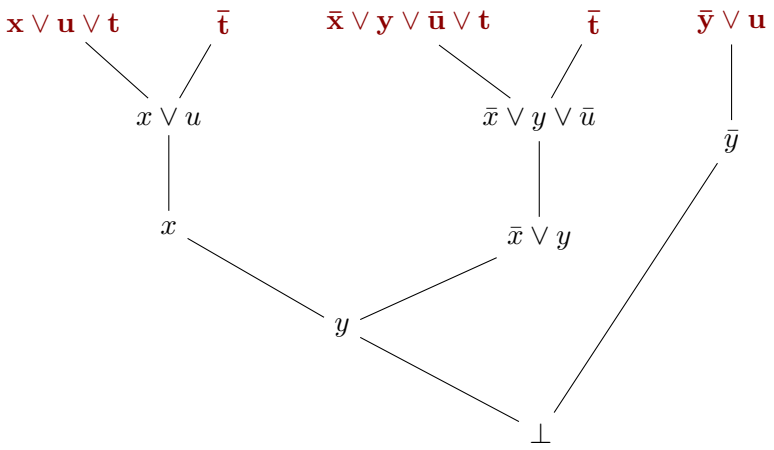
$$\varphi := (x \vee u \vee t) \wedge (\bar{x} \vee y \vee \bar{u} \vee t) \wedge (\bar{y} \vee u) \wedge \bar{t}.$$

This QBF is false, since we can construct a universal winning strategy represented by the following assignment tree, in which each path depicts a (partial) assignment that falsifies the axiom written in red:



By flipping the tree upside down, we can transform this strategy into a Q-Res refutation. Each branching on existential variables corresponds to a resolution step, while the assignments of universal variables represent reduction steps:

[5] An assignment tree is a tree is a representation for winning strategies. Each node is an assignment of one variable. Assignment trees for universal winning strategies branch on existential variables, while trees for existential winning strategies branch on universal variables. For each path from the root to a leaf, if a variable x_1 is assigned earlier than x_2, then either they are in the same quantifier block, or x_1 is left of x_2. If we combine all assigments of a path to one assignment σ, then σ falsifies the matrix if the tree represented a universal winning strategy, otherwise it satisfies the matrix.



The plain version of QBF resolution is rather restricted as we are only allowed to resolve over existential variables and we must not derive clauses that contain universal tautologies. However, we can extend Q-Res to more liberal and potentially stronger proof systems. One natural extension is QU-Resolution [72, 99] or QU-Res for short, which allows resolution steps over both existential and universal variables.

Definition 2.21 (QU-Resolution)**.** *A* QU-Res *proof π from a QCNF Φ of a clause E is a sequence of clauses $\pi = (C_1, \ldots, C_m)$, in which each C_i is derived by the inference rules of* Q-Res*, or we have $C_i = C_j \overset{u}{\bowtie} C_k$ for a universal variable $u \in C_j$ with $\bar{u} \in C_k$ and $j, k < i$.*

Another extension of Q-Res is called long-distance Q-Resolution [3], or LD-Q-Res for short, and allows the derivation of a special kind of universal tautologies[6], which are referred to as *merged literals*.

Definition 2.22 (long-distance Q-Resolution)**.** *An* LD-Q-Res *proof π from a QCNF Φ of a clause E is a sequence of clauses $\pi = (C_1, \ldots, C_m)$, in which each C_i is derived by the inference rules of* Q-Res*, but the condition $\{u, \bar{u}\} \not\subseteq C_i$ for each $u \in var_\forall(\Phi)$ does not longer need to hold. Instead, if we derive a clause C_i via $C_i = C_j \overset{x}{\bowtie} C_k$ with an existential variable x, we only require $x <_\Phi u$ for each universal literal $u \in C_j$ such that $\bar{u} \in C_k$. We will refer to the tautology $u \vee \bar{u}$ in C_i as a* merged literal *and write u^* instead.*

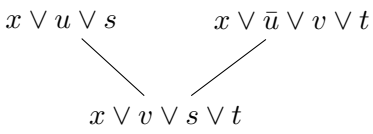
We will refer to resolution steps $C_j \overset{x}{\bowtie} C_k$ with an existential variable x, in which there exists a universal literal $u \in C_j$ with $\bar{u} \in C_k$ with $x <_\Phi u$, as long-distance steps.

QU-Res and LD-Q-Res can be combined into a new proof system called LD-QU-Res [5]. In LD-QU-Res, we are allowed to do both resolution steps over universal variables as well as long-distance steps, but not at the same time. That means that long-distance

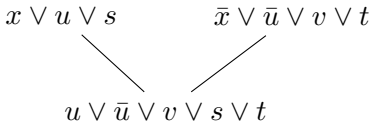
[6] While these expressions are formally tautologies, they should instead be treated as partial strategies. If the two players play their game on an LD-Q-Res refutation, these tautologies will vanish at some point as they cannot occur in the first quantifier block.

resolution steps are only allowed over existential variables and universal resolution steps are not allowed to derive merged literals.

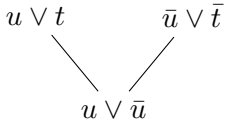
Example 2.23. *Let $\exists x, y \forall u, v \exists s, t$ be a prefix for some QBF. Then*



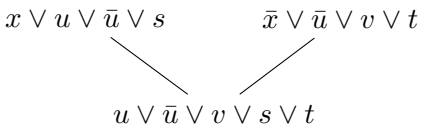
is allowed in QU-Res *and* LD-QU-Res*, but disallowed in* LD-Q-Res*.*



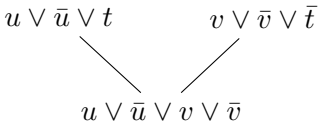
is allowed in LD-Q-Res *and* LD-QU-Res*, but disallowed in* QU-Res*.*



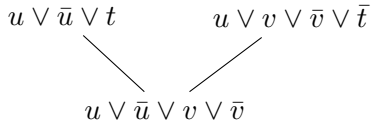
is disallowed in QU-Res*,* LD-Q-Res *and* LD-QU-Res*.*



is allowed in LD-Q-Res *and* LD-QU-Res*, but disallowed in* QU-Res*.*



is allowed in LD-Q-Res *and* LD-QU-Res*, but disallowed in* QU-Res*.*



is disallowed in QU-Res*,* LD-Q-Res *and* LD-QU-Res*.*

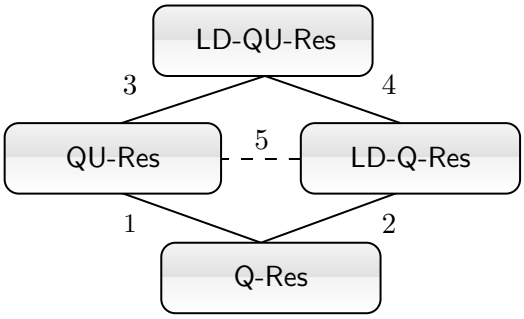
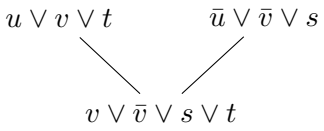


Figure 2.1: Hasse diagram of the simulation order of Q-Res, QU-Res, LD-Q-Res and LD-QU-Res. Solid lines represent p-simulations and exponential separations, while dashed lines represent incomparabilities. For detailed information, see Tables 2.4 and 2.2.



is disallowed in QU-Res*,* LD-Q-Res *and* LD-QU-Res*.*

By definition, both QU-Res and LD-Q-Res p-simulate Q-Res. Also, LD-QU-Res p-simulates both QU-Res and LD-Q-Res. In [10, 43, 99] it was shown that QU-Res and LD-Q-Res are both exponentially stronger than Q-Res. In [5] it was proven that LD-QU-Res is exponentially stronger than QU-Res and LD-Q-Res. Furthermore, the systems QU-Res and LD-Q-Res are incomparable to each other. All relations between these four proof systems are depicted in Figure 2.1.

	Simulation	Separation			
No	Theorem	Formula	easy for	hard for	Theorem
1	by Def.	$\mathtt{KBKF}_n$	QU-Res	Q-Res	[43, 99]
2	by Def.	$\mathtt{Eq}_n$	LD-Q-Res	Q-Res	[10]
3	by Def.	$\mathtt{KBKF-lqu}_n$	LD-QU-Res	QU-Res	[5]
4	by Def.	$\mathtt{KBKF-lq}_n$	LD-QU-Res	LD-Q-Res	[5]

Table 2.1: P-simulations and separations of proof systems from Figure 2.1 (i.e. the solid lines).

	Separation			
No	Formula	easy for	hard for	Theorem
5	$\mathtt{KBKF} - \mathtt{lq}_n$	QU-Res	LD-Q-Res	[5]
	$\mathtt{KBKF} - \mathtt{qu}_n$	LD-Q-Res	QU-Res	[5]

Table 2.2: Incomparable proof systems from Figure 2.1 (i.e. the dashed lines).

The completeness of QU-Res and LD-Q-Res simply follows from the completeness of Q-Res. The soundness of QU-Res and LD-Q-Res can be proven via strategy extraction. We will omit the technicalities and instead refer to [28].

Unlike SAT solving, in which the satisfiability of a formula can simply be proven by providing a satisfying assignment, proving that a QCNF is true is much more difficult. A naive approach might be negating a QCNF and transforming the obtained QDNF into a QCNF again via Tseitin transformation before constructing a refutation for this new QCNF.

Another approach that does not rely on a second Tseitin transformation includes proofs that consist of cubes instead of clauses. Such proofs are constructed analogously to resolution-like proofs and are called *consensus proofs*. We can define the proof systems Q-Con, QU-Con, LD-Q-Con and LD-QU-Con similar to their resolution-like counterparts by flipping the notions 'clause' and 'cube' as well as 'existential' and 'universal'. That means that in QU-Con, we can resolve cubes over universal variables. In QU-Con, we are allowed to resolve cubes over existential variables as well, while in LD-Q-Con we can derive cubes that include a partiular kind of existential tautology. Instead of deriving $\bot$ as the empty clause, we can derive $\top$ in consensus proofs, in which case we will call the proof a *verification* (instead of refutation).

Consensus-like proofs on QDNFs are dual to resolution-like proofs on QCNFs. However, since we normally start with a QCNF, we want to define consensus proofs particularly on QCNFs. The only inference rule we need to redefine is the axiom rule, since QCNFs do not contain any cubes that can be used as axioms. Instead, we will obtain our axiom cubes via assignments that satisfy the matrix.

Let $\Phi = \mathcal{Q} \cdot \varphi$ be a QCNF. If σ is a partial assignment of $\mathrm{var}(\Phi)$ that satisfies φ, then the cube that represents σ can be used as an axiom cube in a consensus proof for Φ. Basically, this cube is the conjunction of all literals which are set to 1 by σ. Formally, say $\sigma \in \langle V \rangle$ for some $V \subseteq \mathrm{var}(\Phi)$, then we can write σ as the cube $\bigwedge_{v \in V} v^{\sigma(v)}$. We will refer to these cubes in a consensus proof as *initial cubes*.

In order to comprehend the soundness of consensus-like proofs on QCNFs, it makes sense how these initial cubes are related to the formula syntactically and semantically. But first we need to introduce a new structure of QBFs in which we can add both clauses and cubes.

Definition 2.24 (augmented QBFs)**.** *A QBF* $\Phi = \mathcal{Q} \cdot \psi \vee \chi$ *is called an* augmented

QBF, *or* AQBF *for short, if* ψ *is a CNF and* χ *is a DNF. In this case, we will use the notations* $\mathfrak{C}(\Phi) := \psi$ *and* $\mathfrak{D}(\Phi) := \chi$.

Note that each QCNF and each QDNF could also be treated as an AQBF and that the notations $\mathfrak{C}(\Phi)$ and $\mathfrak{D}(\Phi)$ are used for QCNFs and QDNFs, as well.

Let us now assume that we start with a (true) QCNF $\Phi = \mathcal{Q} \cdot \varphi$. Let us also assume that we have already constructed a consensus verification π with I_π as the set of initial cubes in π. More precisely, each $D \in I_\pi$ is a satisfying assignment of φ. Hence, the QCNF $\mathcal{Q} \cdot \varphi$ is obviously equivalent to the AQBF $\mathcal{Q} \cdot \varphi \vee \bigvee_{D \in I_\pi} D$, because $\bigvee_{D \in I_\pi} D$ implies φ. The proof π then is simply a standard consensus-like verification of the QDNF $\mathcal{Q} \cdot \bigvee_{D \in I_\pi} D$, which also proves that Φ is true.

As for the completeness, assume that we again start with a QCNF $\Phi = \mathcal{Q} \cdot \varphi$. In the worst case, we can simply use all satisfying assignments of φ as initial cubes. Let us define J_φ as the set of all satisfying assignments of φ, written as cubes. Since assignments from existential winning strategies need to satisfy the matrix, they will also always satisfy at least one initial cube. That means that each existential winning strategy of Φ is also an existential winning strategy of $\mathcal{Q} \cdot \bigvee_{D \in J_\varphi} D$. Hence, $\mathcal{Q} \cdot \bigvee_{D \in J_\varphi} D$ is true and there exists a Q-Con verification π of it. But then π is essentially a Q-Con verification of the QCNF Φ with J_φ as the set of all initial cubes in π, which shows the completeness.

Chapter 3

SAT and QBF Solving

In the previous chapter, we have established proof systems that can prove the satisfiability of propositional formulas, or the truth or falsity of QBFs. The completeness of these proof systems guarantees the existence of such proofs. What remains open is how to find these proofs.

We demonstrated the completeness of Resolution by associating each assignment with a clause and then resolve clauses whose assignments only differ in one variable. Although this is obviously a method of finding a Resolution refutation, it is not polynomially bounded as the number of assignments is exponential in the number of variables. Our goal is therefore to find a method that has at least the potential to be polynomial on some instances. Remember that the SAT problem is NP-complete, hence we will not find an approach for SAT solving that is polynomially bounded on all instances unless P=NP.

Again, we refer to [29] for a more detailed introduction to SAT and QBF solving.

3.1 DPLL

One of the earliest algorithms for SAT solving was introduced 1961 (cf. [53]), called the *Davis-Putnam-Logemann-Loveland algorithm*, or *DPLL* for short. DPLL is a search algorithm that expects a propositional formula as input and outputs a satisfying assignment in case the formula is satisfiable, or returns 'unsatisfiable' otherwise.

Instead of going through each assignment separately like the naive truth table approach, DPLL uses two procedures that are able to exclude particular assignments from the search space which have no chance to satisfy the formula anyway. These two procedures are called *Unit Propagation* and *Pure Literal Elimination*.

For Unit Propagation, we have to detect unit clauses first. A *unit clause* in the propositional case is simply a clause consisting of only one literal. We will often write unit clauses in brackets. For example, (x) is a unit clause containing the literal x. We will denote the empty clause as $\bot$ or $(\bot)$.

Let us assume we want to determine if a CNF φ is satisfiable and say that φ contains

a unit clause (x). Then φ can obviously only be satisfied by assignments setting x to 1. Hence, assignments that set x to 0 can be ignored and we have now cut our search space in half.

Unit Propagation will not only be applied at the beginning, but at any point during the search. Usually, we will even detect chains of Unit Propagations. For example, a formula that contains the clauses (x_1), $\bar{x}_1 \vee x_2$, $\bar{x}_2 \vee x_3$, $\bar{x}_3 \vee x_4$ will trigger the propagation of x_1, then x_2 and x_3 and finally x_4.

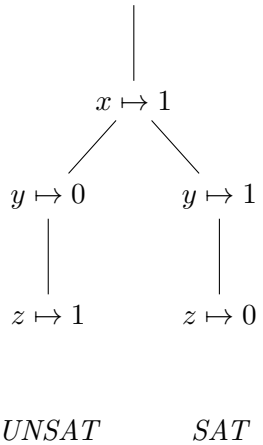
In Pure Literal Elimination, we try to detect literals that appear in only one polarity, called *pure literals*. For example, the CNF $(x \vee y) \wedge (\bar{y} \vee z) \wedge (\bar{y} \vee \bar{z})$ contains the pure literal x because $\bar{x}$ does not appear in the formula. If the CNF is satisfiable, then it suffices to find a satisfying assignment that sets x to 1 since setting it to 0 will not be of advantage. Pure Literal Elimination therefore forces us to set x to 1 and move on with the search.

As Unit Propagation, Pure Literal Elimination will also be applied in the middle of a search. In the example above, Pure Literal Elimination forces us to set x to 1, which leads us to the CNF $(\bar{y} \vee z) \wedge (\bar{y} \vee \bar{z})$. But now $\bar{y}$ has become a pure literal, as well. Therefore, we are force to set $\bar{y}$ to 1 (resp. we set y to 0) and obtain a satisfying assignment.

Unlike Unit Propagation, Pure Literal Elimination might delete satisfying assignments from the search space. However, for each deleted satisfying assignment there will remain at least one satisfying assignment in the search space. Hence, Pure Literal Elimination is still a valid approach for SAT solving.

Clearly, there will be situations in which neither Unit Propagation nor Pure Literal Elimination is applicable. We then choose a branching variable[1] x and consider the cases $x \mapsto 0$ and $x \mapsto 1$ separately. If we have found a satisfying assignment in the case $x \mapsto 0$, it is obviously not necessary to consider $x \mapsto 1$ any more.

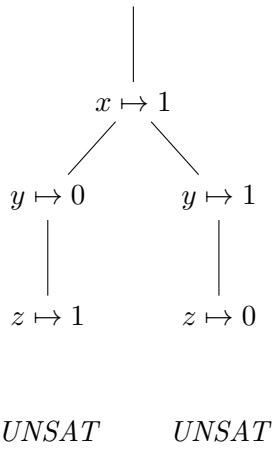
Example 3.1. *Let $\varphi = (x) \wedge (\bar{x} \vee y \vee z) \wedge (\bar{y} \vee \bar{z}) \wedge (y \vee z) \wedge (y \vee \bar{z})$ be a CNF which we want to check for satisfiability. A DPLL run can be expressed as a tree as follows:*



[1]The choice of the branching variable might matter in terms of running time.

We start with setting $x \mapsto 1$ via Unit Propagation and obtain the CNF $(y \vee z) \wedge (\bar{y} \vee \bar{z}) \wedge (y \vee z) \wedge (y \vee \bar{z})$. Because neither Unit Propagation nor Pure Literal Elimination is applicable, we branch over y. Setting $y \mapsto 0$ leads to the CNF $(z) \wedge (\bar{z})$, which forces us to set $z \mapsto 1$ by Unit Propagation. But this obviously falsifies the clause $(\bar{z})$, which means we have to consider the other branch $y \mapsto 1$. On this branch, we obtain the CNF $(\bar{z})$, which forces us to set $z \mapsto 0$ via Pure Literal Elimination or Unit Propagation. We have now constructed a satisfying assignment for φ with $x \mapsto 1$, $y \mapsto 1$ and $z \mapsto 0$.

Let us now consider a different formula $\psi = (x) \wedge (\bar{x} \vee y \vee z) \wedge (\bar{y} \vee \bar{z}) \wedge (y \vee z) \wedge (y \vee \bar{z}) \wedge (\bar{y} \vee z)$. The formula ψ is simply φ with the additional clause $\bar{y} \vee z$. Then we can obtain the following run:



On the $y \mapsto 0$ branch, everything is the same as before. However, in the $y \mapsto 1$ branch, we now obtain the CNF $(\bar{z}) \wedge (z)$, forcing us to set $z \mapsto 0$. But this falsifies the clause (z). Since both branches result in unsatisiability, we conclude that ψ is unsatisfiable.

Whenever a branch results in falsifying assignments, we need to go back a branch which we have visited yet. This is also called *backtracking*. An important property of DPLL is that its backtracking is chronological, meaning that we always have to return to the last unchecked branch. This restriction might be disadvantageous in some cases because going back to earlier branching may sometimes be more beneficial.

Because of this restriction, DPLL runs on false formulas only correspond to treelike Resolution refutations. In fact, by flipping the assignment tree upside down and identifying each leaf with a clause that got falsified by the respective assignment, we obtain a treelike proof in which each branching matches a resolution step.

3.2 CDCL and QCDCL

Conflict Driven Clause Learning, or *CDCL* for short, could be interpreted as an extension of DPLL, although one might argue that it follows a conceptually new approach.

Proposed in 1996 by Marques-Silva and Sakallah [79], CDCL utilizes two new mechanics to overcome the restrictions of DPLL.

One of such mechanics is Clause Learning. In DPLL, whenever we falsify the formula, we return 'UNSAT' for the current branch and continue with a different one. However, in CDCL we will learn a new clause before backtracking. This learned clause is supposed to contain the information of why we were not able to satisfy the formula and prevents us from visiting the exact same branch again.

For that reason we no longer have to track which branches lead to an unsatisfied state, and hence we do not need to consider all branches chronologically. The learned clauses will lead to new branches more or less automatically via Unit Propagation. Therefore, CDCL uses a non-chronological type of backtracking. In particular, we do not necessarily need to return to the last non-visited branching — we are rather allowed to backtrack back to pretty much any point in the assignment. We can even backtrack back to the beginning by undoing all assignments. This special case of backtracking is called a *restart*.

If we construct a satisfying assignment at any point in the run, we will return 'SAT'. For each falsifying assignment we find, we will learn a new clause. Since we can only learn finitely many new clauses (due to the fact that the variables are finite), we will learn the empty clause at some point. Learning the empty clause tells us that 'nothing' caused the formula to get falsified, which essentially means that the formula was unsatisfiable all along.

An important aspect of CDCL is that its Clause Learning is simulated by Resolution, meaning that each learned clause can be derived via some Resolution proof that follows the exact same steps as Clause Learning, but in reverse order. We demonstrate this in the following example:

Example 3.2. *Consider the CNF* $\varphi = (\bar{a}) \wedge (a \vee b) \wedge (c \vee d) \wedge (\bar{b} \vee c \vee \bar{d} \vee e) \wedge (g \vee h) \wedge (e \vee \bar{g} \vee \bar{h}) \wedge (e \vee g \vee \bar{h}) \wedge (\bar{g} \vee h) \wedge (\bar{g} \vee \bar{h}) \wedge (\bar{e} \vee k \vee \ell) \wedge (k \vee \bar{\ell}) \wedge (\bar{k} \vee \bar{\ell})$*. A CDCL run can be represented via implication graphs.*

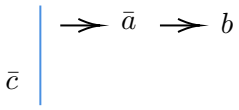
Each implication graph represents an assignment and consists of literals and arrows. The assignment is given by setting all literals appearing in the graph to 1*. The arrows indicate which assignments of literals triggered which Unit Propagations. At the beginning, the only unit clause in* φ *is* $(\bar{a})$*, hence the propagation of* $\bar{a}$ *was caused by 'nothing' and we can write:*

$$\rightarrow \bar{a}$$

Setting a *to* 0 *triggers the propagation of* b *because* $a \vee b$ *becomes the unit clause* (b)*:*

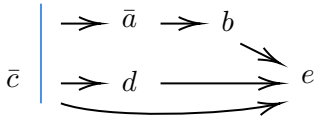
$$\rightarrow \bar{a} \rightarrow b$$

We can detect no more Unit Propagations, an since no literal became pure in the meantime, we need to do something which we called branching before. In CDCL, we simply call it a decision *because technically we do not work with branchings any more. Let us assume we set* c *to* 0 *by decision. Since nothing caused or triggered this decision, we do not use any arrows.*

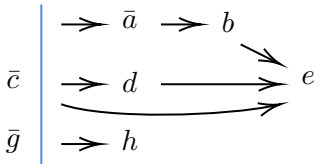


To separate decisions from propagations, we always write decisions in a new row and draw a blue line between them. Each row in the graph denotes a separate decision level. *In our example, the first row represents decision level* 0, *because all assignments on this level were caused without any decision. The assignment of* $\bar{c}$ *then introduces decision level* 1, *since* $\bar{c}$ *is the first decision in the graph.*

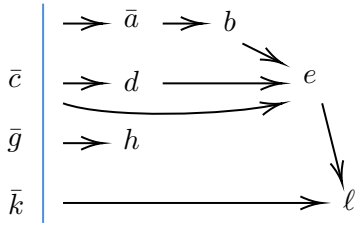
Now $\bar{c}$ *triggers the propagation of* d *via* $c \vee d$, *before* b, $\bar{c}$ *and* d *imply* e *via* $\bar{b} \vee c \vee \bar{d} \vee e$.



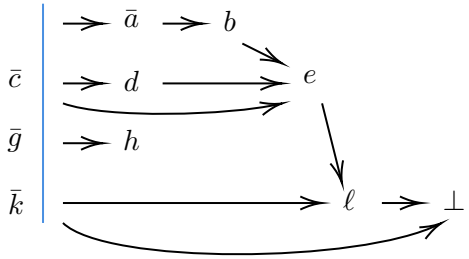
No more propagations are applicable, therefore we have to decide another literal. Say we set g *to* 0, *which causes* h *to be propagated by* $g \vee h$.



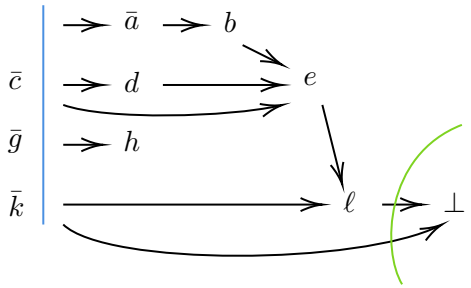
After that, we have to make yet another decision. We set k *to* 0 *and trigger the propagation of* ℓ *with the clause* $\bar{e} \vee k \vee \ell$.



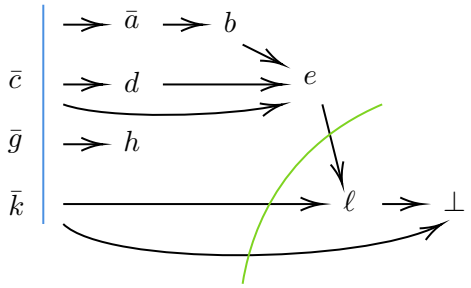
But now we have falsified the clause $k \vee \bar{\ell}$ *by having set* k *to* 0 *and* ℓ *to* 1. *We call this a* conflict *(on the clause* $k \vee \bar{\ell}$*). Such a conflict is represented by* $\bot$ *in the graph as if it was a literal propagated by* $\bar{k}$ *and* ℓ.



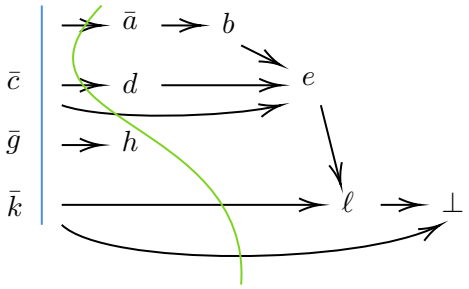
Since we have detected a conflict, we stop assigning more variables and start Clause Learning. We can depict Clause Learning by dividing the graph into the conflict side *and the* reason side*. Intuitively, the conflict side can be interpreted as the conflict itself as it will always contain $\bot$. The reason side provides the cause of the conflict. In other words: The assignments from the reason side cause the propagations of the conflict side that eventually lead to the conflict.*



At the beginning, the conflict side only consists of $\bot$ itself. The green cut not only separates both sides, but also tells us which assignments led to the conflict via the crossed arrows. In our example, the cut went through the arrows that start at $\bar{k}$ and ℓ, meaning that $\bar{k}$ and ℓ led to the conflict. We could learn the clause $k \vee \bar{\ell}$. However, because our conflict side only consists of the conflict itself, the learned clause is already contained in the CNF. More precisely, the first learnable clause will always be the conflict clause (i.e., the clause that got falsified). We therefore have to move the cut to the left by moving each propagated literal from the reason side to the conflict side in reverse order of propagation. In our case, we can move ℓ to the conflict side.

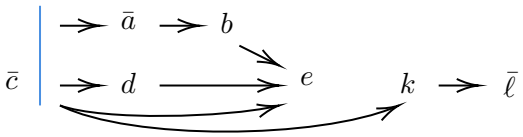


Now we can read off e and $\bar{k}$ as the conflict reason. We could learn the clause $\bar{e} \vee k$ from this cut and stop Clause Learning for now, but we are also allowed to move on. Let us continue moving the cut as far to the left as possible. Note that only propagated literals are allowed in the conflict side.

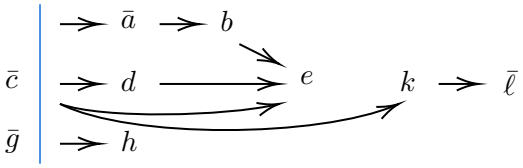


We do not need to move h to the conflict side since it did not contribute to the conflict. We have now obtained a cut for which only decisions caused the conflict essentially. We can now learn the clause $c \vee k$.

This clause is added to the CNF, such that we get $(\bar{a}) \wedge (a \vee b) \wedge (c \vee d) \wedge (c \vee \bar{b} \vee \bar{d} \vee e) \wedge (g \vee h) \wedge (e \vee \bar{g} \vee \bar{h}) \wedge (e \vee g \vee \bar{h}) \wedge (\bar{g} \vee h) \wedge (\bar{g} \vee \bar{h}) \wedge (\bar{e} \vee k \vee \ell) \wedge (k \vee \bar{\ell}) \wedge (\bar{k} \vee \bar{\ell}) \wedge (c \vee k)$. We can backtrack back to pretty much any point in the graph (although some points are more beneficial than others). In our case, it makes sense to backtrack back to a point before we decided $\bar{k}$ to trigger a new propagation of k.



The assignment of k then triggers the propagation of ℓ. After that, we again have to make a decision to move on. Let us decide $\bar{g}$, which then leads to h via Unit Propagation as before.

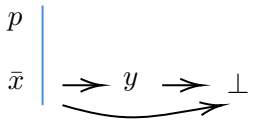


Now we have assigned all variables and not falsified any clause, which means we have found a satisfying assignment.

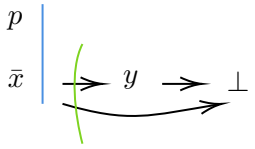
Let us take a look at another example. This time, we consider an unsatisfiable CNF.

Example 3.3. *Let $\varphi = (x \vee y) \wedge (x \vee \bar{y}) \wedge (\bar{x} \vee y) \wedge (\bar{x} \vee \bar{y}) \wedge (p \vee z) \wedge (p \vee \bar{z})$. We start with setting p to 1 via Pure Literal Elimination.*

Literals that are assigned via Pure Literal Elimination can be treated as decisions. Note that the assignment of pure literals will never trigger any propagations by definition. Hence, we have to continue with another decision.

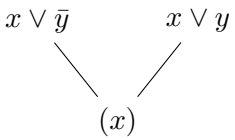


After propagating y, we detect a conflict on the clause $x \vee \bar{y}$. We move y to the conflict side and get the following:



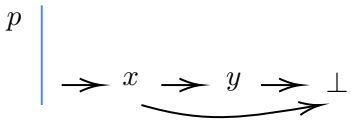
We obtain the learned clause (x). This time, we do not only want to add the learned clause to the formula, but we will also provide a Resolution *proof of this new clause. We can construct this proof by using all axiom clauses that triggered the propagations and conflicts on the conflict side. More precisely, we will resolve them in reverse order of their appearance in the graph.*

In our example, the clause which caused the conflict was $x \vee \bar{y}$. The propagation before the conflict was triggered by $x \vee y$. It is clear that these two clauses must be resolvable, since the propagated literal y has to appear in two different polarities, otherwise the propagation caused by one clause would not have caused the falsification of the other one. After resolving these two clauses over y, we obtain the learned clause $(x) = (x \vee \bar{y}) \overset{y}{\bowtie} (x \vee y)$:

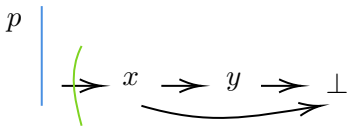


In fact, learned clause can always be derived via Resolution *by resolving all clauses that contributed to the propagations of the conflict side.*

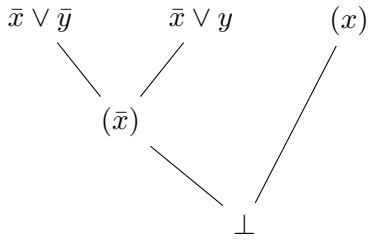
We now add the learned clause to the formula and get $(x \vee y) \wedge (x \vee \bar{y}) \wedge (\bar{x} \vee y) \wedge (\bar{x} \vee \bar{y}) \wedge (p \vee z) \wedge (p \vee \bar{z}) \wedge (x)$. We backtrack back to the point before we decided $\bar{x}$ and obtain:



The unit clause (x) not triggers the propagation of x without any decision, which then causes the propagation of y via $\bar{x} \vee y$ and finally the conflict on $\bar{x} \vee \bar{y}$. During Clause Learning, we will move y and x to the conflict side and get the following cut:

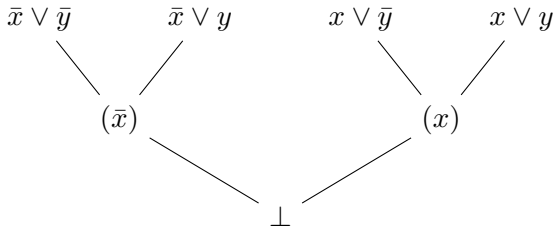


Now, the cause for the conflict was 'nothing', hence we can learn the empty clause $\bot$*. In fact, we can derive* $\bot$ *by resolving the three used clauses as follows:*



Alternatively, we can write $\bot = ((\bar{x} \vee \bar{y}) \overset{y}{\bowtie} (\bar{x} \vee y)) \overset{x}{\bowtie} (x)$.

Although we have learned the empty clause, we have not yet found a Resolution *refutation of* φ *since* (x) *is no axiom clause. However, we already have derived* (x) *via* Resolution *during Clause Learning by using only original clauses as axioms. We can stick together these two proofs and obtain a* Resolution *refutation of the original formula* φ*:*



Like Resolution, the CDCL paradigm can be lifted to the QBF problem. This extension of CDCL is called *QCDCL* or *QBF CDCL*.

Given a QCNF Φ, we want to determine the truth value of Φ by constructing a resolution-like refutation or verification (in the standard version, we want to find a proof in LD-Q-Res). We will still do Unit Propagation as propagations correspond to resolution steps in the proof. We could also perform Pure Literal Elimination, but we will omit it for now as we will analyse its influence on QCDCL later (cf. Chapter 6). In order to lift CDCL to QCDCL, we have to deal with two QBF-specific aspects: reductions and true formulas.

In QCDCL, reductions affect both Unit Propagation and Clause Learning. During Clause Learning, we will perform universal reduction after each resolution step. Such reductions will never be disadvantageous as they will simply shorten the learned clause, which might cause finding further propagations or conflicts even sooner.

Besides Clause Learning, we will also perform reductions during Unit Propagation. A clause that becomes unit after a reduction is essentially a unit clause in the context of a two-player game. For example, consider the clauses $(\bar{y}) \wedge (\bar{y} \vee \bar{t}) \wedge (x \vee u \vee t)$ under the prefix $\exists x, y \forall u \exists t$. The existential player has to set y to 0, otherwise they lose the game immediately. But then $y \vee \bar{t}$ becomes the unit clause $(\bar{t})$. Therefore, the existential player has to set t to 0 independently from u (technically, the existential player has to assign t

only after the universal player assigned u, but the assignment of t is essentially already determined directly after assigning y). We then obtain the clause $x \vee u$, which is, by the definition of unit clauses in the propositional case, not a unit clause. However, we get the unit clause (x) after universally reducing $x \vee u$. In fact, the existential player has to set x to 1, otherwise the universal player can easily win the game. Therefore, we will call a clause C *unit (in the sense of QBF)* if $\mathrm{red}_\forall(C)$ is a unit clause in the propositional sense. Analogously, a clause C is *falsified (in the sense of QBF)*, if $\mathrm{red}_\forall(C) = \bot$.

As another difference, the decisions have to be made along the quantifier prefix in the standard version of QCDCL. More precisely, an existential variable can only be decided after all universal variables left of it are already assigned. Similarly, universal variables can only be decided once all existential variables to the left are assigned. Simply put, the decisions have to respect the order of the two-player game. We will later analyse different versions of QCDCL, in which less restricting decision rules are established (cf. Chapter 7).

Let us take a look at an example that demonstrates the changes explained above.

Example 3.4. *Let Φ be the QCNF with the prefix*

$$\exists x_1, x_2 \forall u_1, u_2 \exists t_1, t_2$$

and the matrix

$$\begin{aligned}&(x_1 \vee u_1 \vee t_1) \wedge (\bar{x}_1 \vee \bar{u}_1 \vee t_1) \wedge \\ &(x_2 \vee u_2 \vee t_2) \wedge (\bar{x}_2 \vee \bar{u}_2 \vee t_2) \wedge \\ &(\bar{t}_1 \vee \bar{t}_2).\end{aligned}$$

No Unit Propagations are available, hence we have to decide a variable from the first quantifier block. Let us assume we set x_1 to 0.

$$\bar{x}_1 \mid$$

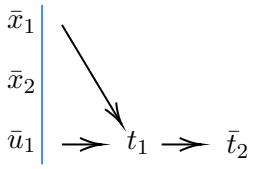
Since nothing happens, we can continue deciding by setting x_2 to 0 as well.

$$\begin{aligned}\bar{x}_1 &\mid \\ \bar{x}_2 &\mid\end{aligned}$$

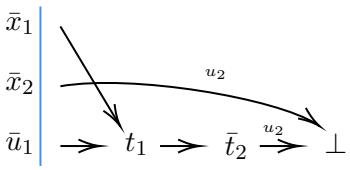
We have still not triggered any propagations, hence we move on to the next block and decide $\bar{u}_1$.

$$\begin{aligned}\bar{x}_1 &\mid \\ \bar{x}_2 &\mid \\ \bar{u}_1 &\mid\end{aligned}$$

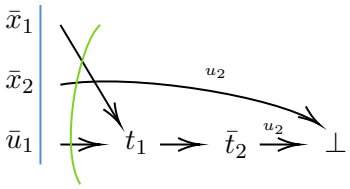
This causes the propagation of t_1 *and the clause* $(\bar{t}_1 \vee \bar{t}_2)$ *becomes unit, which forces us to set* t_2 *to* 0.



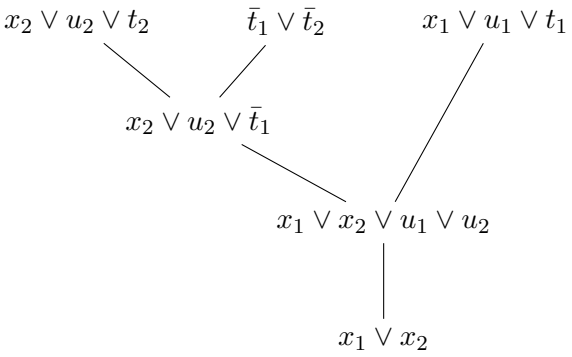
After propagating $\bar{t}_2$*, we obtain the clause* (u_2)*. However, since this clause can be universally reduced to* $\bot$*, this already causes a conflict.*



We write u_2 *on the arrows leading to* $\bot$ *to indicate that* u_2 *was reduced during this Unit Propagation (or conflict). We now start Clause Learning by dividing the graph into conflict and reason side as before. We then move the cut as far to the left as possible (of course, we could also stop Clause Learning earlier if we want).*



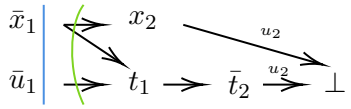
Since we cross an arrow with the u_2 *annotation, the learned clause corresponding to that cut might also contain* u_2 *and not only variables from the reason side. However, this time* u_2 *gets reduced away in the* LD-Q-Res *derivation of the learned clause nevertheless:*



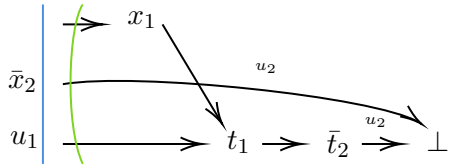
Note that reductions during Unit Propagation do not necessarily correlate with reductions during Clause Learning. For example, if we would have stopped Clause Learning at the clause $x_2 \vee u_2 \vee \bar{t}_1$, the universal literal u_2 would have not been able to be reduced.

We then move on by adding $x_1 \vee x_2$ to the matrix and backtrack back to a suitable point just like in the propositional case. Say we backtrack back to the decision of $\bar{x}_1$.

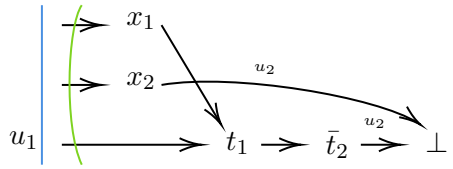
We sketch the following iterations by listing implication graphs and learned clauses only:



We learn the unit clause (x_1) and do a restart.



We learn the unit clause (x_2) and backtrack back to the point before we decided $\bar{x}_2$.



We can now learn the empty clause $\bot$ from this cut and end the run. As before, the LD-Q-Res *refutation (in this case even a* Q-Res *refutation) can be obtained by sticking together the derivations of the learned clauses.*

In SAT solving, as is known, the satisfiability of a propositional formula can be proven by providing a satisfying assignment. The same does obviously not hold for true QBFs. Hence, we need QCDCL not only to find refutations, but verifications as well.

Satisfying the matrix of a QCNF during QCDCL does not determine the truth value of the formula, hence the run is not supposed to stop after assigning all variables without running into a conflict. Instead, as with conflicts, we need to learn a constraint after having satisfied all clauses. However, in place of clauses, we will learn cubes that represent such a satisfying assignment. These cubes correspond precisely to the initial cubes from LD-Q-Con verifications.

Learning a cube from a QCNF and adding it to the matrix converts the formula into an AQBF. We can therefore say that QCDCL is technically applied to AQBFs and not just QCNFs.

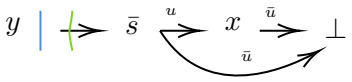
Cubes behave dually to clauses in QCDCL. Unlike clauses, cubes are used for propagating universal literals. A unit cube with the universal literal u will be denoted as $[u]$.

We can run into conflicts on cubes by satisfying them, which triggers Cube Learning that works similar to Clause Learning. The empty cube is written as $\top$ or $[\top]$. We can use $[u]$ to propagate $\bar{u}$ via Unit Propagation, otherwise $[u]$ will be satisfied immediately and we would obtain the empty cube. Instead of reducing universally, we will perform existential reduction during Unit Propagation on cubes.

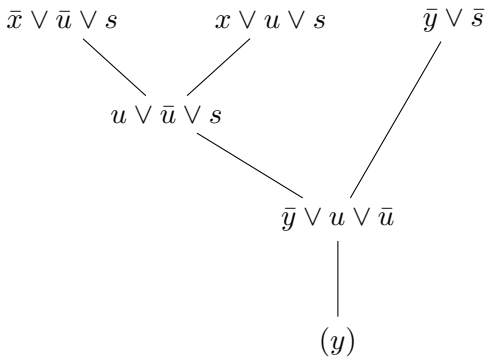
In Clause Learning, literals that were propagate via cubes will be treated as decisions since we cannot resolve clauses and cubes. Analogously, literals that were propagated via clauses will be treated as decisions during Cube Learning.

In the next example, we demonstrate how we use Cube Learning to verify a true QBF via QCDCL.

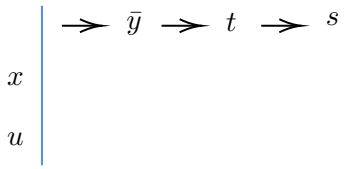
Example 3.5. *Let* $\Phi := \exists x, y \forall u \exists s, t \cdot (x \vee u \vee s) \wedge (\bar{x} \vee \bar{u} \vee s) \wedge (y \vee t) \wedge (s \vee \bar{t}) \wedge (\bar{y} \vee \bar{s})$ *be the given QCNF. We start propagating and deciding literals as before.*



Note that it is still possible to run into conflicts even on true formulas. Here, we can learn the unit clause $(\bar{y})$*, which is even derived by genuine long-distance steps this time:*



We restart and can start with assigning $\bar{y}$ *via Unit Propagation next.*



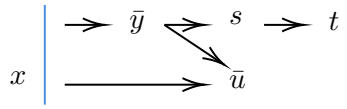
Now we have found a satisfying assignment of the matrix. We can write this assignment as the cube $x \wedge \bar{y} \wedge u \wedge s \wedge t$ *and add it to the matrix*[2]*. Since we can reduce*

[2]In this example, we could even learn the cube $\bar{y} \wedge s \wedge t$, since the corresponding assignment already satisfies the matrix. This cube can now be existentially reduced to $\top$, which would immediately end the run because we already constructed an LD-Q-Con verification of Φ. However, for the sake of interestingness and due to the fact that QCDCL solvers are not necessarily expected to learn optimal cubes in practice, we will continue with $x \wedge \bar{y} \wedge u \wedge s \wedge t$.

existentially on cubes, we will instead learn the shorter cube $x \wedge \bar{y} \wedge u$*. The formula now looks as follows:*

$$\exists x, y \forall u \exists s, t \cdot \big((x \vee u \vee s) \wedge (\bar{x} \vee \bar{u} \vee s) \wedge (y \vee t) \wedge (s \vee \bar{t}) \wedge (\bar{y} \vee \bar{s}) \wedge (\bar{y})\big) \vee (x \wedge \bar{y} \wedge u)$$

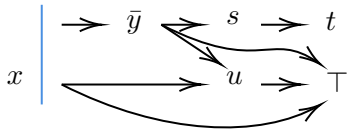
We undo the decision u *such that we can propagate* $\bar{u}$ *via* $x \wedge \bar{y} \wedge u$*.*



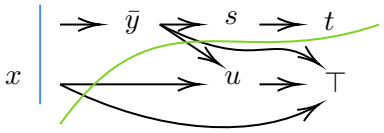
This satisfies the matrix again and we can learn another initial cube, say $x \wedge \bar{y} \wedge \bar{u}$*. The matrix now looks as follows:*

$$((x \vee u \vee s) \wedge (\bar{x} \vee \bar{u} \vee s) \wedge (y \vee t) \wedge (s \vee \bar{t}) \wedge (\bar{y} \vee \bar{s}) \wedge (\bar{y})) \vee (x \wedge \bar{y} \wedge u) \vee (x \wedge \bar{y} \wedge \bar{u})$$

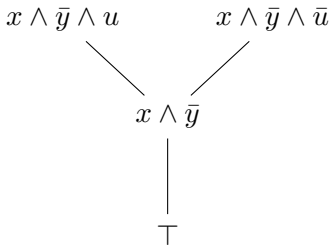
We can undo the last propagation and run into a conflict on one of the learned cubes.



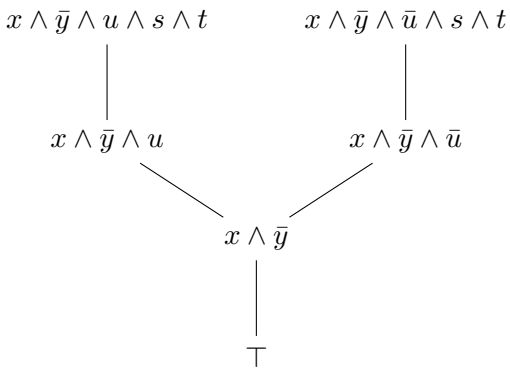
We now start Cube Learning similar to Clause Learning by dividing the graph into conflict side and reason side. The propagated literals $\bar{y}$*,* s *and* t *are treated as decisions because they got propagated via clauses, so we can only move* u *to the conflict side.*



We resolve the two cubes over u *and obtain* $x \wedge \bar{y}$*, which can be reduced to the empty cube.*



Note that the LD-Q-Res *derivation of* $(\bar{y})$ *does not appear in the final* LD-Q-Con *verification of* Φ*. Technically, the whole* LD-Q-Con *verification looks as follows, although we will usually omit the non-reduced initial cubes:*



Remark 3.6. *While we focus on the regular clause learning method for QCDCL, in which clauses and cubes are resolved in the reversed order as they appear in the implication graph, starting at the conflict, an alternative approach called* QPUP *(*QBF pseudo unit propagation*, cf. [78]) allows the derivation of learned clauses via* Q-Res *instead of* LD-Q-Res*. In QPUP, resolution steps are performed in the same order as the corresponding constraints appear in the implication graphs until the conflict clause is reached.*

For a more formal definition of QCDCL as well as completeness and soundness of QCDCL as a proof system, we refer to the next chapter.

Chapter 4

Formalizations of QCDCL

In the previous chapter, we have introduced QCDCL via implication graphs and cuts. However, from this chapter onward, we will represent QCDCL runs in a more compact and formal way. In order to analyse the complexity of QCDCL, we need to formalise it as a proof system like Q-Res.

Instead of a sequence of clauses or cubes, we will interpret QCDCL runs as sequences of so-called *trails*. In these trails we store all relevant information of a QCDCL run. Since QCDCL uses several runs and potentially also restarts, a QCDCL proof will typically consist of many trails.

Definition 4.1 (trails)**.** *A* trail $\mathcal{T}$ *for a QCNF or AQBF* Φ *is a (finite) sequence of pairwise distinct literals from* Φ*, including the empty literals* $\bot$ *and* $\top$*. Each two literals in* $\mathcal{T}$ *have to correspond to pairwise distinct variables from* Φ*. In general, a trail has the form*

$$\mathcal{T} = (p_{(0,1)}, \dots, p_{(0,g_0)}; \mathbf{d_1}, p_{(1,1)}, \dots, p_{(1,g_1)}; \dots; \mathbf{d_r}, p_{(r,1)}, \dots, p_{(r,g_r)}), \tag{4.1}$$

where the d_i *are* decision literals *and* $p_{(i,j)}$ *are* propagated literals*. Decision literals are written in* ***boldface****. We use a semicolon before each decision to mark the end of a decision level. If one of the empty literals* $\bot$ *or* $\top$ *is contained in* $\mathcal{T}$*, then it has to be the last literal* $p_{(r,g_r)}$*. In this case, we say that* $\mathcal{T}$ *has* run into a conflict*.*

Trails can be interpreted as non-tautological sets of literals, and therefore as (partial) assignments. We write $x <_\mathcal{T} y$ *if* $x, y \in \mathcal{T}$ *and* x *is left of* y *in* $\mathcal{T}$*. Furthermore, we write* $x \leq_\mathcal{T} y$ *if* $x <_\mathcal{T} y$ *or* $x = y$*.*

As trails are produced gradually from left to right in an algorithm, we define $\mathcal{T}[i,j]$ *for* $i \in \{0, \dots, r\}$ *and* $j \in \{0, \dots, g_i\}$ *as the subtrail that contains all literals from* $\mathcal{T}$ *up to (and excluding)* $p_{(i,j)}$ *(resp.* d_i*, if* $j = 0$*) in the same order. Intuitively,* $\mathcal{T}[i,j]$ *is the state of the trail before we assigned the literal at the point* $[i,j]$ *(which is* $p_{(i,j)}$ *or* d_i*). For the sake of simplicity, we define* $\mathcal{T}[0,0]$ *as the empty trail.*

For each propagated literal $p_{(i,j)} \in \mathcal{T}$ *there has to be be a clause (or cube)* $ante_\mathcal{T}(p_{(i,j)})$ *such that* $red(ante_\mathcal{T}(p_{(i,j)})|_{\mathcal{T}[i,j]}) = (p_{(i,j)})$ *(or* $[p_{(i,j)}]$*). We call such a clause (cube) the* antecedent clause (cube) *of* $p_{(i,j)}$*.*

We state some general facts about trails and antecedent clauses/cubes.

Remark 4.2. *Let $\mathcal{T}$ be a trail, $\ell \in \mathcal{T}$ a propagated literal and $A := ante_{\mathcal{T}}(\ell)$.*

- *If ℓ is existential, then $\ell \in A$ and for each existential literal $x \in A$ with $x \neq \ell$ we need $\bar{x} <_{\mathcal{T}} \ell$.*
- *If ℓ is universal, then $\bar{\ell} \in A$ and for each universal literal $u \in A$ with $u \neq \bar{\ell}$ we need $u <_{\mathcal{T}} \ell$.*

Definition 4.3 (natural trails)**.** *We call a trail $\mathcal{T}$* natural *for formula Φ, if for each $i \in \{0, \ldots, r\}$ the formula $red(\Phi|_{\mathcal{T}[i,0]})$, contains no unit or empty constraints. Furthermore, the formula $red(\Phi|_{\mathcal{T}[i,j]})$ must not contain empty constraints for each $i \in \{1, \ldots, r\}$, $j \in \{1, \ldots, g_i\}$, except $[i, j] = [r, g_r]$. Intuitively, this means that decisions are only made if there are no more propagations on the same decision level possible. Also, conflicts must be immediately taken care of.*

Definition 4.4 (learnable constraints)**.** *Let $\mathcal{T}$ be a trail for Φ of the form* (4.1) *with $p_{(r,g_r)} \in \{\bot, \top\}$. Starting with $ante_{\mathcal{T}}(\bot)$ (resp. $ante_{\mathcal{T}}(\top)$) we reversely resolve with the antecedent clauses (cubes) that were used to propagate the existential (universal) variables, until we stop at some point. Literals that were propagated via cubes (clauses) will be interpreted as decisions. If a resolution step cannot be performed at some point due to a missing pivot, we simply skip that antecedent. The clause (cube) we so derive is a* learnable constraint*. We denote the sequence of learnable constraints by $\mathfrak{L}(\mathcal{T})$.*

We can also learn cubes from trails that did not run into conflict. If $\mathcal{T}$ is a total assignment of the variables from Φ, then we define the set of learnable constraints as the set of cubes $\mathfrak{L}(\mathcal{T}) := \{red^{\exists}(D) |\, D \subseteq \mathcal{T}$ and D satisfies $\mathfrak{C}(\Phi)\}$.

Formally, if $\mathcal{T}$ is a trail for Φ of the form

$$\mathcal{T} = (p_{(0,1)}, \ldots, p_{(0,g_0)}; \mathbf{d_1}, p_{(1,1)}, \ldots, p_{(1,g_1)}; \ldots; \mathbf{d_r}, p_{(r,1)}, \ldots, p_{(r,g_r)}),$$

with $p_{(r,g_r)} \in \{\bot, \top\}$. Then we will denote the sequence of learnable constraints *$\mathfrak{L}(\mathcal{T})$ as*

$$\mathfrak{L}(\mathcal{T}) := (C_{(r,g_r)}, \ldots, C_{(r,1)}, \ldots, C_{(0,g_0)}, \ldots, C_{(0,1)}),$$

in which the clauses or cubes $C_{(i,j)}$ are recursively defined as:

If $p_{(r,g_r)} = \bot$, then

- $C_{(r,g_r)} := red^{\forall}(ante(\bot))$.
- *For $i \in \{0, \ldots, r\}$, $j \in \{1, \ldots, g_i - 1\}$, if $\bar{p}_{(i,j)} \in C_{(i,j+1)}$ and $p_{(i,j)}$ existential, then*

$$C_{(i,j)} := red^{\forall}\left(C_{(i,j+1)} \overset{p_{(i,j)}}{\bowtie} red^{\forall}\left(ante(p_{(i,j)})\right)\right),$$

otherwise $C_{(i,j)} := C_{(i,j+1)}$.

- *For $i \in \{0, \dots, r-1\}$, if $\bar{p}_{(i,g_i)} \in C_{(k,1)}$ and $p_{(i,g_i)}$ existential, then*

$$C_{(i,g_i)} := red^{\forall}\left(C_{(k,1)} \overset{p_{(i,g_i)}}{\bowtie} red^{\forall}\left(ante(p_{(i,g_i)})\right)\right),$$

  otherwise $C_{(i,g_i)} := C_{(k,1)}$ where $k := \min\{i < h \leq r |\, g_h > 0\}$ (note that always $g_r > 0$).

If $p_{(r,g_r)} = \top$, then

- $C_{(r,g_r)} := red^{\exists}(ante(\top))$.
- *For $i \in \{0, \dots, r\}$, $j \in \{1, \dots, g_i - 1\}$, if $p_{(i,j)} \in C_{(i,j+1)}$ and $p_{(i,j)}$ universal, then*

$$C_{(i,j)} := red^{\exists}\left(C_{(i,j+1)} \overset{p_{(i,j)}}{\bowtie} red^{\exists}\left(ante(p_{(i,j)})\right)\right),$$

  otherwise $C_{(i,j)} := C_{(i,j+1)}$.
- *For $i \in \{0, \dots, r-1\}$, if $p_{(i,g_i)} \in C_{(k,1)}$ and $p_{(i,g_i)}$ universal, then*

$$C_{(i,g_i)} := red^{\exists}\left(C_{(k,1)} \overset{p_{(i,g_i)}}{\bowtie} red^{\exists}\left(ante(p_{(i,g_i)})\right)\right),$$

  otherwise $C_{(i,g_i)} := C_{(k,1)}$ where $k := \min\{i < h \leq r |\, g_h > 0\}$.

Generally, we allow to learn an arbitrary constraint. However, for the characterisations, it suffices to concentrate on Clause Learning. Additionally, most of the time we will simply learn the clause which we obtain after resolving over every available literal in the trail. This clause can only consist of negated decision literals, and literals that were reduced during unit propagation. Since this is the last clause we can derive during Clause Learning in a trail $\mathcal{T}$, we will refer to that clause as the *rightmost clause in* $\mathfrak{L}(\mathcal{T})$.

Definition 4.5 (QCDCL proof systems)**.** *A* QCDCL *proof ι from a QCNF $\Phi = \mathcal{Q} \cdot \varphi$ of a clause or cube C is a (finite) sequence of triples*

$$\iota := [(\mathcal{T}_i, C_i, \pi_i)]_{i=1}^{m},$$

where $C_m = C$, each $\mathcal{T}_i$ is a trail for Φ_i,each $C_i \in \mathfrak{L}(\mathcal{T}_i)$ is one of the constraints we can learn from each trail and π_i is the proof from Φ_i of C_i we obtain by performing the steps described in Definition 4.4, where Φ_i are AQBFs that are defined recursively by setting $\Phi_1 := \mathcal{Q} \cdot (\mathfrak{C}(\Phi) \vee \emptyset)$ and

$$\Phi_{j+1} := \begin{cases} \mathcal{Q} \cdot ((\mathfrak{C}(\Phi_j) \wedge C_j) \vee \mathfrak{D}(\Phi_j)) & \text{if } C_j \text{ is a clause,} \\ \mathcal{Q} \cdot (\mathfrak{C}(\Phi_j) \vee (\mathfrak{D}(\Phi_j) \vee C_j)) & \text{if } C_j \text{ is a cube,} \end{cases}$$

for $j = 1, \dots, m-1$. If necessary, we set $\pi_i := \emptyset$.

