Quantified CDCL and Dependency Schemes: A proof-theoretic study

By

Abhimanyu Choudhury

MATH10201805002

The Institute of Mathematical Sciences, Chennai

A thesis submitted to the

Board of Studies in Mathematical Sciences

In partial fulfillment of requirements

for the Degree of

DOCTOR OF PHILOSOPHY

of

HOMI BHABHA NATIONAL INSTITUTE

January 2026

Homi Bhabha National Institute

Recommendations of the Viva Voce Committee

As members of the Viva Voce Committee, we certify that we have read the dissertation prepared by Abhimanyu Choudhury entitled "Quantified CDCL and Dependency Schemes: A proof-theoretic study" and recommend that it may be accepted as fulfilling the thesis requirement for the award of Degree of Doctor of Philosophy.

Chairman - Dr. Saket Saurabh — Date: 29-1-26

Guide/Convenor - Dr. Meena Mahajan — Date: 29/1/26

Examiner - Dr. Supratik Chakraborty — Date: 29 Jan 2026

Member 1 - Dr. Vikram Sharma — Date: 29/01/26

Member 2 - Dr. Prakash Saivasan — Date: 29/1/26

Member 3 - Dr. B. Srivathsan — Date: 29/1/26

Final approval and acceptance of this thesis is contingent upon the candidate's submission of the final copies of the thesis to HBNI.

I hereby certify that I have read this thesis prepared under my direction and recommend that it may be accepted as fulfilling the thesis requirement.

Date: 29 January 2026

Place: Chennai

Guide

STATEMENT BY AUTHOR

This dissertation has been submitted in partial fulfillment of requirements for an advanced degree at Homi Bhabha National Institute (HBNI) and is deposited in the Library to be made available to borrowers under rules of the HBNI.

Brief quotations from this dissertation are allowable without special permission, provided that accurate acknowledgement of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the Competent Authority of HBNI when in his or her judgement the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

Abhimanyu Choudhury

DECLARATION

I hereby declare that the investigation presented in the thesis has been carried out by me. The work is original and has not been submitted earlier as a whole or in part for a degree / diploma at this or any other Institution / University.

Abhimanyu Choudhury

LIST OF PUBLICATIONS ARISING FROM THE THESIS

Journal

- **Published**
  1. Choudhury, A. and Mahajan, M., 2024. Dependency schemes in CDCL-based QBF solving: a proof-theoretic study. Journal of Automated Reasoning, 68(3), p.16:1-24.
- **Submitted**
  1. Choudhury, A. and Mahajan, M., On the Interplay of Cube Learning and Dependency Schemes in QCDCL Proof Systems. (to the Journal of Automated Reasoning)

Conference

- **Published**
  1. Choudhury, A. and Mahajan, M., 2023. Dependency Schemes in CDCL-Based QBF Solving: A Proof-Theoretic Study. In 43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023) (pp. 38:1-18). Schloss Dagstuhl–Leibniz-Zentrum für Informatik.
  2. Choudhury, A. and Mahajan, M., 2025. On the Interplay of Cube Learning and Dependency Schemes in QCDCL Proof Systems. In 45th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2025) (pp. 25:1-19). Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

List of presentations at conferences, workshops and research visits

1. *"Dependency Schemes in CDCL-Based QBF Solving: A Proof-Theoretic Study"*. IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).

2. *"Dependency Schemes in QCDCL"*. Proof Complexity and Beyond at Mathematisches Forschungsinstitut Oberwolfach (MFO) , Germany (2024).

3. *"Proof Systems for QCDCL with Dependency Schemes"*. International Workshop on Quantified Boolean Formulas and Beyond (Co-located with SAT 2024).

4. *"Quantified CDCL Solving for QBFs and Dependency Schemes"*.Research Visit at TCS Research, Pune (2024).

5. *"On the Interplay of Cube Learning and Dependency Schemes in QCDCL Proof Systems"*. IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2025).

Thesis dedicated to - friends and family

ACKNOWLEDGEMENTS

First of all, I want to thank my advisor, Meena Mahajan, without whom I would have been completely lost. I had no idea what I was doing when I started, and she patiently (and miraculously) helped me find some direction. Thank you for tolerating my mistakes, delays, and general academic chaos and for always pushing me to aim higher.

To my friends at IMSc thank you for infusing my time here with joy, unpredictability, and countless caffeine-fueled conversations. You made the long nights and hard days easier. A special shoutout to my football seniors, whose warmth and camaraderie helped me find my footing, both on the field and off it. You made this place feel like home.

To my batchmates and juniors a thank you for keeping the spirit of sport alive. The football field was more than just a patch of grass; it was my escape, my therapy, and my reset button. Your presence and passion made it all the more special.

To Koyena, this journey would have been infinitely harder, and far less meaningful, without you. Thank you for your strength, patience, and your unshakeable belief in me even in the moments I couldn't believe in myself. You've witnessed both my chaos and calm, my hope and despair, and stood by me through it all. And to Nebbi and Snowy for your comforting presence, the joy and the new experiences you brought into my life.

Lastly, to my family my parents, my brother, my kaku, my mashi-mesho, and all my cousins a big thank you for your love, your encouragement, and your faith in me. Even from afar, you were a constant source of strength and grounding. I couldn't have done this without you in my corner.

@bhimanyu

Contents

Abstract

Quantified Conflict Driven Clause Learning (QCDCL) is one of the main approaches to solving Quantified Boolean Formulas (QBF). Cube-learning is employed in this approach to ensure that true formulas can be verified. Dependency Schemes help to detect spurious dependencies that are implied by the variable ordering in the quantifier prefix of QBFs but are not essential for constructing (counter)models. This detection can provably shorten refutations in specific proof systems, and is expected to speed up runs of QBF solvers.

The simplest underlying proof system $\mathtt{QCDCL}$ [BB23a], formalises the reasoning in the QCDCL approach on false formulas, when neither cube-learning nor dependency schemes is used. The work of [BPB24] further incorporates cube-learning. This thesis is the first work that incorporates the dependency scheme heuristic in the $\mathtt{QCDCL}$ proof system.

The usage of dependency schemes in $\mathtt{QCDCL}$ proof system with and without cube-learning are formalised and these new family of systems, the $D_1 + \mathtt{QCDCL}^{\mathtt{ORD}}(\mathtt{ClausePol}, \mathtt{CubePol})$ proof systems, which incorporates dependency schemes into the proof system, and show it to be sound and complete. When the decisions are restricted to follow level order, but dependency schemes are used in propagation and learning, in conjunction with cube-learning, the resulting proof systems using the dependency schemes $\mathtt{D}^{\mathtt{std}}$ and $\mathtt{D}^{\mathtt{rrs}}$ are investigated in detail and their relative strengths are analysed.

List of Figures

List of Tables

Chapter 1

Introduction

Computational complexity is a foundational area of theoretical computer science that studies the inherent difficulty of computational problems by classifying them according to the resources required to solve them. These resources typically include time and space, measured with respect to input size, and are analyzed using formal models of computation. The primary goal of computational complexity theory is to determine the fundamental limits of efficient computation and to understand the boundary between tractable and intractable problems.

Central to this field is the classification of problems into complexity classes. A particularly important distinction is between the classes P and NP. The class P consists of problems that can be solved in polynomial time, while NP contains problems whose solutions can be verified in polynomial time. The famous P vs. NP question—whether every problem with efficiently verifiable solutions also has an efficient algorithm for finding those solutions—remains one of the most profound open problems in computer science. Its resolution would have significant implications across cryptography, optimization, artificial intelligence, and beyond. Broader complexity classes such as PSPACE (problems solvable using polynomial space) and EXPTIME (problems solvable in exponential time) help categorize harder problems based on

their computational resource requirements.