For each decision literal d_j in some trail $\mathcal{T}_i$, we need that $lv_{\Phi|_{\mathcal{T}_i[j,0]}}(d_j) = 1$. I.e., decisions are level-ordered[1]. We require that $\mathcal{T}_1$ is a natural trail and for each $2 \leq i \leq m$ there is a point $[a_i, b_i]$ such that $\mathcal{T}_i[a_i, b_i] = \mathcal{T}_{i-1}[a_i, b_i]$ and $\mathcal{T}_i \backslash \mathcal{T}_i[a_i, b_i]$ has to be a natural trail for $\Phi_i|_{\mathcal{T}_i[a_i,b_i]}$. This process is called backtracking. *If $\mathcal{T}_{i-1}[a_i, b_i] = \emptyset$, then this is also called a* restart. *If $\mathcal{T}_{i-1}$ has run into a conflict, we require that $\mathcal{T}_{i-1}[a_i, b_i]$ does not contain all decision literals from $\mathcal{T}_{i-1}$ (i.e., at least the last decision that contributed to the conflict needs to be undone during backtracking).*

If $C = C_m = (\bot)$, then ι is called a QCDCL refutation *of Φ. If $C = C_m = [\top]$, then ι is called a* QCDCL verification *of Φ. The proof ends once we have learned $(\bot)$ or $[\top]$.*

If C is a clause, we can stick together the LD-Q-Res *derivations from $\{\pi_1, \dots, \pi_m\}$ and obtain a* LD-Q-Res *proof from Φ of C, which we call $\mathfrak{R}(\iota)$. Similarly, if C is a cube, we can stick together the* LD-Q-Con *derivations and obtain a* LD-Q-Con *proof $\mathfrak{R}(\iota)$ from Φ of C.*

The size *of ι is defined as $|\iota| := \sum_{i=1}^{m} |\mathcal{T}_i|$. Obviously, we have $|\mathfrak{R}(\iota)| \in \mathcal{O}(|\iota|)$.*

Definition 4.6. *A* QCDCL *proof in which all learnt constraints are clauses is called a* QCDCL$^{w/o\ Cubes}$ *proof. Equivalently, all trails of this proof have to run into a conflict on clauses.*

Lemma 4.7. *It is not possible to create clauses (cubes) that contain existential (universal) tautologies $x \vee \bar{x}$ during the derivation of learnable clauses as described in Definition 4.4.*

Proof. Let $\mathcal{T}$ be a QCDCL trail and $\mathfrak{L}(\mathcal{T})$ be the sequence of learnable constraints as described in Definition 4.4. It suffices to concentrate on the case $C_{(i,j)}$, in particular

$$C_{(i,j)} = \operatorname{red}\left(C_{(i,j+1)} \overset{p_{(i,j)}}{\bowtie} \operatorname{red}(\operatorname{ante}(p_{(i,j)}))\right)$$

for $i \in \{0, \dots, r\}$, $j \in [g_i - 1]$. This is analogous to the case for $C_{(i,g_i)}$ (the case $C_{(r,g_r)}$ is trivial).

Assume that $C_{(i,j)}$ is a clause (cube) and there is an existential (universal) literal x with $\operatorname{var}(x) \neq \operatorname{var}(p_{(i,j)})$ such that $x \in C_{(i,j+1)}$ and $\bar{x} \in \operatorname{red}(\operatorname{ante}_{\mathcal{T}}(p_{(i,j)}))$. It holds $\operatorname{red}(A|_{\mathcal{T}[i,j]}) = (p_{(i,j)})$ for $A := \operatorname{ante}_{\mathcal{T}}(p_{(i,j)})$. Since x is existential (universal), we conclude $x \in \mathcal{T}[i,j]$ ($\bar{x} \in \mathcal{T}[i,j]$) since existential (universal) literals cannot be reduced in clauses (cubes).

On the other hand, we have $x \in C_{(i,j)}$, where $C_{(i,j)}$ is a learnable clause (cube) which is derived with the aid of antecedent clauses of literals occurring right of $p_{(i,j)}$ in the trail. In particular, we can find some $p_{(k,m)}$ right of $p_{(i,j)}$ in the trail with $x \in \operatorname{ante}_{\mathcal{T}}(p_{(k,m)})$.

[1] In other words, existential decisions can only be made if all universal variables that are quantified earlier are already assigned. Analogously, universal decisions can only be made if all existential variables that are quantified earlier are already assigned. We will later define QCDCL proof systems for which this restriction is not necessary. However, for the classic QCDCL variant QCDCL, decisions always need to be level-ordered.

Because of $x \in \mathcal{T}[i,j]$ ($\bar{x} \in \mathcal{T}[i,j]$), this gives a contradiction since $\text{ante}_\mathcal{T}(p_{(k,m)})$ must not become true (false) before propagating $p_{(k,m)}$. □

Proposition 4.8. *Let Φ be a QCNF and let*

$$\mathcal{T} = (p_{(0,1)}, \dots, p_{(0,g_0)}; \mathbf{d_1}, p_{(1,1)}, \dots, p_{(1,g_1)}; \dots; \mathbf{d_r}, p_{(r,1)}, \dots, p_{(r,g_r)}).$$

be a trail for Φ. Then the corresponding derivation of any learned clause is a valid LD-Q-Res *or* LD-Q-Con *derivation.*

Proof. Let

$$\mathcal{L}_\mathcal{T} := (C_{(r,g_r)}, \dots, C_{(r,1)}, \dots, C_{(1,g_1)}, \dots, C_{(1,1)}, C_{(0,g_0)}, \dots, C_{(0,1)})$$

be the sequence of learnable constraints. Because of Lemma 4.7 it remains to show that the derivation of clauses (cubes) with universal (existential) tautologies are sound.

Assume otherwise. That means there are $i \in \{0, \dots, r\}$, $j \in \{1, \dots, g_i\}$ and some universal (existential) $u \in \text{var}(\Phi)$ such that $\text{lv}(u) < \text{lv}(p_{(i,j)})$ and one of the following four cases holds, where we resolve over $p_{(i,j)}$:

1. $u \in C_{(i,j+1)}$ and $\bar{u} \in \text{ante}_\mathcal{T}(p_{(i,j)})$,
2. $u \vee \bar{u} \in C_{(i,j+1)}$ and $\bar{u} \in \text{ante}_\mathcal{T}(p_{(i,j)})$,
3. $u \in C_{(i,j+1)}$ and $u \vee \bar{u} \in \text{ante}_\mathcal{T}(p_{(i,j)})$,
4. $u \vee \bar{u} \in C_{(i,j+1)}$ and $u \vee \bar{u} \in \text{ante}_\mathcal{T}(p_{(i,j)})$.

Consider case 1. Since $u \in C_{(i,j+1)}$ there has to be a propagated literal $p_{(k,m)}$ right of $p_{(i,j)}$ in the trail such that $u \in \text{ante}(p_{(k,m)})$. In order to become unit, the u in $\text{ante}(p_{(k,m)})$ needed to vanish. We distinguish two cases:

Case (i). $\bar{u}$ (resp. u) was decided before $p_{(k,m)}$ was propagated, i.e., $\bar{u} \in \mathcal{T}[k,m]$ ($u \in \mathcal{T}[k,m]$).

Then we have $u \notin \mathcal{T}$ ($\bar{u} \notin \mathcal{T}$) since each variable can occur at most once in $\mathcal{T}$. That means reducing $\bar{u}$ (resp. u) is the only way $\text{ante}_\mathcal{T}(p_{(i,j)})$ could have become unit. But for the soundness of reduction we need $\text{lv}(u) > \text{lv}(p_{(i,j)})$. This gives a contradiction.

Case (ii). $u \in \text{ante}_\mathcal{T}(p_{(k,m)})$ has vanished via reduction.

Assume that u (resp. $\bar{u}$) was decided before $p_{(i,j)}$ was propagated, i.e., $u \in \mathcal{T}[i,j]$. But then $\text{ante}(p_{(k,m)})$ would have become true under $\mathcal{T}[k,m] \supseteq \mathcal{T}[i,j]$. Therefore $\text{ante}_\mathcal{T}(p_{(k,m)})$ could not have been used for unit propagation. Thus $\bar{u} \in \text{ante}_\mathcal{T}(p_{(i,j)})$ must have vanished via reduction, which implies $\text{lv}(u) > \text{lv}(p_{(i,j)})$, a contradiction.

The same reasoning works for case 2. Cases 3 and 4 are easier since the only way for $u \vee \bar{u}$ to vanish in $\text{ante}_\mathcal{T}(p_{(i,j)})$ is via reduction. Then we get the contradiction $\text{lv}(u) > \text{lv}(p_{(i,j)})$ as well.

Also the same argumentation works if we consider $C_{(i+1,1)}$ and $\text{ante}_\mathcal{T}(p_{(i,g_i)})$ instead of $C_{(i,j+1)}$ and $\text{ante}_\mathcal{T}(p_{(i,j)})$. □

Note that the above proof does not use the fact that decisions are level-ordered. That means that even with arbitrary decisions, generated derivations are always valid. In fact, we have proven that all trail-based (resp. QCDCL-like) proof systems are sound.

We combine the two results above to an argument of soundness of the defined QCDCL proof system.

Theorem 4.9. QCDCL *is p-simulated by* LD-Q-Res *(resp.* LD-Q-Con *for true formulas). In fact, the derivations of all learned constraints are valid* LD-Q-Res *(*LD-Q-Con*) proofs.*

Hence, for each QCDCL *proof ι from a QBF Φ of a clause (cube) C, the extracted proof $\mathfrak{R}(\iota)$ is a valid* LD-Q-Res *(*LD-Q-Con*) proof from Φ of C.*

In particular, QCDCL *and* QCDCL$^{w/o\ \textsc{Cubes}}$ *are sound.*

An important concept in QBF solving is searching for learnable *asserting* constraints. Learning such asserting constraints is desired in QBF (and SAT) solving since it yields some kind of progression by enabling new unit propagations.

Definition 4.10. *Let $\mathcal{T}$ be some trail in ι. A clause (cube) $C \in \mathfrak{L}(\mathcal{T})$ is called* asserting *(with respect to $\mathcal{T}$) if there exists a point $[i, j]$ such that i is not the last decision level in $\mathcal{T}$ and red$(C|_{\mathcal{T}[i,j]})$ is a unit clause (cube).*

In other words, learnable clauses (cubes) are called asserting whenever it is possible to backtrack back to a point in the trail where this newly learned constraint would become unit. However, the theoretical benefit of asserting constraints is yet to be considered. So far, by learning asserting constraints we are guaranteed to learn new constraints in each turn, which is the main argument for proving completeness of QCDCL. At first, we will show that asserting constraints are always new.

Lemma 4.11. *Let $\iota := [(\mathcal{T}_i, C_i, \pi_i)]_{i=1}^m$ be a* QCDCL *proof from a QCNF $\Phi = \mathcal{Q} \cdot \varphi$. Let $j \in \{1, \dots m\}$ and C_j be an asserting clause (cube) with respect to $\mathcal{T}_j$. Then $C_j \notin \mathfrak{C}(\Phi_j)$ (resp. $C_j \notin \mathfrak{D}(\Phi_j)$).*

Proof. W.l.o.g. let C_j be an asserting clause (the case for asserting cubes is similar). Let $\mathcal{T}_j$ be the trail

$$\mathcal{T}_j = (p_{(0,1)}, \dots, p_{(0,g_0)}; \mathbf{d_1}, p_{(1,1)}, \dots, p_{(1,g_1)}; \dots; \mathbf{d_r}, p_{(r,1)}, \dots, p_{(r,g_r)} = \bot).$$

Since we learned an asserting clause, we obviously have $r > 0$ by definition. Assume that $C_j \in \mathfrak{C}(\Phi)$. Then there is a point such that $C_j|_{\mathcal{T}_j[s,t]}$ is a unit clause with an existential literal ℓ. Because of Definition 4.3 we are forced to propagate ℓ (or run into a conflict) not later than level s. Therefore $\ell \in \mathcal{T}_j[s+1, 0]$. However, this is a contradiction because we need $\ell \in C_j$, which is only possible if $\bar{\ell} \in \mathcal{T}_j$ by definition of Clause Learning. □

In practice, there are different approaches in finding such an asserting learnable constraint, which we will refer to as *asserting learning schemes*. A widely used asserting learning scheme is the so-called *UIP* (*unit implication point*) learning scheme (cf. [104]

for more details). However, in our theoretical setting, it suffices to work with an arbitrary, fixed asserting learning scheme. Formally, an asserting learning scheme can be interpreted as a function $\mathfrak{A}$ that maps a QCNF Φ and a trail $\mathcal{T}$ from some QCDCL proof to a constraint $\mathfrak{A}(\mathcal{T}, \Phi) \in \mathfrak{L}(\mathcal{T})$ that is asserting whenever it is non-empty (we can safely assume that such a scheme always choses an empty constraint if possible).

Since obviously not every learnable constraint is asserting in general, the definition of asserting learning schemes requires the existence of at least one asserting learnable constraint for each trail in a QCDCL proof.

Proposition 4.12. *Let ι be a* QCDCL *proof and let $\mathcal{T}$ be some trail in ι that has run into a conflict. If $(\bot) \notin \mathfrak{L}(\mathcal{T})$ and $[\top] \notin \mathfrak{L}(\mathcal{T})$, then $\mathfrak{L}(\mathcal{T})$ contains an asserting clause or cube[2].*

Proof. W.l.o.g. let the conflict be existential, which means $(\bot) \in \mathcal{T}$. Let $\mathcal{T}$ be of the form

$$\mathcal{T} = (p_{(0,1)}, \dots, p_{(0,g_0)}; \mathbf{d_1}, p_{(1,1)}, \dots, p_{(1,g_1)}; \dots; \mathbf{d_r}, p_{(r,1)}, \dots, p_{(r,g_r)} = \bot).$$

Universal literals that are propagated via cubes will be treated as decisions. That means that universal decisions might be out of order, while existential decisions are still level-ordered (i.e., all universal variables that are quantified left of an existential decision literal d_k must be assigned before d_k in $\mathcal{T}$).

Consider the sequence of learnable clauses

$$\mathcal{L}_\mathcal{T} := (C_{(r,g_r)}, \dots, C_{(r,1)}, \dots, C_{(1,g_1)}, \dots, C_{(1,1)}, C_{(0,g_0)}, \dots, C_{(0,1)}).$$

If $r = 0$ or if all decision literals are universal, we can just take the rightmost clause in $\mathcal{L}_\mathcal{T}$, which is $(\bot)$ and therefore a contradiction. Hence we can assume that $r > 0$ and there exists at least one existential decision literal.

Let $k \in \{1, \dots, r\}$ be maximal such that $\bar{d}_k$ is existential and contained in some clause from $\mathfrak{L}_\mathcal{T}$. Such a literal must exist since otherwise we could resolve over all propagation literals $p_{(i,j)}$ and reduce all universal literals during the learning process. Then we would be able to learn $(\bot)$. Intuitively, d_k is the last existential decision that contributed to the conflict. Let $p_{(s,t)}$ be the next propagated literal right of d_k in $\mathcal{T}$ (this does not necessarily have to be $p_{(k,1)}$). Set $D := C_{(s,t)}$.

We claim that this clause D is asserting with respect to $\mathcal{T}$ with the point $[k, 0]$. It is easy to see that for all existential literals $y \in D \backslash \{\bar{d}_k\}$ we need $\bar{y} \in \mathcal{T}[k, 0]$, because all the other existential literals right of d_k in the trail either got resolved away during Clause Learning, or the corresponding resolution step was skipped due to the pivot missing in the learned clause. In either case, only literals that are assigned prior to d_k can occur as existential literals in D.

All universal variables u with $\mathrm{lv}(u) < \mathrm{lv}(d_k)$ are assigned earlier than d_k in $\mathcal{T}$ because the existential decisions are level-ordered. Now consider $E := D|_{\mathcal{T}[k,0]}$. The only type of literals that can occur in E, aside from $\bar{d}_k$, are universal literals u with $\mathrm{lv}(u) > \mathrm{lv}(d_k)$.

[2]This proposition only holds if decisions are indeed level-ordered.

But then we can conclude $\mathrm{red}(E) = (\bar{d}_k)$ and therefore E is a unit clause and D is asserting. □

Note that we can only guarantee learnable asserting constraints on trails that run into a conflict. Technically, however, after learning a cube that represents a satisfying assignment, we could always immediately trigger a conflict on this cube by not doing any backtracking (i.e., backtracking zero steps, which is allowed on trails without conflicts). From this conflict, we can then learn again an asserting cube.

We combine all results above and prove the completeness of QCDCL and its restricted variant $\mathsf{QCDCL}^{\mathsf{w/o\ Cubes}}$.

Theorem 4.13. *The proof system* QCDCL *is complete on both false and true QCNFs. The variant* $\mathsf{QCDCL}^{\mathsf{w/o\ Cubes}}$ *is at least complete on false QCNFs.*

Proof. Let $\Phi := \mathcal{Q} \cdot \varphi$ be a QCNF. We distinguish two cases.

Case 1. Φ is a false formula.

It suffices to consider $\mathsf{QCDCL}^{\mathsf{w/o\ Cubes}}$ as we only need to learn clauses. Assume that we already started to construct a $\mathsf{QCDCL}^{\mathsf{w/o\ Cubes}}$ proof ι for Φ (note that ι might be empty) and already learned the clauses $C_1, \ldots, C_j$. We show that we can add a trail $\mathcal{T}$ to ι, from which we can learn a new clause C (i.e. $C \notin$ and C was not learned in ι).

Let $\Psi := \mathcal{Q} \cdot \varphi \wedge C_1 \wedge \ldots \wedge C_j$. Independently from the last trail in ι (which might not exist since ι can be empty), we do a restart and construct a natural trail $\mathcal{T}$ for Ψ as follows:

Fix some universal winning strategy for Ψ. Such a strategy must exist since Ψ is false. Whenever a universal decision has to be made in $\mathcal{T}$, the corresponding decision literal has to be assigned according to the strategy. Existential decisions can be done arbitrarily. Because we can only propagate existential literals, we will never assign a universal literal the wrong way (with respect to the strategy). Since all decisions are level-ordered, all literals on which any universal decision literal depends are already assigned.

At some point we will run into a conflict because the strategy falsifies the formula. If we cannot learn the empty clause (which would end the proof), we can at least learn an asserting clause by Proposition 4.12. By Lemma 4.11, such a clause is not contained in $\mathfrak{C}(\Psi)$.

Therefore, we can always add new trails to ι from which we can learn new clauses or the empty clause. Since there are only finitely many clauses over the variables of Φ, we will learn the empty clause at some point.

Case 2. Φ is a true formula.

Analogously to Case 1, we construct trails from which we can always learn new constraints. In contrast to Case 1, it does not matter how the decision literals are assigned. Whenever we run into a conflict (both cube and clause conflicts), we can simply learn an asserting constraint by Proposition 4.12 as long as we cannot learn an empty constraint. If we do not run into a conflict, we simply learn a cube that represents a satisfying assignment as described in Definition 4.4. This cube must be new as well, otherwise we would have gotten a cube conflict on it.

Since there exist only finitely many clauses and cubes over variables from Φ, we will learn an empty constraint at some point (more precisely, we will learn the empty cube because Φ is true). $\square$

Chapter 5

Gauge Lower Bound

In this chapter we will devise the first dedicated lower bound technique for QCDCL. In contrast to previous lower bounds for QCDCL, our technique does not import Q-Resolution hardness and thus applies to different formulas, regardless of whether they are hard for Q-Resolution or not (note that basically all QCDCL lower bounds were imported from long-distance Q-Resolution and hence are lower bounds also for Q-Res). We already mention at this point though, that our technique is not completely general, but is restricted to Σ_3^b-formulas that meet a certain XT-condition.

We define a property of LD-Q-Res proofs, which we call *quasi level-ordered*. This is inspired by the notion of level-ordered proofs, introduced in [68], where the order of resolution steps in proofs must follow the quantification order in the prefix. Quasi level-order proofs relax that condition (Definition 5.5).We then introduce another property of Q-Resolution proofs, called *primitivity*. As we will show in Chapter 6, some QCDCL variants generate primitive proofs under some circumstances, which can then be transformed into quasi level-ordered proofs of the same size (Theorem 5.10).

Our lower bound technique then rests on two steps: (1) We show that for Σ_3^b-formulas with the XT-condition, QCDCL generates primitive proofs (cf. Chapter 6) which can be translated into quasi level-ordered Q-Resolution proofs. (2) We define a new measure called the *gauge* of a QBF and show that large (i.e. linear) gauge implies exponential size in quasi level-ordered Q-resolution. Together, (1) and (2) imply that formulas with the XT-property and large gauge are hard for QCDCL (our main Theorem 5.17 as well as Corollary 6.5).

Our technique can be interpreted as a generalization of the QCDCL lower bound of the QBF $\mathtt{CR}_n$ shown in [64]. Proving this lower bound consisted of two steps: Showing that QCDCL generates level-ordered proofs on $\mathtt{CR}_n$, and then showing the hardness of $\mathtt{CR}_n$ in level-ordered Q-Resolution. We demonstrate that the requirements for this results can be weakened and therefore applied to a wider class of formulas.

We illustrate our technique on a couple of examples on which computing the gauge is fairly straightforward. Thus, though showing (1) and (2) above is rather technical, the lower bound technique itself is quite easily applicable.

It is also interesting to mention that our new notion of gauge is some kind of width

measure on clauses. Showing proof size lower bounds via width lower bounds is a very well-explored theme in proof complexity, both propositionally [8] and in QBF [12, 22]. We show, however, that gauge and proof width are not related in general.

We note that in this chapter we will not apply this technique to QCDCL just yet. All lower bounds proven in this chapter are for solely theoretical proof systems so far.

5.1 Quasi Level-ordered Proofs and Primitivity

We recall the notions of a path in a proof and level-ordered proofs. The exponential lower bound for $\mathtt{CR}_n$ relies on the fact that QCDCL generates level-ordered proofs on this formula. Obviously, extracted proofs from QCDCL are not always level-ordered in general. We want to generalize this notion later while preserving the properties we need to show lower bounds on even more formulas.

Definition 5.1. *Let $\pi = (C_1, \dots, C_m)$ be a* LD-Q-Res *proof. A* path *$P = (C_{i_1}, \dots, C_{i_k})$ is a subsequence of π such that each C_{i_j} is a parent clause of $C_{i_{j+1}}$ in π. We say that P is a* path from C_{i_1} to C_{i_k}.

Note that a proof can contain the same constraint more than once. We will therefore always identify the position of a clause in the proof by the index. Hence, by considering a clause C_i in a proof, we will only consider the clause at position i, even if there were other clauses C_j with $i \neq j$ and $C_i = C_j$.

Definition 5.2 ([68])**.** *An* LD-Q-Res *proof π from a QCNF Φ of a clause C is called* level-ordered *if for each path P in π and two resolution steps in P over variables ℓ_1 and ℓ_2 the following holds: if the resolution over ℓ_1 is closer to the root C than the resolution over ℓ_2, then $lv(\ell_1) \leq lv(\ell_2)$.*

While the above definitions can be applied to arbitrary formulas, we will restrict our results to Σ_3^b QCNFs with the prefix $\exists X \forall U \exists T$. The notion of a level-ordered proof then simplifies to the property that there can be no resolution step over T after a resolution step over X.

Before we define quasi level-ordered proofs, we will introduce some notions that make it easier to identify different kinds of clauses of a Σ_3^b formula. We also introduce the XT-property, which is a necessary condition for formulas to which the lower bound technique can be applied.

Definition 5.3. *Let Φ be a Σ_3^b QCNF with the prefix $\exists X \forall U \exists T$, for sets of variables X, U, T. A variable from X (resp. U or T) is called an X-variable (resp. U- or T-variable). Analogously, we call a literal an X-literal (resp. U- or T-literal) if the corresponding variable is an X-variable (resp. U- or T-variable). Further, we define certain types of clauses:*

- *X-clauses consist of X-literals only (analogously we define U-clauses and T-clauses; the empty clause is considered an X-clause and no U- or T-clause),*

- *XT-clauses consist of at least one X- and at least one T-literal, but no U-literals,*
- *XUT-clauses consist of at least one X-, U- and T-literal.*

We say that Φ fulfils the XT-property *if its matrix contains no XT-clauses as well as no unit T-clauses and there do not exist two T-clauses in the matrix that are resolvable.*

Intuitively, the XT-property ensures that trails will be somewhat "level-ordered" (i.e., in each trail we first assign X-literals, then U-literals, and lastly T-literals), which was a necessity for proving the lower bound of $\mathtt{CR}_n$ in [64]. In detail, it prevents direct propagations of T-literals without using assignments of X- or U-literals. This restriction might seen rather strict at first, but one could motivate it as follows: Each Σ_3^b QCNF encodes a two-player game, in which the existential player controls the X- and T-variables while the universal player controls the U-variables. However, since there is no universal block right of T, the T-variables are representing the winning conditions of the game rather than allowing an "interesting" move of the existential player. Forbidding XT-clauses or unit T-clauses just implies that the winning conditions should not contain "redundancies" or "unfairness".

We will show next that the XT-property is hereditary with respect to derived or learned clauses.

Lemma 5.4 ([14])**.** *Let Φ be a QCNF that fulfils the XT-property. Then the following holds:*

(i) It is not possible to derive XT-clauses by LD-Q-Res.

(ii) It is not possible to derive new T-clauses (i.e., T-clauses not contained in $\mathfrak{C}(\Phi)$) by LD-Q-Res.

(iii) If a non-axiom X-clause was not derived by an X-resolution, then it was derived by a universal reduction.

In particular, the XT-property is maintained even after adding clauses derived by LD-Q-Res.

Proof. (i) Assume that it is possible to derive an XT-clause. Let π be a LD-Q-Res proof of an XT-clause C from Φ. W.l.o.g. let C be the first XT-clause in π (otherwise we consider the subproof of the first XT-clause). Since the matrix of Φ contains no XT-clauses, C has to be derived by a resolution or reduction step. Because C consists of at least one T-literal, it is not derived by a reduction. That means C is derived via some resolution step over an X-literal x or T-literal t. In the first case, say $C = D \overset{x}{\bowtie} E$ for $D, E \in \pi$, neither D nor E can contain U-literals. But both D and E must contain an X-literal and at least one of them needs to contain a T-literal, meaning that one of them has to be an XT-clause. This contradicts our assumption that C is the first XT-clause in π. The case in which the resolution step is performed over a T-literal can be handled analogously.

(ii) Assume that there is an LD-Q-Res proof π from Φ of a new T-clause C. W.l.o.g. let C be the first new T-clause in π. Because there is at least one T-literal in C (the empty clause is no T-clause by definition), C cannot be derived by a reduction step. If C was derived by a resolution step over X, say $C = D \overset{x}{\bowtie} E$, then both D and E must contain an X-literal, none can contain U-literals and at least one has to contain a T-literal. We conclude that one of these two clause has to be an XT-clause, which is not possible by (i).

That means C was derived by a T-resolution step, say $C = D \overset{t}{\bowtie} E$. Then neither D nor E can contain X- or U-literals, and both of them need to consist of at least one T-literal. Hence, both D and E are T-clauses. Since C is the first new T-clause in π, both D and E must be axioms of Φ. But by definition of the XT-property, axiom T-clauses are not resolvable, which is a contradiction.

(iii) It suffices to show that such an X-clause cannot be derived by a T-resolution. For the sake of contradiction, let C be an X-clause derived by a T-resolution, say $C = D \overset{t}{\bowtie} E$. Then neither D nor E can contain U-literals and both of them have to consist of at least one T-literal. Because of (i), neither of them can be an XT-clause. We conclude that both of them are T-clauses, which contradicts the XT-property or (ii).

□

We can now finally introduce the notion of quasi level-ordered proofs.

Definition 5.5. *A* LD-Q-Res *refutation π of a Σ_3^b formula with prefix $\exists X \forall U \exists T$ is called* quasi level-ordered*, if for each assignment $\tau \in \langle X \rangle$ there exists an X-clause $C_\tau \in \pi$ which is falsified by τ and the subproof $\pi_{C_\tau} \subseteq \pi$ of C_τ is level-ordered.*

One can easily show that each level-ordered proof is also quasi level-ordered (by setting each C_τ to the empty clause). However, we cannot guarantee that QCDCL always generates quasi level-ordered proofs (even on our restricted class of formulas).

Hence, we introduce two further notions: fully reduced and primitive proofs. While QCDCL always generates fully reduced proofs by design, primitivity depends on the internal structure of a formula, but can still be proven on many hand-crafted formulas.

Definition 5.6 (fully reduced proofs [33, 38])**.** *A* LD-Q-Res *refutation π of a QCNF Φ is* fully reduced*, if for each clause $C \in \pi$ that contains universal literals that are reducible, the reduction step is performed immediately and C is not used otherwise in the proof.*

Definition 5.7 (primitive proofs [33, 38])**.** *A* LD-Q-Res *proof π from a Σ_3^b formula is* primitive*, if there are no two XUT-clauses in π that are resolved over an X-variable.*

Note that fully reduced primitive LD-Q-Res proofs cannot derive merged literals and therefore never perform any long-distance steps. Hence, from now on we will only consider fully reduced primitive Q-Res proofs.

Before moving on, let us briefly explain the idea of our lower bound technique. This technique will be applicable to all Σ_3^b QCNFs that fulfil the XT-property and

on which QCDCL generates primitive proofs. However, the lower bound itself does not originate from primitivity directly, but from quasi level-ordered proofs (specifically, from the existence of the C_τ clauses, cf. Definition 5.5).

What remains to show is that fully reduced primitive proofs somewhat "imply" quasi level-ordered proofs. We will do this by proving that each fully reduced primitive refutation can be transformed into a quasi level-ordered proof of the same size. Note that this transformation does not need to be efficient as long as we can guarantee that there will be now blow up in the proof size.

The transformation will be performed as follows: We will first detect all X-clauses in a given fully reduced primitive Q-Res refutation. We then check whether or not the derivations of these X-clauses are level-ordered. If they are not, we find "minimal witnesses" for the parts that are not level-ordered. We call these witnesses *disorders* and will gradually get rid of them during the transformation until no disorders are left.

Definition 5.8. *Let $\pi = C_1, \ldots, C_m$ be a fully reduced primitive* Q-Res *proof of a Σ_3^b QCNF Φ that fulfils the XT-property and let $C_c \in \pi$ be a clause that was derived by a universal reduction. A* disorder *of C_c is a path $(C_{a_1}, \ldots, C_{a_k}, C_c)$ with $k \geq 2$ such that C_{a_1} was derived by an X-resolution between an X- and an XUT-clause, say $C_{a_1} = C_d \overset{x}{\bowtie} C_e$ for some X-literal x and clauses $C_d, C_e \in \pi$, and C_{a_j} was derived by a T-resolution, say $C_{a_j} = C_{a_{j-1}} \overset{r_j}{\bowtie} C_{b_{j-1}}$ for some clause $C_{b_{j-1}}$ and $j = 2, \ldots, k$. Obviously, we have $red(C_{a_k}) = C_c$.*

In the next lemma, we show that disorders are actual witnesses of the parts of the proof that are not level-ordered, meaning that there can only be disorders if the derivation of at least one X-clause is not level-ordered.

Lemma 5.9. *In the situation above, if $(C_{a_1}, \ldots, C_{a_k}, C_c)$ is a disorder, then the subproof π_{C_c} of C_c in π is not level-ordered. More precisely, if π contains no disorders, then the subproofs of all X-clauses from π are level-ordered.*

Proof. The first statement is obvious and follows directly by definition.

For the second statement, let us fix some X-clause $C_c \in \pi$. Because of the XT-property, C_c cannot be derived by a T-resolution, otherwise one of the parent clauses would be an XT-clause. We distinguish the two remaining cases:

Case 1. C_c was derived by a universal reduction.

Assume that the subproof of C_c in π is not level-ordered. Let C_a be the rightmost clause in this subproof that was derived by an X-resolution. Such a clause must exist, otherwise the subproof would be trivially level-ordered. The clause left of C_c was derived by a T-resolution, otherwise the reduction could have been performed earlier, which contradicts that π is fully reduced. We conclude that C_c must have been the leftmost clause right of C_a that was derived by a reduction.

therefore all clauses between C_a and C_c were derived via T-resolutions. Since π cannot contain XT-clauses and no two XUT-clauses can be resolved over an X-literal, one of the parent clauses of C_a must be an X- and the other one an XUT-clause. Hence $(C_a, \ldots, C_c)$ is a disorder, which is a contradiction.

Case 2. C_c was derived by an X-resolution.
If the subproof of C_c in π was not level-ordered, then we could find a path

$$(C_{j_1}, \ldots, C_{j_q}, C_c)$$

in π such that some C_{j_i} was derived by an X-resolution and $C_{j_{i+1}}$ was derived by a T-resolution. W.l.o.g. let i be maximal. Let C_{j_s} be the leftmost clause right of $C_{j_{i+1}}$ that was derived by a reduction. Such a clause must exist since C_{j_i} and $C_{j_{i+1}}$ must contain a U-literal (otherwise one of the parent clauses of C_{i_j} would be an XT-clause). Because π is fully reduced primitive and i was maximal, one of the parent clauses of C_{j_i} is an X- and the other an XUT-clause, and all clauses between $C_{j_{i+1}}$ and C_{j_s} were derived via T-resolutions, and therefore the path $(C_{j_i}, C_{j_{i+1}}, \ldots, C_{j_s})$ is a disorder of C_{j_s}, which is a contradiction. □

Note that Lemma 5.9 was the only result for which primitivity was needed. In detail, primitivity reduces the property of not being level-ordered to the existence of a disorder. The following theorem demonstrates how fully reduced primitive refutations can be transformed into quasi level-ordered proofs by "fixing" each disorder one after another without increasing the proof size. The main approach of the transformation is partly based on the algorithm presented in [33].

Theorem 5.10. *Let Φ be a Σ_3^b QCNF that fulfils the XT-property. Then, for each fully reduced primitive* Q-Res *refutation π there exists a quasli level-ordered* Q-Res *proof π' from Φ with $|\pi'| \leq |\pi|$.*

Proof. We sketch the proof first. We will construct a sequence of proofs

$$(\pi =: \pi_0, \pi_1, \ldots, \pi_h)$$

with $|\pi_i| = |\pi|$ for each $i = 0, \ldots, h$. We will show that for each $\tau \in \langle X \rangle$ there exists a clause $D_\tau \in \pi_h =: \pi'$ with $\tau(D_\tau) = 0$. Furthermore, π_h will contain no disorders, meaning that the subproofs of all D_τ will be level-ordered by Lemma 5.9.

Note that the length h of this sequence might be exponential, hence there is no guarantee that π' can be constructed in polynomial time. However, the size of π' will still be bounded by the original input size, which is sufficient for the gauge lower bound.

At first, we will determine some notations. For each $i = 0, \ldots, h$, the j^{th} clause of π_j will be denoted as C_j^i and the number of paths in π_i will be denoted as p_i. We define the following two sets:

$$\Lambda_i := \{c \in \{1, \ldots, |\pi|\} : C_c^i \text{ is an X-clause}\},$$
$$\Delta_c^i := \{P : P \text{ is a disorder of } C_c^i \text{ in } \pi_i\} \text{ for } c \in \Lambda_i$$

Formally, at the end we want $\Delta_c^h = \emptyset$ for all $c \in \Lambda_h$ and for all $\tau \in \langle X \rangle$ there has to exist some $c_\tau \in \Lambda_h$ with $\tau(C_{c_\tau}^h) = 0$.

At the beginning, fix a lexicographic ordering on the set of all paths in π (each path can be interpreted as a tuple of natural numbers). This ordering can be extended to each

π_i. We will later exploit the property that for each two paths we have $(C^i_{j_1}, \ldots, C^i_{j_q}) < (C^i_{m_1}, \ldots, C^i_{m_s})$ whenever $j_1 < m_1$.

We will now inductively construct the proofs π_i for all $i = 1, \ldots, h$. Therefore, let us assume that we have already constructed π_i.

Let $c := \arg\max_{a \in \Lambda_i} \max \Delta^i_a$ and $P := (C^i_{a_1}, \ldots, C^i_{a_k}, C^i_c) := \max_{a \in \Lambda_i} \max \Delta^i_a$. Intuitively, P is the lexicographically last disorder in π_i and C^i_c is the corresponding clause of which P is a disorder. More precisely, we have $C^i_{a_1} = C^i_d \overset{x}{\bowtie} C^i_e$ with an XUT-clause C^i_d, X-clause C^i_e and an X-literal x such that $x \in C^i_d$ and $\bar{x} \in C^i_e$. Further, there are clauses $C^i_{b_j} \in \pi_i$ and T-literals r_j such that $C^i_{a_{j+1}} = C^i_{a_j} \overset{r_j}{\bowtie} C^i_{b_j}$ for $j = 1, \ldots, k-1$ and at the end $C^i_c = \mathrm{red}(C^i_{a_k})$.

We now construct π_{i+1} by setting $C^{i+1}_j := C^i_j$ for all $j < a_2$. Then we set $C^{i+1}_{a_2} := C^{i+1}_d \overset{r_1}{\bowtie} C^{i+1}_{b_1}$, meaning that we skip the resolution with the X-clause C^{i+1}_e in the derivation of $C^{i+1}_{a_2}$. All clauses C^{i+1}_j with $j > a_2$ are derived in the same way as C^i_j in π_i but might still be affected by the modification of the proof. In fact, we have $C^{i+1}_j \subseteq C^i_j \vee x$ for all $j > a_2$. However, we will show that we will not hit on problems by having this extra literal x on some clauses after $C^{i+1}_{a_2}$.

Note that it might also happen that several literals will be missing on clauses after a_2. If, because of that, some resolution steps become impossible, we will simply skip the resolution and copy the parent clause with the missing pivot to the position of the resolvent. For example, assume π_i includes the resolution step $C^i_z = C^i_v \overset{\ell}{\bowtie} C^i_w$ with some literal ℓ such that $\ell \in C^i_v$ and $\bar{\ell} \in C^i_w$. If the literal ℓ is now missing in C^{i+1}_v, then we simply set $C^{i+1}_z := C^{i+1}_v$.

The purpose of this transformation is to get rid of the disorder P. In fact, P is now no proper path in π_{i+1} anymore. We want to show that the number of paths is actually decreasing with this transformation (i.e., $p_{i+1} < p_i$). For any path

$$P' = (C^{i+1}_{m_1}, \ldots, C^{i+1}_{m_s})$$

in π_{i+1} that starts with $C^{i+1}_{a_2}$, or does not contain $C^{i+1}_{a_2}$ at all, the path

$$P'' = (C^i_{m_1}, \ldots, C^i_{m_s})$$

is a path in π_i. If a path P' in π_{i+1} contains but does not start with a_2, then we distinguish two cases (depending on the choice of the parent clause of $C^{i+1}_{a_2}$ that is contained in P'):

$$\begin{aligned} P' =& (C^{i+1}_{j_1}, \ldots, C^{i+1}_{j_q}, C^{i+1}_d, C^{i+1}_{a_2}, C^{i+1}_{m_1}, \ldots, C^{i+1}_{m_s}) \text{ or} \\ P' =& (C^{i+1}_{j_1}, \ldots, C^{i+1}_{j_q}, C^{i+1}_{b_1}, C^{i+1}_{a_2}, C^{i+1}_{m_1}, \ldots, C^{i+1}_{m_s}) \end{aligned}$$

In the first case, the path

$$P'' := (C^i_{j_1}, \ldots, C^i_{j_q}, C^i_d, C^i_{a_1}, C^i_{a_2}, C^i_{m_1}, \ldots, C^i_{m_s})$$

is a path in π_i. In the second case, the path

$$P'' := (C^i_{j_1}, \ldots, C^i_{j_q}, C^i_{b_1}, C^i_{a_2}, C^i_{m_1}, \ldots, C^i_{m_s})$$

is also a path in π_i.

Therefore we can define an injective map from the set of all paths in π_{i+1} to the set of all paths in π_i such that P has no preimage. We conclude that $p_{i+1} < p_i$. In particular, we have shown that $h \leq p_0 < \infty$, therefore this transformation will stop after finitely many iterations. In detail, the transformation stops as soon as $\bigcup_{c \in \Lambda_i} \Delta^i_c = \emptyset$.

Next, we will show that, in the transformation from π_i to π_{i+1} above, the additional x will not cause any problems in future transformations. First of all, x is an X-literal and therefore contained in the first quantifier block. In particular, adding x to some clauses will never prevent any universal reductions. Although adding x might cause some existential tautologies in π_{i+1}, the proof is technically still sound.

It remains to show that π' is in fact quasi level-ordered. For that, we need to fix a total assignment $\tau \in \langle X \rangle$ and find a clause $D_\tau \in \pi'$ such that its subproof in π' is level-ordered and $\tau(D_\tau) = 0$. In particular, D_τ has to be an X-clause. We detect such a D_τ by constructing a sequence of clauses $(C^{i_1}_{c_1}, \ldots, C^{i_g}_{c_g})$ with $c_j \in \{1, \ldots, |\pi|\}$, $i_j \in \{0, \ldots, h\}$ and $i_g = h$ such that $C^{i_j}_{c_j} \in \Lambda_{i_j}$ and $\tau(C^{i_j}_{c_j}) = 0$ for each $j = 1, \ldots, g$. Since the last proof $\pi_h = \pi'$ contains no disorders, the subproof of $D_\tau := C^{i_g}_{c_g}$ is level-ordered by Lemma 5.9.

We start with $i_1 := 0$ and $c_1 := |\pi|$, hence $C^{i_1}_{c_1} = (\bot) \in \pi_0 = \pi$. Assume that we have already detected $C^{i_j}_{c_j}$. We will now explain how to detect $C^{i_{j+1}}_{c_{j+1}}$.

If the subproof of $C^{i_j}_{c_j}$ is already level-ordered, then we can simply set $c_{j+1} := c_j$ and $i_{j+1} := i_j + 1$ since the transformation from π_{i_j} to π_{i_j+1} will not affect any level-ordered subproofs. Hence, $C^{i_j+1}_{c_{j+1}} = C^{i_j}_{c_j}$ and its subproof is still level-ordered in π_{i_j+1}.

Let us now assume that the subproof of $C^{i_j}_{c_j}$ is not level-ordered in π_{i_j}. Because $C^{i_j}_{c_j}$ is a non-axiom X-clause, it is either derived by an X-resolution or a universal reduction by Lemma 5.4. We distinguish these two cases:

Case 1. $C^{i_j}_{c_j}$ was derived by an X-resolution.

Let $C^{i_j}_{d_j}$ and $C^{i_j}_{e_j}$ be the two parent clauses and x the pivot, which means that $C^{i_j}_{c_j} = C^{i_j}_{d_j} \overset{x}{\bowtie} C^{i_j}_{e_j}$ with $x \in C^{i_j}_{d_j}$ and $\bar{x} \in C^{i_j}_{e_j}$. If $\tau(x) = 0$, we set $c_{j+1} := d_{j+1}$ and $i_{j+1} = i_j$, otherwise we set $c_{j+1} := e_{j+1}$ and $i_{j+1} = i_j$. Then we get $\tau(C^{i_j+1}_{c_{j+1}}) = 0$.

Case 2. $C^{i_j}_{c_j}$ was derived by a universal reduction.

If $C^{i_j+1}_{c_j} \subseteq C^{i_j}_{c_j}$, we can set $c_{j+1} := c_j$ and $i_{j+1} := i_j + 1$. So let us assume that $C^{i_j+1}_{c_j} \not\subseteq C^{i_j}_{c_j}$. This only happens when the transformation removes some disorder from π_{i_j}. Let $c := \arg\max_{a \in \Lambda_{i_j}} \max \Delta^{i_j}_a$ and $P_1 := (C^{i_j}_{a_1}, \ldots, C^{i_j}_{a_k}, C^{i_j}_c) := \max_{a \in \Lambda_{i_j}} \max \Delta^{i_j}_a$. Then there is a path P_2 from $C^{i_j}_{a_2}$ to $C^{i_j}_{c_j}$ which cannot contain any disorders (otherwise P_1 would not be maximal). But that means that all clauses of P_2 except $C^{i_j}_{c_j}$ have to be derived by T-resolutions and the path $P_3 := (C^{i_j}_{a_1}, P_2)$ is a disorder of $C^{i_j}_{c_j}$ itself. Consequently, the transformation from π_{i_j} to π_{i_j+1} not only got rid of P_1, but also P_3 (note that $P_1 = P_3$ is possible).

Now, let $C^{i_j}_{d_j}$ and $C^{i_j}_{e_j}$ be the two parent clauses of $C^{i_j}_{a_1}$, such that $C^{i_j}_{d_j}$ is the XUT-clause, $C^{i_j}_{e_j}$ is the X-clause and $C^{i_j}_{a_1} = C^{i_j}_{d_j} \overset{x}{\bowtie} C^{i_j}_{e_j}$ with an X-literal x, $x \in C^{i_j}_{d_j}$ and $\bar{x} \in C^{i_j}_{e_j}$. If $\tau(x) = 0$, we can set $c_{j+1} := c_j$ and $i_{j+1} := i_j + 1$. Because the extra x is falsified by τ, we will still get $\tau(C^{i_j+1}_{c_{j+1}}) = 0$. If $\tau(x) = 1$, we set $c_{j+1} := e_j$ and $i_{j+1} := i_j + 1$ since $C^{i_j+1}_{e_j} = C^{i_j}_{e_j}$ and all X-literals from $C^{i_j}_{e_j}$, except $\bar{x}$, are still contained in $C^{i_j}_{c_j}$, hence $\tau(C^{i_j+1}_{e_j}) = 0$.

By construction, it holds $i_j - c_j < i_{j+1} - c_{j+1}$ for each j. In particular, after finitely many steps we have successfully detected D_τ in π'. We conclude that π' is quasi level-ordered. □

Note that π' does not necessarily need to be a refutation or even a connected proof. On an intuitive level, it is not obvious anymore what is actually "proven" by π'. The only properties we need is $|\pi'| = |\pi|$ and the existence of the D_τ clauses whose subproofs are level-ordered.

Let us now demonstrate the transformation on an example.

Example 5.11. *Let Φ be the QCNF that consists of the prefix*

$$\exists x, y \forall u \exists s, t$$

and the clauses

$$y \vee u \vee s, \quad \bar{x} \vee u \vee \bar{s}, \quad \bar{s} \vee \bar{t}, \quad u \vee \bar{s} \vee t$$
$$x \vee u \vee s, \quad \bar{y} \vee u \vee s, \quad u \vee \bar{s}.$$

Obviously, Φ is a Σ^b_3 QBF that fulfils the XT-property. Let π_0 be the Q-Res *refutation of Φ as depicted in Figure 5.1. Because all reductions are performed as soon as possible and there are not X-resolutions between two XUT-clauses, π_0 is fully reduced and primitive.*

In the first step, all X-clauses (cf. Figure 5.2) as well as all disorders (cf. Figure 5.3) are detected. Some X-clauses might already have level-ordered subproofs (e.g. C^0_4). The maximal (with respect to the lexicographic order) disorder is the first one to be fixed, which is $(C^0_{14}, C^0_{16}, C^0_{17})$. We have $C^0_{14} = C^0_{12} \overset{y}{\bowtie} C^0_{13}$, where C^0_{12} is the X-clause and C^0_{13} is the XUT-clause, and $C^0_{16} = C^0_{14} \overset{s}{\bowtie} C^0_{15}$. In the next iteration π_1, we have to change the derivation of the second clause in the disorder by skipping the resolution with the X-clause C^1_{14} and resolving the XUT-clause C^1_{13} with C^1_{15} directly. Hence, we set $C^1_{16} := C^1_{13} \overset{s}{\bowtie} C^1_{15}$ (cf. Figure 5.4).

The clause C^1_{12} still has two disorders that need to be fixed. We again detect the maximal disorder, which is $(C^1_6, C^1_{10}, C^1_{11}, C^1_{12})$. We change the derivation of C^2_{10} in π_2 by setting $C^2_{10} := C^2_5 \overset{s}{\bowtie} C^2_8$ (cf. Figure 5.5). The remaining (and maximal) disorder $(C^2_6, C^2_9, C^2_{11}, C^2_{12})$ is finally fixed by setting $C^3_9 := C^3_5 \overset{s}{\bowtie} C^3_7$ in $\pi_3 =: \pi'$ (cf. Figure 5.6).

At the end, no disorders are left and the subproofs of all X-clauses are level-ordered. For each $\tau \in \langle X \rangle$ it remains to find a clause $D_\tau \in \pi'$ with $\tau(D_\tau) = 0$.

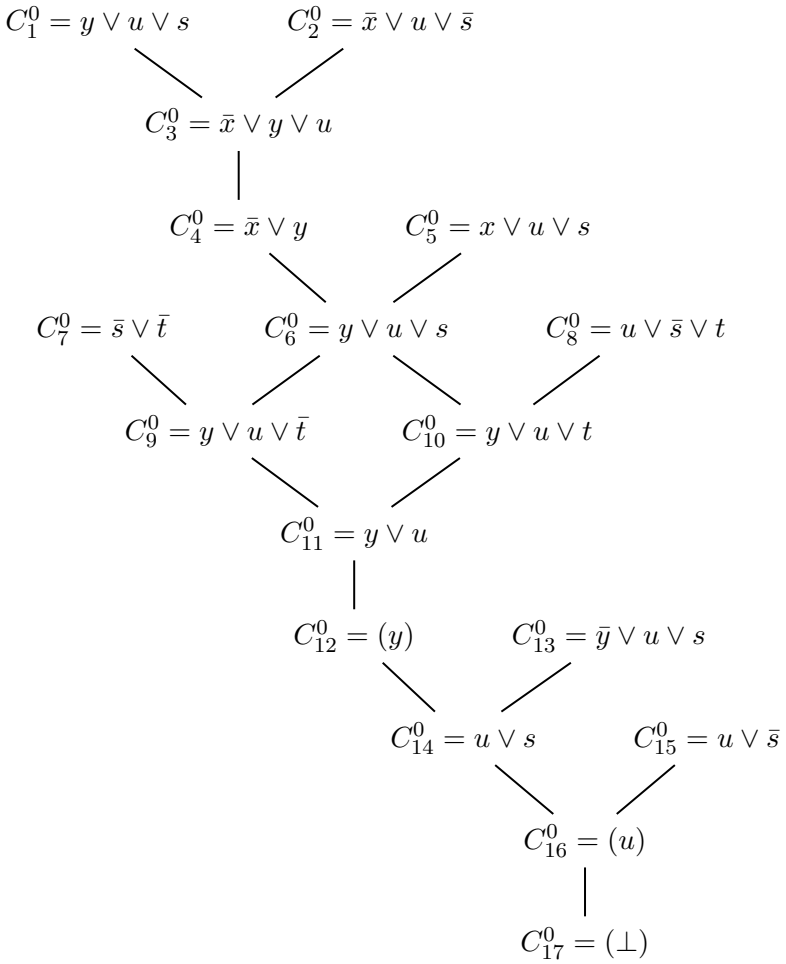


Figure 5.1: Fully reduced primitive Q-Res refutation π_0 of Φ from Example 5.11.

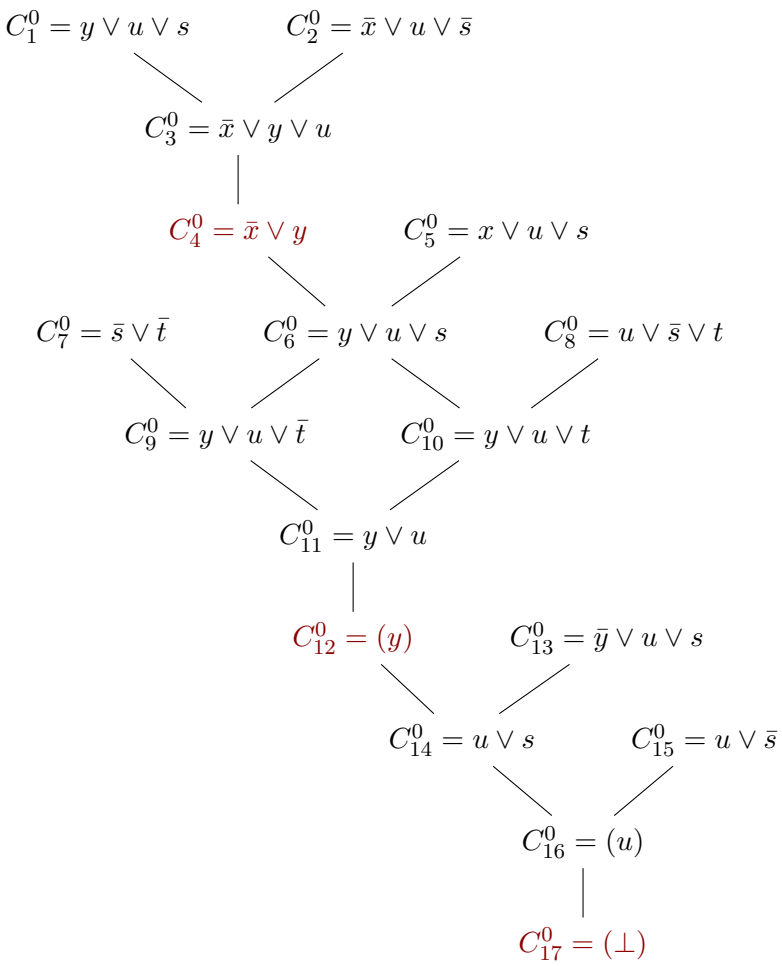


Figure 5.2: Fully reduced primitive Q-Res refutation π_0 of Φ from Example 5.11. X-clauses are colored red.

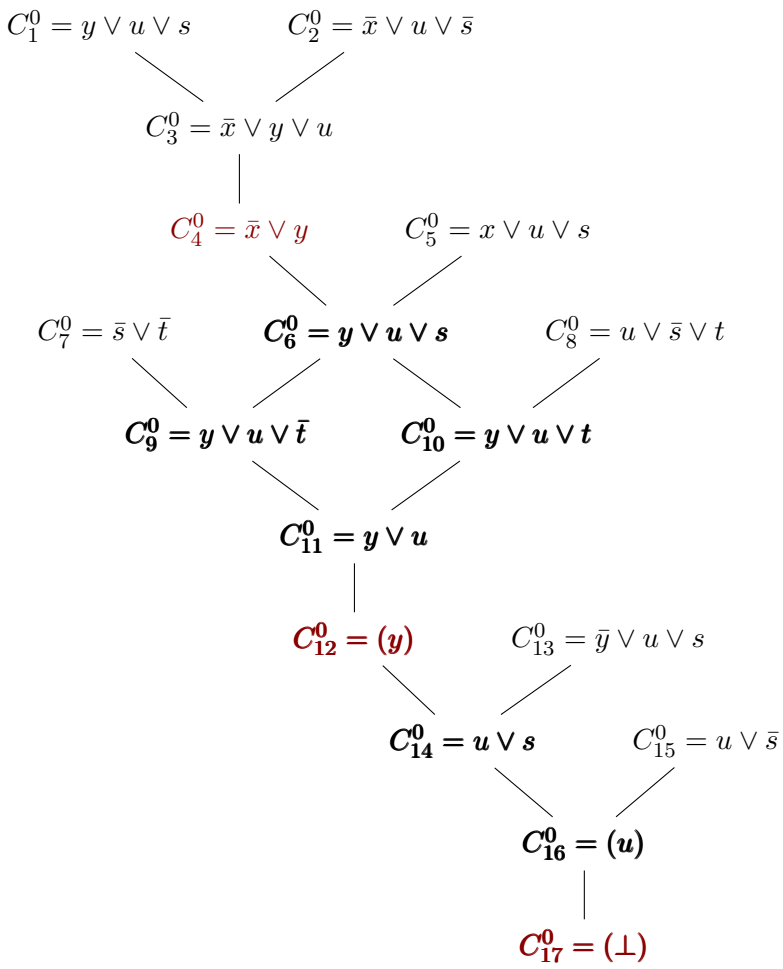


Figure 5.3: Fully reduced primitive Q-Res refutation π_0 of Φ from Example 5.11. X-clauses are colored red. Clauses that belong to disorders are written in bold. The following disorders can be detected: $(C_6^0, C_9^0, C_{11}^0, C_{12}^0), (C_6^0, C_{10}^0, C_{11}^0, C_{12}^0), (C_{14}^0, C_{16}^0, C_{17}^0)$. The lexicographic maximal disorder is $(C_{14}^0, C_{16}^0, C_{17}^0)$.

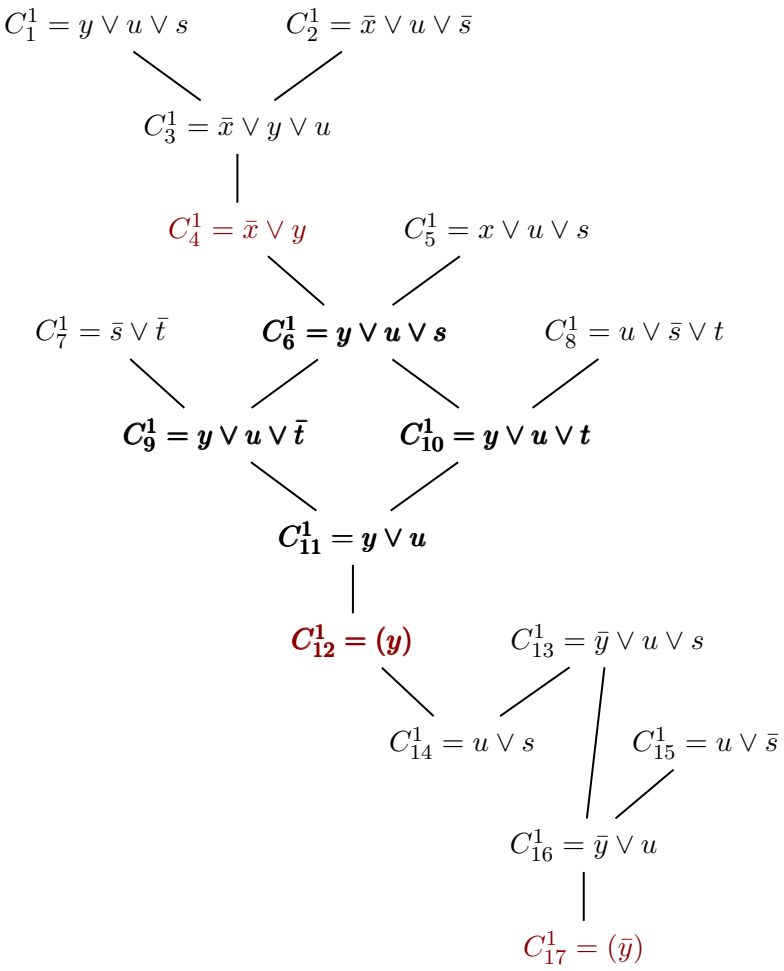


Figure 5.4: Q-Res refutation π_1 of Φ from Example 5.11 after fixing the maximal disorder of π_0. X-clauses are colored red. Clauses that belong to disorders are written in bold. The following disorders can be detected: $(C^1_6, C^1_9, C^1_{11}, C^1_{12}), (C^1_6, C^1_{10}, C^1_{11}, C^1_{12})$. The lexicographic maximal disorder is $(C^1_6, C^1_{10}, C^1_{11}, C^1_{12})$.