Another crucial concept in complexity theory is that of reductions and completeness. A problem is NP-complete if it is among the hardest problems in NP, meaning that an efficient algorithm for solving any NP-complete problem would imply efficient solutions for all problems in NP. Similarly a PSPACE-complete problem would be among the hardest problems in the PSPACE class.

Beyond its theoretical importance, computational complexity has deep practical relevance. For example, modern cryptographic protocols rely on the assumption that certain problems—such as integer factorization and discrete logarithms—are computationally intractable. Likewise, understanding complexity informs the design of heuristics and approximation algorithms for hard optimization problems that arise in logistics, scheduling, and machine learning.

The study of computational complexity not only deepens our understanding of the nature of computation but also guides the development of efficient algorithms and problem-solving strategies. As computational paradigms evolve, including the emergence of quantum computing, a robust understanding of complexity remains crucial for both theoretical advancement and practical application in computer science.

1.1 Propositional Proof Complexity

Propositional proof complexity studies the efficiency of proof systems for propositional logic. Specifically, it seeks to understand the resources, such as proof length and size, required to certify that a given propositional formula is unsatisfiable. Different formal proof systems are analyzed based on their ability to produce short and easily verifiable proofs. A central goal of propositional proof complexity is to establish lower bounds on proof sizes, which in turn relate to fundamental questions about the limits of efficient reasoning and automated theorem proving.

A proof system for propositional logic is a mechanism that verifies the correctness of logical statements. Formally, a propositional proof system can be defined as a polynomial-time computable function f such that for every unsatisfiable formula Φ, there exists a proof π such that $f(\pi) = \Phi$. The length of π determines the efficiency of the proof system [CR79].

One of the simplest and most widely studied propositional proof systems is *resolution*, which is the basis for modern SAT solvers. The resolution system has one single rule which says that from 2 clauses (disjunctions of literals) $A \vee \ell$ and $B \vee \bar{\ell}$, the clause $A \vee B$ can be derived. Using this rule, if one can derive the empty clause ($\square$), then the sequence of derived clauses from the given input formula is a certificate of unsatisfiability of the initial formula. Resolution is sound and complete for proving the unsatisfiability of propositional formulas in conjunctive normal form (CNF). However, it can require exponentially long proofs in the worst case [Hak85].

Other important propositional proof systems include:

- Frege systems: General propositional proof systems that allow derivations using a fixed set of axioms and inference rules.
- Cutting-plane methods: Used in integer programming, and involve reasoning with linear inequalities [CCT87].
- Polynomial calculus: Extends resolution by incorporating algebraic methods to manipulate polynomials representing Boolean functions [CEI96].
- Extended Frege systems: An extension of Frege systems that allows the introduction of abbreviations for formulas, making proofs more concise [Kra95].

SAT solving has many real-life applications, making the development of SAT solvers a field of great interest. The Davis-Putnam-Logemann-Loveland (DPLL) algorithm [DLL62; DP60] was one of the earliest systematic approaches to solving SAT in-

stances. It employs backtracking search with unit propagation and pure literal elimination to systematically explore the search space. Modern SAT solvers, particularly those based on the Conflict-Driven Clause Learning (CDCL) paradigm [SS99], extend DPLL by learning from conflicts and dynamically refining the search process. The Handbook of Satisfiability [BHM09; Bie+21] contains a comprehensive survey of the state-of-the-art.

The quest to further enhance SAT solvers has naturally led to a deeper investigation of the reasoning they employ. This is explained by formal proof systems underlying their operation. Notably, CDCL solvers can be interpreted as generating resolution proofs [SS99; BKS04]. This connection bridges algorithmic SAT solving and proof complexity: lower bounds in the resolution proof system directly translate to limitations in CDCL-based SAT solvers. As a result, proof complexity has become a powerful theoretical tool for analyzing and understanding the strengths and weaknesses of modern SAT solvers, guiding both their development and evaluation.

1.2 Quantified Boolean Formulas

Quantified Boolean Formulas (QBFs) are an extension of propositional logic that introduces quantification over Boolean variables. Propositional satisfiability (SAT) deals with formulas in which all variables are existentially quantified. However, QBFs allow both existential ($\exists$) and universal ($\forall$) quantification over variables. This allows QBFs to express certain complex problems in a manner more succintly than propositional formulas, and thus makes it a powerful framework for modeling complex decision problems in various domains (the handbook [BHM09; Bie+21] gives a comprehensive overview about the state-of-the-art).

Just as the satisfiability of propositional formulas can be analyzed using propositional proof systems like resolution, QBFs require specialized proof systems that

account for both types of quantifiers. Some key QBF proof systems include:

- Q-Resolution (`Q-Res`): An extension of propositional resolution that introduces rules for handling universal quantifiers by allowing the reduction of trailing universal variables from clauses, (with the new rule being called ∀-`red`) [KKF95].
- Expansion-based systems: These transform QBFs into propositional formulas by expanding universal quantifiers into explicit cases, leading to a larger formula with only existential quantifiers, that is a SAT instance [JM15].
- Long Distance Q-Resolution (`LDQ-Res`): An extension of `Q-Res`, that allows tautologies of universal variables to be formed in a resolution step if the universal variable is to the right of the resolution pivot in the quantifier prefix [ZM02; BJ12].

The systems listed above are all refutational proof systems, i.e. they are systems used to prove a given QBF to be false. For true QBFs, there exists proof systems like `Q-TermRes` (the dual of `Q-Res`) and `LDQ-TermRes` which can be used to verify a formula to be true. These systems perform resolution on terms/cubes (conjunction of literals) over a universal pivot, and the reduction rule reduces trailing existential variables.

QBF solvers are tools designed to determine the satisfiability of QBF instances automatically. Unlike SAT solvers, which deal with only existentially quantified variables, QBF solvers must consider alternating quantifiers, making the problem computationally more challenging. Several approaches have been developed for QBF solving, among which one is the extension of the conflict-driven clause learning (CDCL) approach that integrates universal reduction rules for handling quantifiers. This approach is called QCDCL and is the algorithm employed by state-of-the-art QBF solvers like DepQBF and Qute [LE17; PSS19a].

The QCDCL proof system

The QCDCL proof system was introduced by [BB23a] as a method to formalise the reasoning in the QCDCL based QBF solving algorithms. A refutation of a false QBF in the QCDCL proof system is a sequence of triples of the form (T, C, π) where T is a trail (in the QCDCL algorithm) ending in a conflict, C is the clause learnt from this trail, and π is the LDQ-Res derivation of C explaining how C is learnt. From the last triple in the sequence we can learn the empty clause, completing the refutation. Three factors affect the construction of the refutation.

- Decision policy: determines the order and choice of variable assignments during the search. The standard policy is the left-to-right quantifier order (LEV-ORD).
- Propagation policy: governs how unit propagation is carried out. Usually the RED policy is used, which says that an existential variable is propagated from a unit clause, and a clause C is said to be unit if applying $\forall$-red on C restricted by the trail so far yields a single existential literal.
- Learning Policy: specifies which clauses can be learned during conflict analysis. This is often constrained by the use of an underlying resolution system such as LDQ-Res.

An important aspect of QCDCL algorithms is that they are designed not only for refuting false QBFs but also for establishing the truth of satisfiable ones. To this end, QCDCL incorporates both clause learning (used in refutations) and cube learning—the process of learning terms that represent satisfying assignments. Cube learning is essential for ensuring that QCDCL algorithms are complete for all QBFs, regardless of their truth value. The concept of cube learning within QCDCL was formally integrated into the QCDCL proof system in [BPB24].