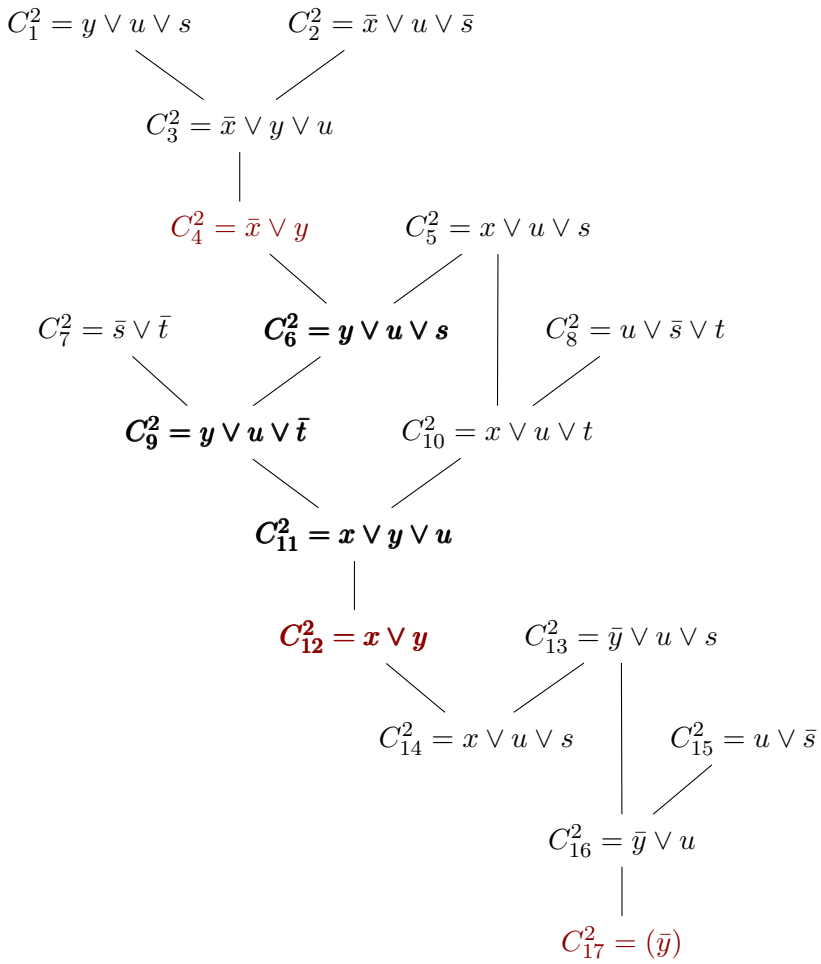


Figure 5.5: Q-Res refutation π_2 of Φ from Example 5.11 after fixing the maximal disorder of π_1. X-clauses are colored red. Clauses that belong to disorders are written in bold. The following disorders can be detected: $(C_6^2, C_9^2, C_{11}^2, C_{12}^2)$. The lexicographic maximal disorder is $(C_6^2, C_9^2, C_{11}^2, C_{12}^2)$.

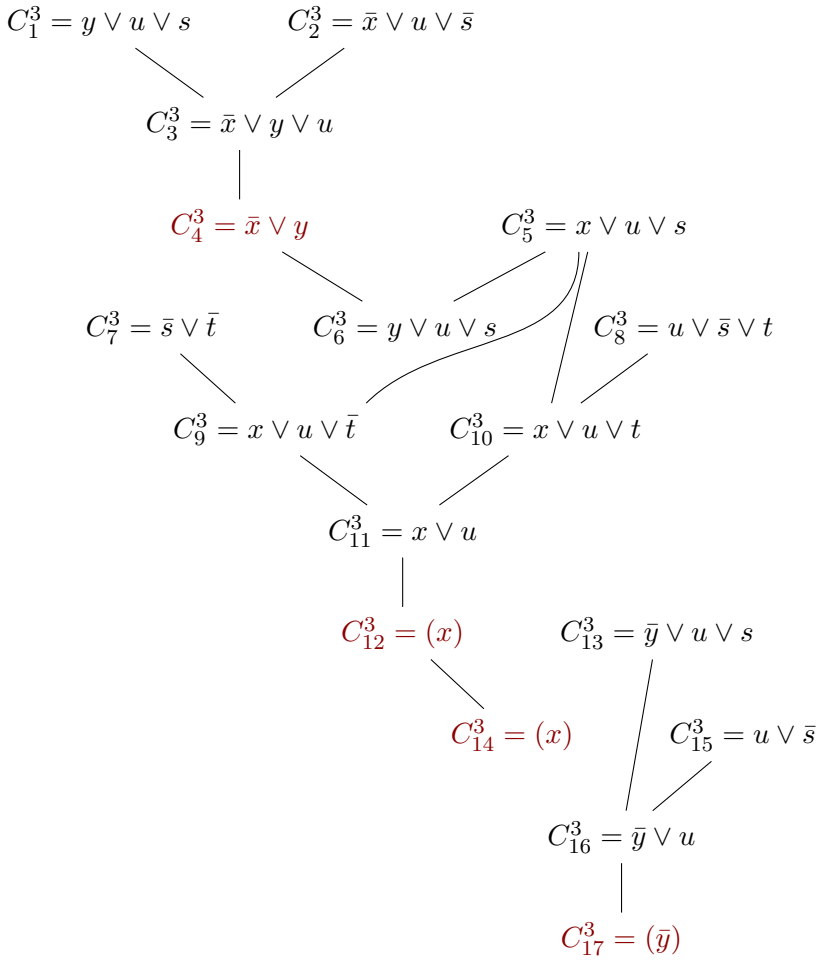


Figure 5.6: Q-Res refutation π_3 of Φ from Example 5.11 after fixing the maximal disorder of π_2. X-clauses are colored red. No disorders can be detected, therefore the proof is quasi level-ordered.

We follow the approach described in the proof of Theorem 5.10. For example, let $\tau_3 \in \langle X \rangle$ *with* $\tau_3(x) = 1$ *and* $\tau_3(y) = 0$*. We construct a sequence of X-clauses that are falsified by* τ_3 *such that the last of these clauses is contained in* π'*. We start with* $C^0_{17} = (\bot)$ *and check if its subproof is already level-ordered, which is not the case. The clause* C^0_{17} *was derived by a reduction and we have* $C^1_{17} \not\subseteq C^0_{17}$ *with* $\tau_3(C^1_{17}) = 1$*. That means in the transformation from* π_0 *to* π_1 *an X-literal was added to the clause which is not falsified by* τ_3*. Therefore, we choose the X-clause from the derivation of the first clause from the corresponding disorder as the next clause in our sequence, which is* C^1_{12}*.*

We repeat the procedure with C^1_{12} *since it is an X-clause whose subproof is not level-ordered. Again, in the step from* π_1 *to* π_2*, a disorder of* C^1_{12} *was fixed, which caused the appearance of an X-literal not falsified by* τ_3*. By the same argument as before, we choose* C^2_4 *as the next clause in our sequence.*

Because the subproof of C^2_4 *is now level-ordered, we can simply choose* C^3_4 *as the next and last clause in our sequence for* τ_3*. We now detected the sequence* $(C^0_{17}, C^1_{12}, C^2_4, C^3_4)$*, from which the last clause is desired. In fact, we have* $C^3_4 = \bar{x} \vee y$ *and* $\tau_3(C^3_4) = 0$*.*

By the same construction, we get sequences for the remaining assignments:

$$\begin{array}{lll}
\tau_1 : x \mapsto 0,\ y \mapsto 0 : & (C^0_{17}, C^1_{12}, C^2_{12}, C^3_{12}) & \Rightarrow D_{\tau_1} := C^3_{12} = (x), \\
\tau_2 : x \mapsto 0,\ y \mapsto 1 : & (C^0_{17}, C^1_{17}, C^2_{17}, C^3_{17}) & \Rightarrow D_{\tau_2} := C^3_{17} = (\bar{y}), \\
\tau_3 : x \mapsto 1,\ y \mapsto 0 : & (C^0_{17}, C^1_{12}, C^2_4, C^3_4) & \Rightarrow D_{\tau_3} := C^3_4 = \bar{x} \vee y, \\
\tau_4 : x \mapsto 1,\ y \mapsto 1 : & (C^0_{17}, C^1_{17}, C^2_{17}, C^3_{17}) & \Rightarrow D_{\tau_4} := C^3_{17} = (\bar{y}).
\end{array}$$

5.2 Formula Gauge

After having described how primitive refutations can be transformed into quasi level-ordered proofs, we will now present the actual lower bound technique for quasi level-ordered proofs. The approach is fairly simple: Quasi level-ordered proofs contain clauses D_τ for each $\tau \in \langle X \rangle$ that are falsified by τ. We introduce a measure which indicates the width of these D_τ clauses. A high (i.e. linear) value will indicate that there only few D_τ will overlap each other, meaning that we can find many of them which results in an exponential lower bound.

This measure is called (formula) gauge and is defined on all Σ^b_3 QCNFs.

Definition 5.12 ([33])**.** *Let* Φ *be a* Σ^b_3 *QCNF with prefix* $\exists X \forall U \exists T$*. We define* W_Φ *as the set of all* Q-Res *proofs* π *from* Φ *of X-clauses* C_π*, such that* π *consists of resolutions over T-literals and reductions only. We define*

$$\mathit{gauge}(\Phi) := \min\{|C_\pi| \ :\ C_\pi \text{ is the root of some } \pi \in W_\Phi\}.$$

Intuitively, the gauge of a formula is the number of different X-literals piled up during the process of getting rid of all T-literals by using T-resolutions only. While the computation of the gauge of a formula is certainly a very hard problem (we have to minimize over proofs), it is rather trivial on many handcrafted formulas as we demonstrate on the following example.

Definition 5.13 ([68]). *The QCNF* $\mathtt{CR}_n$ *consists of the quantifier prefix*

$$\exists x_{(1,1)}, \ldots, x_{(1,n)}, x_{(2,1)}, \ldots, x_{(2,n)}, \ldots, x_{(n,1)}, \ldots, x_{(n,n)} \forall u \exists s_1, \ldots, s_n, t_1, \ldots, t_n$$

and matrix clauses $(x_{(i,j)} \vee u \vee s_i)$, $(\bar{x}_{(i,j)} \vee \bar{u} \vee t_j)$ *for* $i, j \in \{1, \ldots, n\}$ *as well as* $\bigvee_{i \in \{1,\ldots,n\}} \bar{s}_i$ *and* $\bigvee_{i \in \{1,\ldots,n\}} \bar{t}_i$.

The $\mathtt{CR}_n$ formulas describe a 'completion' game on an $(n \times n)$-matrix (cf. [68]): The universal player has to set u to false iff for all i there exists a j such that $x_{(i,j)}$ is set to false. Otherwise, there exists an i such that for each j, the literal $x_{(i,j)}$ is set to true, hence the universal player has to set u to true. In [64], it was shown that $\mathtt{CR}_n$ has polynomial Q-Res refutations, while there is an exponential lower bound on a special version of QCDCL. Perspectively, we want to use the gauge lower bound to show hardness on our QCDCL variants.

It is readily checked that the $\mathtt{CR}_n$ formulas fulfil the XT-property. We can now compute their gauge. Note that according to our convention, the T-variables comprise of all variables $s_1, \ldots, s_n, t_1, \ldots, t_n$.

Lemma 5.14. *We have* $\mathrm{gauge}(\mathtt{CR}_n) = n$.

Proof. Since there are no X-clauses as axioms, we necessarily need to resolve over T somehow. For this we need T-literals of negative polarity, hence each $\pi \in W_{\mathtt{CR}_n}$ contains $\bigvee_{i \in [n]} \bar{s}_i$ or $\bigvee_{i \in [n]} \bar{t}_i$. In each $\pi \in W_{\mathtt{CR}_n}$ every T-literal has to be resolved away. For this reason we need the corresponding clauses $x_{(i,j)} \vee u \vee s_i$ or $\bar{x}_{(i,j)} \vee \bar{u} \vee t_j$. Because we cannot resolve over X in $\pi \in W_{\mathtt{CR}_n}$, there are at least n different X-literals that are piled up and therefore $\mathrm{gauge}(\mathtt{CR}_n) = n$. □

The gauge is used to estimate the minimal width of the X-clauses appearing at the transition from T-resolutions to X-resolutions in a level-ordered proof. We aim to apply this estimate to the derivations of C_τ clauses in quasi level-ordered proofs. The next lemma shows how the size of such a derivation is bounded by $2^{\mathrm{gauge}(\Phi)-c}$, while c is the width of the corresponding C_τ. It is a generalization of a lower bound for level-ordered Q-Res refutations (cf. [64]) by setting $C_\tau := (\bot)$ for all $\tau \in \langle X \rangle$ and $c := 0$.

Lemma 5.15. *Let* Φ *be a* Σ_3^b *QCNF. Let* π *be a level-ordered* Q-Res *proof from* Φ *of a non-tautological X-clause* D *with* $|D| = c$. *Then* $|\pi| \geq 2^{gauge(\Phi)-c}$.

Proof. Let $V := X \backslash \mathrm{var}(D)$. For each assignment $\tau \in \langle V \rangle$ we will find a path P_τ in π by going backwards starting from D. For each resolution step over some $x \in V$ we choose the path whose literals are negated by τ, hence we choose the clause that contains x if $\tau(x) = 0$ and the other clause otherwise. If there are resolution steps over variables from $\mathrm{var}(D)$, then we will always choose the literal from D. If we reach a reduction step, we will just expand the path by this one parental clause. If we detect a resolution step over a T-literal, we stop there.

Let C_τ be the clause at which we stop. Clearly, the subproof π_{C_τ} of C_τ is one of the derivations in W_Φ, hence $|C_\tau| \geq \mathrm{gauge}(\Phi)$. Then C_τ has to be a non-tautological X-clause with at least $\mathrm{gauge}(\Phi)$ different X-literals. Then C_τ contains at least $\mathrm{gauge}(\Phi) - c$

different X-literals whose variables are in V. These literals are negated by the assignment τ.

Now let a be the number of these clauses C_τ by summing over all τ. Since for each C_τ there are at most $|X| - \text{gauge}(\Phi)$ variables that are not contained as some literal in the clause, there are at most $2^{|X|-\text{gauge}(\Phi)}$ paths that can lead to each C_τ. Multiplying with the number of C_τ gives us at least the number of assignments $\tau \in \langle V \rangle$, hence

$$2^{|X|-\text{gauge}(\Phi)} \cdot a \geq 2^{|X|-c}$$
$$\Leftrightarrow a \geq 2^{|X|-c}/2^{|X|-\text{gauge}(\Phi)} = 2^{\text{gauge}(\Phi)-c}.$$

Since each C_τ is a clause from π, we get $|\pi| \geq a \geq 2^{\text{gauge}(\Phi)-c}$. □

By applying the lemma on all C_τ in a quasi level-ordered proof, we get a lower bound for quasi level-ordered Q-Res proofs:

Proposition 5.16. *Each quasi level-ordered* Q-Res *proof of a* Σ_3^b *QCNF* Φ *has size* $2^{\Omega(\mathit{gauge}(\Phi))}$.

Proof. Let π be the shortest quasi level-ordered proof from Φ. By the definition of quasi level-ordered proofs we can find clauses C_τ for each $\tau \in \langle X \rangle$.

Let $h := \min_{\tau \in \langle X \rangle} |C_\tau|$. By Lemma 5.15 we get $|\pi| \geq 2^{\text{gauge}(\Phi)-h}$, hence $h \geq \text{gauge}(\Phi) - \log|\pi|$. Each clause C_τ can have at most $2^{|X|-h}$ assignments $\alpha \in \langle X \rangle$ such that $C_\alpha = C_\tau$. Let $a := |\{C_\tau : \tau \in \langle X \rangle\}|$, then $a \cdot 2^{|X|-h} \geq 2^{|X|}$ and thus

$$|\pi| \geq a \geq 2^h \geq 2^{\text{gauge}(\Phi)-\log|\pi|} = \frac{2^{\text{gauge}(\Phi)}}{|\pi|}.$$

We conclude that $|\pi|^2 \in 2^{\Omega(\text{gauge}(\Phi))}$. □

We combine Theorem 5.10 and Proposition 5.16 and obtain the following lower bound for fully reduced primitive refutations:

Theorem 5.17. *Each fully reduced primitive* Q-Res *refutation of a* Σ_3^b *QCNF* Φ *that fulfils the XT-property has size* $2^{\Omega(\mathit{gauge}(\Phi))}$.

As proofs generated by QCDCL are fully reduced by default, one simply has to prove primitivity of generated refutations in order to apply the lower bound.

5.3 Applications

We now apply our new lower bound technique via gauge to show exponential lower bounds for fully reduced primitive Q-Res proofs for a number of QBF families. First, by combining Lemma 5.14 with Theorem 5.17 we obtain hardness for the $\mathtt{CR}_n$ formulas from [68].

Corollary 5.18. *The formulas* $\mathtt{CR}_n$ *require exponential-size fully reduced primitive* Q-Res *refutations.*

As our second example we introduce the following formulas.

Definition 5.19. *Let* $\texttt{ENarrow}_n := \exists x_1, \ldots, x_{n+1} \forall u_1, \ldots, u_{n+1} \exists t_1, \ldots, t_n \cdot \psi_n$ *with the matrix* ψ_n *containing the clauses:*

$$\begin{array}{ll} x_1 \vee u_1 \vee t_1,\ \bar{x}_1 \vee \bar{u}_1 \vee t_1, & \\ x_i \vee u_i \vee \bar{t}_{i-1} \vee t_i,\ \bar{x}_i \vee \bar{u}_i \vee \bar{t}_{i-1} \vee t_i, & \text{for } i = 2, \ldots, n \\ x_{n+1} \vee u_{n+1} \vee \bar{t}_n,\ \bar{x}_{n+1} \vee \bar{u}_{n+1} \vee \bar{t}_n. & \end{array}$$

It is easy to see that $\texttt{ENarrow}_n$ fulfils the XT-property. $\texttt{ENarrow}_n$ can be interpreted as a variant of $\texttt{Eq}_n$ (cf. [10] or the corresponding definition later), in which the universal player wins by setting all u_i equal to x_i. The only difference is that instead of having a long T-clause, the T-literals are now connected chain-like by only using XUT-clauses. Next we will show an exponential lower bound for $\texttt{ENarrow}_n$ for fully reduced primitive Q-Res.

Lemma 5.20. *We have gauge*$(\texttt{ENarrow}_n) = n + 1$.

Proof. Let $\pi \in W_{\texttt{ENarrow}_n}$ with an X-clause D as root. Define the sets of clauses

$$\begin{aligned} Z_1 &:= \{x_1 \vee u_1 \vee t_1,\ \bar{x}_1 \vee \bar{u}_1 \vee t_1\} \\ Z_i &:= \{x_i \vee u_i \vee \bar{t}_{i-1} \vee t_i,\ \bar{x}_i \vee \bar{u}_i \vee \bar{t}_{i-1} \vee t_i\} \qquad \text{for } i = 2, \ldots, n \\ Z_{n+1} &:= \{x_{n+1} \vee u_{n+1} \vee \bar{t}_n,\ \bar{x}_{n+1} \vee \bar{u}_{n+1} \vee \bar{t}_n\}. \end{aligned}$$

Let C be an axiom clause in π. Then C has to be contained in some set Z_i as above.

Case 1. $C \in Z_1$.

Then we have to get rid of $t_1 \in C$ by resolving over t_1, for which we need a clause from Z_2. In fact, for each axiom clause from Z_i with $i = 2, \ldots, n$ we need to get rid of t_i by using a clause from Z_{i+1}. We conclude that we need a clause from each Z_j for $j = 1, \ldots, n+1$, and for each of these Z_j we will pile up the literal x_j or $\bar{x}_j$ in D. Hence we get $|D| \geq n + 1$.

Case 2. $C \in Z_i$ for some $i \in \{2, \ldots, n\}$.

For each clause from Z_i for $i = 2, \ldots, n$ we need a clause from Z_{i-1} to get rid of $\bar{t}_{i-1}$ and a clause from Z_{i+1} to get rid of t_i. Hence, as in Case 1, we pile up one of the literals x_j or $\bar{x}_j$ in D for $j = 1, \ldots, n+1$.

Case 3. $C \in Z_{n+1}$.

This works similarly to Case 1, except that we start at Z_{n+1} and go backwards. □

Corollary 5.21. *The QBFs* $\texttt{ENarrow}_n$ *require exponential-size fully reduced primitive* Q-Res *refutations.*

The gauge of a formula is obviously some width measure and it seems natural to wonder how it relates to the notion of the existential proof width[1] of LD-Q-Res refutations of a formula as studied in [12, 22, 48]. However, it turns out that these two measures

[1]The existential width of a clause is defined as the number of existential literals in this clause. The existential proof width is defined as the maximal existential width over all clauses in this proof.

are not directly related. On the one hand, it is easy to see that $\texttt{ENarrow}_n$ has LD-Q-Res refutations of constant existential clause width. Hence these formulas have small (constant) existential proof width, but linear gauge.

On the other hand, there are also formulas with constant gauge and linear proof width. For this we revisit the parity formula from [17].

Definition 5.22 ([17])**.** $\texttt{QParity}_n$ *consists of the prefix* $\exists x_1 \dots x_n \forall u \exists t_2 \dots t_n$ *and the matrix*

$$\begin{gathered} x_1 \vee x_2 \vee \bar{t}_2,\ x_1 \vee \bar{x}_2 \vee t_2,\ \bar{x}_1 \vee x_2 \vee t_2,\ \bar{x}_1 \vee \bar{x}_2 \vee \bar{t}_2, \\ x_i \vee t_{i-1} \vee \bar{t}_i,\ x_i \vee \bar{t}_{i-1} \vee t_i,\ \bar{x}_i \vee t_{i-1} \vee t_i,\ \bar{x}_i \vee \bar{t}_{i-1} \vee \bar{t}_i \quad \text{for } i \in \{3, \dots, n\} \\ u \vee t_n,\ \bar{u} \vee \bar{t}_n. \end{gathered}$$

The formulas $\texttt{QParity}_n$ are built on the parity function. In more detail, the universal player only wins by setting u equal to $x_1 \oplus \dots \oplus x_n$. Each t_i encodes the partial sum $x_1 \oplus \dots \oplus x_i$.

It was shown in [12, 22] that $\texttt{QParity}_n$ requires linear proof width. Here we modify this formula such that proof width remains unaffected, but gauge is small. Let $\texttt{mQParity}_n$ be the modified variant of this formula that consists of the prefix

$$\exists x_1, \dots, x_n, y \forall u \exists t_2, \dots, t_n$$

and the matrix

$$(\bar{y}) \wedge \bigwedge_{C \in \texttt{QParity}_n} (y \vee C).$$

Obviously, because of the unit clause $(\bar{y})$, we have $\text{gauge}(\texttt{mQParity}_n) = 1$, but still linear proof width (since we can simply consider the proof width of the proof restricted to the assignment $y \mapsto 0$, which is exactly a refutation of $\texttt{QParity}_n$).

We will see later that we can also use the $\texttt{QParity}_n$ formulas to show that large gauge alone is not sufficient to guarantee QCDCL hardness, but some further assumption such as the XT-condition is needed.

We continue with the equality formula from [10] as a further example of hard formulas for fully reduced primitive Q-Res.

Definition 5.23 ([10])**.** *The formula* $\texttt{Eq}_n$ *is defined as the QCNF*

$$\exists x_1 \dots x_n \forall u_1 \dots u_n \exists t_1 \dots t_n \cdot (\bar{t}_1 \vee \dots \vee \bar{t}_n) \wedge \bigwedge_{i=1}^{n} ((\bar{x}_i \vee \bar{u}_i \vee t_i) \wedge (x_i \vee u_i \vee t_i)).$$

The formula $\texttt{Eq}_n$ obviously fulfils the XT-property and is known to have only one winning strategy: The universal player has to set each u_i equal to x_i. Then the existential player is forced to satisfy each t_i, eventually leading to the falsification of $(\bar{t}_1 \vee \dots \vee \bar{t}_n)$. If the universal player would set some u_i unequal to x_i, then the existential player could win the game by setting t_i to false and all the other T-variables to true.

Proposition 5.24. *We have gauge*$(\mathtt{Eq}_n) = n$*. Consequently the formulas are exponentially hard for fully reduced primitive* Q-Res*.*

Proof. Let $\pi \in W_{\mathtt{Eq}_n}$. Since none of the axioms are X-clauses, we have to resolve over T somehow. For this we need the clause $\bar{t}_1 \vee \ldots \vee \bar{t}_n$. But that means we have to resolve over each t_i at least once in π, and therefore we will pile up all n X-variables.

□

The next example is a formula that follows the same approach as $\mathtt{Eq}_n$. However, we can replace this task with another, more complex one. In our case, the universal player has to detect palindromes in the word that was input by the existential player.

Example 5.25. *Let* $\mathtt{QPalin}_n$ *be the QCNF with prefix*

$$\exists X \forall U \exists T$$

with

$$\begin{aligned} X &= \{x_j : j \in \{1, \ldots, n\}\} \\ U &= \left\{u_{k,i}, v_k : k \in \{0, \ldots n-1\}, i \in \left\{1, \ldots, \left\lfloor \frac{n}{2} \right\rfloor\right\}\right\} \\ T &= \left\{t_{k,i}, s_k : k \in \{0, \ldots n-1\}, i \in \left\{1, \ldots, \left\lfloor \frac{n}{2} \right\rfloor\right\}\right\} \end{aligned}$$

where the indices from the X-variables are interpreted as integers modulo n. Let the matrix of the formula consist of the following clauses:

$$\begin{aligned} & x_{i+k} \vee x_{n-i+1+k} \vee \bar{u}_{k,i} \vee t_{k,i},\ \bar{x}_{i+k} \vee \bar{x}_{n-i+1+k} \vee \bar{u}_{k,i} \vee t_{k,i} \\ & x_{i+k} \vee \bar{x}_{n-i+1+k} \vee u_{k,i} \vee \bar{t}_{k,i},\ \bar{x}_{i+k} \vee x_{n-i+1+k} \vee u_{k,i} \vee \bar{t}_{k,i} \\ & \bar{v}_k \vee \bar{t}_{k,1} \vee \ldots \vee \bar{t}_{k,\lfloor \frac{n}{2} \rfloor} \vee s_k \\ & v_k \vee t_{k,i} \vee s_k,\ \bar{s}_0 \vee \ldots \vee \bar{s}_{n-1} \end{aligned}$$

for $k \in \{0, \ldots n-1\}, i \in \left\{1, \ldots, \left\lfloor \frac{n}{2} \right\rfloor\right\}$.

Intuitively, the X-variables represent the word in which palindromes have to be detected. We not only check the word $x_1 \ldots x_n$ itself, but also shifted variants $x_{1+k} \ldots x_{n+k}$. The $u_{k,i}$ encode whether or not, in the word $x_{1+k} \ldots x_{n+k}$, the i^{th} letter from the left and the i^{th} letter from the right are the same. If the universal player assigns the $u_{k,i}$ correctly, the existential player is forced to set $t_{k,i}$ equal to $u_{k,i}$ in their turn at the end. If and only if the word $x_{1+k} \ldots x_{n+k}$ was a palindrome, the universal player sets v_k to true. Hence, if $x_{1+k} \ldots x_{n+k}$ was a palindrome, v_k as well as $t_{k,i}$ for each i is set to true, hence the existential player has to set s_k to true. Otherwise, if it was not a palindrome, then there was an i such that x_{i+k} and $x_{n-i+1+k}$ were assigned differently, therefore $t_{k,i}$ and v_k are both false and s_k must be set to true, as well. However, setting all s_k to true falsifies the matrix.

In a nutshell: Let τ be a total assignment of X. There is a winning strategy for the universal player by setting $u_{k,i}$ to 1 if and only if $\tau(x_{i+k}) = \tau(x_{n-i+k})$ and v_k to 1 if and only if the word $\tau(x_{1+k}) \dots \tau(x_{n+k})$ is a palindrome. Then the existential player has to set each s_k to 1, negating the last clause in the matrix.

Again, it can be easily shown that $\mathtt{QPalin}_n$ fulfils the XT-property. We will show next that the gauge of $\mathtt{QPalin}_n$ is not exponential but sub-exponential, which implies a sub-exponential lower bound for fully reduced primitive Q-Res.

Lemma 5.26. *We have gauge*$(\mathtt{QPalin}_n) \in \Omega(\sqrt{n})$.

Proof. Let $\pi \in W_{\mathtt{QPalin}_n}$. Since we do not have any X-clauses as axioms, we need to resolve over T-variables at least once. We partition the matrix of $\mathtt{QPalin}_n$ into the following sets:

$$\begin{aligned}
Z^+_{k,i} &:= \{x_{i+k} \vee x_{n-i+1+k} \vee \bar{u}_{k,i} \vee t_{k,i}, \bar{x}_{i+k} \vee \bar{x}_{n-i+1+k} \vee \bar{u}_{k,i} \vee t_{k,i}\} \\
Z^-_{k,i} &:= \{x_{i+k} \vee \bar{x}_{n-i+1+k} \vee u_{k,i} \vee \bar{t}_{k,i}, \bar{x}_{i+k} \vee x_{n-i+1+k} \vee u_{k,i} \vee \bar{t}_{k,i}\} \\
P_k &:= \{\bar{v}_k \vee \bar{t}_{k,1} \vee \ldots \vee \bar{t}_{k,\lfloor \frac{n}{2} \rfloor} \vee s_k\} \\
N_{k,i} &:= \{v_k \vee t_{k,i} \vee s_k\} \\
S &:= \{\bar{s}_0 \vee \ldots \vee \bar{s}_{n-1}\}
\end{aligned}$$

Let $C \in \pi$ be an axiom. We claim that $S \subseteq \pi$, for which we will distinguish four cases.

Case 1. $C \in Z^+_{k,i}$ for some $k \in \{0, \dots, n-1\}$ and $i \in \{1, \dots, \lfloor \frac{n}{2} \rfloor\}$.

Then we have to resolve away $t_{k,i}$, which can only be done with the clause in P_k since the clauses from $Z^-_{k,i}$ are blocked because of the $u_{k,i}$. But now we have introduced s_k which we can only resolve with the clause from S.

Case 2. $C \in Z^-_{k,i}$ for some $k \in \{0, \dots, n-1\}$ and $i \in \{1, \dots, \lfloor \frac{n}{2} \rfloor\}$.

To get rid of $\bar{t}_{k,i}$, we have to use the clause from $N_{k,i}$ since $Z^+_{k,i}$ is blocked as before. But then we have introduces s_k and we will need the clause from S.

Case 3. $C \in P_k$ or $C \in N_{k,i}$ for some $k \in \{0, \dots, n-1\}$ and $i \in \{1, \dots, \lfloor \frac{n}{2} \rfloor\}$.

Then we have $s_k \in C$ and we need to use the clause from S in order to resolve it away.

Case 4. $C \in S$.

This case is trivial.

We have shown that in each case we have $S \subseteq \pi$. That means we have to resolve over each s_k in π. For each $k \in \{0, \dots, n-1\}$ we can choose if we want to use P_k or $N_{k,i}$ to get rid of the literal $\bar{s}_k$. If we choose P_k, then we have to resolve over each $t_{k,i}$ for $i \in \{1, \dots, \lfloor \frac{n}{2} \rfloor\}$ by using the clauses from $Z^+_{k,i}$. However, if we choose $N_{k,i}$, it suffices to resolve over only one $t_{k,i}$ for some particular i. Hence, we only have to use one clause from $Z^-_{k,i}$ for only one i. In the worst case (that means in the case with the least resolutions over $t_{k,i}$), we will always pick $N_{k,i}$. More specific, for each $k \in \{0, \dots, n-1\}$ there exists an $i_k \in \{1, \dots, \lfloor \frac{n}{2} \rfloor\}$ such that π contains at least one clause from Z^+_{k,i_k} or Z^-_{k,i_k}. That means we will pile up at least the X-variables x_{i_k+k}, x_{n-i_k+1+k} for each

$k \in \{0, \ldots, n-1\}$. If we can find a lower bound for the number of these X-variables, then this is also a lower bound for gauge($\texttt{QPalin}_n$).

First of all, it its obvious that for each k we have $x_{i_k+k} \neq x_{n-i_k+1+k}$ since $i_k + k \not\equiv n - i_k + 1 + k \pmod{n}$. Next, we define the following sets of variables:

$$X_k := \{x_{i_k+k}, x_{n-i_k+1+k}\}$$

for $k \in \{0, \ldots, n-1\}$. As we have argued above, we already know that the gauge of $\texttt{QPalin}_n$ is $\Omega(|\bigcup_k X_k|)$. Note that if a pair $\{x_j, x_\ell\}$ is equal to X_k, then their position in the word $x_{1+k} \ldots x_{n+k}$ is symmetric. If n is odd, then each pair $\{x_j, x_\ell\}$ can represent at most one X_k. For example, for $n = 5$ the pair $\{x_3, x_4\}$ can at most be X_3 since they are only symmetric in the word $x_4x_5x_1x_2x_3$. However, if n is even then each pair then each pair $\{x_j, x_\ell\}$ can represent up to two X_k. For example, for $n = 4$ the pair $\{x_1, x_4\}$ is symmetric in both $x_1x_2x_3x_4$ and $x_3x_4x_1x_2$.

We conclude that for odd n we have

$$|\{X_0, \ldots, X_{n-1}\}| = n$$

and for even n we have

$$|\{X_0, \ldots, X_{n-1}\}| \geq \frac{n}{2}.$$

Now, with m different variables we could create at most $\mathcal{O}(m^2)$ different pairs X_k. Hence we need at least $\Omega(\sqrt{n})$ different variables to create $\mathcal{O}(n)$ different pairs X_k, and therefore $|\bigcup_k X_k| \in \Omega(\sqrt{n})$ and also gauge($\texttt{QPalin}_n$) $\in \Omega(\sqrt{n})$. □

Corollary 5.27. *The QCNF* $\texttt{QPalin}_n$ *needs fully reduced primitive* Q-Res *refutations of size* $2^{\Omega(\sqrt{n})}$.

Chapter 6

Analysing Standard QCDCL

Applying the gauge lower bound technique to QCDCL yields plenty of separations not only between several QCDCL variants and their underlying proof systems, but also between different variants of QCDCL. In Chapter 4, we already introduced formalizations of classic QCDCL variants as proof systems with Cube Learning (QCDCL) and without (QCDCL$^{\text{w/o Cubes}}$). We recall another modification for QCDCL called *Pure Literal Elimination*. The idea is simple: Whenever a pure literal (i.e. a literal that appears in only one polarity) is detected, and no Unit Propagation is possible, this literal can be assigned independent of the quantification order. More precisely, an existential pure literal will be set to true out of order, since this is the only logical step the existential player can do. Analogously, a universal pure literal will be set to false immediately because the universal player seeks to falsify the matrix.

We will mark literals that were assigned via Pure Literal Elimination underlined in a trail. For example, in the trail $(\mathbf{x}; \underline{\bar{\mathbf{y}}}, t, \bot)$ the literal $\bar{y}$ was assigned via Pure Literal Elimination and will be treated as a decision. Pure Literal Elimination has a higher priority than regular decisions, meaning that we need to assign pure literals before proceeding with level-ordered decisions. On the other hand, Pure Literal Elimination has a lower priority than Unit Propagation, which means that we can only assign pure literals if no propagations or conflicts are possible.

We will denote QCDCL variants with Pure Literal Elimination as QCDCL$^{\text{PL}}$ (with Cube Learning) or QCDCL$^{\text{PL w/o Cubes}}$ (without Cube Learning). In sum, for this chapter we consider the following four QCDCL variants:

- QCDCL
- QCDCL$^{\text{w/o Cubes}}$
- QCDCL$^{\text{PL}}$
- QCDCL$^{\text{PL w/o Cubes}}$

Since Pure Literal Elimination is technically treated as making decisions out-of-order, we can conclude by Proposition 4.8 that all four variants are sound. In order to prove

that the two new systems $\mathsf{QCDCL}^{\mathsf{P_L}}$ and $\mathsf{QCDCL}^{\mathsf{P_L\ w/o\ Cubes}}$ are complete, we have to show that we are still able to learn a new constraint from each trail.

Proposition 6.1. *Let ι be a $\mathsf{QCDCL}^{\mathsf{P_L}}$ or $\mathsf{QCDCL}^{\mathsf{P_L\ w/o\ Cubes}}$ proof and let $\mathcal{T}$ be some trail in ι that has run into a conflict. If $(\bot) \notin \mathfrak{L}(\mathcal{T})$ and $[\top] \notin \mathfrak{L}(\mathcal{T})$, then $\mathfrak{L}(\mathcal{T})$ contains an asserting clause or cube.*

Proof. The proof is similar to Proposition 4.12. W.l.o.g. let the conflict be existential, which means $(\bot) \in \mathcal{T}$. In the proof of Proposition 4.12, we only needed the fact that the existential decisions are level-ordered.

Universal literals propagated via cubes will still be treated as decisions and do not need to be level-ordered. Literals assigned via Pure Literal Elimination will also treated as decisions. Now, existential literals decided via Pure Literal Elimination do not necessarily need to be level-ordered. However, since they are pure literals, they cannot trigger unit propagations on clauses (because then we would need the literal in two different polarities). Existential pure literals can only trigger unit propagations on cubes, which will be treated as decisions anyways.

Therefore, existential decisions that are relevant for clause are still level-ordered and therefore we can learn the same asserting clause as in the proof of Proposition 4.12. □

In the same way as Theorem 4.13 we can combine the above observations into the following result:

Theorem 6.2. *The proof system $\mathsf{QCDCL}^{\mathsf{P_L}}$ is complete on both false and true QCNFs. The variant $\mathsf{QCDCL}^{\mathsf{P_L\ w/o\ Cubes}}$ is at least complete on false QCNFs.*

6.1 Lower Bounds for QCDCL via Primitivity

We now apply our lower bound technique from Chapter 5 to the four aforementioned variants of QCDCL. We recall the main components for showing lower bounds of a QCNF Φ:

1. Φ has to fulfil the XT-property.
2. gauge(Φ) needs to be "large" (e.g. linear for exponential lower bounds).
3. The QCDCL variant has to generate primitive proofs on Φ.

While the first two points are properties that only depend on the formula Φ itself, the third point is empirically harder to show since it is deeply connected to the policies that are active. Therefore we start with proving some sufficient conditions for primitivity on several QCDCL variants.

Lemma 6.3. *Let $\mathcal{T}$ be a trail in a $\mathsf{QCDCL}^{\mathsf{w/o\ Cubes}}$, QCDCL, $\mathsf{QCDCL}^{\mathsf{P_L\ w/o\ Cubes}}$ or $\mathsf{QCDCL}^{\mathsf{P_L}}$ proof from a QCNF Φ with the XT-property. Then for each T-literal $t_1 \in \mathcal{T}$, which was not decided by Pure Literal Elimination, there is a U-literal $u \in \mathcal{T}$ with $u <_{\mathcal{T}} t_1$.*

Proof. If t_1 was decided regularly, then the situation is clear because we can only decide T-literals if and only if all U-variables were assigned before. Therefore we can assume that there is no T-literal $t' \in \mathcal{T}$ with $t' \leq_\mathcal{T} t_1$ such that t' was a regular decision.

If t_1 was propagated, then it has an antecedent constraint $A := \text{ante}_\mathcal{T}(t_1)$. W.l.o.g. we assume that t_1 is the first propagated T-literal in $\mathcal{T}$. It is clear that A cannot be a cube, otherwise t_1 would have been immediately reduced away during Cube Learning. Hence, A is a clause. If A contains a U-literal, then this U-literal needs to appear left of t_1 in the trail somehow, as we demand.

Therefore, let us assume that A contains no U-literal. Because of the XT-property, A cannot be an XT-clause (and therefore cannot contain X-literals), and it cannot be a unit T-clause, either. That means there is another T-literal $t_1 \neq t \in A$ with $\bar{t} <_\mathcal{T} t_1$. But that is not possible, because $\bar{t}$ can neither be a decision, nor a propagated literal, nor a pure literal (since t already occurs in A positively).

We conclude that in each case, there is a U-literal $u \in T$ with $u <_\mathcal{T} t_1$. □

Lemma 6.3 leads to the following result, which indicates that it is somewhat "hard" to achieve non-primitivity on QCDCL for QBFs that fulfil the XT-property. In fact, we show that QCDCL can only generate primtive proofs (under the XT-property), if the trails are not level-ordered.

Proposition 6.4. *Let ι be a* $\mathsf{QCDCL}^{\text{w/o Cubes}}$, QCDCL, $\mathsf{QCDCL}^{\text{PL w/o Cubes}}$ *or* $\mathsf{QCDCL}^{\text{PL}}$ *refutation of a QCNF Φ that fulfils the XT-property. If $\mathfrak{R}(\iota)$ is not primitive, then there exists a trail in ι such that there is a U-literal $u \in \mathcal{T}$ and an X-literal $x \in \mathcal{T}$ with $u <_\mathcal{T} x$. Additionally, u cannot be a regular decision literal.*

Proof. If $\mathfrak{R}(\iota)$ is not primitive, then there are two XUT-clauses $C, D \in \mathfrak{R}(\iota)$ that are resolved over an X-variable x, say $x \in C$ and $\bar{x} \in D$. One of these clauses has to be an antecedent clause of some trail $\mathcal{T}$ in ι, w.l.o.g. let C be the antecedent clause $\text{ante}_\mathcal{T}(x)$. Let $\bar{t} \in C$ be one of the T-literals from C. In particular, we have $t \in \mathcal{T}$ and $t <_\mathcal{T} x$. Because t was not a pure literal decision (we have $\bar{t} \in C$) and because of Lemma 6.3, there is a U-literal $u \in \mathcal{T}$ with $u <_\mathcal{T} t$. We conclude that also $u <_\mathcal{T} x$ holds.

Since we can only decide U-literals regularly if all X-variables are assigned in some polarity in $\mathcal{T}$, it is impossible for u to be a regular decision literal. □

This result is essentially telling us that for a non-primitive proof $\mathfrak{R}(\iota)$ of some S proof ι, where S is one of our four QCDCL variants, ι needs to consist of a trail that assigns a U-literal out-of-order (i.e., before we have assigned all X-literals). Since neither Cube Learning nor Pure Literal Elimination is allowed in $\mathsf{QCDCL}^{\text{w/o Cubes}}$, we can immediately conclude:

Corollary 6.5. *The extracted* Q-Res *refutation of any* $\mathsf{QCDCL}^{\text{w/o Cubes}}$ *refutation of a QCNF that fulfils the XT-property is always primitive.*

Note that primitivity can still be proven for the other variants with Cube Learning or Pure Literal Elimination. An obvious approach would be considering QCNFs for which

Cube Learning or Pure Literal Elimination do not affect QCDCL runs in a beneficial way (i.e. by triggering non-primitive resolution steps).

After combining Corollary 6.5 with Proposition 5.24 and Corollaries 5.18, 5.21 and 5.27, we obtain:

Corollary 6.6. *It holds:*

(i) $\mathtt{Eq}_n$ *needs* $\mathsf{QCDCL}^{w/o\ \textsc{Cubes}}$ *refutations of size* $2^{\Omega(n)}$.

(ii) $\mathtt{CR}_n$ *needs* $\mathsf{QCDCL}^{w/o\ \textsc{Cubes}}$ *refutations of size* $2^{\Omega(n)}$.

(iii) $\mathtt{ENarrow}_n$ *needs* $\mathsf{QCDCL}^{w/o\ \textsc{Cubes}}$ *refutations of size* $2^{\Omega(n)}$.

(iv) $\mathtt{QPalin}_n$ *needs* $\mathsf{QCDCL}^{w/o\ \textsc{Cubes}}$ *refutations of size* $2^{\Omega(\sqrt{n})}$.

Obviously, all three formulas above fulfil the XT-property. Our next example illustrates that some further condition (such as the XT-property) is indeed required for our lower bound method to work. For this we will take another look at the parity formula $\mathtt{QParity}_n$. These formulas are known to be hard for Q-Res [17]. We will demonstrate that $\mathtt{QParity}_n$ is in fact easy for $\mathsf{QCDCL}^{w/o\ \textsc{Cubes}}$.

Proposition 6.7. $\mathtt{QParity}_n$ *has polynomial-size* $\mathsf{QCDCL}^{w/o\ \textsc{Cubes}}$ *refutations.*

Proof. We construct the trails $(\mathcal{T}_n, \mathcal{U}_n, \mathcal{T}_{n-1}, \mathcal{U}_{n-1}, \ldots, \mathcal{T}_2, \mathcal{U}_2, \mathcal{T}_1, \mathcal{U}_1)$.

Let $\mathcal{T}_n$ be the trail

$$\mathcal{T}_n = (\bar{\mathbf{x}}_\mathbf{1}; \bar{\mathbf{x}}_\mathbf{2}, \bar{t}_2; \bar{\mathbf{x}}_\mathbf{3}, \bar{t}_3; \ldots; \bar{\mathbf{x}}_\mathbf{n-1}, \bar{t}_{n-1}; \bar{\mathbf{x}}_\mathbf{n}, \bar{t}_n, \bot)$$

with antecedent clauses

$$\begin{aligned} \mathrm{ante}_{\mathcal{T}_n}(\bar{t}_2) &= x_1 \vee x_2 \vee \bar{t}_2, \\ \mathrm{ante}_{\mathcal{T}_n}(\bar{t}_i) &= x_i \vee t_{i-1} \vee \bar{t}_i \text{ for } i = 3, \ldots, n, \\ \mathrm{ante}_{\mathcal{T}_n}(\bot) &= u \vee t_n. \end{aligned}$$

After resolving over $\bar{t}_n$ we can derive $C_n := x_n \vee u \vee t_{n-1}$ as a learnable clause and backtrack to $\mathcal{T}_n[n-1, 1]$.

Further, let $\mathcal{U}_n$ be the trail

$$\mathcal{U}_n = (\bar{\mathbf{x}}_\mathbf{1}; \bar{\mathbf{x}}_\mathbf{2}, \bar{t}_2; \bar{\mathbf{x}}_\mathbf{3}, \bar{t}_3; \ldots; \bar{\mathbf{x}}_\mathbf{n-2}, \bar{t}_{n-2}; \mathbf{x}_\mathbf{n-1}, t_{n-1}; \bar{\mathbf{x}}_\mathbf{n}, t_n, \bot)$$

with antecedent clauses

$$\begin{aligned} \mathrm{ante}_{\mathcal{U}_n}(\bar{t}_2) &= x_1 \vee x_2 \vee \bar{t}_2, \\ \mathrm{ante}_{\mathcal{U}_n}(\bar{t}_i) &= x_i \vee t_{i-1} \vee \bar{t}_i \text{ for } i = 3, \ldots, n-2, \\ \mathrm{ante}_{\mathcal{U}_n}(t_{n-1}) &= \bar{x}_{n-1} \vee t_{n-2} \vee t_{n-1}, \\ \mathrm{ante}_{\mathcal{U}_n}(t_n) &= x_n \vee \bar{t}_{n-1} \vee t_n, \\ \mathrm{ante}_{\mathcal{U}_n}(\bot) &= \bar{u} \vee \bar{t}_n. \end{aligned}$$

Symmetrically to $\mathcal{T}_n$, we can learn the clause $D_n := x_n \vee \bar{u} \vee \bar{t}_{n-1}$ by resolving over t_n and backtrack up to $\mathcal{U}_n[n-1,1]$.

We continue with the construction of $\mathcal{T}_{n-1}$.

$$\mathcal{T}_{n-1} = (\bar{\mathbf{x}}_\mathbf{1}; \bar{\mathbf{x}}_\mathbf{2}, \bar{t}_2; \bar{\mathbf{x}}_\mathbf{3}, \bar{t}_3; \ldots; \bar{\mathbf{x}}_\mathbf{n-2}, \bar{t}_{n-2}; \bar{\mathbf{x}}_\mathbf{n-1}, \bar{t}_{n-1}, x_n, t_n, \bot)$$

with antecedent clauses

$$\begin{aligned}
\text{ante}_{\mathcal{T}_{n-1}}(\bar{t}_2) &= x_1 \vee x_2 \vee \bar{t}_2, \\
\text{ante}_{\mathcal{T}_{n-1}}(\bar{t}_i) &= x_i \vee t_{i-1} \vee \bar{t}_i \text{ for } i = 3, \ldots, n-1, \\
\text{ante}_{\mathcal{T}_{n-1}}(x_n) &= C_n = x_n \vee u \vee t_{n-1}, \\
\text{ante}_{\mathcal{T}_{n-1}}(t_n) &= \bar{x}_n \vee t_{n-1} \vee t_n, \\
\text{ante}_{\mathcal{T}_{n-1}}(\bot) &= \bar{u} \vee \bar{t}_n.
\end{aligned}$$

After resolving over t_n, x_n and finally $\bar{t}_{n-1}$ we get the learned clause $C_{n-1} := x_{n-1} \vee u \vee \bar{u} \vee t_{n-2}$ and backtrack up to $\mathcal{T}_{n-1}[n-2,1]$.

As before, we can symmetrically create the next trail $\mathcal{U}_{n-1}$.

$$\mathcal{U}_{n-1} = (\bar{\mathbf{x}}_\mathbf{1}; \bar{\mathbf{x}}_\mathbf{2}, \bar{t}_2; \bar{\mathbf{x}}_\mathbf{3}, \bar{t}_3; \ldots; \bar{\mathbf{x}}_\mathbf{n-3}, \bar{t}_{n-3}; \mathbf{x}_\mathbf{n-2}, t_{n-2}; \bar{\mathbf{x}}_\mathbf{n-1}, t_{n-1}, x_n, \bar{t}_n, \bot)$$

with antecedent clauses

$$\begin{aligned}
\text{ante}_{\mathcal{U}_{n-1}}(\bar{t}_2) &= x_1 \vee x_2 \vee \bar{t}_2, \\
\text{ante}_{\mathcal{U}_{n-1}}(\bar{t}_i) &= x_i \vee t_{i-1} \vee \bar{t}_i \text{ for } i = 3, \ldots, n-3, \\
\text{ante}_{\mathcal{U}_{n-1}}(t_{n-2}) &= \bar{x}_{n-2} \vee t_{n-3} \vee t_{n-2}, \\
\text{ante}_{\mathcal{U}_{n-1}}(t_{n-1}) &= x_{n-1} \vee \bar{t}_{n-2} \vee t_{n-1}, \\
\text{ante}_{\mathcal{U}_{n-1}}(x_n) &= D_n = x_n \vee \bar{u} \vee \bar{t}_{n-1}, \\
\text{ante}_{\mathcal{U}_{n-1}}(\bar{t}_n) &= \bar{x}_n \vee \bar{t}_{n-1} \vee \bar{t}_n, \\
\text{ante}_{\mathcal{U}_{n-1}}(\bot) &= u \vee t_n.
\end{aligned}$$

By resolving over $\bar{t}_n$, x_n and t_{n-1} we can derive $D_{n-1} := x_{n-1} \vee u \vee \bar{u} \vee \bar{t}_{n-2}$ and backtrack up to $\mathcal{U}_{n-1}[n-2,1]$.

We now describe the general step. Let $j \in \{3, \ldots, n-2\}$ and suppose we already learned the clauses $C_n, D_n, C_{n-1}, D_{n-1}, \ldots, C_{j+1}, D_{j+1}$ with the aid of the earlier trails $\mathcal{T}_n, \mathcal{U}_n, \ldots, \mathcal{T}_{j+1}, \mathcal{U}_{j+1}$, where $C_k := x_k \vee u \vee \bar{u} \vee t_{k-1}$ and $D_k := x_k \vee u \vee \bar{u} \vee \bar{t}_{k-1}$ for $k \in \{j+1, \ldots, n-1\}$. Then we can construct $\mathcal{T}_j$ as follows:

$$\mathcal{T}_j = (\bar{\mathbf{x}}_\mathbf{1}; \bar{\mathbf{x}}_\mathbf{2}, \bar{t}_2; \bar{\mathbf{x}}_\mathbf{3}, \bar{t}_3; \ldots; \bar{\mathbf{x}}_\mathbf{j-1}, \bar{t}_{j-1}; \bar{\mathbf{x}}_\mathbf{j}, \bar{t}_j, x_{j+1}, t_{j+1}, x_{j+2}, \bar{t}_{j+2}, \ldots, x_n, t_n^e, \bot)$$

with $e \in \{0,1\}$ such that $e \equiv n-j \pmod 2$ and antecedent clauses

$$\begin{aligned}
\text{ante}_{\mathcal{T}_j}(\bar{t}_2) &= x_1 \vee x_2 \vee \bar{t}_2,\\
\text{ante}_{\mathcal{T}_j}(\bar{t}_i) &= x_i \vee t_{i-1} \vee \bar{t}_i \text{ for } i = 3,\dots,j,\\
\text{ante}_{\mathcal{T}_j}(x_{j+1}) &= C_{j+1} = x_{j+1} \vee u \vee \bar{u} \vee t_j,\\
\text{ante}_{\mathcal{T}_j}(t_{j+1}) &= \bar{x}_{j+1} \vee t_j \vee t_{j+1},\\
\text{ante}_{\mathcal{T}_j}(x_{j+2}) &= D_{j+2} = x_{j+2} \vee u \vee \bar{u} \vee \bar{t}_{j+1},\\
\text{ante}_{\mathcal{T}_j}(\bar{t}_{j+2}) &= \bar{x}_{j+2} \vee \bar{t}_{j+1} \vee \bar{t}_{j+2},\\
&\vdots\\
\text{ante}_{\mathcal{T}_j}(x_n) &= \begin{cases} C_n & = x_n \vee \bar{u} \vee t_{n-1}, \text{ if } n-j \text{ is odd},\\ D_n & = x_n \vee u \vee \bar{t}_{n-1}, \text{ if } n-j \text{ is even}, \end{cases}\\
\text{ante}_{\mathcal{T}_j}(t_n^e) &= \bar{x}_n \vee t_{n-1}^e \vee t_n^e,\\
\text{ante}_{\mathcal{T}_j}(\bot) &= u^{1-e} \vee t_n^{1-e}.
\end{aligned}$$

We resolve over $t_n^e, x_n, \dots, t_{j+1}, x_{j+1}$ and $\bar{t}_j$, derive $C_j := x_j \vee u \vee \bar{u} \vee t_{j-1}$ and backtrack up to $\mathcal{T}_j[j-1,1]$.

In a similar way we will create $\mathcal{U}_j$:

$$\mathcal{U}_j = (\bar{\mathbf{x}}_\mathbf{1}; \bar{\mathbf{x}}_\mathbf{2}, \bar{t}_2; \bar{\mathbf{x}}_\mathbf{3}, \bar{t}_3; \dots; \bar{\mathbf{x}}_{\mathbf{j-2}}, \bar{t}_{j-2}; \mathbf{x}_{\mathbf{j-1}}, t_{j-1}; \bar{\mathbf{x}}_\mathbf{j}, t_j, x_{j+1}, \bar{t}_{j+1}, x_{j+2}, t_{j+2}, \dots, x_n, t_n^f, \bot)$$

with $f \in \{0,1\}$ such that $f \equiv n-j+1 \pmod 2$ and antecedent clauses

$$\begin{aligned}
\text{ante}_{\mathcal{U}_j}(\bar{t}_2) &= x_1 \vee x_2 \vee \bar{t}_2,\\
\text{ante}_{\mathcal{U}_j}(\bar{t}_i) &= x_i \vee t_{i-1} \vee \bar{t}_i \text{ for } i = 3,\dots,j-2,\\
\text{ante}_{\mathcal{U}_j}(t_{j-1}) &= \bar{x}_{j-1} \vee t_{j-2} \vee t_{j-1},\\
\text{ante}_{\mathcal{U}_j}(t_j) &= x_j \vee \bar{t}_{j-1} \vee t_j,\\
\text{ante}_{\mathcal{U}_j}(x_{j+1}) &= D_{j+1} = x_{j+1} \vee u \vee \bar{u} \vee \bar{t}_j,\\
\text{ante}_{\mathcal{U}_j}(\bar{t}_{j+1}) &= \bar{x}_{j+1} \vee \bar{t}_j \vee \bar{t}_{j+1},\\
\text{ante}_{\mathcal{U}_j}(x_{j+2}) &= C_{j+2} = x_{j+2} \vee u \vee \bar{u} \vee t_{j+1},\\
\text{ante}_{\mathcal{U}_j}(t_{j+2}) &= \bar{x}_{j+2} \vee t_{j+1} \vee \bar{t}_{j+2},\\
&\vdots\\
\text{ante}_{\mathcal{U}_j}(x_n) &= \begin{cases} C_n & = x_n \vee \bar{u} \vee t_{n-1}, \text{ if } n-j \text{ is even},\\ D_n & = x_n \vee u \vee \bar{t}_{n-1}, \text{ if } n-j \text{ is odd}, \end{cases}\\
\text{ante}_{\mathcal{U}_j}(t_n^f) &= \bar{x}_n \vee t_{n-1}^f \vee t_n^f,\\
\text{ante}_{\mathcal{U}_j}(\bot) &= u^{1-f} \vee t_n^{1-f}.
\end{aligned}$$

Resolving over $t_n^f, x_n, \dots \bar{t}_{j+1}, x_{j+1}$ and t_j gives us $D_j := x_j \vee u \vee \bar{u} \vee \bar{t}_{j-1}$. We backtrack to $\mathcal{U}_j[j-1,1]$.

We end the refutation with the following four trails whose antecedent clauses are almost as before:

$$\mathcal{T}_2 = (\bar{\mathbf{x}}_\mathbf{1}; \bar{\mathbf{x}}_\mathbf{2}, \bar{t}_2, x_3, t_3, x_4, \bar{t}_4, \ldots, \bot),$$

from which we learn $C_2 := x_1 \vee x_2$, and

$$\mathcal{U}_2 = (\bar{\mathbf{x}}_\mathbf{1}, x_2, t_2, x_3, \bar{t}_4, \ldots, \bot),$$

where we can derive the unit clause (x_1). We continue with

$$\mathcal{T}_1 = (x_1; \bar{\mathbf{x}}_\mathbf{2}, t_2, x_3, \bar{t}_3, x_4, t_4, \ldots, \bot),$$

learn (x_2), and can finally derive the empty clause $(\bot)$ via the last trail

$$\mathcal{U}_1 = (x_1, x_2, \bar{t}_2, x_3, t_3, x_4, \bar{t}_4, \ldots, \bot).$$

□

Nevertheless, we show that $\mathtt{QParity}_n$ has large gauge. Hence this measure alone is not sufficient to imply $\mathsf{QCDCL}^{\mathsf{w/o\ Cubes}}$ hardness.

Proposition 6.8. *We have gauge*$(\mathtt{QParity}_n) = n$.

Proof. We define the following sets of clauses:

$$\begin{aligned} Z_1 &:= \{x_1 \vee x_2 \vee \bar{t}_2,\ x_1 \vee \bar{x}_2 \vee t_2,\ \bar{x}_1 \vee x_2 \vee t_2,\ \bar{x}_1 \vee \bar{x}_2 \vee \bar{t}_2\} \\ Z_i &:= \{x_{i+1} \vee t_i \vee \bar{t}_{i+1},\ x_{i+1} \vee \bar{t}_i \vee t_{i+1},\ \bar{x}_{i+1} \vee t_i \vee t_{i+1},\ \bar{x}_{i+1} \vee \bar{t}_i \vee \bar{t}_{i+1}\} \\ Z_n &:= \{u \vee t_n,\ \bar{u} \vee \bar{t}_n\} \end{aligned}$$

for $i = 2, \ldots, n-1$.

We show that each $\pi \in W_{\mathtt{QParity_n}}$ needs at least one clause from each Z_j as an axiom, hence $\pi \cap Z_j \neq \emptyset$ for every $j \in [n]$.

Assume that there is a proof $\pi \in W_{\mathtt{QParity_n}}$ of an X-clause C and $j \in [n]$ with $\pi \cap Z_j = \emptyset$. Let S be the set of all symmetries σ on $\mathtt{QParity}_n$ such that $\sigma(x_k) \in \{x_k, \bar{x}_k\}$ for each $k \in [n]$ and $\sigma(t_k)$ is chosen such that $\sigma(\mathtt{QParity}_n) \subseteq \mathtt{QParity}_n$ (one has to make sure that $\sigma(t_k) = \sigma(x_1) \oplus \ldots \oplus \sigma(x_k)$).

Then for each such $\sigma \in S$ we can derive $\sigma(C)$ via $\sigma(\pi)$ and still have $\sigma(\pi) \cap Z_j = \emptyset$. But then we could easily construct a refutation by just using $\{\sigma(C) : \sigma \in S\}$. Then $\mathtt{QParity}_n$ without the clauses from Z_j would still be a false QCNF. However, this is not possible since we can construct a winning strategy A for $\mathtt{QParity}_n \backslash Z_j$:

$$\begin{aligned} A(x_k) &:= 0 \text{ for all } k \in \{1, \ldots, n\} \\ A(t_\ell) &:= 0 \text{ for all } \ell \in \{2, \ldots, j\} \\ A(t_{\ell'}) &:= 1 \oplus u \text{ for all } \ell' \in \{j+1, \ldots, n\} \end{aligned}$$

Therefore our assumption is false and we get $\pi \cap Z_j \neq \emptyset$. Using one clause from each Z_j results in piling up all variables $x_1, \ldots, x_n$ in some polarity, hence gauge$(\mathtt{QParity}_n) = n$. □

6.2 Plain QCDCL vs. Cube Learning and Pure Literal Elimination

By applying the gauge lower bound technique to our QCDCL variants, we want to separate the different versions in order to verify the effect of Cube Learning and Pure Literal Elimination. So far, we have only applied the lower bound to the plain QCDCL version $\mathsf{QCDCL}^{\mathsf{w/o\ Cubes}}$. Both Cube Learning and Pure Literal Elimination are able to prevent the generated proofs from becoming primitive.

We start by examining the influence of Cube Learning on our QCDCL model. For false formulas we can always prevent learning cubes by just deciding the universal variables according to a winning strategy for the universal player, which will cause a conflict on the current trail. Thus Cube Learning will never be disadvantageous in principle.

Proposition 6.9. *It holds the following:*

(i) QCDCL *p-simulates* $\mathsf{QCDCL}^{\mathsf{w/o\ Cubes}}$.

(ii) $\mathsf{QCDCL}^{\mathsf{PL}}$ *p-simulates* $\mathsf{QCDCL}^{\mathsf{PL\ w/o\ Cubes}}$.

Proof. A $\mathsf{QCDCL}^{\mathsf{w/o\ Cubes}}$ ($\mathsf{QCDCL}^{\mathsf{PL\ w/o\ Cubes}}$) proof translates into a QCDCL ($\mathsf{QCDCL}^{\mathsf{PL}}$) proof where all trails run into conflict and no cubes are learnt. □

Since Cube Learning is implemented in practical QCDCL by default, it is not surprising that it does not only not have a disadvantageous effect on QBF solving theoretically, it also ensures that true QBFs can be verified. Unexpectedly, it can be shown that Cube Learning benefits QBF solving even on false formulas as the next Proposition shows.

For that we return to the previously defined Equality formulas (cf. Definition 5.23). The idea is as follows: Suitable learnable cubes can be used to propagate universal literals in the correct polarity according to the X-assignments. This destroys the level order inside the trails and prevents primitivity by allowing X-resolutions between XUT-clauses during Clause Learning.

Proposition 6.10. *There exist polynomial-size* QCDCL *refutations of* $\mathtt{Eq}_n$.

Proof. First we learn the cubes $x_i \wedge \bar{u}_i$ and $\bar{x}_i \wedge u_i$ for $i = 1, \ldots, n-1$. In order to learn $x_1 \wedge \bar{u}_1$, we can use the trail

$$\mathcal{T}_1 := (\mathbf{x_1}; \ldots; \mathbf{x_n}; \bar{\mathbf{u}}_\mathbf{1}; \ldots; \bar{\mathbf{u}}_\mathbf{n}; \bar{\mathbf{t}}_\mathbf{1}; \mathbf{t_2}; \ldots; \mathbf{t_n}).$$

Then the partial assignment $x_1 \wedge \bar{u}_1 \wedge \bar{t}_1 \wedge t_2 \wedge \ldots \wedge t_n$ satisfies the matrix of $\mathtt{Eq}_n$. Reducing this cube existentially results in $x_1 \wedge \bar{u}_1$, hence $x_1 \wedge \bar{u}_1 \in \mathfrak{L}(\mathcal{T}_1)$.

Learning $\bar{x}_1 \wedge u_1$ works analogously. Note that the previously learned cube does not interfere with the learning of this cube.

Having already learned the $2i$ cubes from 1 to i, let us now explain how to learn the two cubes for $i+1$. We create the following trail:

$$\mathcal{T}_{i+1} := (\mathbf{x_1}, u_1, t_1; \ldots; \mathbf{x_i}, u_i, t_i; \mathbf{x_{i+1}}; \ldots; \mathbf{x_n}; \bar{\mathbf{u}}_\mathbf{i+1}; \ldots; \bar{\mathbf{u}}_\mathbf{n}; \bar{\mathbf{t}}_\mathbf{i+1}; \mathbf{t_{i+2}}; \ldots; \mathbf{t_n})$$

with

$$\begin{aligned}\text{ante}_{\mathcal{T}_{i+1}}(u_j) &= x_j \wedge \bar{u}_j,\\ \text{ante}_{\mathcal{T}_{i+1}}(t_j) &= \bar{x}_j \vee \bar{u}_j \vee t_j\end{aligned}$$

for $j = 1, \ldots, i$.

Again, the partial assignment $x_{i+1} \wedge \bar{u}_{i+1} \wedge t_1 \wedge \ldots \wedge t_i \wedge \bar{t}_{i+1} \wedge t_{i+2} \wedge \ldots \wedge t_n$ satisfies the matrix of $\texttt{Eq}_n$. This can be reduces to the cube $x_{i+1} \wedge \bar{u}_{i+1}$, which we will learn. As before, learning $\bar{x}_{i+1} \wedge u_{i+1}$ works analogously.

After we have learned all of these $2n-2$ cubes, we will go on with Clause Learning in which we will successively learn the clauses

$$\begin{aligned}L_i &:= \bar{x}_i \vee \bar{u}_i \vee \bigvee_{j=i+1}^{n} (u_j \vee \bar{u}_j) \vee \bigvee_{k=1}^{i-1} \bar{t}_k\\ R_i &:= x_i \vee u_i \vee \bigvee_{j=i+1}^{n} (u_j \vee \bar{u}_j) \vee \bigvee_{k=1}^{i-1} \bar{t}_k\end{aligned}$$

for $i = 2, \ldots, n-1$.

We start with the following trails:

$$\mathcal{U}_{n-1} := (\mathbf{x_1}, u_1, t_1; \ldots; \mathbf{x_{n-1}}, u_{n-1}, t_{n-1}, \bar{t}_n, x_n, \bot)$$

with

$$\begin{aligned}\text{ante}_{\mathcal{U}_{n-1}}(u_j) &= x_j \wedge \bar{u}_j\\ \text{ante}_{\mathcal{U}_{n-1}}(t_j) &= \bar{x}_j \vee \bar{u}_j \vee t_j\\ \text{ante}_{\mathcal{U}_{n-1}}(\bar{t}_n) &= \bar{t}_1 \vee \ldots \vee \bar{t}_n\\ \text{ante}_{\mathcal{U}_{n-1}}(x_n) &= x_n \vee u_n \vee t_n\\ \text{ante}_{\mathcal{U}_{n-1}}(\bot) &= \bar{x}_n \vee \bar{u}_n \vee t_n\end{aligned}$$

for $j = 1, \ldots, n-1$. We resolve over $x_n, \bar{t}_n$ and t_{n-1} and get L_{n-1}. Analogously, we can learn R_{n-1}.

Suppose we have already learned $L_{n-1}, R_{n-1}, \ldots, L_i, R_i$ for some $i \in \{3, \ldots, n-1\}$. Let us now construct trails from which we can learn L_{i-1} and R_{i-1}:

$$\mathcal{U}_{i-1} := (\mathbf{x_1}, u_1, t_1; \ldots; \mathbf{x_{i-1}}, u_{i-1}, t_{i-1}, x_i, \bot)$$

with

$$\begin{aligned}\text{ante}_{\mathcal{U}_{i-1}}(u_j) &= x_j \wedge \bar{u}_j,\\ \text{ante}_{\mathcal{U}_{i-1}}(t_j) &= \bar{x}_j \vee \bar{u}_j \vee t_j\\ \text{ante}_{\mathcal{U}_{i-1}}(x_i) &= R_i\\ \text{ante}_{\mathcal{U}_{i-1}}(\bot) &= L_i\end{aligned}$$

for $j = 1, \ldots, i-1$. We resolve over x_i and t_{i-1} and get L_{i-1}. Again, analogously we can derive R_{i-1}.

After we have finished learning L_2 and R_2, we can create the last two trails as follows:

$$\mathcal{U}_1 := (\mathbf{x_1}, u_1, t_1, x_2, \bot)$$

with

$$\begin{aligned}\text{ante}_{\mathcal{U}_1}(u_1) &= x_1 \wedge \bar{u}_1\\ \text{ante}_{\mathcal{U}_1}(t_1) &= \bar{x}_1 \vee \bar{u}_1 \vee t_1\\ \text{ante}_{\mathcal{U}_1}(x_2) &= R_2\\ \text{ante}_{\mathcal{U}_1}(\bot) &= L_2.\end{aligned}$$

We resolve over x_2 and t_1 and obtain the unit clause $(\bar{x}_1)$. Then the last trail will not contain any decision:

$$\mathcal{U}_1' := (\bar{x}_1, \bar{u}_1, t_1, x_2, \bot)$$

with

$$\begin{aligned}\text{ante}_{\mathcal{U}_1'}(\bar{x}_1) &= (\bar{x}_1)\\ \text{ante}_{\mathcal{U}_1'}(u_1) &= x_1 \wedge \bar{u}_1\\ \text{ante}_{\mathcal{U}_1'}(t_1) &= \bar{x}_1 \vee \bar{u}_1 \vee t_1\\ \text{ante}_{\mathcal{U}_1'}(x_2) &= R_2\\ \text{ante}_{\mathcal{U}_1'}(\bot) &= L_2.\end{aligned}$$

Resolving over all existential variables leads to the empty clause. □

As the formulas $\mathtt{Eq}_n$ require exponential-sized $\mathsf{QCDCL}^{\text{w/o Cubes}}$ refutations by Corollary 6.6 we obtain:

Theorem 6.11. QCDCL *is exponentially stronger than* $\mathsf{QCDCL}^{\text{w/o Cubes}}$.

Next we will look at the influence of Pure Literal Elimination. The effect of Pure Literal Elimination is similar to Cube Learning: they enable out-of-order decisions that can shorten the refutations. This again manifests in $\mathtt{Eq}_n$.

Proposition 6.12. $\mathtt{Eq}_n$ *has poly-size* $\mathsf{QCDCL}^{\text{PL w/o Cubes}}$ *(and* $\mathsf{QCDCL}^{\text{PL}}$*) refutations.*

Proof. The refutation is similar to the one in Proposition 6.10, except that instead of learning cubes, we will use Pure Literal Elimination to decide the universal literals out of order. We will again learn the clauses L_i and R_i for $i = 2, \ldots, n-1$.

We start with the following trails:

$$\mathcal{U}_{n-1} := (\mathbf{x_1}; \underline{\mathbf{u_1}}, t_1; \ldots; \mathbf{x_{n-1}}; \underline{\mathbf{u_{n-1}}}, t_{n-1}, \bar{t}_n, x_n, \bot)$$

with

$$\begin{aligned}\text{ante}_{\mathcal{U}_{n-1}}(t_j) &= \bar{x}_j \vee \bar{u}_j \vee t_j\\ \text{ante}_{\mathcal{U}_{n-1}}(\bar{t}_n) &= \bar{t}_1 \vee \ldots \vee \bar{t}_n\\ \text{ante}_{\mathcal{U}_{n-1}}(x_n) &= x_n \vee u_n \vee t_n\\ \text{ante}_{\mathcal{U}_{n-1}}(\bot) &= \bar{x}_n \vee \bar{u}_n \vee t_n\end{aligned}$$

for $j = 1, \ldots, n-1$. We resolve over $x_n, \bar{t}_n$ and t_{n-1} and get L_{n-1}. In an analogous way we can learn R_{n-1}.

Suppose we have already learned $L_{n-1}, R_{n-1}, \ldots, L_i, R_i$ for some $i \in \{3, \ldots, n-1\}$. Let us now construct trails from which we can learn L_{i-1} and R_{i-1}:

$$\mathcal{U}_{i-1} := (\mathbf{x_1}; \underline{\mathbf{u_1}}, t_1; \ldots; \mathbf{x_{i-1}}; \underline{\mathbf{u_{i-1}}}, t_{i-1}, x_i, \bot)$$

with

$$\begin{aligned}\text{ante}_{\mathcal{U}_{i-1}}(t_j) &= \bar{x}_j \vee \bar{u}_j \vee t_j\\ \text{ante}_{\mathcal{U}_{i-1}}(x_i) &= R_i\\ \text{ante}_{\mathcal{U}_{i-1}}(\bot) &= L_i\end{aligned}$$

for $j = 1, \ldots, i-1$. We resolve over x_i and t_{i-1} and get L_{i-1}. Again, analogously we can derive R_{i-1}. Note that, in our case, the learned clauses will not interfere with Pure Literal Elimination. Once we have learned L_i and R_i, we will not need to make the literals from $u_i, \ldots, u_n$ pure any more. Also, say we learn L_i before R_i, once we decide $\bar{x}_i$ in order to learn R_i, we will also make L_i true. Therefore Pure Literal Elimination behaves (almost) symmetrically.

After we have finished learning L_2 and R_2, we can create the last two trails as follows:

$$\mathcal{U}_1 := (\mathbf{x_1}; \underline{\mathbf{u_1}}, t_1, x_2, \bot)$$

with

$$\begin{aligned}\text{ante}_{\mathcal{U}_1}(t_1) &= \bar{x}_1 \vee \bar{u}_1 \vee t_1\\ \text{ante}_{\mathcal{U}_1}(x_2) = R_2 &= x_2 \vee u_2 \vee \bigvee_{j=3}^{n} (u_j \vee \bar{u}_j) \vee \bar{t}_1\\ \text{ante}_{\mathcal{U}_1}(\bot) = L_2 &= \bar{x}_2 \vee u_2 \vee \bigvee_{j=3}^{n} (u_j \vee \bar{u}_j) \vee \bar{t}_1.\end{aligned}$$

We resolve over x_2 and t_1 and obtain the unit clause $(\bar{x}_1)$. Then the last trail will not contain any decision:

$$\mathcal{U}_1' := (\bar{x}_1, \underline{\bar{\mathbf{u}}_\mathbf{1}}, t_1, x_2, \bot)$$

with

$$\begin{aligned}\text{ante}_{\mathcal{U}'_1}(\bar{x}_1) &= (\bar{x}_1)\\ \text{ante}_{\mathcal{U}'_1}(t_1) &= \bar{x}_1 \vee \bar{u}_1 \vee t_1\\ \text{ante}_{\mathcal{U}'_1}(x_2) &= R_2\\ \text{ante}_{\mathcal{U}'_1}(\bot) &= L_2.\end{aligned}$$

Resolving over all existential variables leads to the empty clause. □

Although Pure Literal Elimination helps to refute $\mathtt{Eq}_n$, it turns out that Pure Literal Elimination can also be disadvantageous. It might be a fallacy to think that pure existential literals should be satisfied in the same way as unit clauses in unit propagation. We will construct formulas in which Pure Literal Elimination thwarts finding a convenient conflict and therefore short refutations.

We construct these formulas in stages, starting with $\mathtt{MirrorCR}_n$. These QBFs are based on $\mathtt{CR}_n$, known to be hard for $\mathsf{QCDCL}^{\text{w/o Cubes}}$ (cf. [64] or Corollary 6.6). The "Mirror"-modification adds new symmetries to the formula, causing pure literals to appear too late to make a difference.

Definition 6.13. *The QCNF* $\mathtt{MirrorCR}_n(X, u, T)$ *with* $X = \{x_{(1,1)}, x_{(1,2)} \ldots, x_{(n,n)}\}$ *and* $T = \{a_1, \ldots, a_n, b_1, \ldots, b_n\}$ *consists of the prefix*

$$\exists x_{(1,1)}, \ldots, x_{(n,n)} \forall u \exists a_1, \ldots, a_n, b_1, \ldots, b_n$$

and the matrix

$$\begin{aligned}&x_{(i,j)} \vee u \vee a_i \quad \bar{a}_1 \vee \ldots \vee \bar{a}_n\\ &\bar{x}_{(i,j)} \vee \bar{u} \vee b_j \quad \bar{b}_1 \vee \ldots \vee \bar{b}_n\\ &x_{(i,j)} \vee \bar{u} \vee \bar{a}_i \quad a_1 \vee \ldots \vee a_n\\ &\bar{x}_{(i,j)} \vee u \vee \bar{b}_j \quad b_1 \vee \ldots \vee b_n \quad \textit{for } i, j \in [n].\end{aligned}$$

It is easy to see that $\mathtt{MirrorCR}_n$ fulfils the XT-property. Additionally, we can show:

Proposition 6.14. *The CNF* $\mathfrak{C}(\mathtt{MirrorCR}_n)$ *is unsatisfiable and gauge*$(\mathtt{MirrorCR}_n) \geq n - 1$.

Proof. We first show the unsatisfiability of the matrix. Assume otherwise. Let σ be a satisfying assignment for $\mathfrak{C}(\mathtt{MirrorCR}_n)$. We can assume that σ is a total assignment. W.l.o.g. let $u \in \sigma$. We distinguish two cases:

Case 1. For all $i \in \{1, \ldots, n\}$ there exists a $j \in \{1, \ldots, n\}$ such that $\bar{x}_{(i,j)} \in \sigma$. Then we need $\bar{a}_i \in \sigma$ for all $i = 1, \ldots, n$, which falsifies the clause $a_1 \vee \ldots \vee a_n$.

Case 2. There is an $i \in \{1, \ldots, n\}$ such that for all $j \in \{1, \ldots, n\}$ we have $x_{(i,j)} \in \sigma$. Then we need $b_j \in \sigma$ for all $j = 1, \ldots, n$, which falsifies the clause $\bar{b}_1 \vee \ldots \vee \bar{b}_n$.

In each case we can conclude that it is not possible to construct a satisfying assignment for $\mathfrak{C}(\mathtt{MirrorCR}_n)$.

We now prove gauge($\mathtt{MirrorCR}_n$) $\geq n-1$.

Since $\mathtt{MirrorCR}_n$ contains no X-clauses as axioms, we have to resolve over some a_i or b_j somehow. Obviously, it is not possible to resolve $x_{(i,j)} \vee u \vee a_i$ and $x_{(i,j)} \vee \bar{u} \vee \bar{a}_i$ or $\bar{x}_{(i,j)} \vee \bar{u} \vee b_j$ and $\bar{x}_{(i,j)} \vee u \vee \bar{b}_j$. That means we have to use the other axioms. Because of the symmetry, we can assume that we use the clause $\bar{a}_1 \vee \ldots \vee \bar{a}_n$ somehow. Then we have to get rid of all $\bar{a}_i$. This can be done via the clauses $x_{(i,j)} \vee u \vee a_i$, or we use the clause $a_1 \vee \ldots \vee a_n$. However, to use the latter clause we have to get rid of at least $n-1$ different a_i in another way first, which is only possible with the aid of the clauses $x_{(i,j)} \vee \bar{u} \vee \bar{a}_i$. We conclude that we will pile up at least $n-1$ different X-literals. □

Applying Theorem 5.17 we infer:

Corollary 6.15. $\mathtt{MirrorCR}_n$ *needs exponential-size fully reduced primitive* Q-Res *refutations.*

All we have to do now is to show that all $\mathsf{QCDCL}^{\mathsf{PL\ w/o\ Cubes}}$ refutations of $\mathtt{MirrorCR}_n$ are primitive. Then the gauge lower bound applies. We will show that for a non-primitive refutation of $\mathtt{MirrorCR}_n$ we would need to decide literals by Pure Literal Elimination, and before each Pure Literal Elimination we have to perform another one, which is a contradiction.

Proposition 6.16. *From each* $\mathsf{QCDCL}^{\mathsf{PL\ w/o\ Cubes}}$ *refutation of* $\mathtt{MirrorCR}_n$ *we can extract a fully reduced primitive* Q-Res *refutation of the same size.*

Proof. Let ι be a $\mathsf{QCDCL}^{\mathsf{PL\ w/o\ Cubes}}$ refutation of $\mathtt{MirrorCR}_n$. We will show that $\mathfrak{R}(\iota)$ is primitive.

Assume not. Then by Proposition 6.4 there exists a trail $\mathcal{T}$ in ι such that there is an X-literal $x \in \mathcal{T}$ and a U-literal $v \in \mathcal{T}$ with $v <_\mathcal{T} x$ and v is not a regular decision literal. Let us say that $\mathrm{var}(x) = x_{(k,m)}$ for some $k, m \in \{1, \ldots, n\}$.

That means we have decided v (which is either u or $\bar{u}$) out of order via Pure Literal Elimination. We show that this is not possible before we have assigned all X-literals.

Claim 1. There is a T-literal t_1 such that $t_1 <_\mathcal{T} v <_\mathcal{T} x$.

W.l.o.g. let $v = \bar{u}$. We need to satisfy the clauses $\bar{x}_{(i,j)} \vee \bar{u} \vee b_j$ and $x_{(i,j)} \vee \bar{u} \vee \bar{a}_i$ for each $i, j \in \{1, \ldots, n\}$ without assigning u. Since we want to propagate x later, we cannot assign the X-variable $x_{(k,m)}$ in order to satisfy these clauses. That means we need to set b_m to true and a_k to false. If we set $t_1 := b_m$, then we get $t_1 <_\mathcal{T} v <_\mathcal{T} x$.

Claim 2. For each T-literal t_j with $t_j <_\mathcal{T} v <_\mathcal{T} x$ there is another T-literal t_{j+1} such that $t_{j+1} <_\mathcal{T} t_j <_\mathcal{T} v <_\mathcal{T} x$.

Because of $t_j <_\mathcal{T} v$, the T-literal t_j cannot be a regular decision. Either t_j was decided as a pure literal, or it was propagated. If it was a pure literal, then we needed to satisfy one of the clauses $\bar{a}_1 \vee \ldots \vee \bar{a}_n$, $\bar{b}_1 \vee \ldots \vee \bar{b}_n$, $a_1 \vee \ldots \vee a_n$ or $b_1 \vee \ldots \vee b_n$. This is only possible if we assigned another T-literal t_{j+1} before, hence $t_{j+1} <_\mathcal{T} t_j <_\mathcal{T} v <_\mathcal{T} x$. However, if t_j was propagated, then there exists the antecedent clause $F := \mathrm{ante}_\mathcal{T}(t_j)$. Due to the XT-property, F cannot be unit. Then there is another literal $t_j \neq \ell \in F$. Because the formula only contains one U-variable, ℓ can only be an X- or a T-literal.

Again, by the XT-property, F cannot be an XT-clause and therefore ℓ has to be a T-literal, which needs to be falsified by the current trail. Therefore, if we set $t_{j+1} := \bar{\ell}$, we get $t_{j+1} <_{\mathcal{T}} t_j <_{\mathcal{T}} v <_{\mathcal{T}} x$.

We proved that $\mathfrak{R}(\iota)$ has to be primitive, otherwise the trail $\mathcal{T}$ would contain infinitely many T-literals t_j. □

Combining Proposition 6.14 and 6.16 with the fact that Cube Learning will not make any difference on $\mathtt{MirrorCR}_n$ (due to the unsatisfiability of its matrix), we conclude:

Corollary 6.17. $\mathtt{MirrorCR}_n$ *requires* $\mathsf{QCDCL}^{\text{w/o Cubes}}$, QCDCL, $\mathsf{QCDCL}^{\text{Pl w/o Cubes}}$ *and* $\mathsf{QCDCL}^{\text{Pl}}$ *refutations of exponential size.*

Next we embed this formula into a new QCNF $\mathtt{PLTrap}_n$.

Definition 6.18. *The QCNF* $\mathtt{PLTrap}_n$ *has the prefix*

$$\exists y, x_{(1,1)}, \ldots, x_{(n,n)} \forall u \exists a_1, \ldots, a_n, b_1, \ldots, b_n, a, b.$$

Its matrix contains all clauses from $\mathtt{MirrorCR}_n$ *and additionally* $(y \vee a)$, $(\bar{a} \vee b)$, $(\bar{a} \vee \bar{b})$, $(a \vee b)$, *and* $(a \vee \bar{b})$.

Proposition 6.19. $\mathtt{PLTrap}_n$ *needs exponential-size* $\mathsf{QCDCL}^{\text{Pl w/o Cubes}}$ *and* $\mathsf{QCDCL}^{\text{Pl}}$ *refutations.*

Proof. Because $\mathfrak{C}(\mathtt{PLTrap}_n)$ contains $\mathfrak{C}(\mathtt{MirrorCR}_n)$, which is unsatisfiable, $\mathfrak{C}(\mathtt{PLTrap}_n)$ is unsatisfiable, as well. Therefore Cube Learning will never be applied and it suffices to consider $\mathsf{QCDCL}^{\text{Pl w/o Cubes}}$ refutations.

Let ι be a $\mathsf{QCDCL}^{\text{Pl w/o Cubes}}$ refutation of $\mathtt{PLTrap}_n$. We will show that each trail of ι can only contain literals from $\mathtt{MirrorCR}_n$ or y. Then ι can be interpreted as a $\mathsf{QCDCL}^{\text{Pl w/o Cubes}}$ refutation of $\mathtt{MirrorCR}_n$ where the only difference is the assignment of y, which does not affect Clause Learning in any form. Then the result follows by Corollary 6.17.

In each $\mathsf{QCDCL}^{\text{Pl w/o Cubes}}$ trail, we will set y to true due to Pure Literal Elimination. That means the clause $y \vee a$ will never become the unit clause (a).

After this, we have to assign the variables from $\mathtt{MirrorCR}_n$. We will show that for each trail $\mathcal{T}$ in ι we have $\{a, \bar{a}, b, \bar{b}\} \cap \mathcal{T} = \emptyset$.

First of all, it is obvious that Pure Literal Elimination of a or b is impossible at any time due to the four clauses $\bar{a} \vee b$, $a \vee b$, $\bar{a} \vee \bar{b}$ and $a \vee \bar{b}$. In fact, if, for example, we would like to make b pure, then we have to satisfy the clauses $\bar{a} \vee \bar{b}$ and $a \vee \bar{b}$, which cannot be done without setting b to false.

Next, let us assume that there is some literal $\ell \in \{a, \bar{a}, b, \bar{b}\}$ that was propagated in some trail $\mathcal{T}$ in ι. In particular, let $\mathcal{T}$ be the first trail in which we propagated a literal $\ell \in \{a, \bar{a}, b, \bar{b}\}$. Since $y \vee a$ can never be used as an antecedent clause for a, we have $\mathrm{ante}_{\mathcal{T}}(\ell) \in \{\bar{a} \vee b, a \vee b, \bar{a} \vee \bar{b}, a \vee \bar{b}\}$. But then we would need another $\ell \neq \ell' \in \{a, \bar{a}, b, \bar{b}\}$ with $\ell' \in \mathcal{T}$ and $\ell' <_{\mathcal{T}} \ell$. If we suppose that ℓ was the first propagation of a literal from $\{a, \bar{a}, b, \bar{b}\}$, then we conclude that ℓ' has to be a regular decision.

We will now argue that we get a contradiction if there is a trail $\mathcal{T}$ in ι in which we have decided a literal $\ell' \in \{a, \bar{a}, b, \bar{b}\}$. Because the decisions are level-ordered, there exists $v \in \{u, \bar{u}\}$ with $v \in \mathcal{T}$ and $v <_{\mathcal{T}} \ell'$. We can only decide v if we have assigned all existential literals left of v. In particular, for each $i, j = 1, \ldots, n$ there is a literal $\ell_{(i,j)} \in \{x_{(i,j)}, \bar{x}_{(i,j)}\}$ with $\ell_{(i,j)} \in \mathcal{T}$ and $\ell_{(i,j)} <_{\mathcal{T}} v$. We now distinguish two cases.

Case 1. For all $i \in \{1, \ldots, n\}$ there exists a $j \in \{1, \ldots, n\}$ with $\ell_{(i,j)} = \bar{x}_{(i,j)}$.

Then if $v = u$, we will gain unit clauses $(\bar{a}_i)$ for $i = 1, \ldots, n$ from the clauses $x_{(i,j)} \vee \bar{u} \vee \bar{a}_i$, which can be used for unit propagations that lead to a conflict in the clause $a_1 \vee \ldots \vee a_n$. Otherwise, if $v = \bar{u}$, then we will get unit clauses (a_i) from the clauses $x_{(i,j)} \vee u \vee a_i$ and a conflict in $\bar{a}_1 \vee \ldots \vee \bar{a}_n$.

Case 2. There exists an $i \in \{1, \ldots, n\}$ such that for all $j \in \{1, \ldots, n\}$ it holds $\ell_{(i,j)} = x_{(i,j)}$.

This case is analogous to Case 1 with unit clauses (b_j) (resp. $(\bar{b}_j)$) and a conflict in $\bar{b}_1 \vee \ldots \vee \bar{b}_n$ (resp. $b_1 \vee \ldots \vee b_n$).

In each case we will get a conflict in some clause. That means the trail $\mathcal{T}$ would run into a conflict before we would have the chance to decide ℓ'. That shows that ℓ' cannot be decided at any point. We conclude that no trail from ι can contain a literal from $\{a, \bar{a}, b, \bar{b}\}$. □

Not having to use Pure Literal Elimination, we can construct short proofs of $\mathtt{PLTrap}_n$ by focusing on the new clauses over a, b.

Proposition 6.20. $\mathtt{PLTrap}_n$ *has polynomial-size* $\mathsf{QCDCL}^{\text{w/o Cubes}}$ *refutations.*

Proof. The shortest refutation only consists of two trails. We start with

$$\mathcal{T} := (\bar{\mathbf{y}}, a, b, \bot)$$

with

$$\begin{aligned} \text{ante}_{\mathcal{T}}(a) &= y \vee a \\ \text{ante}_{\mathcal{T}}(b) &= \bar{a} \vee b \\ \text{ante}_{\mathcal{T}}(\bot) &= \bar{a} \vee \bar{b}. \end{aligned}$$

We resolve over b and learn the unit clause $(\bar{a})$.

The final trail looks as follows:

$$\mathcal{U} := (\bar{a}, b, \bot)$$

with

$$\begin{aligned} \text{ante}_{\mathcal{U}}(\bar{a}) &= (\bar{a}) \\ \text{ante}_{\mathcal{U}}(b) &= a \vee b \\ \text{ante}_{\mathcal{U}}(\bot) &= a \vee \bar{b}, \end{aligned}$$

from which we can learn the empty clause by resolving over everything. □

We conclude that Pure Literal Elimination is advantageous for $\mathtt{Eq}_n$ (cf. Corollary 6.6 and Proposition 6.12), but disadvantageous for $\mathtt{PLTrap}_n$ (cf. Proposition 6.19 and 6.20). Therefore we obtain:

Theorem 6.21. *It holds the following:*

(i) $\mathsf{QCDCL}^{\mathsf{PL\ w/o\ Cubes}}$ *and* $\mathsf{QCDCL}^{\mathsf{w/o\ Cubes}}$ *are incomparable.*

(ii) $\mathsf{QCDCL}^{\mathsf{PL}}$ *and* $\mathsf{QCDCL}^{\mathsf{w/o\ Cubes}}$ *are incomparable.*

6.3 Cube Learning vs. Pure Literal Elimination

As shown in Section 6.2, Cube Learning improves QCDCL, while adding Pure Literal Elimination leads to incomparable systems. Thus it is natural to directly compare Cube Learning and Pure Literal Elimination. Because of the results above, we cannot use $\mathtt{Eq}_n$ for a potential separation. However, we can modify the QBFs such that they remain easy for $\mathsf{QCDCL}^{\mathsf{PL\ w/o\ Cubes}}$, while eliminating the benefits from Cube Learning.

Definition 6.22. *The QCNF* $\mathtt{TwinEq}_n$ *has the prefix*

$$\exists x_1, \ldots, x_n \forall u_1, \ldots, u_n, w_1, \ldots, w_n \exists t_1, \ldots, t_n.$$

Its matrix contains the clauses from $\mathtt{Eq}_n$ *together with* $x_i \vee w_i \vee t_i$ *and* $\bar{x}_i \vee \bar{w}_i \vee t_i$ *for* $i \in [n]$.

The main idea of this twin construction is to ensure that all potential cubes consist of at least two universal variables. We can do the same construction for other QCNFs. Obviously, $\mathtt{TwinEq}_n$ fulfils the XT-property. We compute $\mathrm{gauge}(\mathtt{TwinEq}_n) = n$ and hence infer by Theorem 5.17:

Proposition 6.23. *Fully reduced primitive* Q-Res *refutations of* $\mathtt{TwinEq}_n$ *have exponential size.*

Proof. We need to show $\mathrm{gauge}(\mathtt{TwinEq}_n) = n$, then the result follows by Theorem 5.17.

Since we have to resolve over T somehow, we have to use the clause $\bar{t}_1 \vee \ldots \vee \bar{t}_n$. Hence, we have to resolve over each t_i at least once, and therefore we will pile up x_i or $\bar{x}_i$ in each resolution step due to the XUT-axioms. □

We show that QCDCL refutations of $\mathtt{TwinEq}_n$ are primitive by proving that it is impossible to propagate U-literals before having assigned all X-literals.

Proposition 6.24. *Each* QCDCL *refutation of* $\mathtt{TwinEq}_n$ *has at least exponential size.*

Proof. We will prove that from each QCDCL refutation of $\mathtt{TwinEq}_n$ we can extract a fully reduced primitive Q-Res refutation of the same size. Let ι be a QCDCL refutation of $\mathtt{TwinEq}_n$. We will show that $\mathfrak{R}(\iota)$ is primitive.

Assume not. Then by Proposition 6.4 there exists a trail $\mathcal{T}$ in ι such that there is an X-literal $x \in \mathcal{T}$ and a U-literal $u \in \mathcal{T}$ with $u <_{\mathcal{T}} x$. Also, u cannot be a regular decision in $\mathcal{T}$.

Hence, we have propagated u before x. Universal propagation can only be performed via cubes. Let us now consider how the initial cubes from $\mathtt{TwinEq}_n$ look like.

Assume that the cube A is a (not necessarily total) assignment that satisfies the matrix of $\mathtt{TwinEq}_n$. We have to satisfy the clause $\bar{t}_1 \vee \ldots \vee \bar{t}_n$, hence there is a $j \in \{1, \ldots, n\}$ with $\bar{t}_j \in A$. Then we also have to satisfy the four clauses

$$x_j \vee u_j \vee t_j$$
$$\bar{x}_j \vee \bar{u}_j \vee t_j$$
$$x_j \vee w_j \vee t_j$$
$$\bar{x}_j \vee \bar{w}_j \vee t_j.$$

That means x_j has to appear in some polarity in A, say $x_j \in A$. But then we need to set both u_j and w_j to false, thus $\bar{u}_j, \bar{w}_j \in A$.

We conclude that each (reduced) cube has to contain one of the subcubes

$$x_j \wedge \bar{u}_j \wedge \bar{w}_j$$
$$\bar{x}_j \wedge u_j \wedge w_j$$

for some $j \in \{1, \ldots, n\}$. This also causes that none of these cubes are resolvable.

We observe that all cubes that can be used for universal unit propagation contain at least two universal literals. Since we needed one of these cubes as antecedent cube of some universal literal in our trail $\mathcal{T}$, we would have needed to decide or propagate another universal literal before. Having only finitely many universal literals, we would have needed to decide one universal literal before propagating x, which is a contradiction to the fact that decisions have to be level-ordered.

This shows that $\mathfrak{R}(\iota)$ is indeed primitive. □

Having shown that $\mathtt{TwinEq}_n$ is hard for QCDCL, it remains to prove that it is easy for $\mathsf{QCDCL}^{\mathsf{PL\ w/o\ Cubes}}$.

Proposition 6.25. $\mathtt{TwinEq}_n$ *has polynomial-size* $\mathsf{QCDCL}^{\mathsf{PL\ w/o\ Cubes}}$ *refutations.*

Proof. The proof is similar to the one in Proposition 6.12, except one change: Each time some universal literal is getting pure, say u_i, then also w_i becomes pure as well. That means each time we decide some u_i (resp. $\bar{u}_i$) in the trail by Pure Literal Elimination, we also have to do the same to w_i (resp. $\bar{w}_i$) in the next decision level. However, this does not affect anything concerning unit propagation or Clause Learning.

To give an example: The trail $\mathcal{U}_{n-1}$ from Proposition 6.12 will now look like

$$\mathcal{U}_{n-1} := (\mathbf{x_1}; \underline{\mathbf{u_1}}, t_1; \underline{\mathbf{w_1}}; \ldots; \mathbf{x_{n-2}}; \underline{\mathbf{u_{n-2}}}, t_{n-2}; \underline{\mathbf{w_{n-2}}}; \mathbf{x_{n-1}}; \underline{\mathbf{u_{n-1}}}, t_{n-1}, \bar{t}_n, x_n, \bot).$$

□

For the other separation we use $\mathtt{PLTrap}_n$, which is hard for $\mathsf{QCDCL}^{\mathsf{PL\ w/o\ Cubes}}$, but still easy for QCDCL by Proposition 6.9. Therefore we conclude:

Theorem 6.26. QCDCL *is incomparable to* $\mathsf{QCDCL}^{\mathsf{PL\ w/o\ Cubes}}$.

We have seen earlier that the QCDCL system including Pure Literal Elimination is incomparable to the system without. Now we will prove that this incomparability still holds with Cube Learning turned on. By Proposition 6.9, we obtain that $\mathsf{QCDCL}^{\mathsf{PL}}$ p-simulates $\mathsf{QCDCL}^{\mathsf{PL\ w/o\ Cubes}}$. Therefore we get from Proposition 6.25:

Corollary 6.27. $\mathtt{TwinEq}_n$ *has poly-size* $\mathsf{QCDCL}^{\mathsf{PL}}$ *refutations.*

Since $\mathtt{TwinEq}_n$ is hard for QCDCL, this gives us the first separation between $\mathsf{QCDCL}^{\mathsf{PL}}$ and QCDCL. The other direction can be shown with $\mathtt{PLTrap}_n$.

Proposition 6.28. $\mathtt{PLTrap}_n$ *has poly-size* QCDCL *refutations.*

Proof. The short proofs in QCDCL follow from Propositions 6.9 and 6.20. □

Hence we get:

Theorem 6.29. $\mathsf{QCDCL}^{\mathsf{PL}}$ *and* QCDCL *are incomparable.*

We now consider the relation between $\mathsf{QCDCL}^{\mathsf{PL}}$ and $\mathsf{QCDCL}^{\mathsf{PL\ w/o\ Cubes}}$. We introduce another modification of $\mathtt{Eq}_n$, which we call $\mathtt{BulkyEq}_n$, where we add two clauses.

Definition 6.30. *The QCNF* $\mathtt{BulkyEq}_n$ *is obtained from* $\mathtt{Eq}_n$ *by adding the clauses* $u_1 \vee \ldots \vee u_n \vee t_1 \vee \ldots \vee t_n$ *and* $\bar{u}_1 \vee \ldots \vee \bar{u}_n \vee t_1 \vee \ldots \vee t_n$ *to the matrix.*

As $\mathtt{Eq}_n$, this formula fulfils the XT-property and has a high gauge value ($\geq n-1$). By Theorem 5.17 we infer that $\mathtt{BulkyEq}_n$ needs exponential-size fully reduced primitive Q-Res refutations. Similarly to $\mathtt{MirrorCR}_n$, we can then show that Pure Literal Elimination does not shorten proofs because of the two additional 'bulky' clauses that prevent pure literals to occur early in trails. Therefore $\mathtt{BulkyEq}_n$ is hard for $\mathsf{QCDCL}^{\mathsf{PL\ w/o\ Cubes}}$. On the other hand, we can explicitly construct short proofs in $\mathsf{QCDCL}^{\mathsf{PL}}$. Therefore we get:

Proposition 6.31. *The formula* $\mathtt{BulkyEq}_n$ *has poly-size* $\mathsf{QCDCL}^{\mathsf{PL}}$ *refutations, but needs exponential-size* $\mathsf{QCDCL}^{\mathsf{PL\ w/o\ Cubes}}$ *refutations.*

Proof. Part 1. $\mathtt{BulkyEq}_n$ needs exponential-size $\mathsf{QCDCL}^{\mathsf{PL\ w/o\ Cubes}}$ refutations.

We first prove gauge($\mathtt{BulkyEq}_n$) $\geq n-1$. To derive an X-clause, we have to use $\bar{t}_1 \vee \ldots \vee \bar{t}_n$ somehow. That means we have to resolve over each t_i. We can resolve with $u_1 \vee \ldots \vee u_n \vee t_1 \vee \ldots \vee t_n$ or $\bar{u}_1 \vee \ldots \vee \bar{u}_n \vee t_1 \vee \ldots \vee t_n$ only after we have resolved away at least $n-1$ different T-variables otherwise. That means we have pile up at least $n-1$ different X-literals by using the clauses $x_i \vee u_i \vee t_i$ or $\bar{x}_i \vee \bar{u}_i \vee t_i$. Hence gauge($\mathtt{BulkyEq}_n$) $\geq n-1$.

We will now prove that from each $\mathsf{QCDCL}^{\mathsf{PL\ w/o\ Cubes}}$ refutation of $\mathtt{BulkyEq}_n$ we can extract a fully reduced primitive Q-Res refutation of the same size. Let ι be a $\mathsf{QCDCL}^{\mathsf{PL\ w/o\ Cubes}}$ refutation of $\mathtt{BulkyEq}_n$. We will show that $\mathfrak{R}(\iota)$ is primitive.

Assume not. Then by Proposition 6.4 there exists a trail $\mathcal{T}$ in ι such that there is an X-literal $x \in \mathcal{T}$ and a U-literal $u \in \mathcal{T}$ with $u <_\mathcal{T} x$ and u is not a regular decision literal.

Since Cube Learning is disabled, this universal literal u had to be decided by Pure Literal Elimination. We will show that Pure Literal Elimination of the universal literal u before deciding or propagating all X-variables is not possible. Define $M := \{u_i, \bar{u}_i, t_i, \bar{t}_i : i = 1, \ldots, n\}$.

Claim 1. There exists some $\ell_1 \in M$ such that $\ell_1 <_\mathcal{T} u <_\mathcal{T} x$.

In order to make u pure, we have to satisfy one of the clauses $u_1 \vee \ldots \vee u_n \vee t_1 \vee \ldots \vee t_n$ or $\bar{u}_1 \vee \ldots \vee \bar{u}_n \vee t_1 \vee \ldots \vee t_n$. In particular, we need some $\ell_1 \in M$ with $\ell_1 <_\mathcal{T} u <_\mathcal{T} x$.

Claim 2. For each $\ell_j \in M$ with $\ell_j <_\mathcal{T} u <_\mathcal{T} x$ there exists some $\ell_{j+1} \in M$ such that $\ell_{j+1} <_\mathcal{T} \ell_j <_\mathcal{T} u <_\mathcal{T} x$

If ℓ_j was decided via Pure Literal Elimination, we can use a similar argument as in Claim 1 (now we have satisfy one of the three clauses $u_1 \vee \ldots \vee u_n \vee t_1 \vee \ldots \vee t_n$, $\bar{u}_1 \vee \ldots \vee \bar{u}_n \vee t_1 \vee \ldots \vee t_n$ or $\bar{t}_1 \vee \ldots \vee \bar{t}_n$) and conclude that we need some $\ell_{j+1} \in M$ with $\ell_{j+1} <_\mathcal{T} \ell_j <_\mathcal{T} u <_\mathcal{T} x$. However, if ℓ_j was not decided as a pure literal, then it has to be a T-literal that was propagated. Note that we cannot have decided ℓ_j regularly because of $\ell_j <_\mathcal{T} x$ and $\ell_j <_\mathcal{T} u$. That means there is an antecedent clause $F := \text{ante}_\mathcal{T}(\ell_j)$. Due to the XT-property, F cannot be a unit clause. That means there is another literal $\ell_j \neq \ell \in F$. If ℓ is a U- or a T-literal, then we can set $\ell_{j+1} := \bar{\ell}$. If ℓ is an X-literal, then there is at least one U-literal $v \in F$, again because of the XT-property. Then we can set $\ell_{j+1} := \bar{v}$.

We have proven that if $\mathfrak{R}(\iota)$ is not primitive, then $\mathcal{T}$ has to contain an endless number of literals ℓ_j, which is obviously not possible since the formula only consists of finitely many variables. That means $\mathfrak{R}(\iota)$ has to be primitive.

Part 2. $\texttt{BulkyEq}_n$ has polynomial-size $\mathsf{QCDCL}^{\mathsf{PL}}$ refutations.

We start with the learning of exactly two cubes: $x_1 \wedge \bar{u}_1$ and $\bar{x}_1 \wedge u_1$. We do this via the following two trails:

$$\mathcal{T} := (\mathbf{x_1}; \ldots; \mathbf{x_n}; \bar{\mathbf{u}}_\mathbf{1}; \ldots; \bar{\mathbf{u}}_\mathbf{n}; \bar{\mathbf{t}}_\mathbf{1}; \mathbf{t_2}; \ldots; \mathbf{t_n})$$
$$\mathcal{T}' := (\bar{\mathbf{x}}_\mathbf{1}; \ldots; \bar{\mathbf{x}}_\mathbf{n}; \mathbf{u_1}; \ldots; \mathbf{u_n}; \bar{\mathbf{t}}_\mathbf{1}; \mathbf{t_2}; \ldots; \mathbf{t_n})$$

Unfortunately we cannot continue learning the other cubes as in Proposition 6.10 since this will be blocked by Pure Literal Elimination. This is because the propagation of t_1 satisfies both bulky clauses, which can then no longer prevent the Pure Literal Elimination of u_i once x_i has been decided. However, we can use this effect to our advantage by simulating the missing cubes in this way.

Let us now start the learning of the clauses L_i and R_i for $i = 2, \ldots, n-1$ from the proof of Proposition 6.10.

We begin by constructing the following trail:

$$\mathcal{U}_{n-1} := (\mathbf{x_1}, u_1, t_1; \mathbf{x_2}; \underline{\mathbf{u_2}}, t_2, \ldots, \mathbf{x_{n-2}}; \underline{\mathbf{u_{n-2}}}, t_{n-2};$$
$$\mathbf{x_{n-1}}, \underline{\mathbf{u_{n-1}}}, t_{n-1}, \bar{t}_n, x_n, \bot)$$

with the same antecedent constraints as in Proposition 6.10 (except of the literals $u_2, \dots, u_{n-2}$) and the same learned clause L_{n-1}. Analogously we can learn R_{n-1}.

We go on with the trails $\mathcal{U}_{n-2}, \dots, \mathcal{U}_2$ in the same way as in Proposition 6.10 where we learn $L_{n-2}, \dots, L_2$, except that the literals $u_2, \dots, u_{i-1}$ in $\mathcal{U}_{i-1}$ are now pure literals and not propagated via cubes. However, this does not affect the Clause Learning process in any aspect. The same is obviously true for the analogous trails in which we learn $R_{n-2}, \dots, R_2$.

We finish the proof with the last two trails $\mathcal{U}_1$ and $\mathcal{U}_1'$ exactly as in Proposition 6.10. □

As for the systems without Pure Literal Elimination, we get:

Theorem 6.32. $\mathsf{QCDCL}^{\mathsf{PL}}$ *is exponentially stronger than* $\mathsf{QCDCL}^{\mathsf{PL\ w/o\ Cubes}}$.

6.4 Comparison with Q-Res

In [14] the incomparability of Q-Res and $\mathsf{QCDCL}^{\mathsf{w/o\ Cubes}}$ was shown. We now lift this to the other QCDCL variants introduced here. For one separation, we can use the QCNF $\mathtt{QParity}_n$ from [17], which have short $\mathsf{QCDCL}^{\mathsf{w/o\ Cubes}}$ refutations. These formulas have prefix $\exists x_1 \dots x_n \forall u \exists t_1 \dots t_n$ and clauses

$$\begin{gathered} x_1 \vee \bar{t}_1,\ \ \bar{x}_1 \vee t_1,\ \ u \vee t_n,\ \ \bar{u} \vee \bar{t}_n, \\ x_i \vee t_{i-1} \vee \bar{t}_i,\ \ x_i \vee \bar{t}_{i-1} \vee t_i, \\ \bar{x}_i \vee t_{i-1} \vee t_i,\ \ \bar{x}_i \vee \bar{t}_{i-1} \vee \bar{t}_i \quad \text{for } i \in \{2, \dots, n\}. \end{gathered}$$

Theorem 6.33. $\mathsf{QCDCL}^{\mathsf{w/o\ Cubes}}$, QCDCL, $\mathsf{QCDCL}^{\mathsf{PL\ w/o\ Cubes}}$ *and* $\mathsf{QCDCL}^{\mathsf{PL}}$ *are all incomparable to* Q-Res. *In detail, the QBFs* $\mathtt{QParity}_n$ *have polynomial-size refutations in* $\mathsf{QCDCL}^{\mathsf{w/o\ Cubes}}$, QCDCL, $\mathsf{QCDCL}^{\mathsf{PL\ w/o\ Cubes}}$, *and* $\mathsf{QCDCL}^{\mathsf{PL}}$, *but need exponential-size* Q-Res *refutations. On the other hand,* $\mathtt{MirrorCR}_n$ *have polynomial-size* Q-Res *refutations, but need exponential-size* $\mathsf{QCDCL}^{\mathsf{w/o\ Cubes}}$, QCDCL, $\mathsf{QCDCL}^{\mathsf{PL\ w/o\ Cubes}}$, *and* $\mathsf{QCDCL}^{\mathsf{PL}}$ *refutations.*

Proof. Claim 1. $\mathtt{QParity}_n$ has polynomial-size $\mathsf{QCDCL}^{\mathsf{w/o\ Cubes}}$ and QCDCL refutations.

By Proposition 6.7, $\mathtt{QParity}_n$ has short $\mathsf{QCDCL}^{\mathsf{w/o\ Cubes}}$ refutations. And because of Proposition 6.9, the formula is easy for QCDCL, as well.

Claim 2. $\mathtt{QParity}_n$ has polynomial-size $\mathsf{QCDCL}^{\mathsf{PL\ w/o\ Cubes}}$ and $\mathsf{QCDCL}^{\mathsf{PL}}$ refutations.

We will show that we will never find pure literals while creating $\mathsf{QCDCL}^{\mathsf{PL\ w/o\ Cubes}}$ trails. In fact, the only way in making a literal ℓ pure is to create a unit clause (ℓ), which would immediately lead to the propagation of ℓ or a conflict.

For example, suppose the literal t_i is pure at some point in the trail. Then the clauses $x_i \vee t_{i-1} \vee \bar{t}_i$ and $\bar{x}_i \vee \bar{t}_{i-1} \vee \bar{t}_i$ must have been satisfied by the current assignment of the trail. Since we have not assigned t_i yet, we have to set either x_i to true and t_{i-1} to false, or x_i to false and t_{i-1} to true. In both cases we would obtain the unit clause (t_i) by applying this assignment to either $x_i \vee \bar{t}_{i-1} \vee t_i$ or $\bar{x}_i \vee t_{i-1} \vee t_i$.

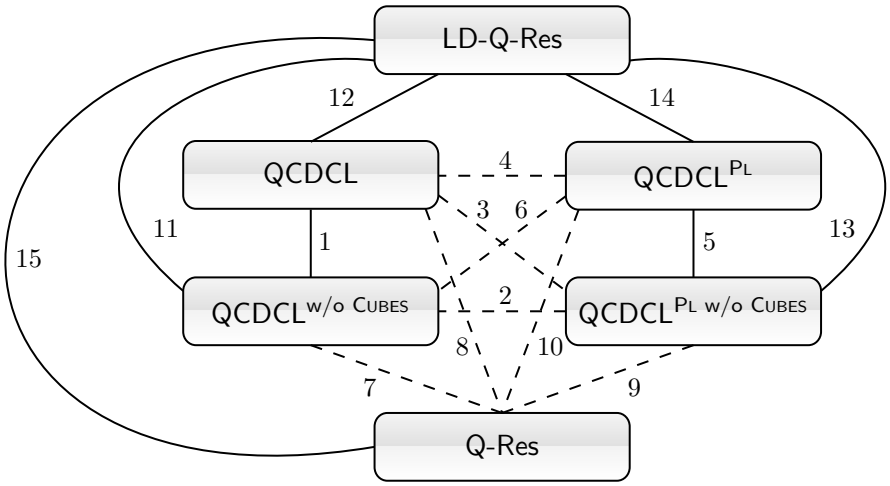


Figure 6.1: Hasse diagram of the simulation order of QCDCL proof systems. Solid lines represent p-simulations and exponential separations (where the system depicted above is the stronger one). Dashed lines represent separations in both directions (i.e., incomparability). Details of the simulations and separations are depicted in Tables 6.1 and 6.2.

The same holds for the universal variable u. For u or $\bar{u}$ to be pure, we need to set t_n to false or true. But then we would obtain the unit clause (u) or $(\bar{u})$, which would immediately lead to a conflict. We conclude that the polynomial-size QCDCL$^{\text{w/o Cubes}}$ refutation of $\texttt{QParity}_n$ is a QCDCL$^{\text{PL w/o Cubes}}$ refutation as well. And because QCDCL$^{\text{PL}}$ p-simulates QCDCL$^{\text{PL w/o Cubes}}$, $\texttt{QParity}_n$ is also easy for QCDCL$^{\text{PL}}$.

Claim 3. $\texttt{QParity}_n$ needs exponential-size Q-Res refutations.

This was already proven in [17].

Claim 4. $\texttt{MirrorCR}_n$ needs refutations of exponential size in all four QCDCL variants QCDCL$^{\text{w/o Cubes}}$, QCDCL, QCDCL$^{\text{PL w/o Cubes}}$ and QCDCL$^{\text{PL}}$.

This is the result of Corollary 6.17.

Claim 5. $\texttt{MirrorCR}_n$ has polynomial-size Q-Res refutations.

This follows directly from the fact that $\texttt{MirrorCR}_n$ extends the original QCNF $\texttt{CR}_n$, which has polynomial-size Q-Res refutations [64]. We will just ignore the clauses that are not contained in $\texttt{CR}_n$. □

Since LD-Q-Res simulates all four QCDCL variants on false formulas by definition, and because all of them are incomparable to Q-Res, which is exponentially weaker than LD-Q-Res, we conclude:

Corollary 6.34. QCDCL$^{\text{w/o Cubes}}$, QCDCL, QCDCL$^{\text{PL w/o Cubes}}$ *and* QCDCL$^{\text{PL}}$ *are all p-simulated by but exponentially weaker than* LD-Q-Res.

	Simulation	Separation			
No	Theorem	Formula	easy for	hard for	Theorem
1	Prop 6.9	$\mathtt{Eq}_n$	QCDCL	$\mathsf{QCDCL}^{\mathsf{w/o\ Cubes}}$	Thm 6.11
5	Prop 6.9	$\mathtt{BulkyEq}_n$	$\mathsf{QCDCL}^{\mathsf{PL}}$	$\mathsf{QCDCL}^{\mathsf{PL\ w/o\ Cubes}}$	Thm 6.32
11	by Def.	$\mathtt{CR}_n$	LD-Q-Res	$\mathsf{QCDCL}^{\mathsf{w/o\ Cubes}}$	Cor 6.6, [64]
12	by Def.	$\mathtt{MirrorCR}_n$	LD-Q-Res	QCDCL	Thm 6.33
13	by Def.	$\mathtt{MirrorCR}_n$	LD-Q-Res	$\mathsf{QCDCL}^{\mathsf{PL\ w/o\ Cubes}}$	Thm 6.33
14	by Def.	$\mathtt{MirrorCR}_n$	LD-Q-Res	$\mathsf{QCDCL}^{\mathsf{PL}}$	Thm 6.33
15	by Def.	$\mathtt{Eq}_n$	LD-Q-Res	Q-Res	[11]

Table 6.1: P-simulations and separations of proof systems from Figure 6.1 (i.e. the solid lines).