By formalizing QCDCL as a proof system, one can analyze and compare its strength against other resolution-based systems, reason about the role of learning schemes, and study how different configurations (e.g., with or without cube learning) affect proof size and solver performance.

Dependency Schemes

In QBFs, the order of variables in the quantifier prefix is more than just a syntactic detail—it has a major impact on what the formula actually means. Changing the order of quantifiers can change which variables depend on which, and that can completely alter whether the formula is true or false. Because of this, QBF proof systems need to be careful to follow the structure of the prefix when reasoning about the formula. But not all dependencies implied by the variable order are truly necessary—some are just there because of how the formula is written, not because they actually matter for evaluating its truth.

Dependency schemes help identify and remove these unnecessary, or spurious, dependencies. They give a more refined way of understanding which variables genuinely depend on which others. When used in a proof system, a dependency scheme can loosen some of the rigid restrictions imposed by the original variable order, making reasoning more flexible and often more efficient, without affecting correctness. This makes dependency schemes a valuable tool in QBF solving, both in theory and in practice.

1.3 Our Contributions

While there have been studies on various heuristics within the `QCDCL` proof system—such as cube learning and pure literal elimination as discussed in [BPB24]—the

impact of incorporating the dependency scheme heuristic into the QCDCL proof system has not been previously explored. This thesis represents the first attempt to systematically investigate the effect of dependency scheme heuristics within the QCDCL framework. Dependency schemes are a particularly compelling object of study because, in resolution-based QBF proof systems, they consistently provide an additional strength: although their benefits may vary depending on the instance, they are never detrimental.

QCDCL systems with dependency – definition, soundness & completeness

To analyze the effect of dependency schemes on QCDCL systems, we identify three key components that may be influenced by their inclusion: the decision policy, unit propagation and learnable clauses, and whether or not cube-learning is used. We examine how these aspects get modified under the addition of dependency schemes.

- Decision Policy: for LEV-ORD decisions, dependency schemes make no effect. We define a decision policy based on dependency schemes, called D-ORD, which states that at a point in a trail, a decision can be made on variable x if all y such that on which x depends according to D have already been assigned in the trail.
- Unit Propagation and Learning: the definition of unit clause changes to clauses being called unit if they become unit under dependency-scheme aware reduction rather than standard reduction. Similarly in learning, dependency scheme aware reduction is used when trying to learn from trails.
- Cube Learning: If cube learning is a part of the system, we show how propagation and learning gets modified if dependency schemes are involved.

To account for all these possibilities, we introduce a new notation for QCDCL proof systems such that they can accommodate dependency schemes as well:

The system $\texttt{QCDCL}^{\texttt{ORD}}(\texttt{ClausePol}, \texttt{CubePol})$ is the QCDCL proof system where

1. ORD denotes the decision policy; e.g. LEV-ORD, D′-ORD, ANY-ORD.
   In D′-ORD, D′ denotes the dependency scheme, such as $\texttt{D}^{\texttt{rrs}}$-ORD, $\texttt{D}^{\texttt{std}}$-ORD.

2. ClausePol is the dependency scheme D used in reduction, propagation, and learning for clauses. Note that this scheme need not be the same as the D′ in D′-ORD.

3. CubePol $\in \{$No-Cube, Cube-LD, Cube-D$\}$ denotes the type of usage of cube learning;

   (a) No-Cube: No cube learning.

   (b) Cube-LD: Cube Learning used, but no dependency (only $\texttt{D}^{\texttt{trv}}$) in propagation, and cube learning using LDQ-TermRes.

   (c) Cube-D: Cube Learning used, dependency scheme D used in propagation, and cube learning is done using Q(D)-TermRes.

An additional, orthogonal way in which dependency schemes may influence a QCDCL system is through *preprocessing* the input formula. This preprocessing step is entirely independent of any dependency scheme used in decision, propagation, or learning within the system itself.