All relations between proof systems relevant for this chapter are depicted in Figure 6.1 and Tables 6.1 and 6.2.

	Separation			
No	Formula	easy for	hard for	Theorem
2	$\mathtt{PLTrap}_n$	QCDCL$^{\text{w/o Cubes}}$	QCDCL$^{\text{P}_\text{L}\text{ w/o Cubes}}$	Prop 6.19, Prop 6.20
	$\mathtt{Eq}_n$	QCDCL$^{\text{P}_\text{L}\text{ w/o Cubes}}$	QCDCL$^{\text{w/o Cubes}}$	Prop 6.12, Cor 6.6
3	$\mathtt{PLTrap}_n$	QCDCL	QCDCL$^{\text{P}_\text{L}\text{ w/o Cubes}}$	Prop 6.19, Prop 6.20
	$\mathtt{TwinEq}_n$	QCDCL$^{\text{P}_\text{L}\text{ w/o Cubes}}$	QCDCL	Prop 6.24, Prop 6.25
4	$\mathtt{PLTrap}_n$	QCDCL	QCDCL$^{\text{P}_\text{L}}$	Prop 6.20, Prop 6.28
	$\mathtt{TwinEq}_n$	QCDCL$^{\text{P}_\text{L}}$	QCDCL	Prop 6.24, Prop 6.25
6	$\mathtt{PLTrap}_n$	QCDCL$^{\text{w/o Cubes}}$	QCDCL$^{\text{P}_\text{L}}$	Prop 6.20, Prop 6.28
	$\mathtt{Eq}_n$	QCDCL$^{\text{P}_\text{L}}$	QCDCL$^{\text{w/o Cubes}}$	Prop 6.12, Cor 6.6
7	$\mathtt{QParity}_n$	QCDCL$^{\text{w/o Cubes}}$	Q-Res	Thm 6.33, [17]
	$\mathtt{CR}_n$	Q-Res	QCDCL$^{\text{w/o Cubes}}$	Cor 6.6, [64]
8	$\mathtt{QParity}_n$	QCDCL	Q-Res	Thm 6.33, [17]
	$\mathtt{MirrorCR}_n$	Q-Res	QCDCL	Thm 6.33
9	$\mathtt{QParity}_n$	QCDCL	Q-Res	Thm 6.33, [17]
	$\mathtt{MirrorCR}_n$	Q-Res	QCDCL	Thm 6.33
10	$\mathtt{QParity}_n$	QCDCL	Q-Res	Thm 6.33, [17]
	$\mathtt{MirrorCR}_n$	Q-Res	QCDCL	Thm 6.33

Table 6.2: Separations between incomparable proof systems from Figure 6.1 (i.e. the dashed lines).

Chapter 7

Comparing Decision Policies

In the previous chapter, we have identified several exponential lower bounds for classic QCDCL variants as well as separations between these variants. The formulas that were used for separating the systems were designed in a way such that they can exploit potential weaknesses of QCDCL — with level-ordered decisions leading the way. In the next chapters, we aim to improve the strength of QCDCL as a proof system by adding modifications that are not yet common in practice. We will refer to these modifications as *policies* and separate them into three categories:

- Decision Policies that determine which variables are allowed for decisions whenever a decision has to be made.
- Reduction Policies that determine which variables are allowed or forced to be reduced during Unit Propagation.
- Propagation Policies that determine which variables are eligible for Unit Propagation itself.

In order to focus on the impact of these policies, we will only consider QCDCL systems in which Cube Learning is allowed, but Pure Literal Elimination is deactivated. Nevertheless, we want to point out that one could still take these two aspects into consideration and obtain four variants for each new variant introduced in this chapter.

In this chapter, we will start with decision policies as it is the most obvious way of improving the performance of QCDCL solvers. The gauge lower bound technique deeply relies on the fact that decisions had to be level-ordered in the classical setting. These level-ordered decisions in combination with the XT-property often lead to level-ordered trails whenever Cube Learning or Pure Literal Elimination were not able to have a beneficial impact (i.e. by assigning U-variables before all X-variables were assigned).

A naive approach would be allowing any kind of decision independent of the quantification order. Although we will demonstrate that it is not advisable to not have any rule which determines what variables are eligible for the next decision, we can show that the decision rule can be relaxed while still ensuring soundness and completeness of the corresponding QCDCL proof system.

Technically, the soundness of QCDCL systems with relaxed decision policies follows directly by the soundness of the base version QCDCL itself (cf. Theorem 4.9). However, for algorithms it is more important to prevent running into a loop at any point than to simply guarantee that there is a way (e.g. by making the right decisions) to prevent a loop.

In QCDCL, our goal is to make 'progress' in each run/trail. Thus, we have to ensure that we can always learn new clauses or cubes from a constructed trail. Since we want to work with QCDCL models that do not necessarily follow the prefix order for decision making, it is not guaranteed that we can even learn new constraints from each trail. We demonstrate that issue on the following example:

Example 7.1. *Consider the trail* $\mathcal{T} = (\mathbf{x}, \bot)$ *for the false QCNF*

$$\forall u \exists x \cdot (u \vee x) \wedge (u \vee \bar{x}) \wedge (\bar{u} \vee x) \wedge (\bar{u} \vee \bar{x}).$$

The only learnable clause is $ante_{\mathcal{T}}(\bot) = \bar{u} \vee \bar{x}$*, which is obviously already known.*

Because the conflict in this example occurred directly after a decision, we were not able to resolve clauses and therefore could not derive a new one. We will prove later that this is essentially the only situation in which the learning of a new constraint could get prevented, meaning that we are able always able to find new learnable clauses whenever we do not run into a conflict directly after a decision. We will refer to this sufficient condition as the *New Constraint Condition*.

Definition 7.2. *A trail* $\mathcal{T}$ *for a formula* Φ *fulfils the* New Constraint Condition *(*NCC *for short), if for each decision* d_i *the formula* $red(\Phi|_{\mathcal{T}[i,0]\cup\{d_i\}})$ *does not contain the empty clause or cube.*

While the NCC ensures that we will always be able to find new learnable clauses and cubes, it does not guarantee that these constraints are asserting (in contrast to the classic QCDCL variants, cf. Proposition 4.12) as shown in the next example.

Example 7.3. *Let*

$$\mathcal{U} := (\mathbf{x}, y; \mathbf{u}, \bar{z}, \bot)$$

be a trail for the false QCNF

$$\forall u \exists x, y, z \cdot (\bar{x} \vee y) \wedge (x \vee y) \wedge (u \vee \bar{y} \vee \bar{z}) \wedge (\bar{u} \vee \bar{y} \vee \bar{z}) \wedge (u \vee \bar{y} \vee z) \wedge (\bar{u} \vee \bar{y} \vee z).$$

There are two new clauses we could learn: $\bar{u} \vee \bar{y}$ *or* $\bar{u} \vee \bar{x}$*. None of the two can become unit after backtracking.*

The reason for not being able to find asserting learnable constraints cannot be worded as easily as the NCC. The fact that we can always find asserting learnable constraints whenever decisions are level-ordered lead to the approach to restrict the decision policy such that it is still more liberal than the classic level-ordered decision policy.

We introduce the idea of 'partially' level-ordered decisions by defining a policy in which only existential decisions are level-ordered, and another one in which only universal decisions are level-ordered. We will later show that the first one can guarantee at least asserting clauses, while the ladder guarantees asserting cubes.

We summarize these three new decision policies, whose corresponding QCDCL variants we will call QCDCL$^{\text{ANY}}$, QCDCL$^{\text{UNI-ANY}}$ and QCDCL$^{\text{EXI-ANY}}$. We formally define these variants via trail properties, for which we consider a general trail

$$\mathcal{T}_j = (p_{(0,1)}, \ldots, p_{(0,g_0)}; \mathbf{d_1}, p_{(1,1)}, \ldots, p_{(1,g_1)}; \ldots; \mathbf{d_r}, p_{(r,1)}, \ldots, p_{(r,g_r)} = \bot).$$

- QCDCL: For each decision d_i we have that $\text{lv}_{\Psi|_{\mathcal{T}[i,0]}}(d_i) = 1$. I.e., decisions are level-ordered.
- QCDCL$^{\text{ANY}}$: Decisions can be made arbitrarily as long as the NCC is fulfilled.
- QCDCL$^{\text{UNI-ANY}}$: An existential decision d_i can only be made if all universal variables that are quantified left of d_i were already assigned in $\mathcal{T}$. Universal decisions can be made in any order as long as the NCC is fulfilled.
- QCDCL$^{\text{EXI-ANY}}$: A universal decision d_i can only be made if all existential variables that are quantified left of d_i were already assigned in $\mathcal{T}$. Existential decisions can be made in any order as long as the NCC is fulfilled.

Note that the NCC not only restricts the way we can choose decisions, but also affect backtracking. In particular, we are not allowed to backtrack to a point directly right of a decision and immediately trigger a conflict. That means that whenever we want to check whether the NCC is fulfillable, we also have to specify a point in the trail to which we can backtrack without violating the condition.

The next result states simulations between systems, cf. Figure 7.1. They all follow by definition.

Proposition 7.4. *Each* QCDCL *proof is also a* QCDCL$^{\text{UNI-ANY}}$ *and* QCDCL$^{\text{EXI-ANY}}$ *proof, and each* QCDCL$^{\text{UNI-ANY}}$ *or* QCDCL$^{\text{EXI-ANY}}$ *proof is also a* QCDCL$^{\text{ANY}}$ *proof.*

7.1 Learning Asserting Constraints

As stated above, our goal is to provide decision policies for which it is impossible to run into a loop no matter which decisions were made. While in theory it suffices to learn new constraints that were not part of the formula before, it is standard practice to learn asserting constraints in a practical setting. We recall some facts of asserting constraints:

- An asserting constraint (with respect to a trail) is a learnable constraint for which there exists a backtracking point after which the constraint becomes unit (cf. Definition 4.10.

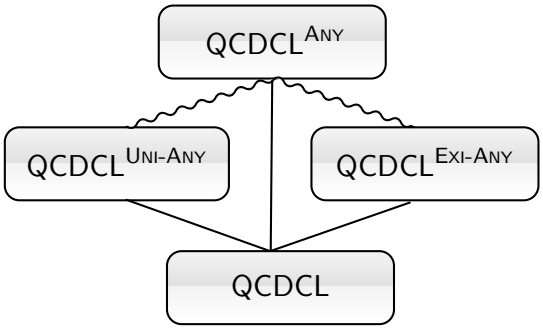


Figure 7.1: Hasse diagram of the simulation order of QCDCL proof systems. Solid lines represent p-simulations and exponential separations. Waved lines represent p-simulations, for which separations are not known.

- There is always at least one learnable asserting constraint in a trail of the base version QCDCL (cf. Proposition 4.12).
- Asserting constraints are always new (cf. Lemma 4.11).

We start by showing that at least one type of learnable asserting constraint does always exist in partially level-ordered QCDCL variants.

Proposition 7.5. *If $\mathcal{T}$ is a trail in a $\mathsf{QCDCL}^{\textsc{Uni-Any}}$ (resp. $\mathsf{QCDCL}^{\textsc{Exi-Any}}$) proof of a formula Φ, and if $\mathcal{T}$ runs into a conflict on a clause (resp. cube), then there exists a new asserting or empty constraint $C \in \mathfrak{L}(\mathcal{T})$.*

Furthermore, if C is non-empty, there exists a point $[i, j]$ in the trail to which we can backtrack after learning C such that the NCC continues to hold.

Proof. We will show the case for conflicts on clauses for $\mathsf{QCDCL}^{\textsc{Uni-Any}}$ proofs, the $\mathsf{QCDCL}^{\textsc{Exi-Any}}$ case is completely dual.

Let the trail $\mathcal{T}$ look like

$$\mathcal{T} = (p_{(0,1)}, \ldots, p_{(0,g_0)}; \mathbf{d_1}, p_{(1,1)}, \ldots, p_{(1,g_1)}; \ldots; \mathbf{d_r}, p_{(r,1)}, \ldots, p_{(r,g_r)}).$$

Then the sequence of learnable clauses is

$$\mathfrak{L}(\mathcal{T}) = (C_{(r,g_r)}, \ldots, C_{(r,1)}, \ldots, C_{(1,g_1)}, \ldots, C_{(1,1)}, C_{(0,g_0)}, \ldots, C_{(0,1)}).$$

We can assume that there exists at least one existential decision literal d_i such that $\bar{d}_i$ is contained in some $C \in \mathfrak{L}(\mathcal{T})$. Otherwise, the rightmost clause in $\mathfrak{L}(\mathcal{T})$ is empty since it contains negated decisions or universal literals only, which will be reduced to the empty clause ($\bot$).

Let $k \in \{1, \ldots, r\}$ be maximal such that an existential $\bar{d}_k$ is contained in some clause from $\mathfrak{L}(\mathcal{T})$. Let $p_{(\ell,m)} \in \mathcal{T}$ be the first propagated (non-empty) literal directly right of d_k in $\mathcal{T}$ and set $D := C_{(\ell,m)}$. Note that $p_{(\ell,m)}$ does not need to be on the same decision level as d_k. Such a $p_{(\ell,m)}$ must exist by the NCC. We will show that D is asserting.

We consider the trail $\mathcal{T}$ at the point $[k, 0]$, that means right before d_k was decided. We will prove that $E := \mathrm{red}^{\forall}_{\Phi}(D|_{\mathcal{T}[k,0]}) = (\bar{d}_k)$.

If there is an existential literal $\bar{d}_k \neq y \in E$, then $\bar{y}$ cannot have been assigned in $\mathcal{T}[k, 0]$, hence we have $d_k <_{\mathcal{T}} \bar{y}$. But that means $\bar{y}$ had to be a decision, otherwise it would have been resolved away during Clause Learning. But this is a contradiction to the maximality of k. We conclude that such a y cannot exist.

Let us now assume there is a universal literal $u \in E$. Then we need $u <_{\Phi} d_k$ since it was not reduced during Clause Learning. But $\mathcal{T}$ was a trail in a $\mathsf{QCDCL}^{\textsf{Uni-Any}}$ proof, hence $\mathrm{lv}_{\Phi|_{\mathcal{T}[k,0]}}(d_k) = 1$ and therefore $\bar{u} \in \mathcal{T}[k, 0]$. Then we get $u \notin E$, contradiction. Thus such a u cannot exist, and E is in fact a unit clause.

We can backtrack to the point $[k, 0]$ (i.e., before we made the decision d_k) and will not hurt the NCC since the only new clause we have learned can only propagate the non-empty literal $\bar{d}_k$.

At the end, we have to show that D is new. In fact, if D was already known, we would get a conflict directly after deciding d_k, which would violate the NCC. Thus, D must be a new clause. □

A similar result holds for the any-order model, albeit with the difference that we might not be able to learn asserting constraints. But at least we can guarantee to learn a new clause/cube.

Proposition 7.6. *If $\mathcal{T}$ is a trail in a $\mathsf{QCDCL}^{\textsf{Any}}$ proof for a formula Φ, that has run into a conflict or in which we assigned all variables, then $\mathfrak{L}(\mathcal{T})$ contains a new clause or cube C that is not contained in Φ.*

Further, if C is non-empty, there exists a point $[i, j]$ in the trail to which we can backtrack after learning C such that the NCC continues to hold.

Proof. ***Case 1.*** $\mathcal{T}$ runs into a conflict.

Let the trail $\mathcal{T}$ look like

$$\mathcal{T} = (p_{(0,1)}, \dots, p_{(0,g_0)}; \mathbf{d_1}, p_{(1,1)}, \dots, p_{(1,g_1)}; \dots; \mathbf{d_r}, p_{(r,1)}, \dots, p_{(r,g_r)}).$$

Then the sequence of learnable clauses is

$$\mathfrak{L}(\mathcal{T}_i) = (C_{(r,g_r)}, \dots, C_{(r,1)}, \dots, C_{(1,g_1)}, \dots, C_{(1,1)}, C_{(0,g_0)}, \dots, C_{(0,1)}).$$

By the NCC, we have that $g_r > 1$. We will show that $C_{(r,1)}$ (which is the clause/cube we get after resolving over $p_{(r,g_r-1)}, \dots, p_{(r,1)}$) is a new clause (cube).

Assume not. Consider the restricted clause (cube) $E := C_{(r,1)}|_{\mathcal{T}_i[r,1]}$. Suppose that there is an existential (universal) literal $x \in E \subseteq C_{(r,1)}$. That means that x is contained in at least one antecedent clause (cube) after (and including) $p_{(r,1)}$. In particular, we need $\bar{x} \in \mathcal{T}$ (resp. $x \in \mathcal{T}$). Because x is still contained in $C_{(r,1)}$, it cannot have been resolved away during learning, hence $\bar{x} \in \mathcal{T}[r, 1]$ (resp. $x \in \mathcal{T}[r, 1]$). This is a contradiction to the definition of E.

We conclude that E can only contain universal (existential) literals, hence $\mathrm{red}^{\forall}_{\Phi}(E) = (\bot)$ (resp. $\mathrm{red}^{\exists}_{\Phi}(E) = [\top]$). But then we would have got a conflict directly after d_r, which is impossible by the NCC. That means that $C_{(r,1)}$ must a new clause (cube).

We can backtrack to the point where we undo the rightmost existential (universal) literal in $\mathcal{T}$ that is contained in $C_{(r,1)}$. At this point, $C_{(r,1)}$ will not be falsified since it still includes at least this one literal.

Case 2. $\mathcal{T}$ does not run into a conflict, but we assigned all variables in $\mathcal{T}$.

Assume that we cannot find such a C (which would need to be a cube). Then there exists a $C \in \mathfrak{L}(\mathcal{T})$ such that $C \in \mathfrak{D}(\Phi_a)$, where Φ_a is the current formula for $\mathcal{T}$. That means there exists a cube $E \subseteq \mathcal{T}$ such that $\mathrm{red}^{\exists}_{\Phi_a}(E) = C$ and E satisfies $\mathfrak{C}(\Phi_a)$. In particular, we have $\mathrm{red}^{\exists}_{\Phi_a}(C|_{\mathcal{T}}) = [\top]$, which means that $\mathcal{T}$ should have run into a conflict. This is a contradiction.

We can backtrack to the point where we undo the rightmost universal literal in $\mathcal{T}$, that is contained in C. Then C will just propagate this universal literal and not an empty one. If this point is on the last decision level, we can alternatively restart. □

Remark 7.7. *To illustrate the relevance of the NCC, we give an example of a* QCDCL$^{\textsc{Any}}$ *trail — violating the NCC — from which we cannot learn a new clause. Consider the trail* $\mathcal{T} = (\mathbf{x}, \bot)$ *for the false QCNF* $\forall u \exists x \cdot (u \vee x) \wedge (u \vee \bar{x}) \wedge (\bar{u} \vee x) \wedge (\bar{u} \vee \bar{x})$*. The trail violates the NCC, as we got a conflict directly after the decision* x*. The only learnable clause is* $\mathrm{ante}_{\mathcal{T}}(\bot) = \bar{u} \vee \bar{x}$*, which is obviously already known.*

Another example illustrates the case where we can learn a new clause but no asserting clause. Let the trail be $\mathcal{U} := (\mathbf{x}, y; \mathbf{u}, \bar{z}, \bot)$ *for the false QCNF* $\forall u \exists x, y, z \cdot (\bar{x} \vee y) \wedge (x \vee y) \wedge (u \vee \bar{y} \vee \bar{z}) \wedge (\bar{u} \vee \bar{y} \vee \bar{z}) \wedge (u \vee \bar{y} \vee z) \wedge (\bar{u} \vee \bar{y} \vee z)$*. There are two new clauses we could learn:* $\bar{u} \vee \bar{y}$ *or* $\bar{u} \vee \bar{x}$*. None of the two can become unit after backtracking since we used the decision heuristic for* QCDCL$^{\textsc{Exi-Any}}$*, although we followed the NCC.*

Figure 7.2 provides an overview of the four systems and their ability to learn asserting clauses and cubes. As a consequence of always learning new constraints, we infer that our models are all complete and terminating proof methods.

Theorem 7.8. QCDCL, QCDCL$^{\textsc{Any}}$, QCDCL$^{\textsc{Uni-Any}}$ *and* QCDCL$^{\textsc{Exi-Any}}$ *are sound and complete proof systems*[1]*. Additionally, as long as we follow the rules of decision making (especially the NCC), we will always learn the empty clause or cube at some point, no matter what decisions were made.*

Proof. By Propositions 7.5 and 7.6 as well as Lemma 4.11 we conclude that from each trail (that has either run into a conflict or assigned all variables) we can always learn a new clause or cube. Note that these results have to be interpreted in the context of Proposition 7.4.

[1] I.e., they are proof systems for the language of false and true QBFs in the setting of [50]. Technically, in order not to trivialise the notion of such a proof system, we could consider proof systems for the language L of the marked union of true and false QBFs, i.e., $L = \{0\Phi \mid \Phi \text{ is a false QBF}\} \cup \{1\Phi \mid \Phi \text{ is a true QBF}\}$. In this way, L is still PSPACE complete.

	asserting clauses	**only new clauses**
asserting cubes	QCDCL	QCDCL$^{\text{Exi-Any}}$
only new cubes	QCDCL$^{\text{Uni-Any}}$	QCDCL$^{\text{Any}}$

Figure 7.2: Overview of guaranteed learnable constraints after a trail conflict in the corresponding models.

Since a given formula only consists of finitely many variables, we can only learn finitely many new clauses and cubes. We finish the proof as soon as we learn the empty clause or cube, which will happen at some point. Therefore all four systems are complete.

The soundness results from the fact that from each QCDCL, QCDCL$^{\text{Any}}$, QCDCL$^{\text{Uni-Any}}$ and QCDCL$^{\text{Exi-Any}}$ proof ι we can extract a LD-Q-Res or LD-Q-Con proof $\mathfrak{R}(\iota)$ for the same formula. □

7.2 Separation on True Formulas

The advantage of QCDCL$^{\text{Exi-Any}}$ (compared to QCDCL) is to decide existential literals out of order while still learning asserting cubes. Since cubes are important for verifications of true formulas, it makes sense to use true QBFs for the separation.

First, we discuss two generic modifications for QBFs. The twin construction, which we already used for the QBF $\mathtt{TwinEq}_n$ (cf. Definition 6.22), doubles all universal variables. For all clauses with universal variables a copy is created in the twin variables.

Definition 7.9 (twin formulas)**.** *Let $\Phi = \exists X \forall U \exists T \cdot \mathfrak{C}(\Phi)$ be a QCNF. Let $U = \{u_1, \ldots, u_m\}$ and let $v_1, \ldots, v_m$ be variables not occuring in Φ. Then the* twin formula *of Φ is the QCNF $\mathtt{Twin}\Phi$ defined as*

$$\mathtt{Twin}\Phi := \exists X \forall (U \cup \{v_1, \ldots, v_m\}) \exists T \cdot \mathfrak{C}(\Phi) \wedge \bigwedge_{C \in \mathfrak{C}(\Phi)} C[u_1/v_1, \ldots, u_m/v_m],$$

where u_i/v_i indicates that all occurrences of u_i are substituted by v_i.

The second modification is the *reversion* of a formula.

Definition 7.10. *If $\Phi = \mathcal{Q}_1 V_1 \ldots \mathcal{Q}_k V_k \cdot \bigwedge_{j=1}^{m} C_j$ is a QCNF with $\mathcal{Q}_i \in \{\exists, \forall\}$ and disjoint sets of variables V_i for $i = 1, \ldots, k$, then the* reversion $\mathtt{Rev}(\Phi)$ *of Φ is the QCNF*

$$\mathcal{Q}'_1 V_1 \ldots \mathcal{Q}'_k V_k \forall w \exists c_1, \ldots, c_m \cdot (\bar{c}_1 \vee \ldots \vee \bar{c}_m) \wedge \bigwedge_{j=1}^{m} \bigwedge_{\ell \in C_j} (\bar{\ell} \vee w \vee c_j) \wedge (\bar{\ell} \vee \bar{w} \vee c_j)$$

where $\mathcal{Q}'_i = \forall$ if $\mathcal{Q}_i = \exists$, and $\mathcal{Q}'_i = \exists$ if $\mathcal{Q}_i = \forall$, and $w, c_1, \ldots, c_m$ are new variables not contained in Φ.

It is easy to prove that there exists a duality between the truth values of Φ and $\mathtt{Rev}(\Phi)$.

Lemma 7.11. *If Φ is a QCNF, then* $\texttt{Rev}(\Phi)$ *is true if and only if Φ is false.*

Proof. ***Case 1.*** Φ is false.

Then there exists a winning strategy for the universal player of Φ. We will show that $\texttt{Rev}(\Phi)$ has an existential winning strategy.

The existential player for $\texttt{Rev}(\Phi)$ can just follow the universal winning strategy for Φ. That means there is at least one clause $C_j \in \mathfrak{C}(\Phi)$ falsified by the total assignment that consists of the assignment from the universal player and the corresponding response determined by the winning strategy. Then the clauses $\bar{\ell} \vee w \vee c_j$ and $\bar{\ell} \vee \bar{w} \vee c_j$ for each $\ell \in C_j$ are satisfied (for this particular j) by this assignment. Note that it does not matter how w was assigned. Therefore, the existential player for this modified formula can just set c_j to true and all the other c_i to false.

Case 2. Φ is true.

This case is analogous to Case 1. The universal player for the modified $\texttt{Rev}(\Phi)$ version follows the existential winning strategy for Φ. Then the universal player can set w to true (it does not matter, actually). For each $j \in \{1, \ldots, m\}$ the clause C_j is satisfied, hence at least one literal $\ell \in C_j$ is set to true. Therefore for each c_j, the clause $\bar{\ell} \vee \bar{w} \vee c_j$ becomes the unit clause (c_j) at some point under the assignment determined by the strategy. That means the existential player for $\texttt{Rev}(\Phi)$ has to set each c_j to true, falsifying the clause $\bar{c}_1 \vee \ldots \vee \bar{c}_m$.

We now have constructed a universal winning strategy for $\texttt{Rev}(\Phi)$. □

We will use the reversion to lift hardness from false to true QCNFs. To verify a true formula, we need to create a proof using cubes. We will show that $\texttt{Rev}(\Phi)$ is designed such that its initial cubes are basically the negated axiom clauses of Φ. Thus, a verification of $\texttt{Rev}(\Phi)$ can be transformed into a refutation of Φ.

Our reversion was inspired by the notion of the *negation* from [69]. The only change we made is adding the variable w. We did this to prevent a direct connection between an X- or U-block and an auxiliary variable c_j from the last block. Our lower bound technique is based on the fact that on certain formulas we cannot have direct connections (hence: cannot directly propagate) between outer and inner quantifier blocks. The added variable w helps to maintain this property.

The next two results shows how we can transform verifications of $\texttt{Rev}(\Phi)$ into refutations of Φ by interpreting the cubes from the verification as negated clauses of a refutation.

Lemma 7.12. *Let $\Phi = \mathcal{Q} \cdot \bigwedge_{j=1}^{m} C_j$ be a QCNF and let σ be an assignment that satisfies $\mathfrak{C}(\texttt{Rev}(\Phi))$. Then there exists some $C \in \mathfrak{C}(\Phi)$ with $\overline{C} \subseteq \sigma$.*

Proof. Since we have to satisfy the clause $(\bar{c}_1 \vee \ldots \vee \bar{c}_m)$, there is some $j \in \{1, \ldots, m\}$ with $\bar{c}_j \in \sigma$. Then the clauses $\bar{\ell} \vee w \vee c_j$ and $\bar{\ell} \vee \bar{w} \vee c_j$ have to be satisfied for each $\ell \in C_j$. We do not need to assign w, but we need to set each ℓ to false, hence $\overline{C}_j \subseteq \sigma$. □

Proposition 7.13. *If Φ is a false QCNF and ρ is a* LD-Q-Con *verification of* $\texttt{Rev}(\Phi)$*, then ρ can be transformed into a fully reduced* LD-Q-Res *refutation π of Φ with $|\pi| \leq |\rho|$.*

More precisely, for each clause $C \in \pi$ there is a cube $C' \in \rho$ with $\overline{C} \subseteq C'$. Furthermore, for each two clauses C, D that are resolved in π, the corresponding cubes C', D' are resolved in ρ, as well.

Proof. By Lemma 7.12, for each initial cube $D \in \rho$ there is a clause $C \in \mathfrak{C}(\Phi)$ such that $\mathrm{red}^{\exists}_{\mathtt{Rev}(\Phi)}(\overline{C}) \subseteq D$ (note that the assignment from Lemma 7.12 can still be reduced). We substitute each initial cube D with its corresponding $\mathrm{red}^{\exists}_{\mathtt{Rev}(\Phi)}(\overline{C})$ and shorten the proof, if necessary (i.e., delete redundant resolutions and reductions). We obtain a subproof $\pi' \subseteq \rho$, that is still a verification.

After that, we negate all cubes in π' and obtain a proof π that consists of clauses. If we interpret π as a proof for Φ (or $\mathrm{red}(\Phi)$ to be precise), all resolutions and reductions are still sound because the quantifiers were flipped, as well.

We can assume that in π we will reduce as soon as possible, otherwise we could shorten the proof even more. Obviously, the last clause in π has not obtained any additional literals, therefore π is a LD-Q-Res refutation of Φ. □

For our next results, we need an even stronger property than the XT-property: We require, that clauses from the formula contain at least one U- and T-literal.

Lemma 7.14. *If Φ is a Σ^b_3 QCNF, in which all clauses contain at least one U- and T-literal, then Φ fulfils the XT-property.*

Proof. Obviously, Φ does not contain any XT- or T-clauses and therefore the XT-property is fulfilled. □

We combine the results above and obtain a new lower bound technique for true formulas, which builds on the gauge technique for false formulas.

Theorem 7.15. *Let Φ be a false Σ^b_3. Additionally, let all clauses $C \in \mathfrak{C}(\Phi)$ contain at least one U- and one T-literal. If the QCNF $\mathtt{Twin}\Phi$ needs fully reduced primitive Q-Res refutations of size s, then QCDCL verifications for $\mathtt{Rev}(\mathtt{Twin}\Phi)$ also need size s.*

Proof. Let ι be a QCDCL verification for $\mathtt{Rev}(\mathtt{Twin}\Phi)$. We will show that there exists a fully reduced primitive Q-Res refutation π for $\mathtt{Twin}\Phi$ with $|\pi| \leq |\mathfrak{R}(\iota)|$.

Let π be the LD-Q-Res refutation of $\mathtt{Twin}\Phi_n$ as described in Proposition 7.13. Then π is fully reduced. We will show that π is primitive.

Assume not. Then there are two XUT-clauses $B_1, B_2 \in \pi$ that are resolved over some $x \in X$. By the construction of π described in Proposition 7.13, we can find two cubes $D_1, D_2 \in \mathfrak{R}(\iota)$ such that $\mathrm{var}(D_i) \cap U \neq \emptyset$ and $\mathrm{var}(D_i) \cap T \neq \emptyset$ for $i = 1, 2$ which are resolved over x. One of these cubes was an antecedent cube for x in some trail $\mathcal{T} \in \mathfrak{T}(\iota)$, say $D_1 = \mathrm{ante}_{\mathcal{T}}(x)$ (that means $\bar{x} \in D_1$).

In particular, there is some T-literal $t \in D_1$ such that $t <_{\mathcal{T}} x$ because D_1 must become unit. Remember that t is universal in $\mathtt{Rev}(\mathtt{Twin}\Phi)$ and we can only reduce cubes existentially. Then either t was a regular decision, or a propagation.

Case 1. t was decided.

This is only possible if all U-variables were assigned before. Hence, for each $u \in U$ there is a literal ℓ_u with $\mathrm{var}(\ell_u) = u$ and $\ell_u <_\mathcal{T} t <_\mathcal{T} x$. Because decisions have to be level-ordered in QCDCL, all ℓ_u had to have been propagated.

Let ℓ_u be the leftmost U-literal in $\mathcal{T}$. Consider its antecedent clause $A := \mathrm{ante}_\mathcal{T}(\ell_u)$.

Claim. If ℓ_u is the leftmost U-literal in $\mathcal{T}$, then there exists an $i \in \{1, \ldots, m\}$ such that $c_i \in \mathrm{var}(\mathrm{ante}_\mathcal{T}(\ell_u))$ (where $c_1, \ldots, c_m$ are the variables from $\mathtt{Rev}(\mathtt{Twin}\Phi)$ as in Definition 7.10).

> *Proof of the claim.* Assume not. We will show that $A := \mathrm{ante}_\mathcal{T}(\ell_u)$ has to contain at least two different U-literals.
>
> Assume that A only contains one U-literal, namely ℓ_u itself. Let Φ consist of the clauses $C_1, \ldots, C_{m'}$ and let $\mathtt{Twin}\Phi$ consist of the clauses $C_1, \ldots, C_m$ with $m > m'$. We can assume that ℓ_u is a copy of a literal from Φ by the construction of a twin formula. In particular, ℓ_u (and $\bar{\ell}_u$) cannot be contained in the clauses $C_1, \ldots, C_{m'}$.
>
> Let ρ be the LD-Q-Res derivation of A that was constructed in ι, but not used for $\mathfrak{R}(\iota)$ since verifications can only make use of cubes. By assumption, A does not contain any c_i or $\bar{c}_i$. However, each axiom clause from $\mathtt{Rev}(\mathtt{Twin}\Phi)$ includes at least one c_i or $\bar{c}_i$. Hence, we have to resolve over these variables somehow. In particular, we need $\bar{c}_1 \vee \ldots \vee \bar{c}_m \in \rho$ since this is the only axiom clause where these variables occur in a negative polarity.
>
> We will now construct another LD-Q-Res derivation ρ' by substituting $\bar{c}_1 \vee \ldots \vee \bar{c}_m$ with $\bar{c}_1 \vee \ldots \vee \bar{c}_{m'}$ in ρ and gradually deleting all redundant clauses. In particular, all clauses from $\mathtt{Rev}(\mathtt{Twin}\Phi)$ that contain ℓ_u or $\bar{\ell}_u$ will be deleted because the corresponding c_i is missing. Let A' be the last clause in ρ', hence ρ' is a LD-Q-Res proof of A' from $\mathtt{Rev}(\Phi)$. Obviously, we get $A' \subseteq A$ and $\ell_u \notin A'$ as well as $c_i, \bar{c}_i \notin A'$ for all $i = 1, \ldots, m$. Since ℓ_u was the only U-literal in A, the clause A' cannot have any U-literals. Therefore A' is a clause consisting of universal literals only. Reducing A' universally gives us the empty clause ($\bot$), which means that we can extend ρ' to a refutation of $\mathtt{Rev}(\Phi)$. But this is a contradiction to the fact that $\mathtt{Rev}(\Phi)$ is a true formula (by Lemma 7.11).
>
> That shows that A must contain more than one U-literal. Let $\ell_u \neq z \in A$ be another U-literal. Then we need $\bar{z} <_\mathcal{T} \ell_u$ since z is existential. However, this contradicts the choice of ℓ_u, which finishes the proof of the claim.

We want to create a contradiction by applying the claim, for which we need to show that A does not contain any literal from $\{c_r, \bar{c}_r \mid r = 1, \ldots, m\}$.

Assume that there is such a literal. That means we can find the leftmost literal $c \in \{c_r, \bar{c}_r \mid r = 1, \ldots, m\}$ in $\mathcal{T}$, hence $c <_\mathcal{T} \ell_u <_\mathcal{T} t <_\mathcal{T} x$. Now, c cannot have been a decision since decisions must be level-ordered. That means that c has been propagated by an antecedent clause $F := \mathrm{ante}_\mathcal{T}(c)$. Because c was leftmost, F cannot be the clause $\bar{c}_1 \vee \ldots \vee \bar{c}_m$. It is easy to see that F then has to contain either w or $\bar{w}$ by the structure

of a reversion (see Definition 7.10). W.l.o.g. let $w \in F$. Then we need $\bar{w} <_{\mathcal{T}} c <_{\mathcal{T}} \ell_u$. Because of the quantification order, $\bar{w}$ cannot be a decided literal. Hence $\bar{w}$ must have been propagated by some antecedent cube $E := \text{ante}_{\mathcal{T}}(\bar{w})$. Let ρ be the subproof of E from $\mathfrak{R}(\iota)$. Then there exists an initial cube $G \in \rho$ with $w \in G$, which is not getting resolved away in ρ. Furthermore, G is also an initial cube in $\mathfrak{R}(\iota)$. By Lemma 7.12, there exists some $H \in \mathfrak{C}(\mathtt{Twin}\Phi)$ such that $\overline{H} \subseteq G$. Since each clause of Φ contains a U-literal, there is such a U-literal $v \in \overline{H} \subseteq G$ and also $v \in E$ because it cannot be resolved or reduced away. This means we need $v <_{\mathcal{T}} \bar{w} <_{\mathcal{T}} \ell_u$, which is a contradiction to the choice of ℓ_u.

We have now shown that A does not contain any $c_r, \bar{c}_r$, $r \in \{1, \ldots, m\}$. However, this is impossible by our claim. We conclude that Case 1 cannot occur.

Case 2. t was propagated.

Consider the antecedent cube $J := \text{ante}_{\mathcal{T}}(t)$. Let τ be the subproof of J in $\mathfrak{R}(\iota)$. Then the first cubes in τ were (reduced) satisfying assignments for $\mathtt{Rev}(\mathtt{Twin}\Phi_n)$. At least one of these initial cubes in τ contains $\bar{t}$ which will not get resolved away since it appears in J. Let $I \in \tau$ be an initial cube with $\bar{t} \in I$ that does not get resolved away in τ. By Lemma 7.12, there exists a clause $K \in \mathfrak{C}(\mathtt{Twin}\Phi_n)$ such that $\overline{K} \subseteq I$. By our assumption, K contains at least one U- and one T-literal. But then also I contains at least one U-literal ℓ. Because ℓ is blocked by $\bar{t}$ all the time, it does not get reduced away in τ, hence $\ell \in J$.

Due to $\ell <_{\mathtt{Rev}(\mathtt{Twin}\Phi_n)} t$, we need $\ell <_{\mathcal{T}} t$ in order for J to become unit. W.l.o.g. let ℓ be the leftmost U-literal in $\mathcal{T}$ (the fact that $\ell \in J$ is not important anymore from this point on). Because of $x <_{\mathtt{Rev}(\mathtt{Twin}\Phi_n)} \ell$, the literal ℓ cannot be a regular decision. That means it must have been propagated.

We can repeat the argument from Case 1. We conclude that such an ℓ does not exist. Thus Case 2 does not occur and we get a contradiction regarding our assumption that π was not primitive. □

We now construct specific QBFs that meet the conditions of Theorem 7.15. We already know from Proposition 5.24 that the equality formulas $\mathtt{Eq}_n$ have linear gauge and therefore need exponential-size fully reduced primitive Q-Res refutations. However, not all clauses from $\mathtt{Eq}_n$ contain a U-literal. We modify the formulas by adding an artificial U-literal p to the relevant clauses:

Definition 7.16. *The QCNF* $\mathtt{ModEq}_n$ *consists of the prefix*

$$\exists x_1, \ldots, x_n \forall u_1, \ldots, u_n, p \exists t_1, \ldots, t_n$$

and the matrix

$$\begin{aligned} &x_i \vee u_i \vee t_i, \\ &\bar{x}_i \vee \bar{u}_i \vee t_i, \\ &p \vee \bar{t}_1 \vee \ldots \vee \bar{t}_n, \\ &\bar{p} \vee \bar{t}_1 \vee \ldots \vee \bar{t}_n \end{aligned}$$

for $i = 1, \ldots, n$.

Neither this nor the Twin modification changes the gauge of the formulas. Hence we get:

Proposition 7.17. *It holds gauge*($\texttt{TwinModEq}_\texttt{n}$) $= n$. *Therefore,* $\texttt{TwinModEq}_\texttt{n}$ *needs exponential-size fully reduced primitive* Q-Res *refutations.*

Proof. Since all axiom clauses contain T-literals, we have to get rid of them somehow. The only four clauses that contain T-literals in a negative polarity are the clauses $p \vee \bar{t}_1 \vee \ldots \vee \bar{t}_n$, $\bar{p} \vee \bar{t}_1 \vee \ldots \vee \bar{t}_n$, $q \vee \bar{t}_1 \vee \ldots \vee \bar{t}_n$ and $\bar{q} \vee \bar{t}_1 \vee \ldots \vee \bar{t}_n$, where q is the copy of p. Hence, we have to use at least one of them in order to derive an X-clause. In particular, we have to resolve over each t_i. The only four clauses in which t_i occurs in a positive polarity are $x_i \vee u_i \vee t_i$, $\bar{x}_i \vee \bar{u}_i \vee t_i$, $x_i \vee v_i \vee t_i$ and $\bar{x}_i \vee \bar{v}_i \vee t_i$, where v_i is the copy of u_i. In each case we will pile up x_i or $\bar{x}_i$ for each resolution over t_i. Therefore, our X-clause at the end will contain at least n different X-literals.

Hence gauge($\texttt{TwinModEq}_\texttt{n}$) $= n$. The second claim then follows from Theorem 5.17. □

The lower bound for the true QBFs then follows with Theorem 7.15.

Corollary 7.18. $\texttt{Rev}(\texttt{TwinModEq}_n)$ *needs exponential-size* QCDCL *verifications.*

We now use a direct construction to show that there exists a short $\mathsf{QCDCL}^{\textsc{Exi-Any}}$ refutation for $\texttt{Rev}(\texttt{TwinModEq}_n)$.

Proposition 7.19. $\texttt{Rev}(\texttt{TwinModEq}_n)$ *has polynomial-size* $\mathsf{QCDCL}^{\textsc{Exi-Any}}$ *verifications.*

Proof. Let us first list all the clauses of $\texttt{TwinModEq}_n$. It consists of the prefix

$$\exists x_1, \ldots, x_n \forall u_1, \ldots, u_n, p, v_1, \ldots, v_n, q \exists t_1, \ldots, t_n$$

and the matrix

$$\begin{aligned} C_{(i,1)} &:= x_i \vee u_i \vee t_i & C_1 &:= p \vee \bar{t}_1 \vee \ldots \vee \bar{t}_n \\ C_{(i,2)} &:= \bar{x}_i \vee \bar{u}_i \vee t_i & C_2 &:= \bar{p} \vee \bar{t}_1 \vee \ldots \vee \bar{t}_n \\ C_{(i,3)} &:= x_i \vee v_i \vee t_i & C_3 &:= q \vee \bar{t}_1 \vee \ldots \vee \bar{t}_n \\ C_{(i,4)} &:= \bar{x}_i \vee \bar{v}_i \vee t_i & C_4 &:= \bar{q} \vee \bar{t}_1 \vee \ldots \vee \bar{t}_n \end{aligned}$$

for $i = 1, \ldots, n$.

Then the true QCNF $\texttt{Rev}(\texttt{TwinModEq}_n)$ consists of the prefix

$$\forall x_1, \ldots, x_n \exists u_1, \ldots, u_n, p, v_1, \ldots, v_n, q \forall t_1, \ldots, t_n, w \exists M,$$

with $M := \{c_{(i,j)}, c_j \mid i = 1, \ldots, n,\ j = 1, \ldots, 4\}$, and the matrix

$$E := \bigvee_{i=1}^{n} \bigvee_{j=1}^{4} \bar{c}_{(i,j)} \vee \bigvee_{k=1}^{4} \bar{c}_k$$

$$
\begin{array}{llll}
\bar{x}_i \vee w \vee c_{(i,1/3)} & \bar{u}_i \vee w \vee c_{(i,1)} & \bar{v}_i \vee w \vee c_{(i,3)} & \\
\bar{x}_i \vee \bar{w} \vee c_{(i,1/3)} & \bar{u}_i \vee \bar{w} \vee c_{(i,1)} & \bar{v}_i \vee \bar{w} \vee c_{(i,3)} & \\
x_i \vee w \vee c_{(i,2/4)} & u_i \vee w \vee c_{(i,2)} & v_i \vee w \vee c_{(i,4)} & \\
x_i \vee \bar{w} \vee c_{(i,2/4)} & u_i \vee \bar{w} \vee c_{(i,2)} & v_i \vee \bar{w} \vee c_{(i,4)} & \\
\bar{t}_i \vee w \vee c_{(i,1/2/3/4)} & & & \\
\bar{t}_i \vee \bar{w} \vee c_{(i,1/2/3/4)} & & & \\
\bar{p} \vee w \vee c_1 & p \vee w \vee c_2 & \bar{q} \vee w \vee c_3 & q \vee w \vee c_4 \\
\bar{p} \vee \bar{w} \vee c_1 & p \vee \bar{w} \vee c_2 & \bar{q} \vee \bar{w} \vee c_3 & q \vee \bar{w} \vee c_4 \\
t_i \vee w \vee c_{1/2/3/4} & & & \\
t_i \vee \bar{w} \vee c_{1/2/3/4} & & &
\end{array}
$$

for $i = 1, \ldots, n$, where variables like $c_{(i,1/3)}$ decode two versions of this clause: One clause with $c_{(i,1)}$ and the other with $c_{(i,3)}$ (analogously with $c_{(i,2/4)}$, $c_{(i,1/2/3/4)}$ and $c_{1/2/3/4}$).

Let us now construct a polynomial size $\mathsf{QCDCL}^{\textsf{Exi-Any}}$ verification. At first, we would like to learn the cubes

$$
\begin{aligned}
D_{(i,1)} &:= \bar{x}_i \wedge \bar{u}_i \wedge \bar{t}_i \\
D_{(i,2)} &:= x_i \wedge u_i \wedge \bar{t}_i \\
D_1 &:= \bar{p} \wedge t_1 \wedge \ldots \wedge t_n
\end{aligned}
$$

for $i = 1, \ldots, n$. In order to learn $D_{(i,1)}$, we will make (level-ordered) decisions that satisfy all literals from $D_{(i,1)}$, but falsify all the other $D_{(i',1)}$ for $i' \neq i$. For example, we set x_i, u_i and t_i to false, and we can assign all the other variables left of w arbitrarily. Note that until we reach w, we will never make any propagations since w or $\bar{w}$ is blocking them. After having decided all variables left of w, we will decide w and potentially trigger some propagations. However, the variable $c_{(i,1)}$ will never be propagated because all clauses containing it are already satisfied. After this we will set $c_{(i,1)}$ to false and all the remaining variables to true.

We now have satisfied the clause E. Furthermore, we have set all $c_{(i',j)}$ and c_k to true except $c_{(i,1)}$. Hence we have satisfied all clauses except the four clauses containing $c_{(i,1)}$. But these two clauses were already satisfied because we have satisfied the cube $D_{(i,1)}$ with the decisions left of w.

Let $\mathcal{T}_{(i,1)}$ be the trail we have constructed now. We can extract the cube

$$
\bar{x}_i \wedge \bar{u}_i \wedge \bar{t}_i \wedge \bar{c}_{(i,1)} \wedge \bigwedge_{(i',j) \in (\{1,\ldots,n\} \times \{1,2,3,4\}) \setminus \{(i,1)\}} c_{(i',j)} \wedge \bigwedge_{k=1}^{4} c_k,
$$

which, as an assignment, already satisfies all clauses from $\mathtt{Rev}(\mathtt{TwinModEq}_n)$. This cube can be existentially reduced to $D_{(i,1)}$, which is the cube we learn from $\mathcal{T}_{(i,1)}$. Analogously, we can learn the cubes $D_{(i,2)}$ for $i = 1, \ldots, n$ via some analogue trails $\mathcal{T}_{(i,2)}$.

It remains to learn the cube D_1, which represents the clause $C_1 \in \mathfrak{C}(\texttt{TwinModEq}_n)$. We will construct a trail $\mathcal{T}_1$ which includes (level-ordered) decisions that satisfy D_1. But now we have to make sure not to trigger propagations via $D_{(i,1)}$ or $D_{(i,2)}$ since we must not set t_i to false. This can be done by setting all x_i to false and all u_i to true. Then we can set p to false and all t_i to true. The remaining variables left of w can again be decided arbitrarily. Then we set w to true and potentially trigger some propagations of $c_{(i,j)}$ or c_k, which is not a problem since c_1 will never be propagated (the clauses containing c_1 are already satisfied). Then we set c_1 to false and all remaining variables can be set to true.

As with $\mathcal{T}(i,1)$, we have satisfied all clauses from $\texttt{Rev}(\texttt{TwinModEq}_n)$. We can extract the cube

$$\bar{p} \wedge t_1 \wedge \ldots \wedge t_n \wedge \bar{c}_1 \wedge \bigwedge_{i=1}^{n} \bigwedge_{j=1}^{4} c_{(i,j)} \wedge \bigwedge_{k=2}^{4} c_k,$$

from $\mathcal{T}_1$, which already satisfies the matrix and can be existentially reduced to D_1.

We will now define the cubes

$$R_i := \bar{x}_i \wedge \bar{u}_i \wedge \bar{p} \wedge \bigwedge_{k=i+1}^{n} (u_k \wedge \bar{u}_k) \wedge \bigwedge_{\ell=1}^{i-1} t_\ell$$
$$L_i := x_i \wedge u_i \wedge \bar{p} \wedge \bigwedge_{k=i+1}^{n} (u_k \wedge \bar{u}_k) \wedge \bigwedge_{\ell=1}^{i-1} t_\ell$$

for $i = 2, \ldots, n-1$. We will construct trails $\mathcal{U}_{n-1}, \mathcal{V}_{n-1}, \ldots, \mathcal{U}_2, \mathcal{V}_2$ with which we will gradually learn the clauses $R_{n-1}, L_{n-1}, \ldots, R_2, L_2$.

We start with

$$\mathcal{U}_{n-1} := (\bar{\mathbf{p}}; \bar{\mathbf{x}}_\mathbf{1}; \bar{\mathbf{u}}_\mathbf{1}, t_1; \ldots; \bar{\mathbf{x}}_\mathbf{n-1}; \bar{\mathbf{u}}_\mathbf{n-1}, t_{n-1}, \bar{t}_n, \bar{x}_n, \top)$$

with antecedent cubes

$$\begin{aligned} \text{ante}_{\mathcal{U}_{n-1}}(t_j) &= D_{(j,1)} \\ \text{ante}_{\mathcal{U}_{n-1}}(\bar{t}_n) &= D_1 \\ \text{ante}_{\mathcal{U}_{n-1}}(\bar{x}_n) &= D_{(n,2)} \\ \text{ante}_{\mathcal{U}_{n-1}}(\top) &= D_{(n,1)} \end{aligned}$$

for $j = 1, \ldots, n-1$. We learn the cube $R_{n-1} = \Big(\Big(D_{(n,1)} \overset{x_n}{\bowtie} D_{(n,2)}\Big) \overset{t_n}{\bowtie} D_1\Big) \overset{t_{n-1}}{\bowtie} D_{(n-1,1)}$.

Analogously, by flipping some polarities, we construct the trail $\mathcal{V}_{n-1}$ and learn the cube $L_{n-1} = \Big(\Big(D_{(n,1)} \overset{x_n}{\bowtie} D_{(n,2)}\Big) \overset{t_n}{\bowtie} D_1\Big) \overset{t_{n-1}}{\bowtie} D_{(n-1,2)}$. Note that R_{n-1} will not interfere with the assignments in $\mathcal{V}_{n-1}$.

Assume we have already learned the clauses $R_{n-1}, L_{n-1}, \ldots, R_i, L_i$ for some $i \in \{3, \ldots, n-1\}$. Then we can construct the following trail:

$$\mathcal{U}_{i-1} := (\bar{\mathbf{p}}; \bar{\mathbf{x}}_\mathbf{1}; \bar{\mathbf{u}}_\mathbf{1}, t_1; \ldots; \bar{\mathbf{x}}_\mathbf{i-1}; \bar{\mathbf{u}}_\mathbf{i-1}, t_{i-1}, x_i, \top)$$

with antecedent cubes

$$\begin{aligned}\mathrm{ante}_{\mathcal{U}_{i-1}}(t_j) &= D_{(j,1)}\\ \mathrm{ante}_{\mathcal{U}_{i-1}}(x_i) &= R_i\\ \mathrm{ante}_{\mathcal{U}_{i-1}}(\top) &= L_i\end{aligned}$$

for $j = 1, \ldots, i-1$. We learn the cube $R_{i-1} = \left(L_i \overset{x_i}{\bowtie} R_i\right) \overset{t_{i-1}}{\bowtie} D_{(i-1,1)}$. Analogously, we can construct the trail $\mathcal{V}_{i-1}$ and learn $L_{i-1} = \left(L_i \overset{x_i}{\bowtie} R_i\right) \overset{t_{i-1}}{\bowtie} D_{(i-1,2)}$.

After having learned the cubes $R_{n-1}, L_{n-1}, \ldots, R_2, L_2$, we construct two more trails, namely

$$\mathcal{U}_1 := (\bar{\mathbf{p}}; \bar{\mathbf{x}}_\mathbf{1}; \bar{\mathbf{u}}_\mathbf{1}, t_1, x_2, \top)$$

with antecedent cubes

$$\begin{aligned}\mathrm{ante}_{\mathcal{U}_1}(t_1) &= D_{(1,1)}\\ \mathrm{ante}_{\mathcal{U}_1}(x_2) &= R_2\\ \mathrm{ante}_{\mathcal{U}_1}(\top) &= L_2,\end{aligned}$$

from which we learn $[\bar{x}_1] = \left(L_2 \overset{x_2}{\bowtie} R_2\right) \overset{t_1}{\bowtie} D_{1,1}$, and the trail

$$\mathcal{V}_1 := (x_1; \bar{\mathbf{p}}; \mathbf{u}_\mathbf{1}, t_1, x_2, \top)$$

with antecedent cubes

$$\begin{aligned}\mathrm{ante}_{\mathcal{V}_1}(x_1) &= [x_1]\\ \mathrm{ante}_{\mathcal{V}_1}(t_1) &= D_{(1,2)}\\ \mathrm{ante}_{\mathcal{V}_1}(x_2) &= R_2\\ \mathrm{ante}_{\mathcal{V}_1}(\top) &= L_2,\end{aligned}$$

from which we learn the empty cube $[\top] = \mathrm{red}^{\exists}_{\mathtt{Rev(TwinModEq_n)}} \left(\left(L_2 \overset{x_2}{\bowtie} R_2\right) \overset{t_1}{\bowtie} D_{(1,2)}\right) \overset{x_1}{\bowtie} [x_1]$.

All in all, we have constructed a $\mathsf{QCDCL}^{\textsc{Exi-Any}}$ verification using the $4n-1$ trails

$$\mathcal{T}_{(1,1)}, \ldots, \mathcal{T}_{(n,1)}, \mathcal{T}_{(1,2)} \ldots, \mathcal{T}_{(n,2)}, \mathcal{T}_1, \mathcal{U}_{n-1}, \mathcal{V}_{n-1}, \ldots, \mathcal{U}_1, \mathcal{V}_1.$$

□

Corollary 7.20. QCDCL *and* $\mathsf{QCDCL}^{\textsc{Exi-Any}}$ *are exponentially separated on true formulas.*

7.3 Separation on False Formulas

For separating QCDCL and QCDCL$^{\text{Uni-Any}}$, we recall the completion principle $\mathtt{CR}_n$ (cf. Definition 5.13). For the lower bound, we will use the modification $\mathtt{TwinCR}_n$. As we show, Cube Learning becomes rather useless with the `Twin` modification. This fact helps us to ensure that QCDCL refutations of $\mathtt{TwinCR}_n$ are primitive, and thus we can apply the gauge lower-bound method.

Similarly as in Proposition 7.17 we can compute the gauge.

Lemma 7.21. *It holds* $\mathit{gauge}(\mathtt{TwinCR}_n) = n$.

Proof. For the derivation of an X-clause we need at least one of the clauses $\bar{a}_1 \vee \ldots \vee \bar{a}_n$ or $\bar{b}_1 \vee \ldots \vee \bar{b}_n$ since we have to get rid of all T-literals. In particular, w.l.o.g. we have to resolve over each a_i. For this, we need one of the clauses $x_{(i,j)} \vee u \vee a_i$ or $x_{(i,j)} \vee v \vee a_i$ for each i. That means for each i we will pile up at least one $x_{(i,j)}$ for some j. Therefore $\mathrm{gauge}(\mathtt{TwinCR}_n) = n$. □

The main work is to check that QCDCL refutations of $\mathtt{TwinCR}_n$ are primitive.

Proposition 7.22. *If* ι *is a* QCDCL *refutation of* $\mathtt{TwinCR}_n$*, then* $\mathfrak{R}(\iota)$ *is fully reduced and primitive.*

Proof. It suffices to show that $\mathfrak{R}(\iota)$ is primitive. Assume not.

Then there exists two XUT-clauses $C, D \in \mathfrak{R}(\iota)$ that are resolved over an X-literal, say x. One of these two clauses has to be the antecedent clause of x by the definition of Clause Learning, say $C = \mathrm{ante}_{\mathcal{T}}(x)$ for some trail $\mathcal{T} \in \mathfrak{T}(\iota)$. Let $t_1 \in C$ be one of the T-literals. We want to show, that there exists a U-literal w with $w <_{\mathcal{T}} x$.

Assume that no such w exists. Since C had to become unit at the propagation of x, we need $\bar{t}_1 <_{\mathcal{T}} x$. The literal $\bar{t}_1$ cannot be a decision in $\mathcal{T}$, since this would mean that we assigned all U-variables earlier in the trail, which contradicts our assumption. Hence $\bar{t}_1$ must have been a propagation.

Starting with $i = 1$, we define $F_i := \mathrm{ante}_{\mathcal{T}}(\bar{t}_i)$. Now, F_i cannot contain U-literals since we cannot falsify these literals before assigning $\bar{t}_i$. Because of the XT-property (and Lemma 5.4), F_i cannot contain X-literals, as well (otherwise it would be an XT-clause). But if the XT-property is fulfilled, we cannot derive unit T-clauses, therefore F_i has to contain at least one additional T-literal, say $t_{i+1} \in F_i$.

This argument can be repeated for each $i \in \mathbb{N}$, which means we could find an infinite amount of T-literals $\bar{t}_i$ that must be all contained in $\mathcal{T}$, which is obviously not possible. This shows that our assumption was false and we can indeed find such a U-literal $w <_{\mathcal{T}} \bar{t}_1 <_{\mathcal{T}} x$.

W.l.o.g. let w be the first (leftmost) U-literal in $\mathcal{T}$. Since it was assigned before x, it cannot be a decision literal. Hence, it was propagated. Define $A := \mathrm{ante}_{\mathcal{T}}(w)$. Clearly, A is a cube. We will show that A contains at least two different U-literals. Then, since w was the first U-literal in $\mathcal{T}$, A cannot become unit until at least one U-literal was assigned, which would be a contradiction.