When prepocessing is involved by using a dependency scheme D″, we denote the system as $\texttt{D}'' + \texttt{QCDCL}^{\texttt{ORD}}(\texttt{ClausePol}, \texttt{CubePol})$.

The above definitions allow us to define a huge family of dependency-scheme-aware QCDCL proof systems. We show that for the class of *normal* dependency schemes [PSS19b], the $\texttt{D}'' + \texttt{QCDCL}^{\texttt{D}'\text{-}\texttt{ORD}}(\texttt{D}, \texttt{CubePol})$ proof systems are sound and complete.

Having established that these systems are sound and complete, we study these systems in more detail to analyse their strengths and weaknesses.

QCDCL systems with dependency – relative strengths of systems

Our study focuses on LEV-ORD systems, as they represent the most fundamental decision policy within the QCDCL proof system, as identified in [BB23a]. Among the various dependency schemes, our primary interest lies in $\mathtt{D}^{\mathtt{rrs}}$ and $\mathtt{D}^{\mathtt{std}}$ [SS16]. The $\mathtt{D}^{\mathtt{rrs}}$ scheme is notable for being one of the most widely studied and was the first dependency scheme shown to provide a strict advantage over $\mathtt{D}^{\mathtt{trv}}$ within the Q-Res proof system. On the other hand, $\mathtt{D}^{\mathtt{std}}$ is of particular relevance due to its practical implementation in the state-of-the-art QBF solver DepQBF.

For this purpose of our study, we analyze several well-known formulas from the existing QBF literature and establish bounds for them within the proposed dependency-scheme-aware QCDCL proof systems. In addition, we carefully construct a set of new formulas specifically designed to highlight the strengths and limitations of this family of systems, and we rigorously prove bounds for these systems on the newly introduced formulas.

We formally establish a number of results regarding the relative strength and incomparability of these QCDCL proof systems with and without cube learning. A key takeaway is that incorporating dependency schemes into the decision policy consistently yields an advantage over the baseline LEV-ORD policy. However, when restricting attention to LEV-ORD systems, the addition of $\mathtt{D}^{\mathtt{rrs}}$ may strengthen or even weaken the system in some cases. Furthermore, the different ways of incorporating $\mathtt{D}^{\mathtt{rrs}}$ are incomparable in proof strength to each other.

The new ideas and results presented in this thesis appear in [CM24] and [CM25].

Organization of thesis

The rest of this thesis is structured as follows: Chapter 2 presents various QBF proof systems, describes the concept of dependency schemes, and discusses their

integration into certain resolution-based QBF systems that have been studied in existing literature. Chapter 3 explores the novel idea of incorporating dependency schemes into the QCDCL proof system, offering formal definitions and establishing soundness and completeness results. Chapter 4 revisits well-known formulas from the QBF literature, introduces several newly constructed formulas, and establishes proof complexity bounds for these formulas within QCDCL systems enhanced with dependency schemes. Chapter 5 analyzes the relative strengths of these new families of dependency-scheme-parameterized QCDCL proof systems using the bounds from Chapter 4. Finally, Chapter 6 provides a conclusion.

Chapter 2

QBF Proof Systems and Dependency Schemes

This chapter introduces the notations used throughout the thesis, following which we first present existing resolution based QBF proof systems and the `QCDCL` proof system. Then we discuss dependency schemes, and how they can be added to resolution based systems. All of the defintions, theorems, and results in this chapter have been established in existing literatue.

2.1 Basic Notation

A literal ℓ is a Boolean variable x or its negation $\bar{x}$, and $\texttt{var}(\ell)$ denotes the associated variable x. A clause is a disjunction of literals; a term or a cube is a conjunction of literals. With some abuse of notation we sometimes view clauses and cubes as the set of literals they contain; the interpretation assumed will be clear from context. For a clause or cube C, $\texttt{var}(C)$ denotes the set $\{\texttt{var}(\ell) \mid \ell \in C\}$. A propositional formula φ is built from variables using conjunction, disjunction, and negation; it is in conjunctive normal form (CNF) if it is a conjunction of clauses. For a formula

φ, the set of variables in it is denoted by $\texttt{var}(\varphi)$. For a variable x in $\texttt{var}(\varphi)$, and a Boolean value a, $\varphi|_{x=a}$ refers to the formula obtained by substituting $x = a$ everywhere in φ. For a set S of clauses and a literal ℓ, we use shorthand $\ell \vee S$ to denote the set of clauses $\{\ell \vee C \mid C \in S\}$. The empty clause is denoted $\square$ and is unsatisfiable; the empty cube is denoted $\top$ and is always true. A clause (cube) is said to be tautological (contradictory) if for some variable x it contains both x and $\bar{x}$.

For clauses $A' = A \vee x$ and $B' = B \vee \bar{x}$, their resolvent is the clause $A \vee B$ denoted as $\texttt{res}(A', B', x)$ or $\texttt{res}(B', A', x)$. For cubes $A' = A \wedge x$ and $B' = B \wedge \bar{x}$, their resolvent is the cube $A \wedge B$, also denoted as $\texttt{res}(A', B', x)$ or $\texttt{res}(B', A', x)$.

Let $G = G(V, E)$ denote an undirected graph with vertex set V and edge set E, where $E \subseteq \binom{V}{2}$. For a set $V' \subseteq V$ the subgraph induced in G by V' is denoted $G[V']$; it has vertex set V' and edge set E', where $E' = \{(x, y) \in E \mid x, y \in V'\}$.

Quantified Boolean Formulas

A Quantified Boolean Formula (QBF) in prenex conjunction normal form (PCNF) is a prefix with a list of variables, each quantified either existentially or universally, and a matrix, which is a set (conjunction) of clauses over these variables. That is, it has the form

$$\Phi = \mathcal{Q}\vec{x} \cdot \varphi = Q_1 x_1 Q_2 x_2 \dots Q_n x_n \;\; \varphi(x_1, x_2, \dots, x_n)$$

where φ is a propositional formula in CNF, and each Q_i is in $\{\exists, \forall\}$. We denote by $\texttt{var}_\exists(\varphi)$ or $X_\exists$ ($\texttt{var}_\forall(\varphi)$ or $X_\forall$ respectively) the set of all variables quantified existentially (resp. universally). For variables x and y, $(x, y) \in R_\Phi$ if y is to the right of x in the quantifier prefix, and $R_\Phi(x)$ denotes the set of all variables to the right of x in the quantifier prefix.

If we unite adjacent quantifiers of the same type, we can express the quantifier prefix

in the form

$$\mathcal{Q} = Q'_1 X_1 Q'_2 X_2 \dots Q'_m X_m$$

where $X'_i s$ are non-empty sets of variables, called blocks, such that $\cup_{i=1}^{m} X_i = \{x_1, x_2, \cdots, x_n\}$ and $Q'_i \neq Q'_{i+1}$ for $i \in [m-1]$. Using this, we can define the notion of the quantifier level of a variable x in the formula by $\texttt{lev}(x) = i$ if $x \in X_i$. The level of two variables can be compared by saying for $\texttt{lev}(x) = i$ and $\texttt{lev}(y) = j$, if $i < j$, then $\texttt{lev}(x) < \texttt{lev}(y)$.