Now, A is a cube that was derived during Cube Learning from cubes that represent satisfying (partial) assignments of the matrix of $\mathtt{TwinCR_n}$. Let D be a cube that satisfies the matrix of $\mathtt{TwinCR_n}$. Because we have to satisfy the clauses $\bar{a}_1 \vee \ldots \vee \bar{a}_n$ and $\bar{b}_1 \vee \ldots \vee \bar{b}_n$, there exists an $r \in \{1, \ldots, n\}$ with $\bar{a}_r \in D$ and an $s \in \{1, \ldots, n\}$ with $\bar{b}_s \in D$. Furthermore, we have to satisfy the clauses $x_{(r,s)} \vee u \vee a_r$, $x_{(r,s)} \vee v \vee a_r$, $\bar{x}_{(r,s)} \vee \bar{u} \vee b_s$ and $\bar{x}_{(r,s)} \vee \bar{v} \vee b_s$. That means we have to assign u in some polarity. W.l.o.g. let $u \in D$. Then we have to set $x_{(r,s)}$ to false, hence $\bar{x}_{(r,s)} \in D$. In order to satisfy $x_{(r,s)} \vee v \vee a_r$, we have to set v to true, as well. Therefore we get $v \in D$.

We conclude that $u \in D$ if and only if $v \in D$, and analogously $\bar{u} \in D$ if and only if $\bar{v} \in D$. This means that we will never be able to resolve such two learned cubes in ι since we cannot create universal tautologies in cubes. In particular, we have proven that A contains at least two U-literals, which leads to a contradiction as described above. □

Applying Theorem 5.17 then yields the lower bound.

Corollary 7.23. $\mathtt{TwinCR}_n$ *needs exponential-sized* QCDCL *refutations.*

On the other hand, $\mathtt{TwinCR}_n$ is easy for $\mathsf{QCDCL}^{\textsc{Uni-Any}}$. Basically, we can simulate the Q-Res refutation of $\mathtt{CR}_n$ from [64], because we can decide universal literals out of order.

Proposition 7.24. $\mathtt{TwinCR}_n$ *has polynomial-sized* $\mathsf{QCDCL}^{\textsc{Uni-Any}}$ *refutations.*

Proof. For each $k = 1, \ldots, n$ we construct the trail

$$\mathcal{T}_k := (\bar{\mathbf{x}}_{(\mathbf{1},\mathbf{k})}; \ldots; \bar{\mathbf{x}}_{(\mathbf{n},\mathbf{k})}; \bar{\mathbf{u}}, a_1, \ldots, a_n, \bot)$$

with antecedent clauses

$$\mathrm{ante}_{\mathcal{T}_k}(a_i) = x_{(i,k)} \vee u \vee a_i, \quad \mathrm{ante}_{\mathcal{T}_k}(\bot) = \bar{a}_1 \vee \ldots \vee \bar{a}_n,$$

for $i = 1, \ldots, n$.

Resolving $\bar{a}_1 \vee \ldots \vee \bar{a}_n$ over each $\mathrm{ante}_{\mathcal{T}_k}(a_i)$ gives us the clause $E_k := x_{(1,k)} \vee \ldots \vee x_{(n,k)}$, which we will learn. Note that the trails and the learned clauses will not affect each other, hence the order in which we construct these n trails does not matter. Next, we construct the trails $\mathcal{U}_1, \ldots, \mathcal{U}_{n-1}$ (in that order). From each $\mathcal{U}_k$ we learn the clause $C_k := \bar{u} \vee b_k$. While constructing $\mathcal{U}_k$, we assume that $C_1, \ldots, C_{k-1}$ were already learned. Then, $\mathcal{U}_k$ looks as follows:

$$\mathcal{U}_k := (\mathbf{u}, b_1, \ldots, b_{k-1}; \mathbf{v}; \bar{\mathbf{b}}_{\mathbf{k}}, \bar{x}_{(1,k)}, \ldots, \bar{x}_{(n,k)}, \bot)$$

with antecedent clauses

$$\mathrm{ante}_{\mathcal{U}_k}(b_j) = C_j, \quad \mathrm{ante}_{\mathcal{U}_k}(\bar{x}_{(i,k)}) = \bar{x}_{(i,k)} \vee \bar{u} \vee b_k, \quad \mathrm{ante}_{\mathcal{U}_k}(\bot) = E_k,$$

for $i = 1, \ldots, n$ and $j = 1, \ldots, k-1$. Resolving E_k over each $\mathrm{ante}_{\mathcal{U}_k}(\bar{x}_{(i,k)})$ leads to the learnable clause C_k. Having learned the clauses $C_1, \ldots, C_{n-1}$, we continue with the trail $\mathcal{V}$, which will be the last one. It looks as follows:

$$\mathcal{V} := (\mathbf{u}, b_1, \ldots, b_{n-1}, \bar{b}_n, \bar{x}_{(1,n)}, \ldots, \bar{x}_{(n,n)}, \bot)$$

with antecedent clauses

$$\text{ante}_{\mathcal{V}}(b_j) = C_j, \quad \text{ante}_{\mathcal{V}}(\bar{b}_n) = \bar{b}_1 \vee \ldots \vee \bar{b}_n, \quad \text{ante}_{\mathcal{V}}(\bar{x}_{(i,n)}) = \bar{x}_{(i,n)} \vee \bar{u} \vee b_n,$$
$$\text{ante}_{\mathcal{V}}(\bot) = E_n,$$

for $i = 1, \ldots, n$ and $j = 1, \ldots, n-1$. Since we only made a universal decision, we can learn the empty clause ($\bot$) from $\mathcal{V}$ by resolving over everything.

Thus we constructed a $\mathsf{QCDCL}^{\textsc{Uni-Any}}$ refutation using $2n+1$ trails. □

We summarize the results from Corollary 7.23 and Proposition 7.24:

Corollary 7.25. $\texttt{TwinCR}_n$ *is hard for* QCDCL*, but easy for* $\mathsf{QCDCL}^{\textsc{Uni-Any}}$*.*

Besides $\texttt{TwinCR}_n$, we can find further separations between QCDCL and $\mathsf{QCDCL}^{\textsc{Uni-Any}}$. We recall the $\texttt{MirrorCR}_n$ formulas from Definition 6.13. We have already shown that $\texttt{MirrorCR}_n$ consists of an unsatisfiable matrix (cf. Proposition 6.14) needs exponential-sized QCDCL refutations (cf. Corollary 6.17). We prove now that these formulas are in fact easy for $\mathsf{QCDCL}^{\textsc{Uni-Any}}$.

Proposition 7.26. $\texttt{MirrorCR}_n$ *has polynomial-sized* $\mathsf{QCDCL}^{\textsc{Uni-Any}}$ *refutations.*

Proof. At first, we will derive the clauses $A_k := x_{(1,k)} \vee \ldots \vee x_{(n,k)}$ for each $k = 1, \ldots, n$. Suppose, we have already learned $A_1, \ldots, A_{k-1}$. We construct the trail $\mathcal{T}_k$ as follows:

$$\mathcal{T}_k := (\bar{\mathbf{x}}_{(\mathbf{1},\mathbf{k})}; \ldots; \bar{\mathbf{x}}_{(\mathbf{n},\mathbf{k})}; \bar{\mathbf{u}}, a_1, \ldots, a_n, \bot)$$

with

$$\text{ante}_{\mathcal{T}_k}(a_i) = x_{(i,k)} \vee u \vee a_i$$
$$\text{ante}_{\mathcal{T}_k}(\bot) = \bar{a}_1 \vee \ldots \vee \bar{a}_n$$

for $i = 1, \ldots, n$. From this trail we can learn E_k by resolving over all a_i and then we restart.

Our next goal is to learn the clauses $B_k := \bar{u} \vee b_k$ for each $k = 1, \ldots, n-1$. We now suppose that we have already learned $A_1, \ldots, A_n$ and $B_1, \ldots, B_{k-1}$. We construct the trail $\mathcal{U}_k$ as follows:

$$\mathcal{U}_k := (\mathbf{u}, b_1, \ldots, b_{k-1}; \bar{\mathbf{b}}_{\mathbf{k}}, \bar{x}_{(1,k)}, \ldots, \bar{x}_{(n,k)}, \bot)$$

with

$$\text{ante}_{\mathcal{U}_k}(b_j) = B_j$$
$$\text{ante}_{\mathcal{U}_k}(\bar{x}_{(i,k)}) = \bar{x}_{(i,k)} \vee \bar{u} \vee b_k$$
$$\text{ante}_{\mathcal{U}_k}(\bot) = A_k$$

for $j = 1, \ldots, k-1$ and $i = 1, \ldots, n$. We learn B_k by resolving A_k over all $x_{(i,k)}$. After this we backtrack back to the point where we decided $\bar{b}_k$.

Our last trail, from which we plan to learn the empty clause, looks as follows:

$$\mathcal{U}_n := (\mathbf{u}, b_1, \ldots, b_n, \bot)$$

with

$$\begin{aligned} \mathrm{ante}_{\mathcal{U}_n}(b_j) &= B_j \\ \mathrm{ante}_{\mathcal{U}_n}(\bot) &= \bar{b}_1 \vee \ldots \vee \bar{b}_n. \end{aligned}$$

We resolve over all b_j and obtain $(\bot)$. □

Combining Corollary 6.17 and Proposition 7.26 shows that $\texttt{MirrorCR}_n$ yields another separation between QCDCL and $\mathsf{QCDCL}^{\textsc{Uni-Any}}$.

Corollary 7.27. $\texttt{MirrorCR}_n$ *is hard for* QCDCL*, but easy for* $\mathsf{QCDCL}^{\textsc{Uni-Any}}$*.*

Corollary 7.28. QCDCL *and* $\mathsf{QCDCL}^{\textsc{Uni-Any}}$ *are exponentially separated on false formulas.*

We combine both separations into our main result:

Theorem 7.29. *(i)* $\mathsf{QCDCL}^{\textsc{Uni-Any}}$ *is exponentially stronger than* QCDCL *on false formulas.*

(ii) $\mathsf{QCDCL}^{\textsc{Exi-Any}}$ *is exponentially stronger than* QCDCL *on true formulas.*

(iii) $\mathsf{QCDCL}^{\textsc{Any}}$ *is exponentially stronger than* QCDCL *both on false and true formulas.*

Chapter 8

Comparing Reduction and Propagation Policies

Besides using more liberal decision heuristics, there are other approaches to modify QCDCL variants in order to improve their performance. While arbitrarily ordered decisions seem to be an obvious advantage even from a high-level perspective, the case for Reduction or Propagation Policies is not quite as clear.

Our main goal will be to propose these types of policies in order to characterise the underlying well-known proof systems like Q-Res, LD-Q-Res or QU-Res (cf. Chapter 9). Both kinds of policies alter the resolution steps that are performed during constraint learning and proof generation in QCDCL.

We introduce Propagation Policies that determine what kinds of constraints are allowed to propagate which types of literals. In the classic setting, we can only use unit clauses to propagate existential literals and unit cubes for propagating universal literals. We can extend these rules by allowing the propagation of universal literals via unit cubes and existential literals via unit clauses. Hence, the extracted proofs now not only contain resolutions between clauses over existential literals, but over universal literals as well. Likewise we can also resolve cubes over existential literals.

As for Reduction Policies, we will show that the way we reduce literals during unit propagation deeply effects the existence of tautological clauses and contradictory cubes in the extracted proofs (i.e., long-distance steps). Roughly speaking, the more often we reduce during unit propagation, the 'more' the proof produced during constraint learning looks like a genuine LD-Q-Res proof and less like a Q-Res proof.

In some cases it makes sense to forego reductions during unit propagation — for example to prevent the learned clause from becoming tautological and therefore difficult to use for further propagations. Sometimes we even want to refrain from doing reductions for the whole QCDCL run because we explicitly want to generate Q-Res proofs (i.e., without any tautological clauses). In order to make reductions more 'controllable', we introduce the notion of *partial reductions*.

Definition 8.1 (partial reduction)**.** *Let K be a non-contradictory set of literals and let*

$C := \ell_1 \vee \ldots \vee \ell_m$ be a clause from a QCNF Φ such that

$$\{\ell_k, \ldots, \ell_m\} = \{\ell \in C |\ \ell \in K,\ \ell \text{ is universal and } x <_\Phi \ell \text{ for all existential } x \in C\}.$$

Then we can partially reduce C by K and obtain

$$red^{\forall}_{\Phi,K}(C) := \ell_1 \vee \ldots \vee \ell_{k-1}.$$

Intuitively, we will reduce all reducible literals that are also contained in K. As before, we simply write red_K instead of $red^{\forall}_{\Phi,K}$ if the context is clear.

Analogously, we can do partial existential reductions of a cube D, which we will denote as $red^{\exists}_{\Phi,K}(D)$ or simply $red^{\forall}_{\Phi,K}(D)$.

We will now apply the idea of partial reductions to our QCDCL framework. Consider a trail

$$\mathcal{T} = (p_{(0,1)}, \ldots, p_{(0,g_0)}; \mathbf{d_1}, p_{(1,1)}, \ldots, p_{(1,g_1)}; \ldots; \mathbf{d_r}, p_{(r,1)}, \ldots, p_{(r,g_r)}),$$

from some QCDCL proof. We now demand that for each point $[i, j]$ in the trail there must exist a set of literals $K_{(i,j)}$ which we call the *reductive set at point* $[i, j]$. Then for each propagated literal $p_{(i,j)} \in \mathcal{T}$ there has to be be a clause $\text{ante}_\mathcal{T}(p_{(i,j)})$ such that

$$\text{red}_{K_{(i,j)}}(\text{ante}_\mathcal{T}(p_{(i,j)})|_{\mathcal{T}[i,j]}) = (p_{(i,j)}),$$

or a cube $\text{ante}_\mathcal{T}(p_{(i,j)})$ such that

$$\text{red}_{K_{(i,j)}}(\text{ante}_\mathcal{T}(p_{(i,j)})|_{\mathcal{T}[i,j]}) = [\bar{p}_{(i,j)}].$$

As before, we call such a clause/cube the *antecedent clause/cube* of $p_{(i,j)}$. Obviously, if $\mathcal{T}$ is a trail for a QCNF Φ, by setting each $K_{(i,j)} := \text{var}(\Phi) \cup \overline{\text{var}(\Phi)}$, we obtain the classic QCDCL.

We will restrict the Decision Policies to level-ordered and arbitrarily ordered decisions since the other types of policies might affect the learnability of asserting constraints and therefore the purpose of partial level-ordered decision policies might not longer apply. The policy for level-ordered decisions will be named LEV-ORD and the one for arbitrary decisions will be named ANY-ORD.

We collect all relevant policies that can be applied to QCDCL in this chapter:

- Decision policies:
  - LEV-ORD: For each decision d_i we have that $\text{lv}_{\Phi|_{\mathcal{T}[i,0]}}(d_i) = 1$. I.e., decisions are level-ordered.
  - ANY-ORD: Decisions can be made arbitrarily in any order (as long as the NCC is fulfilled).
- Reduction policies:

  - ALL-RED: All $K_{(i,j)}$ are set to $\mathrm{var}(\Phi) \cup \overline{\mathrm{var}(\Phi)}$. This is the classic setting – we have to reduce all reducible literals during unit propagation.
  - NO-RED: All $K_{(i,j)}$ are set to $\emptyset$. We are not allowed to reduce during unit propagation at all. There is one exception: Combined with ALL-PROP, we are allowed (but not forced) to reduce universal unit clauses (existential unit cubes) and immediately obtain a conflict. This is due to reasons of completeness which will be explained later.
  - ANY-RED: The sets $K_{(i,j)}$ can be set arbitrarily. Hence, we can choose after each propagation or decisions step which literals are to be reduced next.

- Propagation policies:

  - EXI-PROP: Unit clauses can only propagate existential literals. Universal unit clauses will be reduced to the empty clause if allowed by the reduction policy.
    Analogously, unit cubes can only propagate universal literals. Existential unit cubes will be reduced to the empty cube if allowed by the reduction policy.
  - ALL-PROP: Universal unit clauses will lead to the propagation of the universal unit literal. Analogously, existential unit cubes will lead to the propagation of the existential unit literal.
    This policy is nullified if combined with ALL-RED. If combined with NO-RED, we are allowed to reduce universal unit clauses and existential unit cubes instead of doing a unit propagation. This is due to reasons of completeness.

Having chosen $\mathsf{D} \in \{\text{LEV-ORD}, \text{ANY-ORD}\}$, $\mathsf{R} \in \{\text{ALL-RED}, \text{NO-RED}, \text{ANY-RED}\}$ and $\mathsf{P} \in \{\text{EXI-PROP}, \text{ALL-PROP}\}$, we will denote QCDCL trails that fulfil the corresponding conditions above as $\mathsf{QCDCL}^{\mathsf{D}}_{\mathsf{R},\mathsf{P}}$ trails and use the notion of $\mathsf{QCDCL}^{\mathsf{D}}_{\mathsf{R},\mathsf{P}}$ proofs, refutations and verifications as usual.

Remark 8.2. *We allow (but not force) a QCDCL solver that uses NO-RED together with ALL-PROP to reduce universal unit clauses to the empty clause instead of using them for unit propagation. The following example will explain this tweak:*

Consider the QBF $\forall u \cdot (u)$ and assume, we would not be allowed to reduce universal unit clauses. Then we would need to propagate u as this is the only action available. We will not obtain a conflict and therefore learn the cube $[u]$. After backtracking, we must first propagate $\bar{u}$ via $[u]$, followed by a conflict on (u), which allows us to learn the empty clause.

However, if we would have first propagated u via (u), we would have got a cube conflict on $[u]$, from which we would not be able to learn something new. This might lead to unwanted loops, which should be avoided.

Additionally, this tweak ensures the completeness of the corresponding model on false formulas without the necessity to perform cube learning, which would be otherwise a very unnatural property.

We can show that all combinations of the above policies lead to sound and complete proof systems (and algorithms).

Proposition 8.3. *All defined QCDCL variants are sound and complete.*

Proof. It suffices to show completeness for the weakest combinations. Hence, we can use LEV-ORD and choose between ALL-RED and NO-RED, as both are subsumed by ANY-RED. For EXI-PROP, the version $\mathsf{QCDCL}^{\textsf{LEV-ORD}}_{\textsf{ALL-RED},\textsf{EXI-PROP}}$ is exactly the classic QCDCL variant, for which completeness was proven in Theorem 4.13. The completeness of $\mathsf{QCDCL}^{\textsf{LEV-ORD}}_{\textsf{NO-RED},\textsf{EXI-PROP}}$ can be shown as follows:

Let $\mathcal{T}$ be a trail in a $\mathsf{QCDCL}^{\textsf{LEV-ORD}}_{\textsf{NO-RED},\textsf{EXI-PROP}}$ proof for a QCNF Φ and let $d_1, \ldots, d_r \in \mathcal{T}$ be the decision literals in $\mathcal{T}$. Suppose we run into a clause conflict (the case for cube conflicts is analogue). Then we can consider the rightmost clause C in $\mathfrak{L}(\mathcal{T})$, which can only consist of negated decisions or universal literals that were reduced during unit propagation since we have resolved over everything. Since no reductions were made during unit propagation, C only contains negated decisions and hence must be a new clause, otherwise we would not have been able to make the last decision d_i that also appears as $\bar{d}_i$ in C.

If $\mathcal{T}$ does not run into a conflict, we simply learn an arbitrary cube as described in Definition 4.4. This cube is a new cube, because otherwise we would have got a conflict on exactly that cube.

We conclude that we are always able to find new learnable constraints, hence at some point we will derive the empty clause if Φ is false or the empty cube if Φ is true.

For ALL-PROP, we distinguish two cases:

(i) ALL-RED: Then we will never propagate universal literals via clauses, as they will always be directly reduced to the empty clause. Analogously, we will never propagate existential literals via cubes, as they will always be directly reduced to the empty cube. Hence, this system is the same as if we would have chosen EXI-PROP.

(ii) NO-RED: As described in Remark 8.2, we are not forced to do universal propagations via clauses or existential propagations via cubes. Therefore, the version with EXI-PROP is already simulated by this combination system.

The soundness follows from the soundness of LD-QU-Res (resp. LD-QU-Con) proofs, which can be extracted from the trails of all QCDCL variants defined here. □

While all $2 \times 3 \times 2 = 12$ combinations of policies are sound and complete, we point out that we will not consider all of them. On the one hand, combining the policies ALL-RED and ALL-PROP leads to two QCDCL variants that collapse to the respective versions with EXI-PROP. This is because ALL-RED prevents ALL-PROP from enabling propagations of universal literals via clauses and existential literals via cubes as universal unit clauses and existential unit cubes would be immediately reduced to empty constraints.

On the other hand, we also omit two further combinations using ALL-PROP, namely $\mathsf{QCDCL}^{\textsf{ANY-ORD}}_{\textsf{ANY-RED},\textsf{ALL-PROP}}$ and $\mathsf{QCDCL}^{\textsf{LEV-ORD}}_{\textsf{ANY-RED},\textsf{ALL-PROP}}$ since our simulation method (cf. Chapter 9) does not work for systems with ANY-RED and ALL-PROP. In particular, the proof of Lemma 9.5 from the next chapter, which is crucial for showing the characterisations, requires that ANY-RED is not combined with ALL-PROP due to some technicalities. We

conjecture that $\mathsf{QCDCL}^{\textsf{Any-Ord}}_{\textsf{Any-Red},\textsf{All-Prop}}$ might characterise a proof system such as LD-QU-Res, but we would need a completely different approach.

That said, we consider all remaining eight variants that are included in Figure 9.1. In particular, we analyse all possible variants with the Exi-Prop policy, which is also the standard policy in practical QCDCL.

Obviously, Any-Red covers (hence: simulates) both All-Red an No-Red, as we can simply choose to reduce everything or nothing. We want to prove now that both All-Red and No-Red are exponentially worse than Any-Red on some family of QBFs. I.e., we want to show that there exist formulas where we need to reduce *some* but not *all* literals during unit propagation.

These formulas will be hand-crafted, consisting of two already well-known QCNFs: $\texttt{MirrorCR}_n$ (cf. Definition 6.13) and $\texttt{QParity}_n$ (cf. Definition 5.22).

Proposition 8.4. *The QBFs* $\texttt{MirrorCR}_n$ *have polynomial-size* Q-Res *refutations.*

Proof. As $\texttt{MirrorCR}_n$ is an extension of the Completion Principle ($\texttt{CR}_n$), which is known to be easy for Q-Res [68], we can simply reuse the exact same refutation from [68]. Note that we do not need all axiom clauses to refute the formula. □

In order to combine both formulas, we will rename the variables for $\texttt{QParity}_n$ such that they do not overlap those from $\texttt{MirrorCR}_n$:

The QCNF $\texttt{QParity}_n(Y, w, S)$ consists of the prefix $\exists Y \forall w \exists S$, where $Y := \{y_1, \ldots, y_n\}$ and $S := \{s_2, \ldots, s_n\}$, and the matrix

$$\begin{array}{llll} y_1 \vee y_2 \vee \bar{s}_2 & y_1 \vee \bar{y}_2 \vee s_2 & \bar{y}_1 \vee y_2 \vee s_2 & \bar{y}_1 \vee \bar{y}_2 \vee \bar{s}_2 \\ y_i \vee s_{i-1} \vee \bar{s}_i & y_i \vee \bar{s}_{i-1} \vee s_i & \bar{y}_i \vee s_{i-1} \vee s_i & \bar{y}_i \vee \bar{s}_{i-1} \vee \bar{s}_i \quad \text{for } i \in \{2, \ldots, n\}, \\ s_n \vee w & \bar{s}_n \vee \bar{w}. & & \end{array}$$

When introduced in Chapter 6, it was shown that $\texttt{MirrorCR}_n$ is hard for all QCDCL models with level-ordered decisions considered in Corollary 6.17. We generalize this result and show that the lower bound for $\texttt{MirrorCR}_n$ indeed only depends on the decision policy used and also holds for our new models introduced here.

Proposition 8.5. *The QBFs* $\texttt{MirrorCR}_n(X, u, T)$ *need exponential-sized refutations in all our QCDCL variants with the* Lev-Ord *policy.*

Proof. In Corollary 6.17, it was shown that $\texttt{MirrorCR}_n$ is hard for the QCDCL variant $\mathsf{QCDCL}^{\textsf{Lev-Ord}}_{\textsf{All-Red},\textsf{Exi-Prop}}$ by proving that the QCDCL model generates primitive Q-Res refutations for that formula. This proof is completely independent from the applied reduction policy. Hence, one can easily show that the same result holds for No-Red and Any-Red, as well.

It remains to show that the policy All-Prop does not affect the hardness. This is because QCDCL with Lev-Ord will never be able to propagate universal literals for $\texttt{MirrorCR}_n$.

Assume, for the sake of contradiction, that QCDCL with Lev-Ord propagates a universal literal v in some trail $\mathcal{T}$. W.l.o.g. let this be the first trail and the first time in the trail where this happens. Its antecedent clause $\text{ante}_{\mathcal{T}}(v)$ must contain at least some

T-literal ℓ_1, otherwise v would have been reduced away during the learning of $\text{ante}_{\mathcal{T}}(v)$. But then we need $\bar{\ell}_1 \in \mathcal{T}$ and $\bar{\ell}_1$ has to be assigned before v. It could not be done by decision, otherwise this would contradict the LEV-ORD rule. Therefore $\bar{\ell}_1$ was propagated and there exists its antecedent clause $A_1 := \text{ante}_{\mathcal{T}}(\bar{\ell}_1)$.

Since $\texttt{MirrorCR}_n$ fulfils the XT-property, A_1 cannot be a unit clause or a clause that consists of X- and T-literals, but no universal literals. Because the only universal variable of $\texttt{MirrorCR}_n$ will be assigned later, and we are not allowed to reduce universal literals for the propagation of T-literals, we conclude that A_1 cannot contain universal literals must therefore contain another T-literal t_2.

We can repeat the argument above and find the antecedent clause $A_2 := \text{ante}_{\mathcal{T}}(\bar{t}_2)$, another T-literal t_3, another clause A_3 and so on. Therefore we would detect an infinite amount of different T-literals, which is obviously impossible.

Hence, we have shown that QCDCL cannot propagate universal literals for $\texttt{MirrorCR}_n$ and each QCDCL variant with ALL-PROP behaves the same as the corresponding version with EXI-PROP on that formula. □

With the QBFs $\texttt{QParity}_n$ one obtains one direction of the incomparability between classical QCDCL (here called $\mathsf{QCDCL}^{\textsf{LEV-ORD}}_{\textsf{ALL-RED},\textsf{EXI-PROP}}$) and Q-Res, being easy for the former (cf. Proposition 6.7) and hard for the latter system (cf. [19]).

Proposition 8.6 ([19])**.** *The QBFs* $\texttt{QParity}_n$ *need exponential-sized* Q-Res *and* QU-Res *refutations, but have polynomial-sized* $\mathsf{QCDCL}^{\textsf{LEV-ORD}}_{\textsf{ALL-RED},\textsf{EXI-PROP}}$ *refutations.*

We combine the MirrorCR and QParity formulas into a new one, using auxiliary variables.

Definition 8.7. *The QBF* $\texttt{MiPa}_n$ *consists of the prefix* $\forall z \exists X \forall u \exists T \forall p \exists Y \forall w \exists S \forall v \exists r$ *such that* X*,* u*,* T *are the variables for* $\texttt{MirrorCR}_n(X, u, T)$*, and* Y*,* w*,* S *are the variables for* $\texttt{QParity}_n(Y, w, S)$*. The matrix of* $\texttt{MiPa}_n$ *contains the clauses*

$$\left.\begin{array}{l} C \vee p \vee v \vee r \\ C \vee p \vee \bar{v} \vee r \\ C \vee \bar{p} \vee v \vee r \\ C \vee \bar{p} \vee \bar{v} \vee r \end{array}\right\} \text{for } C \in \mathfrak{C}(\texttt{MirrorCR}_n(X, u, T)),$$

$$\left.\begin{array}{l} p \vee D \\ \bar{p} \vee D \end{array}\right\} \text{for } D \in \mathfrak{C}(\texttt{QParity}_n(Y, w, S)).$$

We show next that $\texttt{MiPa}_n$ needs indeed ANY-RED in order to admit polynomial-size refutations in QCDCL. The idea is that ALL-RED will always lead to refutations of $\texttt{MirrorCR}_n$, and NO-RED will alternatively lead to Q-Res refutations of $\texttt{QParity}_n$, which are both of exponential size.

Theorem 8.8. *The QBFs* $\texttt{MiPa}_n$

(i) need exponential-size $\mathsf{QCDCL}^{\textsf{LEV-ORD}}_{\textsf{ALL-RED},\textsf{EXI-PROP}}$ *refutations,*

(ii) need exponential-size $\mathsf{QCDCL}^{\textsc{Lev-Ord}}_{\textsc{No-Red},\textsc{Exi-Prop}}$ *refutations,*

(iii) but have polynomial-size $\mathsf{QCDCL}^{\textsc{Lev-Ord}}_{\textsc{Any-Red},\textsc{Exi-Prop}}$ *refutations.*

Proof. For (i), since the formula has no unit clauses, we have to start by deciding the variable z. Because z occurs symmetrically in $\mathtt{MiPa}_n$, we can assume that we set z to true. This always triggers the unit propagation of $\bar{r}$ via the clause $\bar{z} \vee \bar{r}$. After that, we are forced to assign the variables from X, $U := \{u\}$ and T along the quantification order. Since the matrix of $\mathtt{MirrorCR}_n$ is unsatisfiable, and we need to reduce all literals if possible, we will detect a conflict at the same time as we would get the conflict in $\mathtt{MirrorCR}_n$ itself. The proof we can extract from the trails is essentially a $\mathsf{QCDCL}^{\textsc{Lev-Ord}}_{\textsc{All-Red},\textsc{Exi-Prop}}$ refutation of $\mathtt{MirrorCR}_n$, except that it additionally contains the variables z, p, v and r in some polarities. However, this does not change the fact that we can still not resolve two clauses that contain X-, U-, and T-variables over any X-variable. Therefore, if we shorten the proof by assigning r to false and z to true, we get a refutation of $\mathtt{MirrorCR}_n$, in which we never resolve two clauses that contain X-, U-, and T-variables over an X-variable – meaning that this refutation has to be primitive (and fully reduced by default). By Corollary 6.15, this refutation has exponential size.

For (ii), we start in the same way as in (i), but we do not get a conflict once we assigned all variables of $\mathtt{MirrorCR}_n$. Next, we need to decide p in some polarity, but nothing will happen for the moment. We then start assigning the variables of $\mathtt{QParity}_n$ along the quantification order. Now we have to distinguish two cases:

Case 1. We get a conflict in $\mathtt{QParity}_n$.

But then, because of NO-RED, we can only extract Q-Res derivations of learned clauses. And if we get enough conflicts in $\mathtt{QParity}_n$, we can essentially extract a Q-Res refutation of $\mathtt{QParity}_n$, which has exponential size.

Case 2. We do not get a conflict in $\mathtt{QParity}_n$.

This might happen when the universal player assigns the variable w the "wrong" way. Then the only unassigned variable is v. After deciding it in any polarity, we will always get a conflict in $\mathtt{MirrorCR}_n$. If we find enough conflicts in $\mathtt{MirrorCR}_n$, we can essentially extract an exponential-size fully reduced primitive Q-Res refutation of $\mathtt{MirrorCR}_n$ as in (i).

Note that it is possible to get both kind of conflicts. However, it is only important with what kind of conflicts we were able to derive the empty clause.

Finally, for (iii), we can construct a polynomial-size $\mathsf{QCDCL}^{\textsc{Lev-Ord}}_{\textsc{Any-Red},\textsc{Exi-Prop}}$ proof by only reducing the literals w and $\bar{w}$. After deciding z, propagating $\bar{r}$, assigning all variables from X, u and T and deciding p arbitrarily, we can simply copy the polynomial-size $\mathsf{QCDCL}^{\textsc{Lev-Ord}}_{\textsc{All-Red},\textsc{Exi-Prop}}$ proof of $\mathtt{QParity}_n$ (note that ALL-RED only applies to w and $\bar{w}$). At some point, we will derive the clause (p) or $(\bar{p})$, which can be reduced to the empty clause. □

One of our motivations is to find a way to p-simulate LD-Q-Res refutations of QCNFs by certain variants of QCDCL. However, it appears that not all resolution steps that are allowed in LD-Q-Res can be recreated with QCDCL proofs. In LD-Q-Res proofs that are

extracted from QCDCL, one can easily observe that for each resolution step $C_1 \overset{\ell}{\bowtie} C_2$, at least one parent clause C_i has to be an antecedent clause for ℓ or $\bar{\ell}$ in the corresponding trail. In particular, there must be a partial assignment τ and a set of literals K such that $\mathrm{red}_K(C_i|_\tau)$ becomes unit, i.e. $\mathrm{red}_K(C_i|_\tau) = (\ell)$ (resp. $(\bar{\ell})$). This is not possible if there are tautologies left of ℓ in C_i that cannot be reduced.

Motivated by this observation, we introduce a new proof system similar to LD-Q-Res, but with the restriction that such a situation as described above is not allowed.

Definition 8.9. *A* LD-Q-Res *proof is called a* mLD-Q-Res proof*, if it does not contain a resolution step between two clauses D and E, such that $C = D \overset{x}{\bowtie} E$ for an existential variable x and there are universal variables u, w such that $u^* \in D$, $w^* \in E$ and $lv_\Phi(u), lv_\Phi(w) < lv_\Phi(x)$.*

With this definition in place, we can show that mLD-Q-Res proofs can be extracted from runs of most variants of QCDCL that we defined. Further, for some QCDCL paradigms, stricter simulations hold.

Proposition 8.10. *The following holds on false QCNFs:*

(i) Q-Res *p-simulates* $\mathsf{QCDCL}^{\textsc{Any-Ord}}_{\textsc{No-Red},\textsc{Exi-Prop}}$.

(ii) QU-Res *p-simulates* $\mathsf{QCDCL}^{\textsc{Any-Ord}}_{\textsc{No-Red},\textsc{All-Prop}}$.

(iii) mLD-Q-Res *p-simulates* $\mathsf{QCDCL}^{\textsc{Any-Ord}}_{\textsc{Any-Red},\textsc{Exi-Prop}}$.

Proof. For (i), we already know that the classic version QCDCL generates LD-Q-Res refutations by the definition of Clause Learning (cf. Definition 4.4). Because of No-Red, all literals in the learnable clauses have to appear negatively in the corresponding trail. Since trails never contain variables in two different polarities, learnable clauses can never include universal tautologies. Hence, the extracted proofs are always Q-Res proofs.

For (ii), because of the All-Prop policy, we might propagate (and resolve) over universal literals, which can be handled by QU-Res. It remains to show that No-Red prevents the derivation of tautological clauses. This holds because we only use antecedent clauses for Clause Learning. Let us assume that we learn a tautological clause C from a $\mathsf{QCDCL}^{\textsc{Any-Ord}}_{\textsc{No-Red},\textsc{All-Prop}}$ trail $\mathcal{T}$. Then there would be two antecedent clauses $D := \mathrm{ante}_\mathcal{T}(\ell_1)$ and $E := \mathrm{ante}_\mathcal{T}(\ell_2)$ such that there exists a universal literal u with $u \neq \ell_1$, $\bar{u} \neq \ell_2$, $u \in D$ and $\bar{u} \in E$. We need $\bar{u} \in \mathcal{T}$ for D to become unit and at the same time we need $u \in \mathcal{T}$ for E to become unit, which is not possible. Therefore, we will never derive tautological clauses.

Let us now prove (iii). By definition, via Clause Learning we can extract LD-Q-Res proofs from $\mathsf{QCDCL}^{\textsc{Any-Ord}}_{\textsc{Any-Red},\textsc{Exi-Prop}}$ trails (note that we only propagate existential literals, hence we also only resolve over existential variables during Clause Learning). It remains to show that the kind of resolution step that is forbidden in mLD-Q-Res (but allowed in LD-Q-Res) will never occur during Clause Learning.

Assume it does. Then we have derived a clause C by resolving two clauses D and E over some literal x (hence $C = D \overset{x}{\bowtie} E$), such that there exists universal tautologies

$u^* \in D$ and $w^* \in E$ with $u^* \neq w^*$ and $\mathrm{lv}(u^*), \mathrm{lv}(w^*) < \mathrm{lv}(x)$. Then at least one of these parent clauses needs to be an antecedent clause for a trail $\mathcal{T}$, say $D = \mathrm{ante}_{\mathcal{T}}(x)$. But then D can never become the unit clause (x), because we cannot reduce u^* since it is blocked by x, and we cannot falsify it by the previous trail assignment since it is a tautology. This is a contradiction that shows that all resolution and reduction steps are allowed in mLD-Q-Res. □

We could formulate analogous results on true QCNFs using the notation of consensus proofs. However, we will omit this as all separations and characterisations will be performed on false QCNFs and resolution proofs.

One can easily show that the separation between Q-Res and LD-Q-Res transfers to a separation between Q-Res and mLD-Q-Res.

Corollary 8.11. mLD-Q-Res *p-simulates and is exponentially stronger than* Q-Res.

Proof. The simulation follows by definition. The separation follows by Proposition 8.6 and 8.10 (iii). □

In fact, all currently known upper bounds for LD-Q-Res can be easily transformed into mLD-Q-Res upper bounds. However, we leave open the question of whether LD-Q-Res is stronger than or equivalent to mLD-Q-Res.

At the end, we want to show that the two policies ALL-RED and NO-RED are in fact incomparable to each other (at least as long as LEV-ORD and EXI-PROP are activated). We already know that $\mathtt{QParity}_n$ is easy for $\mathsf{QCDCL}^{\textsf{LEV-ORD}}_{\textsf{ALL-RED},\textsf{EXI-PROP}}$ and hard for Q-Res by Proposition 8.6.

Since Q-Res p-simulates $\mathsf{QCDCL}^{\textsf{ANY-ORD}}_{\textsf{NO-RED},\textsf{EXI-PROP}}$ and therefore $\mathsf{QCDCL}^{\textsf{LEV-ORD}}_{\textsf{NO-RED},\textsf{EXI-PROP}}$ as well, we conclude that $\mathtt{QParity}_n$ must also be hard for $\mathsf{QCDCL}^{\textsf{LEV-ORD}}_{\textsf{NO-RED},\textsf{EXI-PROP}}$, which separates ALL-RED from NO-RED in one direction.

For the other direction, it suffices to construct a formula that is easy with NO-RED, but hard with ALL-RED. We can follow an approach similar to the $\mathtt{MiPa}_n$ formulas:

Definition 8.12. *The QBF* $\mathtt{MiEasy}_n$ *consists of the prefix* $\forall z \exists X \forall u \exists T \forall p \exists y \forall v \exists r$ *such that* X, u, T *are the variables for* $\mathtt{MirrorCR}_n(X, u, T)$. *The matrix of* $\mathtt{MiEasy}_n$ *contains the clauses*

$$
\begin{array}{l}
z \vee \bar{r}, \quad \bar{z} \vee \bar{r} \\
\left.\begin{array}{l} C \vee p \vee v \vee r \\ C \vee p \vee \bar{v} \vee r \\ C \vee \bar{p} \vee v \vee r \\ C \vee \bar{p} \vee \bar{v} \vee r \end{array}\right\} \textit{for } C \in \mathfrak{C}(\mathtt{MirrorCR}_n(X, u, T)), \\
p \vee y \\
\bar{p} \vee y \\
p \vee \bar{y} \\
\bar{p} \vee \bar{y}
\end{array}
$$

Proposition 8.13. *The QBFs* $\texttt{MiEasy}_n$ *need exponential-size* $\mathsf{QCDCL}^{\textsc{Lev-Ord}}_{\textsc{All-Red},\textsc{Exi-Prop}}$ *refutations, but have polynomial size* $\mathsf{QCDCL}^{\textsc{Lev-Ord}}_{\textsc{No-Red},\textsc{Exi-Prop}}$ *refutations.*

Proof. If ALL-RED is activated, we have to start deciding z somehow, say we set z to true. Then we propagate $\bar{r}$, which now no longer blocks the universal variables p and v in the clauses that contain clauses from $\texttt{MirrorCR}_n$. Next, we need to assign the variables from X, u and T, either by decision or propagation. No matter how we assign these variables, we will always get a conflict since one of the $C \in \mathfrak{C}(\texttt{MirrorCR}_n)$ will get falsified and we are forced to reduce p and v. That means we essentially have to refute $\texttt{MirrorCR}_n$ on $\mathsf{QCDCL}^{\textsc{Lev-Ord}}_{\textsc{All-Red},\textsc{Exi-Prop}}$, which is hard by Proposition 8.5.

However, if NO-RED is activated, we no longer have to reduce p and v and can move to the clauses that contain y, which are easy to refute. In fact, we can refute the formula in just one trail:

$$\mathcal{T} := (\mathbf{z}, r; \underbrace{\ldots}_{\text{arbitrary decisions of } X,\ u \text{ and } T}; \mathbf{p}, y, \bot)$$

with

$$\begin{aligned} \text{ante}_{\mathcal{T}}(r) &= \bar{z} \vee \bar{r}, \\ \text{ante}_{\mathcal{T}}(y) &= \bar{p} \vee y, \\ \text{ante}_{\mathcal{T}}(\bot) &= \bar{p} \vee \bar{y}. \end{aligned}$$

By resolving over y, we can learn $(\bar{p} \vee \bar{y}) \overset{y}{\bowtie} (\bar{p} \vee y) = (\bar{p})$ and reduce it to $(\bot)$. Note that NO-RED does only affect reductions during Unit Propagation and not during Clause Learning. □

Corollary 8.14. $\mathsf{QCDCL}^{\textsc{Lev-Ord}}_{\textsc{All-Red},\textsc{Exi-Prop}}$ *and* $\mathsf{QCDCL}^{\textsc{Lev-Ord}}_{\textsc{No-Red},\textsc{Exi-Prop}}$ *are incomparable.*

Chapter 9

QCDCL Characterisations

In this chapter, we show that all the simulations in Proposition 8.10 can be tightened to equivalences. For this we will characterise Q-Res, mLD-Q-Res and QU-Res by the specific variants of QCDCL mentioned in Proposition 8.10. However, we leave open whether we can extend these characterisations to LD-Q-Res. This will depend on whether it is possible to polynomially transform the 'forbidden' resolution steps that can occur in LD-Q-Res, but cannot be created by QCDCL, into mLD-Q-Res steps.

We will show the characterisations of mLD-Q-Res and QU-Res parallelly as the characterisation of Q-Res is a special case for the former. Hence, we will temporarily only concentrate on the QCDCL variants $\mathsf{QCDCL}^{\textsf{Any-Ord}}_{\textsf{Any-Red},\textsf{Exi-Prop}}$ and $\mathsf{QCDCL}^{\textsf{Any-Ord}}_{\textsf{No-Red},\textsf{All-Prop}}$ as well as their underlying proof systems mLD-Q-Res and QU-Res.

Besides choosing the "correct" decisions, one of the main challenges of characterising mLD-Q-Res is to find suitable reductive sets. On the one hand, they have to trigger the derivation of desired tautological formulas, but on the other hand they need to ensure that we do not learn too many tautological clauses since they can become useless for unit propagation quite quickly.

Although it is allowed to define all reductive sets $K_{(i,j)}$ differently, it might make sense from a practical perspective to weaken these possibilities. We point out three nuances of partial reduction in QCDCL that are interesting to consider:

(i) We change the reductive set after each propagation or decision step. That means that all sets $K_{(i,j)}$ might be different. This is the strongest possible version of partial reduction.

(ii) We only update the reductive set after backtracking. That means the sets $K_{(i,j)}$ are constant for each trail. This version is enough to simulate all resolution steps for the characterisation of mLD-Q-Res (cf. Theorem 9.6).

(iii) We never change the reductive set. That means that the sets $K_{(i,j)}$ remain constant throughout the whole QCDCL proof. This version is enough for the separation between systems with and systems without partial reduction (cf. Theorem 8.8).

In particular that means that for the most results regarding the characterisation of mLD-Q-Res, it is enough to fix the literals that are going to be reduced throughout the whole trail. Thus, we introduce the notion of *L-reductive trails*.

Definition 9.1 (L-reductive trails)**.** *Let L be a set of literals. A trail $\mathcal{T}$ is called* L-reductive*, if for each propagation step in $\mathcal{T}$ the literals that were selected to be reduced are exactly the literals in L. Formally, this means that for each $p_{(i,j)}$ there is an antecedent clause (resp. cube) $ante_{\mathcal{T}}(p_{(i,j)})$ such that $red_L(ante_{\mathcal{T}}(p_{(i,j)})|_{\mathcal{T}[i,j]}) = (p_{(i,j)})$ (resp. $[\bar{p}_{(i,j)}]$).*

Before starting with a new L-reductive trail, we always need to consider the choice of the reductive set L. Since $\mathsf{QCDCL}^{\textsf{Any-Ord}}_{\textsf{No-Red,Exi-Prop}}$ is already simulated by Q-Res, we can conclude that in some sense the only purpose of reductions during unit propagation is to create tautological clauses. Therefore we will distinguish between the tautological and the non-tautological part of a clause.

Definition 9.2. *Let C be a clause. Let $G(C) := \{u \in C : u \text{ is universal and } \bar{u} \in C\}$. This set is the* tautological part *of C. The* non-tautological part *$H(C)$ of C is defined as $H(C) := C \backslash G(C)$.*

Note that for each QU-Res proof π and $C \in \pi$ we have $G(C) = \emptyset$ as all the clauses are non-tautological.

Our next notion is similar to the concepts of *unreliable* [14] and *1-empowering* [86].

Definition 9.3 (Blockades)**.** *Let $\mathsf{S} \in \{\mathsf{QCDCL}^{\textsf{Any-Ord}}_{\textsf{Any-Red,Exi-Prop}}, \mathsf{QCDCL}^{\textsf{Any-Ord}}_{\textsf{No-Red,All-Prop}}\}$ and C be a clause. A tuple $(\mathcal{U}, \alpha, \ell, K)$, where $\mathcal{U}$ is a trail, ℓ is a literal, α is a non-tautological set of literals and K is a set of universal literals, is called a* blockade of C with respect to S *for a QCNF $\Phi = \mathcal{Q} \cdot \varphi$, if $\mathcal{U}$ is a K-reductive S trail with decisions α, such that $\ell \in C$, $\ell \in \mathcal{U} \backslash \alpha$, $\alpha \subseteq \overline{C} \backslash \{\bar{\ell}\}$, $K \subseteq G(C)$ and $\alpha \cap K = \emptyset$. In particular, ℓ is a propagated literal in $\mathcal{U}$.*

For $\mathsf{S} = \mathsf{QCDCL}^{\textsf{Any-Ord}}_{\textsf{Any-Red,Exi-Prop}}$, we additionally require that ℓ is an existential literal and α consists of only existential literals.

Example 9.4. *Blockades occur when we are not able to choose all decisions from a pre-defined non-tautological set α. For example, consider the QCNF*

$$\exists x, y \forall u, v \exists z \ (\bar{y} \vee \bar{z}) \wedge (\bar{x} \vee \bar{u} \vee z) \wedge (x \vee y \vee v \vee z) \wedge (y \vee \bar{v} \vee z).$$

Assume that we use $\mathsf{QCDCL}^{\textsf{Any-Ord}}_{\textsf{Any-Red,All-Prop}}$. Then the clause $C := \bar{x} \vee \bar{y} \vee u \vee \bar{u} \vee z$ has a blockade $(\mathcal{U}, \alpha, \ell, K)$ with $\mathcal{U} := (\mathbf{y}, \bar{z}, \bar{x})$, where $ante_{\mathcal{U}}(\bar{z}) = \bar{y} \vee \bar{z}$, $ante_{\mathcal{U}}(\bar{x}) = \bar{x} \vee \bar{u} \vee z$, as well as $\ell := \bar{x} \in C$, $\alpha := \{y\} \subseteq \overline{C} \backslash \{\bar{\ell}\}$ and $K := \{\bar{u}\}$.

Intuitively, this means that although the clause C is not directly contained in the formula, we are still able to detect the implication $(\alpha \wedge \overline{K} \to \ell) = (y \wedge u) \to \bar{x}$ (which is equivalent to $\bar{y} \vee \bar{u} \vee \bar{x} \subseteq C$) as a composition of decisions and unit propagations. It turns out that, instead of learning C directly, it is enough to detect a blockade in order to make use of C for unit propagations in later trails.

The next lemma shows, that we can recall trails (and blockades in particular), that were detected and stored at an earlier point, and restore all propagations they contained. This will be important for the characterisations, as we will go through the given proof, find blockades or conflicts for all clauses in that proof, and recall the corresponding trails (by using this Lemma) an all their containing propagations whenever the clauses are needed for another resolution step. In that way, we can virtually store previous trails and recall them later again as natural trails.

Lemma 9.5. *Let $\Phi = \mathcal{Q} \cdot \varphi$ and $\Psi = \mathcal{Q} \cdot \psi$ be QCNFs such that $\psi \subseteq \varphi$.*

Let $\mathsf{S} \in \{\mathsf{QCDCL}^{\textsc{Any-Ord}}_{\textsc{Any-Red},\textsc{Exi-Prop}}, \mathsf{QCDCL}^{\textsc{Any-Ord}}_{\textsc{No-Red},\textsc{All-Prop}}\}$ and let $\mathcal{U}$ be a K-reductive S trail (for NO-RED we set $K = \emptyset$) for the QCNF Ψ with decisions β. Let $\mathcal{T}$ be a natural L-reductive S trail ($L = \emptyset$ for NO-RED) with decisions α for the QCNF Φ such that $K \subseteq L$, $\beta \subseteq \mathcal{T}$ and $\alpha \cap L = \emptyset$. If $\mathcal{T}$ does not run into a clause conflict, then all propagated literals from $\mathcal{U}$ are also contained in $\mathcal{T}$.

Proof. Assume that $\mathcal{T}$ does not run into a clause conflict, but there are some propagated literals from $\mathcal{U}$ that are not contained in $\mathcal{T}$. Let $p_{(a,b)}$ be the literal that is leftmost in $\mathcal{U}$ with this property and define $A := \mathrm{ante}_{\mathcal{U}}(p_{(a,b)})$. Since there are no cubes present, we conclude that A must be a clause, regardless of whether $p_{(a,b)}$ is existential or universal.

Because $p_{(a,b)}$ is leftmost, all other propagated literals before $p_{(a,b)}$ in $\mathcal{U}$ are already contained in $\mathcal{T}$. Since $\mathcal{U}$ was K-reductive, we know that $\mathrm{red}_K(A|_{\mathcal{U}[a,b]}) = (p_{(a,b)})$. Because of $K \subseteq L$ and $\mathcal{U}[a,b] \subseteq \mathcal{T}$ we have either $\mathrm{red}_L(A|_{\mathcal{T}}) \in \{(p_{(a,b)}), (\bot)\}$, or $A|_{\mathcal{T}}$ becomes true. Note that we can set $K := L := \emptyset$ for the rest of our argumentation in the case where $p_{(a,b)}$ is universal. This is because we require the NO-RED policy to be active if ALL-PROP is active as well.

The first case would contradict our assumption (since $\mathcal{T}$ is natural), therefore we have to assume that $A|_{\mathcal{T}}$ becomes true. This means that we can find a literal $p_{(a,b)} \neq u \in A \cap \mathcal{T}$. If u was existential, then we would need $\bar{u} \in \mathcal{U}[a,b]$. But this would also imply $\bar{u} \in \mathcal{T}$ which contradicts the fact that $u \in \mathcal{T}$. Hence u must be universal.

If u was a decision in $\mathcal{T}$, then we would have $u \in \alpha$. Because of $\alpha \cap L = \emptyset$ we conclude $u \notin L$ and also $u \notin K$. In order to make u vanish in $\mathrm{red}_K(A|_{\mathcal{U}[a,b]})$, we need $\bar{u} \in \mathcal{U}[a,b]$, hence also $\bar{u} \in \mathcal{T}$. However, this is a contradiction because we already assumed $u \in \mathcal{T}$.

Therefore, u must have been propagated by an antecedent clause $\mathrm{ante}_{\mathcal{T}}(u)$. But then we have $K = \emptyset$, hence $u \notin K$ and $\bar{u} \in \mathcal{U}[a,b] \subseteq \mathcal{T}$, which is a contradiction again because of $u \in \mathcal{T}$. □

In the next theorem we will prove the main result: There exist two QCDCL variants that can p-simulate mLD-Q-Res and QU-Res respectively. The other direction was already proven in Proposition 8.10, therefore we essentially prove the equivalence of these systems.

Theorem 9.6. *The following holds:*

- $\mathsf{QCDCL}^{\textsc{Any-Ord}}_{\textsc{Any-Red},\textsc{Exi-Prop}}$ *p-simulates* mLD-Q-Res.
- $\mathsf{QCDCL}^{\textsc{Any-Ord}}_{\textsc{No-Red},\textsc{All-Prop}}$ *p-simulates* QU-Res.

In detail: Let $\Phi = \mathcal{Q} \cdot \varphi$ be a QCNF in n variables and $\pi = D_1, \ldots, D_m$ be a mLD-Q-Res *(*QU-Res*) refutation of Φ. Then we can construct a* $\mathsf{QCDCL}^{\textsc{Any-Ord}}_{\textsc{Any-Red},\textsc{Exi-Prop}}$ *(*$\mathsf{QCDCL}^{\textsc{Any-Ord}}_{\textsc{No-Red},\textsc{All-Prop}}$*) refutation ι of Φ with $|\iota| \in \mathcal{O}(n \cdot |\pi|)$. Furthermore, all trails from ι can be constructed such that they run into clause conflicts, meaning that we will only learn clauses in ι.*

Note that the fact that we only learn clauses in the QCDCL proof not only strengthens the result (a possibly weaker QCDCL system suffices for the simulations), but it also simplifies the simulation itself as several auxiliary results below will rely on the assumption that no cubes are learned. Before giving the full proof of Theorem 9.6, which involves several auxiliary results, we will sketch the proof idea.

Proof sketch of Theorem 9.6. Going through a given mLD-Q-Res (QU-Res) refutation π, starting at the axioms, for each $C \in \pi$ we create specific natural trails (where some of them will later be part of the $\mathsf{QCDCL}^{\textsc{Any-Ord}}_{\textsc{Any-Red},\textsc{Exi-Prop}}$ or $\mathsf{QCDCL}^{\textsc{Any-Ord}}_{\textsc{No-Red},\textsc{All-Prop}}$ proof) in which all decisions are negated literals from C, until one of the following events occur:

- We get a conflict and learn a subclause of C.
- We obtain a blockade of C.

When this happens, we either assign the label "subclause" or the label "blockade" to C. When a clause was derived via a resolution or reduction step in π, we simply recall the blockades of its parent clauses by applying Lemma 9.5 to create a blockade for the resolvent or a conflict. If a parent clause does not have a blockade, the clause itself (or a subclause) must have been learned directly and can therefore be used as an antecedent clause for the trail that either becomes a blockade for the resolvent, or that runs into a conflict from which we can learn a subclause of the resolvent.

Since a clause $C \in \pi$ can be derived via resolution (say $C = D \bowtie E$) or reduction (say $C = \mathrm{red}(D)$), we have to consider all possible cases:

(i) resolution, both D and E are labelled "blockade" (cf. Lemma 9.8)

(ii) resolution, D is labelled "blockade", E is labelled "subclause", or vice versa (cf. Lemma 9.9)

(iii) resolution, both D and E are labelled "subclause" (cf. Lemma 9.10)

(iv) reduction, D is labelled "blockade" (cf. Lemma 9.11)

(v) reduction, D is labelled "subclause" (cf. Lemma 9.12)

At the end, each clause in π is either labelled "subclause" or "blockade". In particular, this holds for the empty clause. Because, by definition, there cannot be a blockade of the empty clause (we need at least one literal), the empty clause must be labelled "subclause", which means we have learned the empty clause. □

We now proceed with the full proof of Theorem 9.6. Before doing so, we need a couple of technical lemmas.

The aforementioned creation of specific natural trails is determined by their decisions as well as reductive sets $K_{(i,j)}$. In particular, it is not important in which order propagations are performed as long as they are valid. The following remark explains the way we create these trails in more detail.

Remark 9.7. *Let $\Phi = \mathcal{Q} \cdot \varphi$ be a QCNF and L be a set of universal literals. Let α be a non-tautological set of literals. Then we can construct a trail $\mathcal{T}$ for Φ by choosing α in a specific order as decision literals and propagating literals as soon as the corresponding antecedent clauses become L-reductive unit clauses. We are allowed to update the set L after each propagation. We can even undertake these automatic construction steps after backtracking. However, it is possible that we propagate a literal from α in the same polarity before deciding it. In this case we have to skip the decision. Also, we could reach a conflict before deciding all literals, then we abort the trail as usual.*

If we propagate a literal from $\bar{\alpha}$, then we also abort.

For the next lemmas, we will construct natural trails from a given set α of decision literals. Our goal will to obtain a blockade or a conflict. We will always assume that we start with all existential literals from α before deciding universal literals. In particular, for the simulation of mLD-Q-Res, where we use EXI-PROP instead of ALL-PROP, all blockades $(\mathcal{U}, \alpha, \ell, K)$ will consist of existential decisions α, but universal reductions K. Therefore we can guarantee $\alpha \cap K = \emptyset$.

The first lemma handles the case where we want to simulate a resolution step between two clauses such that we have already detected blockades for both of them (i.e., both parental clauses are labelled "blockade"). This corresponds to Case (i) from the proof sketch of Theorem 9.6.

Lemma 9.8. *Let $\Phi = \mathcal{Q} \cdot \varphi$ be a QCNF. Let further $C \vee x$ and $D \vee \bar{x}$ two clauses such that $C \vee D = C \vee x \overset{x}{\bowtie} D \vee \bar{x}$ is a valid* mLD-Q-Res *(*QU-Res*) step. Suppose that there exists a blockade of $C \vee x$ for $\Psi = \mathcal{Q} \cdot \psi$ and a blockade of $D \vee \bar{x}$ for $\Gamma = \mathcal{Q} \cdot \gamma$ with $\psi, \gamma \subseteq \varphi$. Then there exists a $\mathsf{QCDCL}^{\textsc{Any-Ord}}_{\textsc{Any-Red},\textsc{Exi-Prop}}$ ($\mathsf{QCDCL}^{\textsc{Any-Ord}}_{\textsc{No-Red},\textsc{All-Prop}}$) proof*

$$\iota = [(\mathcal{T}_i, C_i, \pi_i)]_{i=1}^{c}$$

from Φ with a constant c, such that $C_c \subseteq C \vee D$ or there exists a blockade of $C \vee D$ for $\mathcal{Q} \cdot (\varphi \cup \{C_1, \ldots, C_c\})$.

Proof. Let the blockade of $C \vee x$ for Ψ be $(\mathcal{U}_1, \alpha_1, \ell_1, K_1)$. Analogously let $(\mathcal{U}_2, \alpha_2, \ell_2, K_2)$ be the blockade of $D \vee \bar{x}$ for Γ with respect to corresponding QCDCL model.

Case 1. $\ell_1 = x$ and $\ell_2 = \bar{x}$.

We construct a natural $G(C \vee D)$-reductive trail $\mathcal{T}$ with decisions $\alpha := \alpha_1 \cup \alpha_2 \subseteq \overline{C \vee D}$. If we obtain a blockade, we are done. Note that the set of decisions that were actually made and $G(C \vee D)$ is always disjoint, because for the EXI-PROP variant, α consists of existential literals only, while for the ALL-PROP variant, $G(C \vee D)$ is empty.

If we run into a conflict, then we can start Clause Learning and learn the rightmost clause E in $\mathfrak{L}_{\mathcal{T}}$. Then E can only contain literals from $\bar{\alpha} \subseteq C \vee D$ or $G(C \vee D) \subseteq C \vee D$. In this case we are also done.

Suppose that we do not get a blockade and do not run into a conflict. Then we have $\alpha \subseteq \mathcal{T}$. By Lemma 9.5, each propagation from $\mathcal{U}_1$ as well as $\mathcal{U}_2$ is contained in $\mathcal{T}$. Note that we have $K_1 \cup K_2 \subseteq G(C \vee D)$. But then we would have $x, \bar{x} \in \mathcal{T}$, which is a contradiction.

Case 2. $\ell_1 = x$ and $\ell_2 \neq \bar{x}$ (or analogously $\ell_1 \neq x$ and $\ell_2 = \bar{x}$).

We construct a natural $G(C \vee D)$-reductive trail $\mathcal{T}$ with decisions $\alpha := (\alpha_1 \cup \alpha_2 \cup \{\bar{\ell}_2\}) \backslash \{x\} \subseteq \overline{C \vee D}$ (note that x might be contained in α_2). Similar to Case 1, we are done if we get a blockade or run into a conflict.

Otherwise we would have $\alpha \subseteq \mathcal{T}$. By Lemma 9.5, we conclude $\ell_1 = x \in \mathcal{T}$. This means $\alpha_2 \subseteq \mathcal{T}$. Again, by Lemma 9.5 we would get $\ell_2 \in \mathcal{T}$, which is a contradiction to $\bar{\ell}_2 \in \mathcal{T}$. Hence we always get a blockade or a conflict.

Case 3. $\ell_1 \neq x$ and $\ell_2 \neq \bar{x}$.

W.l.o.g. let C not contain a universal tautology $u \vee \bar{u}$ with $\mathrm{lv}(u) < \mathrm{lv}(x)$. We can make this assumption because the resolution step is valid for mLD-Q-Res (and also for QU-Res).

We construct a natural K_1-reductive trail $\mathcal{T}$ with decisions $\alpha := \alpha_1 \cup \{\bar{\ell}_1\}$, but we will decide $\bar{x}$ at the end (if x or $\bar{x}$ does not get propagated before). If we run into a conflict without deciding $\bar{x}$, then we can again learn the rightmost clause E in $\mathfrak{L}_{\mathcal{T}}$ which is a subclause of C and therefore a subclause of $C \vee D$. Assume we get a blockade of $C \vee x$ with a literal $\ell \in \bar{\alpha} \subseteq C \vee x$. If $\ell \neq x$, then this is a blockade of C (and also a blockade of $C \vee D$) since $\bar{x}$ was not decided, yet. If $\ell = x$, then we have propagated x before deciding $\bar{x}$. But then we can go to Case 2 with the trails $\mathcal{T}$ and $\mathcal{U}_2$.

If we do not get a blockade and do not run into a conflict without deciding $\bar{x}$, and if we actually decide $\bar{x}$ at the end, we will show that we will run into a conflict afterwards. Assume not. Then we have $\alpha_1 \subseteq \alpha \subseteq \mathcal{T}$. By Lemma 9.5, we conclude that $\ell_1 \in \mathcal{T}$, which is a contradiction to $\bar{\ell}_1 \in \alpha \subseteq \mathcal{T}$. Therefore we run into a conflict.

Then we again learn the rightmost clause E in $\mathfrak{L}_{\mathcal{T}}$, which is now a subclause of $\bar{\alpha} \vee K_1 \subseteq \bar{\alpha} \vee G(C)$ with $x \in E$ (because $\bar{x}$ was the last decision and the last decision always contributes to the conflict). If $\bar{x}$ was the r^{th} decision, we backtrack back to $\mathcal{T}[r, 0]$ (right before the decision $\bar{x}$ was made). Because $\bar{x}$ was the last decision, we have $\alpha \backslash \{\bar{x}\} \subseteq \mathcal{T}[r, 0]$.

Our precondition at the beginning was that C does not contain a universal tautology left of x. In particular, for all $u \in G(C)$ we have $\mathrm{lv}(u) \geq \mathrm{lv}(x)$. We conclude

$$\mathrm{red}_{K_1}(E|_{\mathcal{T}[r,0]}) = (x).$$

Finally, we propagate x, obtain the new trail $\mathcal{T}'$ (which is $\mathcal{T}[r, 0]$ plus x) and go into Case 2 again. Note that $\mathcal{T}'$ is still a K_1-reductive trail, even after backtracking.

The number of backtracking steps and restarts is obviously bounded by a constant. Note that for QU-Res, we construct $\emptyset$-reductive trails because $G(C \vee D) = \emptyset$, which means that we can activate NO-RED. □

In the next lemma, we want to simulate a resolution step between a clause for which we have already detected a blockade, and a second clause that is subsumed by a previously learned clause or an axiom (i.e., one clause is labelled "blockade" while the other is labelled "subclause"). This corresponds to Case (ii) from the proof sketch of Theorem 9.6.

Lemma 9.9. *Let $\Phi = \mathcal{Q} \cdot \varphi$ be a QCNF. Let further $C \vee x$ and $D \vee \bar{x}$ two clauses such that $C \vee D = C \vee x \overset{x}{\bowtie} D \vee \bar{x}$ is a valid* mLD-Q-Res *(*QU-Res*) step. Suppose that there exists a blockade of $C \vee x$ for $\Psi = \mathcal{Q} \cdot \psi$ with $\psi \subseteq \varphi$. Suppose also there exists a subclause $D' \subseteq D \vee \bar{x}$ with $D' \in \varphi$. Then there exists a* $\mathsf{QCDCL}^{\textsc{Any-Ord}}_{\textsc{Any-Red},\textsc{Exi-Prop}}$ *(*$\mathsf{QCDCL}^{\textsc{Any-Ord}}_{\textsc{No-Red},\textsc{All-Prop}}$*) proof*

$$\iota = [(\mathcal{T}_i, C_i, \pi_i)]_{i=1}^{c}$$

from Φ with a constant c, such that $C_c \subseteq C \vee D$ or there exists a blockade of $C \vee D$ for $\mathcal{Q} \cdot (\varphi \cup \{C_1, \ldots, C_c\})$.

Proof. Let the blockade of $C \vee x$ be $(\mathcal{U}_1, \alpha_1, \ell_1, K_1)$.

Case 1. $\ell_1 = x$.

Construct a natural $G(C \vee D)$-reductive trail $\mathcal{T}$ with decisions $\alpha := (\alpha_1 \cup \bar{D}) \backslash G(C \vee D)$ such that we decide existential literals first (again, this is only important for mLD-Q-Res). If we get a blockade, we are done, as for mLD-Q-Res we could have only decided existential literals from $\alpha \cup \bar{D} \subseteq \overline{C \vee D}$. If we run into a conflict, we can learn the rightmost clause E in $\mathfrak{L}_{\mathcal{T}}$ which is a subclause of $C \vee D$.

Assume now that we do not get a blockade and not run into a conflict. Then we have $\alpha \subseteq \mathcal{T}$. By Lemma 9.5, all propagations from $\mathcal{U}_1$ are contained in $\mathcal{T}$, in particular $\ell_1 = x \in \mathcal{T}$. Consider the clause

$$A := \mathrm{red}_{G(C \vee D)}(D'|_{\mathcal{T}}).$$

The negations of all literals from $D \backslash G(C \vee D)$ are contained in $\mathcal{T}$. Hence A can only consist of literals from $G(C \vee D)$. But these literals can be reduced away. Therefore $A = (\bot)$ and we would be able to run into a conflict, which is a contradiction. All in all we run into a conflict or obtain a blockade of $C \vee D$.

Case 2. $\ell_1 \neq x$.

Case 2.1. $C \vee x$ does not contain a universal tautology $u \vee \bar{u}$ with $\mathrm{lv}(u) < \mathrm{lv}(x)$ (we are always in this case if we consider QU-Res).

Note that in this case for all literals $w \in G(C)$ we have $\mathrm{lv}(w) > \mathrm{lv}(x)$. This case is similar to Case 3 of the previous Lemma. We construct a natural $G(C)$-reductive trail $\mathcal{T}$ with decisions $\alpha := \alpha_1 \cup \{\bar{\ell}_1\}$, whereby we decide $\bar{x}$ at the end (if $\bar{x} \in \alpha_1$). If we run into a conflict before deciding $\bar{x}$, we can learn the rightmost clause E in $\mathfrak{L}_{\mathcal{T}}$, which is a subclause of C. Assume we get a blockade with a literal $\ell \in \bar{\alpha} \subseteq C \vee x$. If $\ell \neq x$, then this is a blockade of C and also $C \vee D$. However, if $\ell = x$, then we can go to Case 1 and replace the blockade that consists of $\mathcal{U}_1$ with the blockade consisting of $\mathcal{T}$.

Suppose we do not run into a conflict or get a blockade before deciding $\bar{x}$. If we somehow propagate $\bar{x}$, we have $\alpha_1 \subseteq \alpha \subseteq \mathcal{T}$ and by Lemma 9.5 we conclude $\ell_1 \in \mathcal{T}$. However, this contradicts $\bar{\ell}_1 \in \mathcal{T}$.

By not running into a conflict or getting a blockade before deciding $\bar{x}$, we are able to actually decide $\bar{x}$ at the end. We would run into a conflict by the same argument as above. We learn the rightmost clause E in $\mathfrak{L}_{\mathcal{T}}$, which is a subclause of $\bar{\alpha} \vee G(C)$ with $x \in E$. We backtrack back to $\mathcal{T}[r, 0]$ (right before deciding $\bar{x}$). As above, we have $\alpha \backslash \{\bar{x}\} \subseteq \mathcal{T}[r, 0]$.

By our precondition, we conclude

$$\mathrm{red}_{G(C)}(E|_{\mathcal{T}[r,0]}) = (x).$$

We propagate x and obtain another blockade of $C \vee x$ such that we can go into Case 1 again.

Case 2.2. $D \vee \bar{x}$ does not contain a universal tautology $u \vee \bar{u}$ with $\mathrm{lv}(u) < \mathrm{lv}(x)$.

We can assume that we only consider mLD-Q-Res and $\mathsf{QCDCL}^{\textsc{Any-Ord}}_{\textsc{Any-Red},\textsc{Exi-Prop}}$.

Now we have $\mathrm{lv}(v) > \mathrm{lv}(x)$ for all $v \in G(D)$. We construct a natural $G(D)$-reductive trail $\mathcal{T}$ with decisions

$$\alpha := (\alpha_1 \cup \{\bar{\ell}_1\} \cup \overline{H(D)}) \backslash \{\bar{x}\}$$

such that we decide the existential literals first. Note that α is non-tautological because $\alpha_1 \cup \{\bar{\ell}_1\}$ consists of existential literals only and $\overline{C \vee D} \supseteq \alpha$ can only have universal tautologies. Also, we still have $\alpha \cap L = \emptyset$ because of $G(D) \cap \overline{H(D)} = \emptyset$.

If we run into a conflict or get a blockade, we are done again. Otherwise, we have decided or propagated all literals from α. I.e., $\alpha \subseteq \mathcal{T}$. Consider the clause

$$A := \mathrm{red}_L(D'|_{\mathcal{T}}).$$

Because of $\bar{x} \notin \overline{H(D)}$, we have $\overline{H(D)} \subseteq \alpha \subseteq \mathcal{T}$ and therefore $A \subseteq G(D) \vee \bar{x}$. By our precondition (all literals from $G(D)$ are right of x), we conclude $A = (\bar{x})$ since we can reduce all universal literals from $D'|_{\mathcal{T}}$. That means we have to propagate $\bar{x}$ in $\mathcal{T}$, hence $\bar{x} \in \mathcal{T}$. But then we have $\alpha_1 \subseteq \mathcal{T}$. By Lemma 9.5, all propagated literals from $\mathcal{U}_1$ have to be contained in $\mathcal{T}$, in particular $\ell_1 \in \mathcal{T}$. However, this is a contradiction to $\bar{\ell}_1 \in \mathcal{T}$.

That means we always have to run into a conflict or get a blockade, as we desired. □

The next lemma covers the last case for the simulation of resolution steps: Both parent clauses are now subsumed by previously learned clauses or axioms (i.e., both clauses are labelled "subclause"). This corresponds to Case (iii) from the proof sketch of Theorem 9.6.

Lemma 9.10. *Let $\Phi = \mathcal{Q} \cdot \varphi$ be a QCNF. Let further $C \vee x$ and $D \vee \bar{x}$ two clauses such that $C \vee D = C \vee x \overset{x}{\bowtie} D \vee \bar{x}$ is a valid* mLD-Q-Res *(*QU-Res*) resolution step. Suppose there exist subclauses $C' \subseteq C \vee x$ and $D' \subseteq D \vee \bar{x}$ with $C', D' \in \varphi$. Then there exists a* $\mathsf{QCDCL}^{\textsc{Any-Ord}}_{\textsc{Any-Red},\textsc{Exi-Prop}}$ *(*$\mathsf{QCDCL}^{\textsc{Any-Ord}}_{\textsc{No-Red},\textsc{All-Prop}}$*) proof*

$$\iota = [(\mathcal{T}_i, C_i, \pi_i)]_{i=1}^{c}$$

from Φ with a constant c, such that $C_c \subseteq C \vee D$ or there exists a blockade of $C \vee D$ for $\mathcal{Q} \cdot (\varphi \cup \{C_1, \ldots, C_c\})$.

Proof. We construct a natural $G(C \vee D)$-reductive trail $\mathcal{T}$ with decisions $\alpha := \overline{H(C \vee D)}$ such that existential decisions are made first. If we get a blockade or run into a conflict, we are done. So suppose we neither get a blockade, nor run into a conflict. Then we have $\alpha \subseteq \mathcal{T}$. W.l.o.g. let $C \vee x$ not contain a universal tautology $u \vee \bar{u}$ with $\mathrm{lv}(u) < \mathrm{lv}(x)$. Consider the clause

$$A := \mathrm{red}_{G(C \vee D)}(C'|_{\mathcal{T}}).$$

The clause $C'|_{\mathcal{T}}$ can only consist of x or universal literals from $G(C \vee D)$ since the rest got negated by α. We now want to prove that all universal literals in $C'|_{\mathcal{T}}$ can be reduced. In detail, for all universal literals $w \in C'|_{\mathcal{T}}$ we need $\mathrm{lv}(w) > \mathrm{lv}(x)$. Suppose we have a universal literal $v \in C'|_{\mathcal{T}} \subseteq C \vee x$ with $\mathrm{lv}(v) < \mathrm{lv}(x)$. We already concluded that this literal v has to be contained in $G(C \vee D)$. Because we do not have universal tautologies in C left of x, we conclude $\bar{v} \notin C$. But then we need $\bar{v} \in D$ since $v \in G(C \vee D)$. However, such a resolution step is not allowed in mLD-Q-Res (not even in LD-Q-Res).

That means all universal literals from $C'|_{\mathcal{T}}$ can be reduced, hence $A \in \{(x), (\bot)\}$. The case $A = (\bot)$ is impossible because we assumed we do not run into a conflict. Therefore $A = (x)$ and we have to propagate x in $\mathcal{T}$. I.e., $x \in \mathcal{T}$.

Now, we consider the clause

$$B := \mathrm{red}_{G(C \vee D)}(D'|_{\mathcal{T}}).$$

Similarly to the situation before, B can only consist of universal literals from $G(C \vee D)$. Note that the $\bar{x}$ that was potentially contained in D' is now vanished. All literals from $D'|_{\mathcal{T}}$ can be reduced, hence $B = (\bot)$. Then $\mathcal{T}$ would run into a conflict, which is a contradiction. □

The following lemma shows how we can simulate a reduction step on a clause for which we have detected a blockade (i.e., the clause is labelled "blockade"). This corresponds to Case (iv) from the proof sketch of Theorem 9.6.

Lemma 9.11. *Let $\Phi = \mathcal{Q} \cdot \varphi$ be a QCNF. Let $D = C \vee u_1 \vee \ldots \vee u_s$ such that red$(D) = C$ is a valid reduction step in* mLD-Q-Res *(*QU-Res*). Suppose there exists a blockade of D for $\Psi = \mathcal{Q} \cdot \psi$ with $\psi \subseteq \varphi$. Then there exists a* $\mathsf{QCDCL}^{\textsc{Any-Ord}}_{\textsc{Any-Red},\textsc{Exi-Prop}}$ *(*$\mathsf{QCDCL}^{\textsc{Any-Ord}}_{\textsc{No-Red},\textsc{All-Prop}}$*) proof*

$$\iota = [(\mathcal{T}_i, C_i, \pi_i)]_{i=1}^{c}$$

from Φ with a constant c, such that $C_c \subseteq C$ or there exists a blockade of C for $\mathcal{Q} \cdot (\varphi \cup \{C_1, \ldots, C_c\})$.

Proof. Let the blockade of D be $(\mathcal{U}_1, \alpha_1, \ell_1, K_1)$. In the case of mLD-Q-Res, we construct a natural $G(C)$-reductive trail $\mathcal{T}$ with decisions $\alpha := \alpha_1 \cup \{\bar{\ell}_1\} \subseteq \bar{D}$. If we run into

a conflict, we can learn the rightmost clause E in $\mathfrak{L}_{\mathcal{T}}$. Then E is a subclause of D. Since Clause Learning automatically reduces all learned clauses, we even get $E \subseteq C$, as desired.

If we get a blockade, then we only used existential literals (i.e., literals from $\bar{C}$) as decisions. Therefore such a blockade is a blockade of C.

If we neither run into a conflict, nor get a blockade, we change the set of reducible literals in $\mathcal{T}$ from $G(C)$ to $G(D)$ as soon as all decisions are contained in $\mathcal{T}$. Note that we are allowed to change this set even midway through a trail. However, we do not make additional decisions. We continue making further propagations with the aid of $G(D)$-reductive unit clauses as long as possible. Assume that we still do not run into a conflict. Then, analogously to Lemma 9.5, we claim that each propagated literal from $\mathcal{U}_1$ is contained in $\mathcal{T}$.

Assume not. Suppose that $p_{(i,j)}$ is the leftmost propagated literal from $\mathcal{U}_1$ that is not contained in $\mathcal{T}$. We obviously have

$$\mathrm{red}_{K_1}(\mathrm{ante}_{\mathcal{U}_1}(p_{(i,j)})|_{\mathcal{U}_1[i,j]}) = (p_{(i,j)})$$

and $\mathcal{U}_1[i,j] \subseteq \mathcal{T}$.

Now we consider the clause

$$A := \mathrm{red}_{G(D)}(\mathrm{ante}_{\mathcal{U}_1}(p_{(i,j)})|_{\mathcal{T}}).$$

We conclude that we either get $A \in \{(p_{(i,j)}), (\bot)\}$, or $\mathrm{ante}_{\mathcal{U}_1}(p_{(i,j)})$ is verified under $\mathcal{T}$. Since we presumed that we do not run into a conflict or propagate $p_{(i,j)}$, we can assume that $\mathcal{T}$ satisfies $\mathrm{ante}_{\mathcal{U}_1}(p_{(i,j)})$. Then we can find a literal $u \in \mathrm{ante}_{\mathcal{U}_1}(p_{(i,j)}) \cap \mathcal{T}$. This literal cannot be existential, since otherwise we would get $\bar{u} \in \mathcal{U}_1[i,j] \subseteq \mathcal{T}$. Thus, u is universal and therefore $u \in \alpha$ (due to ∃-Prop). But this contradicts the fact that all literals in α are existential.

We conclude that all propagated literals from $\mathcal{U}_1$ are contained in $\mathcal{T}$, in particular $\ell_1 \in \mathcal{T}$. However, this is a contradiction since $\bar{\ell}_1 \in \alpha \subseteq \mathcal{T}$. Therefore, we have to run into a conflict in $\mathcal{T}$. From this conflict, we can again learn the rightmost clause E in $\mathfrak{L}(\mathcal{T})$, for which it holds $E \subseteq C$, as desired.

In the case for QU-Res, we construct a natural K_1-reductive trail $\mathcal{T}$ with decisions $\alpha := \alpha_1 \cup \{\bar{\ell}_1\}$ such that the literals $\bar{u}_i$ are decided last for those contained in α. If we get a conflict, we can learn the clause $\mathrm{red}(\bar{\alpha}) \subseteq \mathrm{red}(D) = C$ and we are done. So suppose we get a blockade $(\mathcal{T}, \beta, \ell, K_1)$ with $\beta \subseteq \alpha$. If $\ell \neq u_i$ for each $i = 1, \ldots, m$, then we also have $\bar{u}_i \notin \beta$ for each i because the $\bar{u}_i$ can only be decided last. But then we have a blockade of C and we are done. However, if $\ell = u_i$ for some $i \in \{1, \ldots m\}$, then instead of propagating u_i, we can simply run into a conflict and learn a subclause of $\mathrm{red}(\bar{\beta} \vee u) \subseteq \mathrm{red}(C \vee u) = C$.

If we neither get a blockade, nor run into a conflict, then we have $\alpha \subseteq \mathcal{T}$ and we can make all propagations from $\mathcal{U}_1$ by Lemma 9.5, hence we get $\beta \in \mathcal{T}$. This is a contradiction to $\bar{\ell} \in \alpha \subseteq \mathcal{T}$. □

The next lemma is the last one before we are able to completely prove Theorem 9.6. It handles the case where we want to simulate a reduction step on a clause D

that is subsumed by a previously learned clause or axiom (i.e., the clause is labelled "subclause"). This corresponds to Case (v) from the proof sketch of Theorem 9.6. Note that this case is somewhat special: As learned clauses are already fully universally reduced, there is nothing to show if D is subsumed by a previously learned clause. Hence, we can assume that D is subsumed by an axiom. But then we can also assume that D is an axiom itself, otherwise we could simply shorten the given mLD-Q-Res (QU-Res) refutation that is to be simulated. In particular, we can assume that D is non-tautological.

Lemma 9.12. *Let $\Phi = \mathcal{Q} \cdot \varphi$ be a QCNF. Let $D = C \vee u_1 \vee \ldots \vee u_m$ be non-tautological such that red$(D) = C$. Suppose $D \in \varphi$. Then there exists a* $\mathsf{QCDCL}^{\textsc{Any-Ord}}_{\textsc{Any-Red},\textsc{Exi-Prop}}$ *(resp.* $\mathsf{QCDCL}^{\textsc{Any-Ord}}_{\textsc{No-Red},\textsc{All-Prop}}$*) proof*

$$\iota = [(\mathcal{T}_i, C_i, \pi_i)]_{i=1}^{c}$$

from Φ with a constant c, such that $C_c \subseteq C$ or there exists a blockade of C for $\mathcal{Q} \cdot (\varphi \cup \{C_1, \ldots, C_c\})$.

Proof. We construct a natural ($\emptyset$-reductive) trail $\mathcal{T}$ with decisions $\alpha = \bar{D}$, such that all the $\bar{u}_i$ are decided last. If we run into a conflict, we can learn a subclause of $\mathrm{red}(\bar{\alpha}) = \mathrm{red}(D) = C$.

Assume that we get a blockade $(\mathcal{T}, \beta, \ell, \emptyset)$. If $\ell \neq u_i$ for each i, then we also have $\bar{u}_i \notin \beta$ for each i because the $\bar{u}_i$ are decided last. In that case, this is also a blockade for C and we are done. In the case where $\ell = u_i$ for some i, we can again simply run into a conflict instead of propagating u_i, hence learning a subclause of $\mathrm{red}(\bar{\alpha}) = \mathrm{red}(D) = C$.

Suppose that none of this occurs. Then we have $\alpha \subseteq \mathcal{T}$. But then we would falsify D, hence we would have the opportunity to run into a conflict with the aid of D, which is a contradiction. □

Finally, we are able to formally prove Theorem 9.6.

Theorem 9.6. *It holds the following:*

- $\mathsf{QCDCL}^{\textsc{Any-Ord}}_{\textsc{Any-Red},\textsc{Exi-Prop}}$ *p-simulates* mLD-Q-Res*.*
- $\mathsf{QCDCL}^{\textsc{Any-Ord}}_{\textsc{No-Red},\textsc{All-Prop}}$ *p-simulates* QU-Res*.*

Proof. Let $\mathsf{S} \in \{\mathsf{QCDCL}^{\textsc{Any-Ord}}_{\textsc{Any-Red},\textsc{Exi-Prop}}, \mathsf{QCDCL}^{\textsc{Any-Ord}}_{\textsc{No-Red},\textsc{All-Prop}}\}$. The procedure for both simulations is the same. The plan is going through the whole proof π for each clause D in π either find a blockade, or learn a subclause of D during the construction of S-trails.

Suppose we already considered the clauses $D_1, \ldots, D_{i-1}$ for an $i \in \{1, \ldots, m\}$ and constructed S proofs $\iota_1, \ldots, \iota_{i-1}$. In detail, we have $\iota_j = [(\mathcal{T}_q^{(j)}, C_q^{(j)}, \pi_q^{(j)})]_{q=1}^{c_j}$ for each $j \in [i-1]$ and some constants c_j. Define

$$\varphi_j := \varphi \cup \bigcup_{k=1}^{j-1} \{C_1^{(k)}, \ldots, C_{c_k}^{(k)}\}.$$

One could interpret φ_j as the knowledge base right before considering the clause D_j in π (and right after going through D_{j-1}, if $j > 1$). In particular, we want to show that if for each $h \in [i-1]$ there exists a blockade of D_h for $\mathcal{Q} \cdot \varphi_{h+1}$ or there exists a subclause $D'_h \subseteq D_h$ with $D'_h \in \varphi_{h+1}$, then we can construct an S proof $\iota_i = [(\mathcal{T}_q^{(i)}, C_q^{(i)}, \pi_q^{(i)})]_{q=1}^{c_i}$ from $\mathcal{Q} \cdot \varphi_i$ such that there exists a blockade of D_i for $\mathcal{Q} \cdot \varphi_{i+1}$ or there exists a subclause $D'_i \subseteq D_i$ with $D'_i \in \varphi_{i+1}$. Note that we will only add the trail to our S proof if we learned a clause. The blockade itself will never actually be added to the proof.

If D_i was an axiom (for example if $i = 1$), then we already have a subclause of D_i which is contained in φ_i (in fact, D_i itself). In this case we set $\varphi_{i+1} := \varphi_i$ and do nothing. The proof ι_{i+1} can be defined as the empty proof (or simply left out).

Suppose D_i was the resolvent of two previous clauses D_a and D_b. By induction, we know that

- there exists a blockade of D_a for $\mathcal{Q} \cdot \varphi_{a+1}$ or there exists a subclause $D'_a \subseteq D_a$ with $D'_a \in \varphi_{a+1}$, and
- there exists a blockade of D_b for $\mathcal{Q} \cdot \varphi_{b+1}$ or there exists a subclause $D'_b \subseteq D_b$ with $D'_b \in \varphi_{b+1}$.

Each possibility is covered by some earlier Lemma: Lemma 9.8 or Lemma 9.9 or Lemma 9.10. In each case we can construct an S proof ι_i from $\mathcal{Q} \cdot \varphi_i$ such that $D'_i := C_{c_i}^{(i)} \subseteq D_i$ or we get a blockade of D_i for $\mathcal{Q} \cdot \varphi_{i+1}$. Note that we always have $\varphi_{a+1} \subseteq \varphi_i$ and $\varphi_{b+1} \subseteq \varphi_i$.

Now suppose D_i was derived by a reduction of some previous clause D_a, i.e., $D_i = \mathrm{red}(D_a)$. By induction, we either know that

- there exists a blockade of D_a for $\mathcal{Q} \cdot \varphi_{a+1}$, or
- there exists a subclause $D'_a \subseteq D_a$ with $D'_a \in \varphi_{a+1}$.

The first case is covered by Lemma 9.11. In the second case either D'_a is non-tautological (this case is covered by Lemma 9.12), or D'_a is tautological and hence actually a previously learned clause. In the latter case we already have $D'_a = \mathrm{red}(D'_a) \subseteq \mathrm{red}(D_a) = D_i$ by the definition of Clause Learning. Hence we do not have to construct any trails and therefore ι_i can again be viewed as the empty proof.

Finally we construct proofs ι_i as long as we have not learned the empty clause, yet. In the worst case we will learn $(\bot)$ during ι_m since it is impossible to find a blockade of $D_m = (\bot)$ and therefore we will always learn a subclause of D_m.

We obtain a refutation ι by sticking together all constructed subproofs ι_i. As usual, we restart between two subproofs ι_i and ι_{i+1}. Note that all ι_i have, by construction, linear size and therefore $|\iota| \in \mathcal{O}(n \cdot |\pi|)$. □

Corollary 9.13. $\mathsf{QCDCL}^{\textsc{Any-Ord}}_{\textsc{No-Red},\textsc{Exi-Prop}}$ *p-simulates* $\mathsf{Q\text{-}Res}$.

Proof. Each $\mathsf{Q\text{-}Res}$ refutation is also an $\mathsf{mLD\text{-}Q\text{-}Res}$ refutation without any tautologies. Hence, by Theorem 9.6, $\mathsf{QCDCL}^{\textsc{Any-Ord}}_{\textsc{Any-Red},\textsc{Exi-Prop}}$ simulates $\mathsf{Q\text{-}Res}$. The literals we have to reduce during unit propagation always occur in the tautological part $G(C)$ of some clause

	Simulation		
No	simulating system	simulated system	Theorem
2	LD-Q-Res	mLD-Q-Res	by Def.
4	mLD-Q-Res	$\mathsf{QCDCL}^{\textsc{Any-Ord}}_{\textsc{All-Red},\textsc{Exi-Prop}}$	Prop 8.10

Table 9.1: P-simulations (without separations) of proof systems from Figure 9.1 (i.e. the solid lines).

C from the Q-Res refutation. Since $G(C) = \emptyset$ for each clause C in a Q-Res refutation, we conclude that all reductive sets can be set to $\emptyset$ and therefore Q-Res is even p-simulated by $\mathsf{QCDCL}^{\textsc{Any-Ord}}_{\textsc{No-Red},\textsc{Exi-Prop}}$. □

Proposition 8.10, Theorem 9.6 and Corollary 9.13 yield the following equivalences:

Corollary 9.14. *It holds*

(i) $\mathsf{QCDCL}^{\textsc{Any-Ord}}_{\textsc{No-Red},\textsc{Exi-Prop}} \equiv_p$ Q-Res,

(ii) $\mathsf{QCDCL}^{\textsc{Any-Ord}}_{\textsc{Any-Red},\textsc{Exi-Prop}} \equiv_p$ mLD-Q-Res,

(iii) $\mathsf{QCDCL}^{\textsc{Any-Ord}}_{\textsc{No-Red},\textsc{All-Prop}} \equiv_p$ QU-Res.

Remark 9.15. *Note that our simulations require a particular learning scheme, in which we almost always restart after each conflict. This is also the reason why we get an improved simulation complexity of $\mathcal{O}(n \cdot |\pi|)$ compared to $\mathcal{O}(n^3 \cdot |\pi|)$ from [14], in which arbitrary (asserting) learning schemes were allowed (where we do not necessarily restart every time). Performing our simulation under arbitrary asserting learning schemes might require some additional analysis on asserting clauses under the* Any-Ord *and* Any-Red *rules, as a clause learned from a K_1-reductive trail might not be asserting in K_2-reductive trails anymore. However, if it was clear how to guarantee asserting clauses in our systems, we would be able to obtain similar results as in [14], that is:*

- *For each clause C in the given* mLD-Q-Res *(*QU-Res*) refutation and an arbitrary asserting learning scheme, we need $\mathcal{O}(n^2)$ trails and backtracking steps until we either learn a subclause of C, or we obtain a blockade for C.*
- *Under any arbitrary asserting learning scheme, we can perform the simulation in time $\mathcal{O}(n^3 \cdot |\pi|)$. In particular, we do not need to restart after each conflict.*

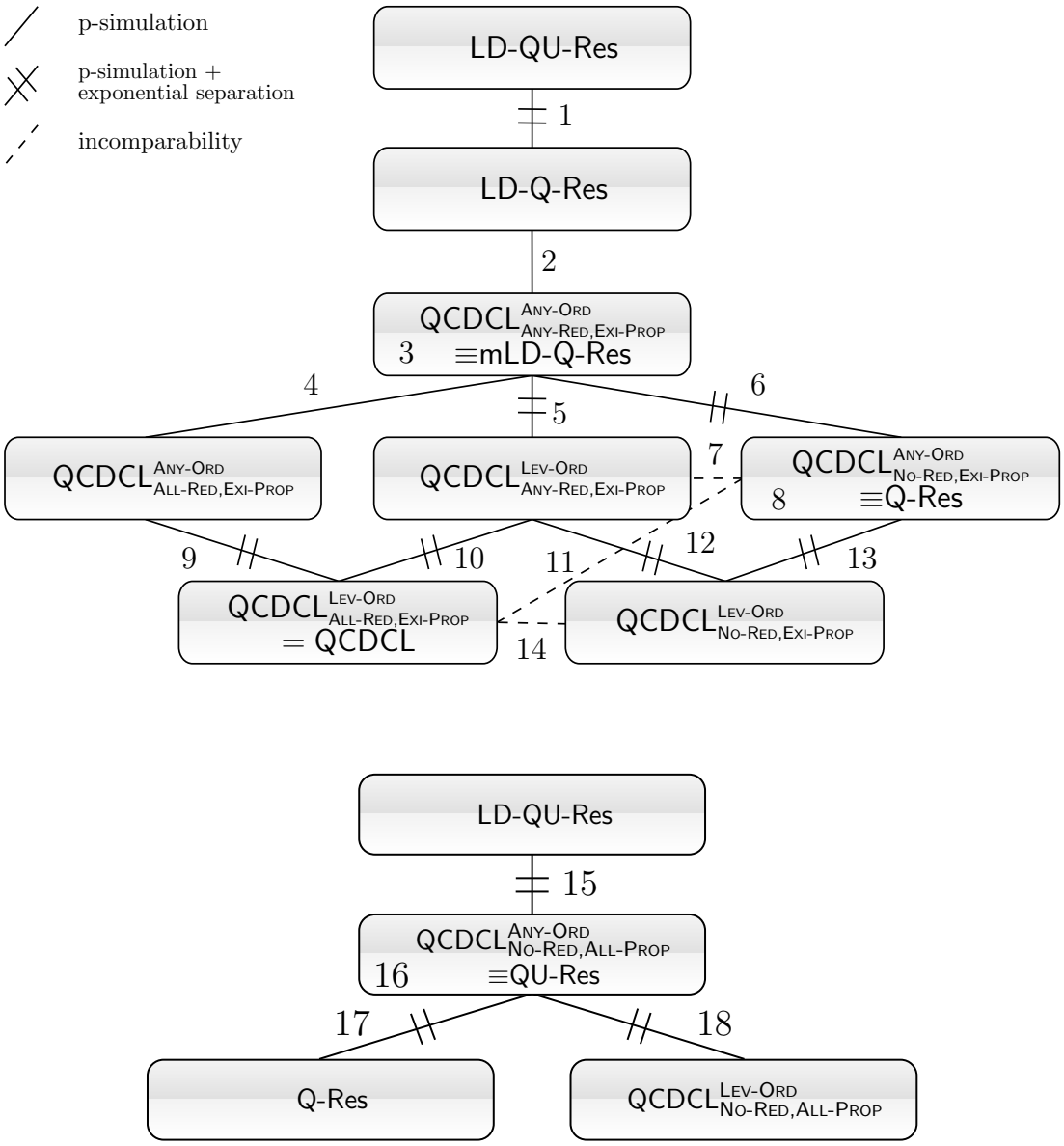


Figure 9.1: Hasse diagrams of the simulation order of QCDCL proof systems using propagation policies Exi-Prop (above) and All-Prop (below) together with corresponding QBF resolution proof systems. Each relation is labelled by a number that represents a reference depicted in the Tables 9.1, 9.2, 9.3 and 9.4.

	Simulation	Separation			
No	Theorem	Formula	easy for	hard for	Theorem
1	by Def.	$\texttt{KBKF-lq}_n$[1]	LD-QU-Res	LD-Q-Res	[5]
5	Prop 8.10	$\texttt{MirrorCR}_n$	mLD-Q-Res	$\text{QCDCL}^{\text{Lev-Ord}}_{\text{Any-Red,Exi-Prop}}$	Prop 8.4, Prop 8.5
6	by Def.	$\texttt{Eq}_n$	mLD-Q-Res	Q-Res	Cor 8.11
9	by Def.	$\texttt{TwinCR}_n$	$\text{QCDCL}^{\text{Any-Ord}}_{\text{All-Red,Exi-Prop}}$	$\text{QCDCL}^{\text{Lev-Ord}}_{\text{All-Red,Exi-Prop}}$	Cor 7.23, Prop 7.24
10	by Def.	$\texttt{MiPa}_n$	$\text{QCDCL}^{\text{Lev-Ord}}_{\text{Any-Red,Exi-Prop}}$	$\text{QCDCL}^{\text{Lev-Ord}}_{\text{All-Red,Exi-Prop}}$	Thm 8.8
12	by Def.	$\texttt{MiPa}_n$	$\text{QCDCL}^{\text{Lev-Ord}}_{\text{Any-Red,Exi-Prop}}$	$\text{QCDCL}^{\text{Lev-Ord}}_{\text{No-Red,Exi-Prop}}$	Thm 8.8
13	Prop 8.10	$\texttt{MirrorCR}_n$	Q-Res	$\text{QCDCL}^{\text{Lev-Ord}}_{\text{No-Red,Exi-Prop}}$	Prop 8.4, Prop 8.5
15	by Def.	$\texttt{KBKF-lqu}_n$[2]	LD-QU-Res	QU-Res	[5]
17	by Def.	$\texttt{KBKF}_n$[3]	QU-Res	Q-Res	[43, 99]
18	Prop 8.10	$\texttt{MirrorCR}_n$	QU-Res	$\text{QCDCL}^{\text{Lev-Ord}}_{\text{No-Red,All-Prop}}$	Prop 8.4, Prop 8.5

Table 9.2: P-simulations and separations of proof systems from Figure 9.1 (i.e. the crossed lines).

	Separation			
No	Formula	easy for	hard for	Theorem
11	$\texttt{MirrorCR}_n$	Q-Res	$\text{QCDCL}^{\text{Lev-Ord}}_{\text{All-Red,Exi-Prop}}$	Prop 8.4, Prop 8.5
	$\texttt{QParity}_n$	$\text{QCDCL}^{\text{Lev-Ord}}_{\text{All-Red,Exi-Prop}}$	Q-Res	Prop 8.6
14	$\texttt{QParity}_n$	$\text{QCDCL}^{\text{Lev-Ord}}_{\text{All-Red,Exi-Prop}}$	$\text{QCDCL}^{\text{Lev-Ord}}_{\text{No-Red,Exi-Prop}}$	Prop 8.6
	$\texttt{MiEasy}_n$	$\text{QCDCL}^{\text{Lev-Ord}}_{\text{No-Red,Exi-Prop}}$	$\text{QCDCL}^{\text{Lev-Ord}}_{\text{All-Red,Exi-Prop}}$	Prop 8.13

Table 9.3: Incomparable proof systems from Figure 9.1 (i.e. the dashed lines).

[1] Defined in [5].
[2] Defined in [5].
[3] Defined in [43].

	Equivalence		
No	Systems		Theorem
3	mLD-Q-Res	$\mathsf{QCDCL}^{\textsf{Any-Ord}}_{\textsf{Any-Red},\textsf{Exi-Prop}}$	Cor 9.14
8	Q-Res	$\mathsf{QCDCL}^{\textsf{Any-Ord}}_{\textsf{No-Red},\textsf{Exi-Prop}}$	Cor 9.14
16	QU-Res	$\mathsf{QCDCL}^{\textsf{Any-Ord}}_{\textsf{No-Red},\textsf{All-Prop}}$	Cor 9.14

Table 9.4: P-equivalent proof systems from Figure 9.1.

Chapter 10

Runtime vs. Extracted Proof Size of QCDCL

For this last chapter, we revisit QCDCL systems with level-ordered decisions only. As we have already shown in Proposition 8.10, the three variants $\mathsf{QCDCL}^{\textsc{Any-Ord}}_{\textsc{No-Red},\textsc{Exi-Prop}}$, $\mathsf{QCDCL}^{\textsc{Any-Ord}}_{\textsc{Any-Red},\textsc{All-Prop}}$ and $\mathsf{QCDCL}^{\textsc{Any-Ord}}_{\textsc{Any-Red},\textsc{Exi-Prop}}$ are p-simulated by Q-Res, QU-Res and mLD-Q-Res respectively. The same is true if we replace ANY-ORD with LEV-ORD because this simulation does not depend on the Decision Policy.

What remains open is whether there is an exponential gap between the number of trails and the size of the extracted proof. We already know that the size of the extracted proof is polynomial in the number of trails (simply by definition of Clause/Cube Learning, cf. Definition 4.4), but there might be cases in which an exponential sized QCDCL refutation generates a polynomial extracted proof.

For this analysis, we consider the following three variants of QCDCL:

- $\mathsf{M}_{\mathsf{Q}} := \mathsf{QCDCL}^{\textsc{Lev-Ord}}_{\textsc{No-Red},\textsc{Exi-Prop}}$
- $\mathsf{M}_{\mathsf{LD}} := \mathsf{QCDCL}^{\textsc{Lev-Ord}}_{\textsc{All-Red},\textsc{Exi-Prop}}$
- $\mathsf{M}_{\mathsf{QU}} := \mathsf{QCDCL}^{\textsc{Lev-Ord}}_{\textsc{No-Red},\textsc{All-Prop}}$

Instead of $\mathsf{QCDCL}^{\textsc{Lev-Ord}}_{\textsc{All-Red},\textsc{Exi-Prop}}$, we could also define M_{LD} as $\mathsf{QCDCL}^{\textsc{Lev-Ord}}_{\textsc{Any-Red},\textsc{Exi-Prop}}$ — all results regarding M_{LD} hold for ALL-RED as well as ANY-RED. While ANY-RED is a theoretically stronger policy (as proven in the last two chapters), ALL-RED is still the default policy for practical QCDCL which yields more relevant results than non-established variants.

Rather than comparing these three variants among themselves, we will compare the QCDCL proof systems with the proofs that are extracted via Clause or Cube Learning. Obviously, the extracted proof cannot be larger than the combined size of all trails since each resolution step corresponds to a unit propagation in a trail.

What is not obvious is whether QCDCL proofs that need exponentially many trails will also necessarily generate exponential proofs. We will give a negative answer to

this question for each considered variant $\mathsf{M_Q}$, $\mathsf{M_{LD}}$ and $\mathsf{M_{QU}}$. In detail, we will show that there exist formulas that need exponentially many trails, but the refutations that are generated via proof extraction can be polynomial in the formula size. That means QCDCL needs to learn exponentially many constraints that will never appear in the extracted proof, but are still relevant for finding a short extracted proof.

10.1 The Separation Technique

Intuitively, we aim to find false formulas for which QCDCL needs to learn "many" cubes in order to find a short refutation with clauses. We recall that Cube Learning can be beneficial even on false formulas (cf. Section 6.2) and that cubes will never appear in (clause) refutations. So far, all formulas that separated variants with and without Cube Learning only needed a polynomial number of learned Cubes in order to be able to find short refutations. We combine this approach with the one from Section 7.2, in which we constructed a lower bound on true formulas.

The idea is as follows: We choose a false formula that is easy to refute with Cube Learning and hard without, and a true formula that is generally hard to verify. We combine these two formulas via conjunction such that the left part is the false formula and the right part is the true one. In particular, the whole formula is false and each (clause) refutation of the formula is a refutation of the left part only. In order to find a short refutation of that left part, we need to do some Cube Learning, but the right part ensures that learning a cube for the left part means essentially verifying the right part, which is hard.

Before we approach our main Theorem 10.2, we will formally define how we construct QBFs by combining two QCNFs into one single QCNF by concatenating the two quantifier prefixes and conjoining both matrices. An important property is that the variables of these two QCNFs are disjoint.

Definition 10.1. *Let $\Phi = \mathcal{Q} \cdot \varphi$ and $\Psi = \mathcal{R} \cdot \psi$ be two QCNFs. Let $\Psi' = \mathcal{R}' \cdot \psi'$ be the QCNF that is obtained after renaming the variables from Ψ such that $var(\Phi) \cap var(\Psi') = \emptyset$. Then we define the* disjoint composition *of Φ and Ψ as the QCNF $dc(\Phi, \Psi) := \mathcal{Q}\mathcal{R}' \cdot \varphi \wedge \psi'$.*

From now on, we will assume that variables from φ and variables from ψ' do not share the same quantifier level in $\mathcal{Q}\mathcal{R}'$. In particular, Φ will always end with an existential quantifier and Ψ will start with a universal quantifier.

The next theorem is our main technical result which will be used for all following main results in the paper.

Theorem 10.2. *Let Φ_n and Ψ_n be two formulas with the following properties:*

1. *Φ_n is a Σ_3^b QCNF that fulfils the XT-property, is false, and has $\mathsf{M_{LD}}$ (resp. $\mathsf{M_Q}$ and $\mathsf{M_{QU}}$) refutations with trail size s.*
2. *(for $\mathsf{M_{QU}}$) Each $\mathsf{QU\text{-}Res}$ refutation of Φ_n is a $\mathsf{Q\text{-}Res}$ refutation. I.e., it is not possible to resolve two clauses that were derived from Φ_n over a universal variable.*

3. *Ψ_n is true, and for each* $\mathsf{M_{LD}}$ *(resp.* $\mathsf{M_Q}$ *and* $\mathsf{M_{QU}}$*) proof ι_E of some cube E from $dc(\Phi_n, \Psi_n)$ with $var(E) \subseteq var(\Phi_n)$ we have $|\iota_E| \geq r$.*

Then for each $\mathsf{M_{LD}}$ *(resp.* $\mathsf{M_Q}$ *and* $\mathsf{M_{QU}}$*) refutation θ of the QCNF $dc(\Phi_n, \Psi_n)$ we have trail-size$(\theta) \in \min(2^{\Omega(gauge(\Phi_n))}, r)$, but there exists a* $\mathsf{M_{LD}}$ *(resp.* $\mathsf{M_Q}$ *and* $\mathsf{M_{QU}}$*) refutation ι of $dc(\Phi_n, \Psi_n)$ with extr-size$(\iota) \leq s$.*

Before proving the theorem, let us provide some intuition on this result and the way we will apply it later in Section 10.2.

One can obtain an exponential separation between the trail size and the extracted size of a proof by choosing Φ_n and Ψ_n in such a way that s is polynomial in n, gauge(Φ_n) is linear in n, and r is exponential in n.

Intuitively, in order to refute $\mathrm{dc}(\Phi_n, \Psi_n)$, we can consider two possibilities:

1. We never learn a cube E with $\mathrm{var}(E) \subseteq \mathrm{var}(\Phi_n)$. Then each learned cube contains some variables from Ψ_n. We will show in the proof of Theorem 10.2 that we will then generate fully reduced primitive proofs and therefore the gauge lower bound applies (Theorem 5.17), resulting in an exponential lower bound.

2. We learn at least one cube E with $\mathrm{var}(E) \subseteq \mathrm{var}(\Phi_n)$. But then the derivation of this cube E itself needs QCDCL proofs of size r, which is exponential by assumption.

For the polynomial upper bound on the extracted proof size, we are allowed to learn and derive any cube that is useful. Since the derivations of these cubes do not appear in the extracted proof, they may even have exponential size without increasing the size of the extracted proof. As we assume that Φ_n is easy to refute in the considered QCDCL variant, we can reproduce this refutation, as verifying Ψ_n comes for free when only measuring the extracted proof size.

A visualisation of this separation is depicted in Figure 10.1.

Proof of Theorem 10.2. Obviously, $\mathrm{dc}(\Phi_n, \Psi_n)$ is a false formula. It suffices to show that each $\mathsf{M_{LD}}$ (resp. $\mathsf{M_Q}$ and $\mathsf{M_{QU}}$) refutation θ of $\mathrm{dc}(\Phi_n, \Psi_n)$, in which each learned cube contains some literals from Ψ_n, has trail-size$(\theta) = 2^{\Omega(\mathrm{gauge}(\Phi_n))}$. We show that such θ generates fully reduced primitive Q-Res refutations of Φ_n. The lower bound then follows by Theorem 5.17.

Assume, for the sake of contradiction, that there is such a $\mathsf{M_{LD}}$ (resp. $\mathsf{M_Q}$ and $\mathsf{M_{QU}}$) refutation θ of $\mathrm{dc}(\Phi_n, \Psi_n)$ such that the extracted proof $\mathfrak{R}(\theta)$ is not a fully reduced primitive Q-Res refutation of Φ_n. By the definition of a disjoint composition, $\mathfrak{R}(\theta)$ is a refutation of Φ_n. By condition 2 of the theorem, $\mathfrak{R}(\theta)$ does not contain a resolution step over a universal variable, i.e. all resolutions are over existential variables. There must be a resolution step in $\mathfrak{R}(\theta)$ between two XUT-clauses over an X-literal. Consider the first trail $\mathcal{T}$ in θ in which such a resolution, say $C \overset{x}{\bowtie} D$ with $x \in C$ and $\bar{x} \in D$, appeared in the learning phase.

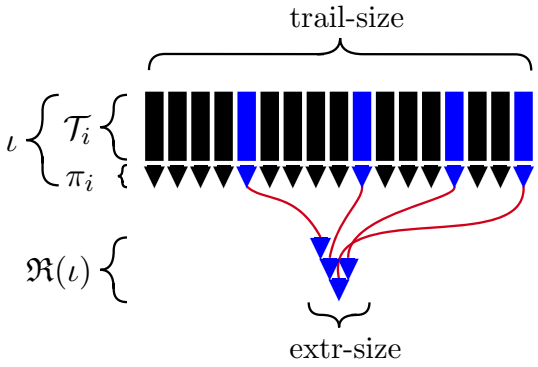


Figure 10.1: Visualisation of the separations following from Theorem 10.2: The rectangles symbolise the trails of a proof, the triangles represent the derivations of the learned constraints. **Black** rectangles and triangles denote trails and derivations for learned **cubes**, while **blue** rectangles and triangles denote trails and derivations of learned **clauses**. As the last learned constraint is empty, i.e. the empty **clause**, all derivations of learned **clauses** can be stuck together to obtain a refutation of the original formula. The derivations of learned **cubes** will not be used for the extracted refutation The separation between the measures trail-size and extr-size occurs when the number of **black** trails is exponential while the number of **blue** trails is polynomial.

Then one of these two clauses must have been an antecedent clause for the pivot, say $\text{ante}_\mathcal{T}(x) = C$. The clause C must contain at least one T-literal, say $t_1 \in C$. Then we need $\bar{t}_1 <_\mathcal{T} x$ and therefore there exists an antecedent clause $A_1 := \text{ante}_\mathcal{T}(\bar{t}_1)$. Because of the XT-property, A_1 cannot be a unit clause, hence it must be either a non-unit T-clause, or a clause with a U-literal. If A_1 is a non-unit T-clause, then we can find another $\bar{t}_1 \neq t_2 \in A_1$, for which we would find another antecedent clause $A_2 := \text{ante}_\mathcal{T}(\bar{t}_2)$. We can repeat this argument, until at some point the antecedent clause $A_j = \text{ante}_T(\bar{t}_j)$ for some T-literal $\bar{t}_j$ contains a U-literal, say $u \in A_j$.

Because $u <_{\Phi_n} t_j$, we need $\bar{u} <_\mathcal{T} \bar{t}_j <_\mathcal{T} x$ in order to propagate $\bar{t}_j$. Because our decisions need to be level-ordered, we conclude that $\bar{u}$ was propagated. It is not possible for $\bar{u}$ to have been propagated by a clause, because otherwise we could perform a universal resolution step in $\mathfrak{R}(\theta)$. Therefore $\bar{u}$ must have been propagated by a cube $F := \text{ante}_\mathcal{T}(\bar{u})$ (note that $u \in F$). From now on, we assume that $\bar{u}$ is the first U-literal that was propagated in $\mathcal{T}$ by a cube and let F be the corresponding antecedent cube (we do not need the connection to A_j anymore).

By our assumption from the beginning of the proof, F contains some literals from Ψ_n. We can safely assume that at least one of these is universal, otherwise all literals from Ψ_n would have been reduced away during the learning of F. Let $w \in F$ be such a universal literal from Ψ_n. Then we need $w <_\mathcal{T} \bar{u}$ because we cannot reduce universally in cubes. That means w was also propagated by some constraint $G := \text{ante}_\mathcal{T}(w)$, which is either a clause or a cube. We distinguish these two cases.

Case 1. G is a cube. This cube G cannot contain any U-literals from Φ_n, otherwise they must have been propagated via a cube (not possible because $\bar{u}$ was the first propagated U-literal via a cube in $\mathcal{T}$) or decided (not possible because decisions need to be level-ordered) before $\bar{u}$ and x in $\mathcal{T}$. We conclude that only existential literals from Φ_n appear in G.

Let π_G be the LD-Q-Con (resp. QU-Con) subproof of G from $\mathrm{dc}(\Phi_n, \Psi_n)$. We can restrict π_G to an LD-Q-Con (resp. QU-Con) proof ρ from Φ_n by just deleting all literals from Ψ_n (and, if necessary, delete redundant cubes). But then ρ is a LD-Q-Con (resp. QU-Con) proof of G', where G' is a cube that only contains existential literals that are also contained in Φ_n. If we reduce G' existentially, we obtain a verification of Φ_n, contradicting the falsity of Φ_n.

Case 2. G is a clause. This case is only relevant for M_{QU}. We can assume w.l.o.g. that G is the first clause that propagates a universal literal in $\mathcal{T}$. Note that this literal has to be a literal from Ψ_n as we cannot resolve over universal variables in refutations of Φ_n by condition 2. Since G was a clause that propagated a universal literal, we could have also run into a conflict in G at that point. Let $\mathcal{T}'$ be the trail that is identical to $\mathcal{T}$ up to the point where w was propagated, but instead of propagating w we run into conflict.

Then we can learn a new clause K by starting with G and resolving over every propagated literal before that conflict. Note that K can only consist of literals from Ψ_n since the two QCNFs are always disjoint on clauses. Furthermore, K cannot contain literals that were propagated via clauses in $\mathcal{T}'$, otherwise they would have been resolved away or skipped if necessary. Because Ψ_n is a true formula, K has to consist of at least one existential literal, say $y \in K$. In particular, this means that $\bar{y} <_{\mathcal{T}} w$.

As decisions are level-ordered, $\bar{y}$ must have been propagated via some cube $H := \mathrm{ante}_T(\bar{y})$. Now, H is a cube that was used to propagate an existential literal. As before, we can define another trail $\mathcal{T}''$ that is identical to $\mathcal{T}$ up to the point where $\bar{y}$ was propagated, but instead of that propagation it runs into conflict on the cube H.

After that we can start Cube Learning and again resolve over each literal that was propagated via a cube (or simply skip them if necessary). Let I be the corresponding learnable cube. Since $\mathrm{dc}(\Phi_n, \Psi_n)$ is a false formula, I needs to contain at least one universal literal v, in particular $v <_{\mathcal{T}} \bar{y}$. Again, by our LEV-ORD policy, v cannot have been decided. Hence, v must have been propagated via a clause. But this is impossible because w was assumed to be the first literal of this type. This contradiction concludes Case 2.

We thus obtain contradictions in both cases. Hence, our assumption that $\mathfrak{R}(\theta)$ was not fully reduced primitive is false. We conclude that $|\theta| \in 2^{\Omega(\mathrm{gauge}(\Phi_n))}$ by Theorem 5.17.

The upper bound can be shown by the following construction: Given an M_{LD} (resp. M_{Q} and M_{QU}) refutation ι' of Φ_n with trail size s, we can essentially reproduce all trails from ι' for a M_{LD} (resp. M_{Q} and M_{QU}) refutation ι of $\mathrm{dc}(\Phi_n, \Psi_n)$. The only difference is that, whenever a cube is learned in ι', we need to verify Ψ_n to learn this cube in ι.

However, as this verification of Ψ_n does not appear in $\mathfrak{R}(\iota)$, we conclude $\mathfrak{R}(\iota) = \mathfrak{R}(\iota')$ and therefore $\mathfrak{R}(\iota)$ is of size at most s. □

10.2 Separations for QCDCL Models

We will now put this general idea into action and construct separations between the trail size and the extracted proof size for each of the three QCDCL variants M_{LD}, M_{Q} and M_{QU}, corresponding to their respective underlying proof systems LD-Q-Res, Q-Res and QU-Res.

10.2.1 QCDCL Based on LD-Q-Res

We start with M_{LD} which corresponds to standard QCDCL as used in modern state-of-the art QBF solvers [76, 82].

First, we need to find a true QCNF that is hard for M_{LD} and fulfils the properties of Ψ_n from Theorem 10.2. We recall the lower bound on true formulas (cf. Theorem 7.15) which uses the notion of the *twin formula* (cf. Definition 7.9) of a QCNF as well as the *reversion* (cf. Definition 7.10).

The next theorem is a generalization of Theorem 7.15. Instead of proving a lower bound on verifications of a particular formula Ψ, we consider derivations of any cube from a formula $\mathrm{dc}(\Gamma, \Psi)$, such that this cube does not contain literals from Ψ. Hence, by choosing Γ as the empty formula, one obtains Theorem 7.15.

Theorem 10.3. *Let Λ be a false Σ_3^b QCNF with the prefix $\exists X \forall U \exists T$ and let Γ be an arbitrary QCNF. Let ι_E be an M_{LD} proof of some cube E from $dc(\Gamma, \mathtt{Rev}(\mathtt{Twin}\Lambda))$ such that $var(E) \subseteq var(\Gamma)$. Additionally, let all clauses $C \in \mathfrak{C}(\Lambda)$ contain at least one U- and one T-literal. If the QCNF $\mathtt{Twin}\Lambda$ needs fully reduced primitive Q-Res refutations of size s, then extr-size$(\iota_E) := |\mathfrak{R}(\iota_E)| \geq s$.*

Proof. We will show that there exists a fully reduced primitive Q-Res refutation π for $\mathtt{Twin}\Lambda$ with $|\pi| \leq |\mathfrak{R}(\iota_E)|$.

Let π be the LD-Q-Res refutation of $\mathtt{Twin}\Lambda$ we can obtain from $\mathfrak{R}(\iota_E)$ as described in Proposition 7.13. Then $|\pi| \leq |\mathfrak{R}(\iota_E)|$ and π is fully reduced. We will show that π is primitive.

Assume not. Then there are two XUT-clauses $B_1, B_2 \in \pi$ that are resolved over some $x \in X$. By the construction of π described in Proposition 7.13, we can find two cubes $D_1, D_2 \in \mathfrak{R}(\iota_E)$ such that $\mathrm{var}(D_i) \cap U \neq \emptyset$ and $\mathrm{var}(D_i) \cap T \neq \emptyset$ for $i = 1, 2$ which are resolved over x. One of these cubes was an antecedent cube for x in some trail $\mathcal{T}$ from ι_E, say $D_1 = \mathrm{ante}_{\mathcal{T}}(x)$ (that means $\bar{x} \in D_1$).

In particular, there is some T-literal $t \in D_1$ such that $t <_{\mathcal{T}} x$ because D_1 must become unit. Remember that t is universal in $\mathtt{Rev}(\mathtt{Twin}\Phi)$ and we can only reduce cubes existentially. Then either t was a decision, or a propagation.

Case 1. t was decided.

This is only possible if all U-variables were assigned before. Hence, for each $u \in U$ there is a literal ℓ_u with $\mathrm{var}(\ell_u) = u$ and $\ell_u <_\mathcal{T} t <_\mathcal{T} x$. Because decisions have to be level-ordered in M_{LD}, all ℓ_u had to have been propagated.

Let ℓ_u be the leftmost U-literal in $\mathcal{T}$. Consider its antecedent clause $A := \mathrm{ante}_\mathcal{T}(\ell_u)$.

Claim. If ℓ_u is the leftmost U-literal in $\mathcal{T}$, then there exists an $i \in \{1, \ldots, m\}$ such that $c_i \in \mathrm{var}(\mathrm{ante}_\mathcal{T}(\ell_u))$ (where $c_1, \ldots, c_m$ are the variables from $\mathtt{Rev}(\mathtt{Twin}\Phi)$ as in Definition 7.10).

Proof of the claim. Assume not. We will show that A has to contain at least two different U-literals.

Assume that A only contains one U-literal, namely ℓ_u itself. Let Λ consist of the clauses $C_1, \ldots, C_{m'}$ and let $\mathtt{Twin}\Lambda$ consist of the clauses $C_1, \ldots, C_m$ with $m > m'$. We can assume that ℓ_u is a copy of a literal from Λ by the construction of a twin formula. In particular, ℓ_u (and $\bar{\ell}_u$) cannot be contained in the clauses $C_1, \ldots, C_{m'}$.

Let ρ_A be the LD-Q-Res derivation of A that was constructed in ι_E, but not used for $\mathfrak{R}(\iota_E)$ since verifications can only make use of cubes. By assumption, A does not contain any c_i or $\bar{c}_i$. Since A contains at least one literal from $\mathtt{Rev}(\mathtt{Twin}\Lambda)$ (namely ℓ_u), and the variables from $\mathtt{Rev}(\mathtt{Twin}\Lambda)$ and Γ are disjoint in $\mathrm{dc}(\Gamma, \mathtt{Rev}(\mathtt{Twin}\Lambda))$, we conclude that all axioms from ρ_A are clauses of $\mathtt{Rev}(\mathtt{Twin}\Lambda)$. However, each axiom clause from $\mathtt{Rev}(\mathtt{Twin}\Lambda)$ includes at least one c_i or $\bar{c}_i$. Hence, we have to resolve over these variables somehow. In particular, we need $\bar{c}_1 \vee \ldots \vee \bar{c}_m \in \rho_A$ since this is the only axiom clause where these variables occur in a negative polarity.

We will now construct another LD-Q-Res derivation ρ' by substituting $\bar{c}_1 \vee \ldots \vee \bar{c}_m$ with $\bar{c}_1 \vee \ldots \vee \bar{c}_{m'}$ in ρ_A and gradually deleting all redundant clauses. In particular, all clauses from $\mathtt{Rev}(\mathtt{Twin}\Lambda)$ that contain ℓ_u or $\bar{\ell}_u$ will be deleted because the corresponding c_i is missing. Let A' be the last clause in ρ', hence ρ' is a LD-Q-Res proof of A' from $\mathtt{Rev}(\Lambda)$. Obviously, we get $A' \subseteq A$ and $\ell_u \notin A'$ as well as $c_i, \bar{c}_i \notin A'$ for all $i = 1, \ldots, m$. Since ℓ_u was the only U-literal in A, the clause A' cannot have any U-literals. Therefore A' is a clause consisting of universal literals only. Reducing A' universally gives us the empty clause ($\bot$), which means that we can extend ρ' to a refutation of $\mathtt{Rev}(\Lambda)$. But this is a contradiction to the fact that $\mathtt{Rev}(\Lambda)$ is a true formula (by Lemma 7.11).

That shows that A must contain more than one U-literal. Let $\ell_u \neq z \in A$ be another U-literal. Then we need $\bar{z} <_\mathcal{T} \ell_u$ since z is existential. However, this contradicts the choice of ℓ_u, which finishes the proof. □

We want to create a contradiction by applying the claim, for which we need to show that A does not contain any literal from $\{c_r, \bar{c}_r \mid r = 1, \ldots, m\}$.

Assume that there is such a literal. That means we can find the leftmost literal $c \in \{c_r, \bar{c}_r \mid r = 1, \ldots, m\}$ in $\mathcal{T}$, hence $c <_\mathcal{T} \ell_u <_\mathcal{T} t <_\mathcal{T} x$. Now, c cannot have been a decision since decisions must be level-ordered. That means that c has been propagated by an antecedent clause $F := \mathrm{ante}_\mathcal{T}(c)$. Because c was leftmost, F cannot be the clause $\bar{c}_1 \vee \ldots \vee \bar{c}_m$. It is easy to see that F then has to contain either w or $\bar{w}$ by the structure of a reversion (see Definition 7.10). W.l.o.g. let $w \in F$. Then we need $\bar{w} <_\mathcal{T} c <_\mathcal{T} \ell_u$. Because of the quantification order, $\bar{w}$ cannot be a decided literal. Hence $\bar{w}$ must have been propagated by some antecedent cube $L := \mathrm{ante}_\mathcal{T}(\bar{w})$. Let ρ_L be the subproof of L

from $\mathfrak{R}(\iota_E)$. Then there exists an initial cube $G \in \rho_L$ with $w \in G$, which is not getting resolved away in ρ_L. Furthermore, G is also an initial cube in $\mathfrak{R}(\iota_E)$. By Lemma 7.12, there exists some $H \in \mathfrak{C}(\mathtt{Twin}\Lambda)$ such that $\overline{H} \subseteq G$. Since each clause of Λ contains a U-literal, there is such a U-literal $v \in \overline{H} \subseteq G$ and also $v \in L$ because it cannot be resolved or reduced away. This means we need $v <_{\mathcal{T}} \bar{w} <_{\mathcal{T}} \ell_u$, which is a contradiction to the choice of ℓ_u.

We have now shown that A does not contain any $c_r, \bar{c}_r$, $r \in \{1, \ldots, m\}$. However, this is impossible by our claim. We conclude that Case 1 cannot occur.

Case 2. t was propagated.

Consider the antecedent cube $J := \mathrm{ante}_{\mathcal{T}}(t)$. Let ρ_J be the subproof of J in $\mathfrak{R}(\iota_E)$. Then the first cubes in ρ_J were (reduced) satisfying assignments for $\mathtt{Rev}(\mathtt{Twin}\Lambda)$. At least one of these initial cubes in ρ_J contains $\bar{t}$ which will not get resolved away since it appears in J. Let $I \in \rho_J$ be an initial cube with $\bar{t} \in I$ that does not get resolved away in ρ_J. By Lemma 7.12, there exists a clause $K \in \mathfrak{C}(\mathtt{Twin}\Phi_n)$ such that $\overline{K} \subseteq I$. By our assumption, K contains at least one U- and one T-literal. But then also I contains at least one U-literal ℓ. Because ℓ is blocked by $\bar{t}$ all the time, it does not get reduced away in ρ_J, hence $\ell \in J$.

Due to $\ell <_{\mathtt{Rev}(\mathtt{Twin}\Lambda)} t$, we need $\ell <_{\mathcal{T}} t$ in order for J to become unit. W.l.o.g. let ℓ be the leftmost U-literal in $\mathcal{T}$ (the fact that $\ell \in J$ is not important anymore from this point on). Because of $x <_{\mathtt{Rev}(\mathtt{Twin}\Lambda)} \ell$, the literal ℓ cannot be a regular decision. That means it must have been propagated.

We can repeat the argument from Case 1. We conclude that such an ℓ does not exist. Thus Case 2 does not occur and we get a contradiction regarding our assumption that π was not primitive. □

Next, we want to find specific formulas to which Theorem 10.3 can be applied. We recall the well-known equality formulas $\mathtt{Eq}_n$ (cf. Definition 5.23), as well as the modification $\mathtt{ModEq}_n$ (cf. Definition 7.16). This modification adds a U-literal to some clauses such that each clause now contains at least one U- and one T-literal, which is a precondition for Theorem 10.3.

Many properties of $\mathtt{Eq}_n$ carry over to $\mathtt{ModEq}_n$ or even $\mathtt{TwinModEq}_n$. This is important for obtaining exponential lower bounds via Theorem 10.2.

Proposition 10.4 ([10, 14, 33, 38, 39])**.** $\mathtt{Eq}_n$ *needs* QU-Res *refutations of size* $2^{\Omega(n)}$ *but has* M_{LD} *refutations of quadratic trail size. Furthermore,* $\mathtt{Eq}_n$ *and* $\mathtt{ModEq}_n$ *fulfil the XT-property and gauge*$(\mathtt{Eq}_n)$ = *gauge*$(\mathtt{TwinModEq_n}) = n$. *Hence,* $\mathtt{Eq}_n$ *and* $\mathtt{TwinModEq_n}$ *need exponential-size fully reduced primitive* Q-Res *refutations.*

Using Theorem 10.3 and Proposition 10.4, we obtain:

Corollary 10.5. *Let* $\mathtt{RTME}_n := \mathtt{Rev}(\mathtt{TwinModEq}_n)$. *Then for each QCNF* Γ_n, *all* M_{LD} *proofs of any cube* E *with* $var(E) \subseteq var(\Gamma_n)$ *from the disjoint composition* $dc(\Gamma_n, \mathtt{RTME}_n)$ *have exponential trail size.*

Definition 10.6. *We define the QCNF* $\mathtt{ERTME}_n$ *as the disjoint composition*

$$\mathtt{ERTME}_n := dc(\mathtt{Eq}_n, \mathtt{RTME}_n).$$

After applying our Master Theorem 10.2 by setting $\Phi_n := \mathtt{Eq}_n$ and $\Psi_n := \mathtt{RTME}_n$, where s is quadratic by Proposition 10.4 and r is exponential by Corollary 10.5, we conclude:

Corollary 10.7. *For each* $\mathsf{M_{LD}}$ *refutation* θ_n *of* $\mathtt{ERTME}_n$ *we have trail-size*$(\theta_n) \in 2^{\Omega(n)}$*, but there exists an* $\mathsf{M_{LD}}$ *refutation* ι_n *of* $\mathtt{ERTME}_n$ *with extr-size*$(\iota_n) \in O(n^2)$*.*

10.2.2 QCDCL Based on Q-Res

For the separation on the QCDCL model $\mathsf{M_Q}$ we need a formula with linear gauge that is still easy for $\mathsf{M_Q}$. Since $\mathsf{M_Q}$ generates Q-Res proofs, $\mathtt{Eq}_n$ will not work because it is hard for Q-Res (and even QU-Res) [10]. Therefore we introduce the simplicity formulas $\mathtt{Sim}_n$, which are similar to $\mathtt{Eq}_n$, but now all universal literals occur in only one polarity.

Definition 10.8 (Simplicity formula)**.** *The QCNF* $\mathtt{Sim}_n$ *consists of the prefix*

and the matrix

$$x_i \vee u_i \vee t_i,$$
$$\bar{x}_i \vee u_i \vee t_i,$$
$$\bar{t}_1 \vee \ldots \vee \bar{t}_n$$

for $i = 1, \ldots, n$*.*

One can easily construct short $\mathsf{M_Q}$ and $\mathsf{M_{QU}}$ refutations for $\mathtt{Sim}_n$ (cf. the appendix).

Proposition 10.9. $\mathtt{Sim}_n$ *has* $\mathsf{M_Q}$ *and* $\mathsf{M_{QU}}$ *refutations with quadratic trail size.*

Proof. We will show both cases simultaneously (the opportunity to propagate universal literals will never arise).

For the sake of clarity, we denote the axiom clauses as follows:

$$P_i := x_i \vee u_i \vee t_i,$$
$$N_i := \bar{x}_i \vee u_i \vee t_i,$$
$$T_n := \bar{t}_1 \vee \ldots \vee \bar{t}_n$$

for $i \in [n]$.

First, we will one by one learn the unit cubes $[u_1], \ldots, [u_n]$. Assume we already learned $[u_1], \ldots, [u_i]$, we can then construct the trail

$$\mathcal{T}_{i+1} := (\bar{u}_1, \ldots, \bar{u}_i; \mathbf{x_1}, t_1; \ldots; \mathbf{x_i}, t_i; \mathbf{x_{i+1}}; \ldots; \mathbf{x_n}; \mathbf{u_{i+1}}; \ldots; \mathbf{u_n}; \bar{\mathbf{t}}_{\mathbf{i+1}}; \mathbf{t_{i+2}}; \ldots; \mathbf{t_n}),$$

with

$$\mathrm{ante}_{\mathcal{T}_{i+1}}(\bar{u_j}) = [u_j],$$
$$\mathrm{ante}_{\mathcal{T}_{i+1}}(t_j) = N_j,$$

for $j = 1, \dots, i$, from which we can learn the initial cube $u_{i+1} \wedge \bar{t}_{i+1} \wedge \bigwedge_{k \in [n] \setminus \{i+1\}} t_k$, which can be existentially reduced to the unit cube $[u_{i+1}]$. Note that in $\mathcal{T}_n$, the literal $\bar{t}_n$ will be propagated by T_n directly after assigning t_{n-1} instead of being decided later. However, we can still learn the cube $[u_n]$ from $\mathcal{T}_n$.

After that, we will learn the clauses

$$L_i := \bar{x}_i \vee u_i \vee \bigvee_{g=i+1}^{n} u_g \vee \bigvee_{h=1}^{i-1} \bar{t}_h,$$
$$R_i := x_i \vee u_i \vee \bigvee_{g=i+1}^{n} u_g \vee \bigvee_{h=1}^{i-1} \bar{t}_h$$

for $i = 2, \dots, n-1$. We start with the trail

$$\mathcal{U}_{n-1} := (\bar{u}_1, \dots, \bar{u}_n; \mathbf{x_1}, t_1; \dots; \mathbf{x_{n-1}}, t_{n-1}, \bar{t}_n, x_n, \bot)$$

with

$$\begin{aligned} \mathrm{ante}_{\mathcal{U}_{n-1}}(\bar{u}_j) &= [u_j], \\ \mathrm{ante}_{\mathcal{U}_{n-1}}(t_k) &= N_j, \\ \mathrm{ante}_{\mathcal{U}_{n-1}}(\bar{t}_n) &= T_n, \\ \mathrm{ante}_{\mathcal{U}_{n-1}}(\bar{x}_n) &= P_n, \\ \mathrm{ante}_{\mathcal{U}_{n-1}}(\bot) &= N_n \end{aligned}$$

for $j \in [n]$ and $k \in [n-1]$. By resolving over x_n, t_n and t_{n-1}, we can learn the clause $L_{n-1} = ((N_n \overset{x_n}{\bowtie} P_n) \overset{t_n}{\bowtie} T_n) \overset{t_{n-1}}{\bowtie} N_{n-1}$. Analogously, by flipping the x_k decisions, we can construct a trail $\mathcal{V}_{n-1}$, from which we can learn $R_{n-1} = ((P_n \overset{x_n}{\bowtie} N_n) \overset{t_n}{\bowtie} T_n) \overset{t_{n-1}}{\bowtie} P_{n-1}$.

Assume we already learned the clauses $L_{n-1}, R_{n-1}, \dots, L_i, R_i$ for some $i \in \{3, \dots, n-1\}$. Then we can construct the trail

$$\mathcal{U}_{i-1} := (\bar{u}_1, \dots, \bar{u}_n; \mathbf{x_1}, t_1; \dots; \mathbf{x_{i-1}}, t_{i-1}, \bar{x}_i, \bot),$$

with

$$\begin{aligned} \mathrm{ante}_{\mathcal{U}_{i-1}}(\bar{u}_j) &= [u_j], \\ \mathrm{ante}_{\mathcal{U}_{i-1}}(t_k) &= N_k, \\ \mathrm{ante}_{\mathcal{U}_{i-1}}(\bar{x}_i) &= L_i, \\ \mathrm{ante}_{\mathcal{U}_{i-1}}(\bot) &= R_i \end{aligned}$$

for $j \in [n]$ and $k \in [i-1]$, from which we can learn $L_{i-1} = (R_i \overset{x_i}{\bowtie} L_i) \overset{t_{i-1}}{\bowtie} N_{i-1}$ by resolving over $\bar{x}_i$ and t_{i-1}. Again, by flipping the polarities of the decisions x_k, we can construct a trail $\mathcal{V}_{i-1}$, from which we can learn $R_{i-1} = (L_i \overset{x_i}{\bowtie} R_i) \overset{t_{i-1}}{\bowtie} P_{i-1}$.

After having learned L_2 and R_2, we create the last two trails as follows:

$$\mathcal{U}_1 := (\bar{u}_1, \ldots, \bar{u}_n; \mathbf{x_1}, t_1, x_2, \bot),$$

with

$$\begin{aligned} \mathrm{ante}_{\mathcal{U}_1}(\bar{u}_j) &= [u_j], \\ \mathrm{ante}_{\mathcal{U}_1}(t_1) &= N_1, \\ \mathrm{ante}_{\mathcal{U}_1}(x_2) &= R_2, \\ \mathrm{ante}_{\mathcal{U}_1}(\bot) &= L_2 \end{aligned}$$

for $j \in [n]$, from which we learn $(\bar{x}_1) = \mathrm{red}((L_2 \overset{x_2}{\bowtie} R_2) \overset{t_1}{\bowtie} N_1)$ by resolving over x_2 and t_1, and then

$$\mathcal{V}_1 := (\bar{u}_1, \ldots, \bar{u}_n, \bar{x}_1, t_1, x_2, \bot)$$

with

$$\begin{aligned} \mathrm{ante}_{\mathcal{V}_1}(\bar{u}_j) &= [u_j], \\ \mathrm{ante}_{\mathcal{V}_1}(\bar{x}_1) &= (\bar{x}_1), \\ \mathrm{ante}_{\mathcal{V}_1}(t_1) &= P_1, \\ \mathrm{ante}_{\mathcal{V}_1}(x_2) &= R_2, \\ \mathrm{ante}_{\mathcal{V}_1}(\bot) &= L_2 \end{aligned}$$

for $j \in [n]$, from which we can learn the empty clause by resolving over every existential variable in $\mathcal{V}_1$.

Obviously, the size of the $\mathsf{M_Q}$ (resp. $\mathsf{M_{QU}}$) refutation consisting of the trails

$$\mathcal{T}_1, \ldots, \mathcal{T}_n, \mathcal{U}_{n-1}, \mathcal{V}_{n-1}, \ldots, \mathcal{U}_1, \mathcal{V}_1$$

is quadratic. □

Yet, the gauge properties of $\mathtt{Eq}_n$ carry over to $\mathtt{Sim}_n$.

Proposition 10.10. *The QCNF* $\mathtt{Sim}_n$ *fulfils the XT-property and gauge*$(\mathtt{Sim}_n) = n$. *Hence, each fully reduced primitive* Q-Res *refutation of* $\mathtt{Sim}_n$ *has exponential size.*

Proof. Clearly, $\mathtt{Sim}_n$ fulfils the XT-property. We can compute gauge$(\mathtt{Sim}_n)$ by counting the number of X-literals that are getting piled up in proofs in which we get rid of all T-literals. Because $\bar{t}_1 \vee \ldots \vee \bar{t}_n$ is the only clause with negative T-literals, we have to get rid of each $\bar{t}_i$ for $i = 1, \ldots, n$. For this we need the literal t_i from the clauses $x \vee u_i \vee t_i$ or $\bar{x}_i \vee \bar{u}_i \vee t_i$. Hence, for each i, we need at least one of these two clauses, from which we will respectively obtain the X-literal x_i or $\bar{x}_i$. We conclude that we will always pile up n different X-literals, hence gauge$(\mathtt{Sim}_n) = n$. □

Now that we have found our candidate for Φ_n in Theorem 10.2, let us construct a suitable formula for Ψ_n. For this, we need to find a true formula that is hard to verify in M_Q. We will again make use of the idea of a reversion of a formula that is already hard for Q-Res (and QU-Res).

Proposition 10.11. *The true QCNF* $\mathtt{Rev}(\mathtt{Eq}_n)$ *needs exponential-trail-size* QU-Con *verifications.*

Next, we have to show that deriving a cube from $\mathrm{dc}(\Gamma, \Psi)$ without variables from Ψ is as hard as verifying Ψ.

Proposition 10.12. *For each QCNF* Γ *and* Ψ*, from each* M_Q *(resp.* M_QU*) proof* ι_E *of some cube* E *from* $dc(\Gamma, \Psi)$ *with* $var(E) \subseteq var(\Gamma)$ *we can extract a* Q-Con *(resp.* QU-Con*) verification* ρ *of* Ψ *with* $|\rho| \leq |\iota_E|$*.*

Proof. Because the variables from Γ and Ψ are disjoint by definition, we can simply delete all literals of Γ from the Q-Con or QU-Con proof $\mathfrak{R}(\iota_E)$ and, if necessary, shorten the proof. We then obtain a Q-Con or QU-Con verification ρ of Ψ with $|\rho| \leq |\mathfrak{R}(\iota_E)| \leq |\iota_E|$. □

From Propositions 10.11 and 10.12 we conclude:

Corollary 10.13. *For each QCNF* Γ_n*, the formula* $dc(\Gamma_n, \mathtt{Rev}(\mathtt{Eq}_n))$ *needs exponential-trail-size* M_Q *and* M_QU *proofs of any cube* E *with* $var(E) \subseteq var(\Gamma_n)$*.*

We combine $\mathtt{Sim}_n$ with $\mathtt{Rev}(\mathtt{Eq}_n)$ and obtain our formula for the separation.

Definition 10.14. *We define the QCNF* $\mathtt{SRE}_n$ *as the disjoint composition*

$$\mathtt{SRE}_n := dc(\mathtt{Sim}_n, \mathtt{Rev}(\mathtt{Eq}_n)).$$

Applying Theorem 10.2 with $\Phi_n := \mathtt{Sim}_n$ and $\Psi_n := \mathtt{Rev}(\mathtt{Eq}_n)$ where s is quadratic by Proposition 10.9 and r is exponential by Corollary 10.13, we conclude:

Corollary 10.15. *For each* M_Q *refutation* θ_n *of* $\mathtt{SRE}_n$ *we have trail-size*$(\theta_n) \in 2^{\Omega(n)}$*, but there exists an* M_Q *refutation* ι_n *of* $\mathtt{SRE}_n$ *with extr-size*$(\iota_n) \in O(n^2)$*.*

10.2.3 QCDCL Based on QU-Res

For the last separation in the QCDCL model M_QU – recently implemented as a QBF solver [95] – we can use the same formulas as for M_Q and Q-Res, as we only have to prove that no genuine QU-Res proofs can be generated.

Lemma 10.16. *Each* QU-Res *refutation of* $\mathtt{Sim}_n$ *is a* Q-Res *refutation.*

Proof. All universal variables from $\mathtt{Sim}_n$ occur only in one polarity, hence we cannot resolve over them. □

As for Corollary 10.15, we conclude:

Corollary 10.17. *For each* M_QU *refutation* θ_n *of* $\mathtt{SRE}_n$ *we have trail-size*$(\theta_n) \in 2^{\Omega(n)}$*, but there exists a* M_QU *refutation* ι_n *of* $\mathtt{SRE}_n$ *with extr-size*$(\iota_n) \in O(n^2)$*.*

Chapter 11

Conclusion

Quantification adds a significant level of complexity to the SAT problem. While CDCL is basically the only paradigm used for practical SAT solving, the situation is not as clear in the QBF case — not least because it is not even obvious which version of QCDCL to use. In each chapter, we have proposed various versions of QCDCL that differ from the classic one, which have led to stronger QCDCL proof systems within our framework.

In practice, however, these modified QCDCL variants only demonstrate the potential to find shorter proofs. It is by no means guaranteed that such short proofs can be actually constructed in a reasonable time. For example, allowing out-of-order decisions can be beneficial *if* the 'right' decisions are made. This is simply due to how we handle polynomial upper bounds: We say that a formula is 'easy' for a proof system, if there exists a polynomial-sized proof, independent of whether such a proof can be easily found or how many short proofs exist for this formula.

In this sense, each proof system is somewhat non-deterministic, and it would be unfair to compare fully deterministic algorithmic proof systems (like practical CDCL or practical QCDCL) with their non-deterministic underlying proof systems (like Resolution or Q-Res), as this would lead to the P vs. NP problem in one way or another. Other interpretations of upper bounds, like worst-case upper bounds, might avoid those issues, but could also be unsuitable for handling resolution- and CDCL-like proof systems as they can become arbitrarily large.

Of course this does not devalue our results regarding stronger QCDCL variants. For example, as one of our main results in Chapter 7 we observed that QCDCL versions with arbitrary decisions are exponentially stronger than variants in which decisions need to be level-ordered. This does not necessarily transfer to the practice one-on-one, because with a less restricted decision policy we also risk making more 'wrong' decisions. However, our results could clarify in which situations it would be advisable to decide out-of-order. The gauge lower bound technique relies on the fact that on some formulas, level-ordered decisions cause the trails to become level-ordered as well. Thus, choosing variables out-of-order when a level-ordered trail is detected may disrupt the sequence and accelerate the refutation by surpassing the gauge lower bound.

Furthermore, as we have seen in Chapter 8, it can be beneficial to turn off reductions

for some variables. Obviously, this means we need suitable heuristics to implement the ANY-RED policy in practice. In the simulation of LD-Q-Res by a particular QCDCL variant, we have ascertained that reductions during Unit Propagations are deeply connected to the existence and derivation of merged literals in LD-Q-Res proofs. If we could measure when a merged literal is needed, this might lead to a heuristic that tells us when to turn off reduction in a QCDCL run.

Finding heuristics for our modifications belongs to a different discipline which can only be dealt with experimentally. In [37, 39] the influence of out-of-order decisions were evaluated via experiments on QCDCL solver Qute [82]. While one could observe a visible improvement of QCDCL's performance on some of our pre-designed separation formulas, the results were not as clear on benchmark formulas as the implementation of our decision policies was done rather ad hoc. In other words, QCDCL sometimes does not seem to find the short proofs that exist in theory. More suitable heuristics could remedy this problem, although adding a fully deterministic heuristic would weaken the QCDCL proof system on the theory side. In any case, further experimental evaluation and the implementation of the proposed modifications poses the most obvious and probably most important next step.

In recent years, so-called Dependency Learning has been established as a method for implementing out-of-order decisions in QBF solving (cf. [82]). QCDCL with Dependency Learning ignores the quantification order while creating a trail and instead learns dependencies in case any unsound resolution steps are detected during constraint learning. These dependencies then affect decision making and Unit Propagation. For example, if a dependency of a variable t on u is learned, QCDCL is no longer allowed to decide t before u or reduce u in clauses in which t is still present during Unit Propagation. In particular, QCDCL is allowed to decide variables arbitrarily as long as no dependencies are learned. In the classic QCDCL model with level-ordered decisions, such dependencies are fully defined by the prefix of the QCNF, meaning that each quantifier block depends on all quantifier blocks left of it.

A more static approach is the implementation of Dependency Schemes [91, 92]. Besides adding such schemes to resolution-based proof systems like Q-Res or LD-Q-Res (cf. [81, 83, 97]), they an be applied algorithmically as well. Instead of learning dependencies gradually as in Dependency Learning, it is also possible to start with a pre-defined Dependency Scheme like the *Standard Dependency Scheme* [92] or the *Resolution Path Dependency Scheme* [54, 96]. Experimental evaluation shows that particular Dependency Schemes can improve the performance of QCDCL despite a slight overhead caused by the computation of the scheme itself (cf. [75]). Quite surprisingly, recent theoretical analysis of QCDCL with Dependency Schemes shows an incomparability to the classic variant, which means that Dependency Schemes could even be disadvantageous on some instances (cf. [46]).

It is important to point out that our policies like ANY-ORD do not necessarily align with dependencies. Hence, such policies could potentially even added to QCDCL on top of Dependency Learning or Dependency Schemes because they allow out-of-order decisions even if genuine dependencies between the corresponding variables are present

and detected. That said, a theoretical analysis of the effect of Dependency Learning on QCDCL as a proof system, with or without additional decision policies, mark an important and significant direction for future research.

There are plenty of other extensions and modifications of both Q-Res and QCDCL which we have not considered so far. For example, *reductionless* Q-Res[1] [11, 31, 84] disallows the reduction rule completely. Instead, reductionless Q-Res refutations are simply derivations of clauses consisting of universal literals only. Q-Res *with symmetries* [70, 71] exploits QBFs with symmetrical structures and strengthens the Q-Res calculus.

Preprocessing represents another important aspect in practical QBF solving (as well as SAT solving). Throughout the years, plenty of preprocessing techniques were introduced [30, 42, 55, 57, 93, 101] that can significantly simplify a given formula before it gets passed to the solver itself. The *QRAT* [62] proof system and its generalization $QRAT^+$ [77], extensions of the so-called *RAT (resolution asymmetric tautology)* proof system [60, 61] from SAT, consists of rules with which clauses can be added or removed from a given formula without changing the truth value. Both QRAT and $QRAT^+$, which turned out to be equivalent as refutational proof systems [45], can be used to take preprocessing into account and certify the correctness of its outcome.

Let us now recall and give answers to our opening questions from the introduction. In the initial situation, QCDCL was not able to even simulate Q-Res, while CDCL was proven to be equivalent to Resolution. No lower bound techniques for QCDCL were known besides lower bounds for its underlying proof system LD-Q-Res or lower bounds on particular formulas like $\mathtt{CR}_n$ (cf. [64]).

On which QBF instances does QCDCL fail and why? We have introduced a lower bound technique specitally for QCDCL (cf. Chapters 5 and 6). We have identified a class of instances (cf. Definition 5.3) which this lower bound technique can be applied on. The technique exploits the fact that decisions are level-ordered in the classic version of QCDCL, which causes the trails for formulas that belong to the aforementioned to class be level-ordered as well.

What are already established modifications of QCDCL and how do they impact its performance? We analysed the influence of Cube Learning and Pure Literal Elimination on QCDCL (cf. Chapter 6). While Cube Learning always strengthens QCDCL (cf. Theorems 6.11 and 6.32), Pure Literal Elimination was disadvantageous on some instances and can therefore be treated as incomparable to versions without it (cf. Theorems 6.21, 6.26 and 6.29). An interesting observation is that Cube Learning even helped with refuting false formulas. More precisely, Cube Learning has the ability to break the level order of decisions and trails and can therefore prevent the effects of the gauge lower bound at least on some formulas. Furthermore, our results are supported by an experimental evaluation in [36, 38][2], in which the influence of Cube Learning and Pure Literal Elimination on the performance of the QCDCL solver DebQBF [76] was

[1] Implemented in the QBF solver GhostQ [73].

[2] The experiments were omitted in this thesis.

analysed. We conclude that Cube Learning should not be omitted, even if we already know that the formula is false.

How can QCDCL be improved with further modifications not yet established? Although we observed that Cube Learning can overcome the gauge lower bound, we can still rule out its benefits via several formula modifications (e.g. via twin-modification, cf. Definition 7.9). Hence, we needed further improvements in order to obtain a QCDCL variant that might potentially simulate its underlying proof system. An important step was to allow arbitrary (i.e. out-of-order) decision making (cf. Chapter 7). We even defined decision policies that at least partially guaranteed the existence of asserting constraints, although it is still unclear whether or not asserting constraints improves the performance of QCDCL in a theoretical sense. The three new QCDCL variants turned out to be exponentially stronger than the classic version (cf. Theorem 7.29). More precisely, one variant is expected to perform better on false formulas, while the other one prefers true formulas. Although the truth values of QBF instances are unknown in the common setting, these versions could still be very helpful for parallel solving or on sets of formulas for which the truth value is known and only proofs are desired. Besides decision policies, we also introduced reduction and propagation policies that alter Unit Propagation (cf. Chapter 8). While the new propagation policy allows the generation of QU-Res proofs, the reduction policies control universal or existential reduction during Unit Propagation. Such reduction policies can be very powerful as we have shown that QCDCL versions with arbitrary reductions (i.e., reductions can be turned on and off for each propagation) result in an exponentially stronger proof system (cf. Theorem 8.8).

Can we modify QCDCL such that it becomes equivalent to its underlying proof system? Yes — by making use of suitable combinations of aforementioned policies, we were able to construct QCDCL variants that can p-simulate Q-Res, mLD-Q-Res[3] or QU-Res refutations, respectively (cf. Chapter 9). The simulation is roughly based on the approach that was used to simulate Resolution by CDCL (cf. [86]).

Although implementability is the most pressing issue, there are still other open questions that we have not covered but that emerge from our results. Throughout this thesis, we almost always made use of the gauge lower bound in order to obtain exponential lower bounds for QCDCL. This technique is restricted to Σ_3^b QCNFs and it is unclear whether or not we could redesign it to be applicable to formula families with arbitrary (or even unbounded) quantifier depth.

Another question is whether the gauge lower bound technique can be extended to true formulas. So far we have found only one way to translate the gauge lower bound to true formulas (cf. Theorem 7.15), but that approach deeply depends on the structure of the input formula and ultimately goes back to the technique for false formulas. Instead, one

[3] mLD-Q-Res is a subsystem of LD-Q-Res which simulates all of our QCDCL variants that are already simulated by LD-QU-Res

could aim to translate the results from Chapter 5 (especially the notions of XT-property, quasi level-ordered proofs and formula gauge) to true formulas.

The main disadvantage of the gauge lower bound is probably the non-applicability to QCDCL versions in which decisions do not need to be level-ordered. So far, we still lack lower bounds (or even techniques) for QCDCL with decision policies like ANY-ORD, UNI-ANY or EXI-ANY (except lower bounds for Q-Res, LD-Q-Res or QU-Res themselves).

In Chapter 9, we were able to characterise Q-Res, QU-Res and mLD-Q-Res, which is a slight modification of LD-Q-Res, via suitable QCDCL versions. One could now assume that characterising LD-QU-Res (or a slight modification of it) should be possible in the same way. However, as it turns out, our simulations technique seems to be incompatible with LD-QU-Res. More precisely, Lemma 9.5 demands that the use of ALL-PROP (which is needed for the generation of QU-Res proofs), or the use of ANY-RED or ALL-RED (which is necessary to create LD-Q-Res proofs) is disallowed. Hence, p-simulating LD-QU-Res by a QCDCL variant (potentially $\mathsf{QCDCL}^{\text{ANY-ORD}}_{\text{ANY-RED,ALL-PROP}}$) would therefore be a problem that would have to be solved with a fundamentally different approach. Furthermore, it is still an open question whether or not LD-Q-Res and mLD-Q-Res can be exponentially separated (otherwise they would be equivalent). Both potential outcomes would be interesting, as it would either mean we characterised LD-Q-Res via QCDCL, or we can conclude that LD-Q-Res cannot be characterised by any of these QCDCL variants whatsoever.

Besides those considered in this thesis, there are other proof systems following a fundamentally different approach. *Polynomial Calculus* (*PC*, cf. [47]) and *Polynomial Calculus with Resolution* (*PCR*, cf. [1]) are algebraic propositional proof systems based on *Hilbert's Nullstellensatz*[4] [6]. This proof system works on polynomials instead of clauses and thereby allows a wider range of arithmetic operations on constraints, such as additions and multiplications. Even though such approaches have received little attention in practical solving, there are some SAT solvers that follow the Polynomial Calculus paradigm [40, 41] by implementing Buchberger's algorithm for finding Gröbner bases [52]. *Cutting Planes* [51] is a geometrical proof system in which constraints are translated into linear inequalities, such that satisfying assignments can be computed via integer linear programming. The *Frege* proof system [50, 89] can be considered as *the* classic textbook proof system, being defined via several axiom and propositional rules.

Although these aforementioned proof systems are propositional, they can be lifted to QBF proof system by adding a reduction rule (cf. [9, 16]). Roughly speaking, these proof systems can be categorized into *line-based proof systems* that follow a similar structure, which allows us to extend them to QBF by applying the same modification [9]. By this rather simple extension, it is possible to obtain QBF PCR [9], QBF Cutting Planes [23] and QBF Frege systems [16]. As most QBF solvers are either resolution-based or expansion-based, developing solvers for these alternative proof systems could, similar to QCDCL, potentially enable deeper characterizations and broaden our understanding of these systems.

[4] *Nullstellensatz* is also the name of a static algebraic propositional proof system.

Bibliography

[1] Michael Alekhnovich, Eli Ben-Sasson, Alexander A. Razborov, and Avi Wigderson. Space complexity in propositional calculus. *SIAM Journal on Computing*, 31(4):1184–1211, 2002.

[2] Valeriy Balabanov and Jie-Hong R. Jiang. Resolution proofs and skolem functions in QBF evaluation and applications. In Ganesh Gopalakrishnan and Shaz Qadeer, editors, *Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings*, volume 6806 of *Lecture Notes in Computer Science*, pages 149–164. Springer, 2011.

[3] Valeriy Balabanov and Jie-Hong R. Jiang. Unified QBF certification and its applications. *Form. Methods Syst. Des.*, 41(1):45–65, 2012.

[4] Valeriy Balabanov, Jie-Hong Roland Jiang, Mikolas Janota, and Magdalena Widl. Efficient extraction of QBF (counter)models from long-distance resolution proofs. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 3694–3701, 2015.

[5] Valeriy Balabanov, Magdalena Widl, and Jie-Hong R. Jiang. QBF resolution systems and their proof complexities. In *Proc. Theory and Applications of Satisfiability Testing (SAT)*, pages 154–169, 2014.

[6] Paul Beame, Russell Impagliazzo, Jan Krajícek, Toniann Pitassi, and Pavel Pudlák. Lower bound on Hilbert's Nullstellensatz and propositional proofs. In *Proc. 35th IEEE Symposium on the Foundations of Computer Science*, pages 794–806, 1994.

[7] Paul Beame, Henry A. Kautz, and Ashish Sabharwal. Towards understanding and harnessing the potential of clause learning. *J. Artif. Intell. Res. (JAIR)*, 22:319–351, 2004.

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

[9] Olaf Beyersdorff, Joshua Blinkhorn, and Luke Hinde. Size, cost, and capacity: A semantic technique for hard random QBFs. In *Proc. Conference on Innovations in Theoretical Computer Science (ITCS'18)*, pages 9:1–9:18, 2018.

[10] 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.

[11] 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.

[12] Olaf Beyersdorff, Joshua Blinkhorn, and Meena Mahajan. Hardness characterisations and size-width lower bounds for QBF resolution. In *Proc. ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 209–223. ACM, 2020.

[13] Olaf Beyersdorff and Benjamin Böhm. Understanding the Relative Strength of QBF CDCL Solvers and QBF Resolution. In *12th Innovations in Theoretical Computer Science Conference (ITCS 2021)*, volume 185 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 12:1–12:20, 2021.

[14] Olaf Beyersdorff and Benjamin Böhm. Understanding the relative strength of QBF CDCL solvers and QBF resolution. *Log. Methods Comput. Sci.*, 19(2), 2023.

[15] Olaf Beyersdorff, Benjamin Böhm, and Meena Mahajan. Runtime vs. extracted proof size: An exponential gap for CDCL on qbfs. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan, editors, *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 7943–7951. AAAI Press, 2024.

[16] Olaf Beyersdorff, Ilario Bonacina, and Leroy Chew. Lower bounds: From circuits to QBF proof systems. In *Proc. ACM Conference on Innovations in Theoretical Computer Science (ITCS'16)*, pages 249–260. ACM, 2016.

[17] 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.

[18] Olaf Beyersdorff, Leroy Chew, and Mikoláš Janota. On unification of QBF resolution-based calculi. In *Proc. 39th Symposium on Mathematical Foundations of Computer Science*, volume 8635 of *Lecture Notes in Computer Science*, pages 81–93, 2014.

[19] Olaf Beyersdorff, Leroy Chew, and Mikoláš Janota. Proof complexity of resolution-based QBF calculi. In *Proc. Symposium on Theoretical Aspects of Computer Science (STACS'15)*, pages 76–89. LIPIcs, 2015.

[20] Olaf Beyersdorff, Leroy Chew, Meena Mahajan, and Anil Shukla. Are short proofs narrow? QBF resolution is not simple. In *Proc. Symposium on Theoretical Aspects of Computer Science (STACS'16)*, pages 15:1–15:14, 2016.

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

[22] 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.

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

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

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

[26] Olaf Beyersdorff, Nicola Galesi, and Massimo Lauria. A characterization of tree-like resolution size. *Information Processing Letters*, 113(18):666–671, 2013.

[27] Olaf Beyersdorff, Nicola Galesi, and Massimo Lauria. Parameterized complexity of DPLL search procedures. *ACM Transactions on Computational Logic*, 14(3):20:1–20:21, 2013.

[28] Olaf Beyersdorff, Mikolás Janota, Florian Lonsing, and Martina Seidl. Quantified Boolean formulas. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, Frontiers in Artificial Intelligence and Applications, pages 1177–1221. IOS Press, 2021.

[29] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability*, Frontiers in Artificial Intelligence and Applications. IOS Press, 2021.

[30] Armin Biere, Florian Lonsing, and Martina Seidl. Blocked clause elimination for QBF. In Nikolaj S. Bjørner and Viorica Sofronie-Stokkermans, editors, *Automated Deduction - CADE-23 - 23rd International Conference on Automated Deduction, Wroclaw, Poland, July 31 - August 5, 2011. Proceedings*, volume 6803 of *Lecture Notes in Computer Science*, pages 101–115. Springer, 2011.

[31] Nikolaj Bjørner, Mikolás Janota, and William Klieber. On conflicts and strategies in QBF. In Ansgar Fehnker, Annabelle McIver, Geoff Sutcliffe, and Andrei Voronkov, editors, *20th International Conferences on Logic for Programming, Artificial Intelligence and Reasoning LPAR 2015*, volume 35 of *EPiC Series in Computing*, pages 28–41. EasyChair, 2015.

[32] Benjamin Böhm and Olaf Beyersdorff. Lower bounds for QCDCL via formula gauge. In Chu-Min Li and Felip Manyà, editors, *Theory and Applications of Satisfiability Testing (SAT)*, pages 47–63, Cham, 2021. Springer International Publishing.

[33] Benjamin Böhm and Olaf Beyersdorff. Lower bounds for QCDCL via formula gauge. *J. Autom. Reason.*, 67(4):35, 2023.

[34] Benjamin Böhm and Olaf Beyersdorff. QCDCL vs QBF Resolution: Further Insights. In Meena Mahajan and Friedrich Slivovsky, editors, *26th International Conference on Theory and Applications of Satisfiability Testing (SAT 2023)*, volume 271 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 4:1–4:17, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[35] Benjamin Böhm and Olaf Beyersdorff. Qcdcl vs qbf resolution: Further insights. *J. Artif. Intell. Res. (JAIR)*, 81:741–769, 2024.

[36] Benjamin Böhm, Tomás Peitl, and Olaf Beyersdorff. QCDCL with cube learning or pure literal elimination - what is best? In Luc De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1781–1787. ijcai.org, 2022.

[37] Benjamin Böhm, Tomás Peitl, and Olaf Beyersdorff. Should decisions in QCDCL follow prefix order? In Kuldeep S. Meel and Ofer Strichman, editors, *25th International Conference on Theory and Applications of Satisfiability Testing (SAT)*, volume 236 of *LIPIcs*, pages 11:1–11:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[38] Benjamin Böhm, Tomás Peitl, and Olaf Beyersdorff. QCDCL with cube learning or pure literal elimination - what is best? *Artif. Intell.*, 336:104194, 2024.

[39] Benjamin Böhm, Tomás Peitl, and Olaf Beyersdorff. Should decisions in QCDCL follow prefix order? *J. Autom. Reason.*, 68(1):5, 2024.

[40] Michael Brickenstein and Alexander Dreyer. Polybori: A framework for gröbner-basis computations with boolean polynomials. *J. Symb. Comput.*, 44(9):1326–1345, 2009.

[41] Michael Brickenstein, Alexander Dreyer, Gert-Martin Greuel, Markus Wedler, and Oliver Wienand. New developments in the theory of gröbner bases and applications to formal verification. *Journal of Pure and Applied Algebra*, 213(8):1612–1635, 2009. Theoretical Effectivity and Practical Effectivity of Gröbner Bases.

[42] Uwe Bubeck and Hans Kleine Büning. Bounded universal expansion for preprocessing QBF. In João Marques-Silva and Karem A. Sakallah, editors, *Theory and Applications of Satisfiability Testing - SAT 2007, 10th International Conference, Lisbon, Portugal, May 28-31, 2007, Proceedings*, volume 4501 of *Lecture Notes in Computer Science*, pages 244–257. Springer, 2007.

[43] H. Kleine Büning and T. Lettmann. *Propositional Logic: Deduction and Algorithms*, volume 48 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1999.

[44] Marco Cadoli, Andrea Giovanardi, and Marco Schaerf. An algorithm to evaluate quantified Boolean formulae. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence and Tenth Innovative Applications of Artificial Intelligence Conference (AAAI)*, pages 262–267, 1998.

[45] Leroy Chew and Judith Clymo. The equivalences of refutational QRAT. In Mikolás Janota and Inês Lynce, editors, *Theory and Applications of Satisfiability Testing - SAT 2019 - 22nd International Conference, SAT 2019, Lisbon, Portugal, July 9-12, 2019, Proceedings*, volume 11628 of *Lecture Notes in Computer Science*, pages 100–116. Springer, 2019.

[46] Abhimanyu Choudhury and Meena Mahajan. Dependency schemes in cdcl-based QBF solving: A proof-theoretic study. *J. Autom. Reason.*, 68(3):16, 2024.

[47] Matthew Clegg, Jeff Edmonds, and Russell Impagliazzo. Using the Groebner basis algorithm to find proofs of unsatisfiability. In *Proc. 28th ACM Symposium on Theory of Computing*, pages 174–183, 1996.

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

[49] Stephen A. Cook. The complexity of theorem proving procedures. In *Proc. 3rd Annual ACM Symposium on Theory of Computing*, pages 151–158, 1971.

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

[51] William Cook, Collette R. Coullard, and György Turán. On the complexity of cutting-plane proofs. *Discrete Applied Mathematics*, 18(1):25–38, 1987.

[52] David A. Cox, John Little, and Donal O'Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Springer-Verlag, Berlin, Heidelberg, 2007.

[53] Martin Davis, George Logemann, and Donald W. Loveland. A machine program for theorem-proving. *Commun. ACM*, 5(7):394–397, 1962.

[54] Allen Van Gelder. Variable independence and resolution paths for quantified boolean formulas. In Jimmy Ho-Man Lee, editor, *Principles and Practice of Constraint Programming - CP 2011 - 17th International Conference, CP 2011, Perugia, Italy, September 12-16, 2011. Proceedings*, volume 6876 of *Lecture Notes in Computer Science*, pages 789–803. Springer, 2011.

[55] Allen Van Gelder, Samuel B. Wood, and Florian Lonsing. Extended failed-literal preprocessing for quantified boolean formulas. In Alessandro Cimatti and Roberto Sebastiani, editors, *Theory and Applications of Satisfiability Testing - SAT 2012 - 15th International Conference, Trento, Italy, June 17-20, 2012. Proceedings*, volume 7317 of *Lecture Notes in Computer Science*, pages 86–99. Springer, 2012.

[56] Mohammad Ghasemzadeh, Volker Klotz, and Christoph Meinel. Embedding memoization to the semantic tree search for deciding qbfs. In Geoffrey I. Webb and Xinghuo Yu, editors, *AI 2004: Advances in Artificial Intelligence, 17th Australian Joint Conference on Artificial Intelligence, Cairns, Australia, December 4-6, 2004, Proceedings*, volume 3339 of *Lecture Notes in Computer Science*, pages 681–693. Springer, 2004.

[57] Enrico Giunchiglia, Paolo Marin, and Massimo Narizzano. squeezebf: An effective preprocessor for qbfs based on equivalence reasoning. In Ofer Strichman and Stefan Szeider, editors, *Theory and Applications of Satisfiability Testing - SAT 2010, 13th International Conference, SAT 2010, Edinburgh, UK, July 11-14, 2010. Proceedings*, volume 6175 of *Lecture Notes in Computer Science*, pages 85–98. Springer, 2010.

[58] Enrico Giunchiglia, Massimo Narizzano, and Armando Tacchella. Clause/term resolution and learning in the evaluation of quantified Boolean formulas. *J. Artif. Intell. Res.*, 26:371–416, 2006.

[59] 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, 2011.

[60] Marijn Heule, Warren A. Hunt Jr., and Nathan Wetzler. Trimming while checking clausal proofs. In *Formal Methods in Computer-Aided Design, FMCAD 2013, Portland, OR, USA, October 20-23, 2013*, pages 181–188. IEEE, 2013.

[61] Marijn Heule, Warren A. Hunt Jr., and Nathan Wetzler. Verifying refutations with extended resolution. In *Automated Deduction - CADE-24 - 24th International Conference on Automated Deduction*, pages 345–359, 2013.

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

[63] Cynthia J. Huffman. *The Game of Ur: An Exercise in Strategic Thinking and Problem Solving and A Fun Math Club Activity.* Open Educational Resources - Math. 15., 2019.

[64] Mikolás Janota. On Q-Resolution and CDCL QBF solving. In *Proc. International Conference on Theory and Applications of Satisfiability Testing (SAT)*, pages 402–418, 2016.

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

[66] Mikolás Janota and João Marques-Silva. Abstraction-based algorithm for 2qbf. In Karem A. Sakallah and Laurent Simon, editors, *Theory and Applications of Satisfiability Testing - SAT 2011 - 14th International Conference, SAT 2011, Ann Arbor, MI, USA, June 19-22, 2011. Proceedings*, volume 6695 of *Lecture Notes in Computer Science*, pages 230–244. Springer, 2011.

[67] Mikolás Janota and João Marques-Silva. On propositional QBF expansions and Q-resolution. In *Theory and Applications of Satisfiability Testing - SAT 2013 - 16th International Conference, Helsinki, Finland, July 8-12, 2013. Proceedings*, pages 67–82, 2013.

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

[69] Mikoláš Janota and Joao Marques-Silva. An Achilles' heel of term-resolution. In Eugénio Oliveira, João Gama, Zita Vale, and Henrique Lopes Cardoso, editors, *Progress in Artificial Intelligence*, pages 670–680, Cham, 2017. Springer International Publishing.

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

[71] Manuel Kauers and Martina Seidl. Symmetries of quantified boolean formulas. In Olaf Beyersdorff and Christoph M. Wintersteiger, editors, *Theory and Applications of Satisfiability Testing - SAT 2018 - 21st International Conference, SAT 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 9-12, 2018, Proceedings*, volume 10929 of *Lecture Notes in Computer Science*, pages 199–216. Springer, 2018.

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

[73] William Klieber, Samir Sapra, Sicun Gao, and Edmund M. Clarke. A non-prenex, non-clausal QBF solver with game-state learning. In Ofer Strichman and Stefan Szeider, editors, *Theory and Applications of Satisfiability Testing - SAT 2010, 13th International Conference, SAT 2010, Edinburgh, UK, July 11-14, 2010. Proceedings*, volume 6175 of *Lecture Notes in Computer Science*, pages 128–142. Springer, 2010.

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

[75] Florian Lonsing. *Dependency Schemes and Search-Based QBF Solving: Theory and Practice*. PhD thesis, Johannes Kepler University Linz, 2012.

[76] Florian Lonsing and Uwe Egly. DepQBF 6.0: A search-based QBF solver beyond traditional QCDCL. In *Proc. International Conference on Automated Deduction (CADE)*, pages 371–384, 2017.

[77] Florian Lonsing and Uwe Egly. QRAT+: generalizing QRAT by a more powerful QBF redundancy property. *CoRR*, abs/1804.02908, 2018.

[78] Florian Lonsing, Uwe Egly, and Allen Van Gelder. Efficient clause learning for quantified Boolean formulas via QBF pseudo unit propagation. In *Proc. International Conference on Theory and Applications of Satisfiability Testing (SAT)*, pages 100–115, 2013.

[79] João P. Marques Silva and Karem A. Sakallah. GRASP - a new search algorithm for satisfiability. In *Proc. IEEE/ACM International Conference on Computer-aided Design (ICCAD)*, pages 220–227, 1996.

[80] Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an efficient SAT solver. In *Proceedings of the 38th Design Automation Conference, DAC 2001, Las Vegas, NV, USA, June 18-22, 2001*, pages 530–535. ACM, 2001.

[81] Tomás Peitl, Friedrich Slivovsky, and Stefan Szeider. Long distance Q-resolution with dependency schemes. In *International Conference on Theory and Applications of Satisfiability Testing (SAT)*, pages 500–518, 2016.

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

[83] Tomás Peitl, Friedrich Slivovsky, and Stefan Szeider. Long-distance Q-resolution with dependency schemes. *J. Autom. Reasoning*, 63(1):127–155, 2019.

[84] Tomás Peitl, Friedrich Slivovsky, and Stefan Szeider. Proof complexity of fragments of long-distance Q-resolution. In Mikolás Janota and Inês Lynce, editors, *Theory and Applications of Satisfiability Testing - SAT 2019 - 22nd International Conference, SAT, Proceedings*, volume 11628 of *Lecture Notes in Computer Science*, pages 319–335. Springer, 2019.

[85] P.A. Piccione. *In Search of the Meaning of Senet*. Archaeological Institute of America, 1980.

[86] Knot Pipatsrisawat and Adnan Darwiche. On the power of clause-learning SAT solvers as resolution engines. *Artif. Intell.*, 175(2):512–525, 2011.

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

[88] Pavel Pudlák and Russell Impagliazzo. A lower bound for DLL algorithms for SAT. In *Proc. 11th Symposium on Discrete Algorithms (SODA)*, pages 128–136, 2000.

[89] Robert A. Reckhow. *On the lengths of proofs in the propositional calculus.* PhD thesis, University of Toronto, 1976.

[90] John Alan Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12:23–41, 1965.

[91] Marko Samer. Variable dependencies of quantified csps. In Iliano Cervesato, Helmut Veith, and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning, 15th International Conference, LPAR 2008, Doha, Qatar, November 22-27, 2008. Proceedings*, volume 5330 of *Lecture Notes in Computer Science*, pages 512–527. Springer, 2008.

[92] Marko Samer and Stefan Szeider. Backdoor sets of quantified Boolean formulas. *Journal of Automated Reasoning*, 42(1):77–97, 2009.

[93] Horst Samulowitz, Jessica Davies, and Fahiem Bacchus. Preprocessing QBF. In Frédéric Benhamou, editor, *Principles and Practice of Constraint Programming - CP 2006, 12th International Conference, CP 2006, Nantes, France, September 25-29, 2006, Proceedings*, volume 4204 of *Lecture Notes in Computer Science*, pages 514–529. Springer, 2006.

[94] Ankit Shukla, Armin Biere, Luca Pulina, and Martina Seidl. A survey on applications of quantified Boolean formulas. In *Proc. IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 78–84, 2019.

[95] Friedrich Slivovsky. Quantified CDCL with universal resolution. In Kuldeep S. Meel and Ofer Strichman, editors, *25th International Conference on Theory and Applications of Satisfiability Testing (SAT)*, volume 236 of *LIPIcs*, pages 24:1–24:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[96] Friedrich Slivovsky and Stefan Szeider. Computing resolution-path dependencies in linear time ,. In Alessandro Cimatti and Roberto Sebastiani, editors, *Theory and Applications of Satisfiability Testing - SAT 2012 - 15th International Conference, Trento, Italy, June 17-20, 2012. Proceedings*, volume 7317 of *Lecture Notes in Computer Science*, pages 58–71. Springer, 2012.

[97] Friedrich Slivovsky and Stefan Szeider. Soundness of Q-resolution with dependency schemes. *Theoretical Computer Science*, 612:83–101, 2016.

[98] L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time. *Proc. 5th ACM Symposium on Theory of Computing*, pages 1–9, 1973.

[99] Allen Van Gelder. Contributions to the theory of practical quantified Boolean formula solving. In *Proc. Principles and Practice of Constraint Programming (CP)*, pages 647–663, 2012.

[100] Marc Vinyals. Hard examples for common variable decision heuristics. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

[101] Ralf Wimmer, Sven Reimer, Paolo Marin, and Bernd Becker. Hqspre - an effective preprocessor for QBF and DQBF. In Axel Legay and Tiziana Margaria, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 23rd International Conference, TACAS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings, Part I*, volume 10205 of *Lecture Notes in Computer Science*, pages 373–390, 2017.

[102] E. Zermelo. *Über eine Anwendung der Mengenlehre auf die Theorie des Schachspiels.* Proceedings of the Fifth Congress of Mathematics, Vol. II, Cambridge 1913.

[103] Lintao Zhang, Conor F. Madigan, Matthew W. Moskewicz, and Sharad Malik. Efficient conflict driven learning in Boolean satisfiability solver. In *Proc. IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 279–285, 2001.

[104] Lintao Zhang and Sharad Malik. Conflict driven learning in a quantified Boolean satisfiability solver. In *Proc. IEEE/ACM International Conference on Computer-aided Design (ICCAD)*, pages 442–449, 2002.