Semantically the truth or falsity of a QBF is defined inductively on the number of variables (n) in the QBF:
For $n = 0$, If φ has no clauses i.e. $\varphi = \emptyset$, then Φ is true, and if $\varphi = \{\Box\}$, the set containing the empty clause, then Φ is false.
For $n \geq 1$, let $\Phi = \mathcal{Q}_1 x \mathcal{Q}\vec{y}\ \ \varphi(x, \vec{y})$, where $\vec{y}$ has $n-1$ variables.
If $\mathcal{Q}_1 = \exists$, then Φ is true if at least one of $\mathcal{Q}\vec{y}\ \ \varphi(0, \vec{y})$ and $\mathcal{Q}\vec{y}\ \ \varphi(1, \vec{y})$ is true.
If $\mathcal{Q}_1 = \forall$, then Φ is true if both of $\mathcal{Q}\vec{y}\ \ \varphi(0, \vec{y})$ and $\mathcal{Q}\vec{y}\ \ \varphi(1, \vec{y})$ are true.

Alternatively we say QBF is true if for each existentially quantified variable x_i, there exists a (Skolem) function s_i, depending only on universally quantified variables x_j with $j < i$, such that substituting these s_i in φ yields a tautology. The set of all s_i's is said to form a model for the given QBF. Similarly, the formula is false if for each universally quantified variable u_i, there is a (Herbrand) function h_i, depending only on existentially quantified variables x_j with $j < i$, such that substituting h_i in φ yields an unsatisfiable formula.The set of all h_i's is said to form a countermodel for the given QBF.

Semantics of QBFs via Two-Player Games

A useful way to understand the semantics of QBFs is through a two-player evaluation game, where two players — the existential player ($\exists$) and the universal player ($\forall$) —

take turns assigning values to their respective variables. The goal of the existential player is to satisfy the inner propositional formula, while the universal player aims to falsify it.

The game proceeds as follows:

- The players take turns selecting values for their assigned variables in the order dictated by the quantifier prefix.
- Once all variables have been assigned, the resulting propositional formula is evaluated.
- If the final propositional formula evaluates to true, the existential player wins; otherwise, the universal player wins.
- A QBF is true if and only if the existential player has a winning strategy, meaning they can always make choices that lead to a satisfying assignment regardless of the universal player's moves. Otherwise, the formula is false.

For example, consider the QBF:

$$\forall x \exists y \ (x \vee y) \wedge (\bar{x} \vee \bar{y})$$

- If $x = 1$, the existential player can set $y = 0$.
- If $x = 0$, the existential player can set $y = 1$.

Since the existential player has a winning strategy, this QBF is true.

On the other hand if the order of the variables in the prefix is exchanged, then the resulting formula $\exists y \forall x \ (x \vee y) \wedge (\bar{x} \vee \bar{y})$ is false because the universal player can win the game by setting $x = y$.

2.2 Introduction to QBF Proof Systems

QBF proof systems are logical frameworks designed to reason about the truth of QBFs. Unlike propositional proof systems, which focus on the satisfiability of existentially quantified Boolean formulas, QBF proof systems must handle the alternation of existential and universal quantifiers, making them significantly more complex. These systems aim to formally verify whether a given QBF is true (i.e., has a winning strategy for the existential player) or false (i.e., the universal player has a winning strategy). There are various types proof systems for QBFs some resolution-based, some expansion-based. In this chapter we will look at a few resolution-based systems that exist in literature.

Formally, a proof system is described by a set of rules which are used to construct a verification if the formula is true and a refutation if the formula is false. A proof system is said to be *sound* if whenever the system has a proof of a given formula to be true (false), the formula must indeed be true (false). A proof system is said to be *complete* if given a true (false) formula the system can always provide a proof showing the formula to be true (false).

Given two proof systems $\mathtt{P}_1$ and $\mathtt{P}_2$ we say that $\mathtt{P}_1$ simulates $\mathtt{P}_2$ if any proof in the latter can be translated to a proof in the former with an atmost polynomial blow-up in size. Further, if this translation can be computed in polynomial time then we say $\mathtt{P}_1$ p-simulates $\mathtt{P}_2$.

2.3 Resolution based QBF proof systems

Resolution is one of the very well studied propositional proof systems, so a natural way to create QBF proof systems was to try and extend the resolution logic to handle QBFs. Here we will talk about three such QBF proof systems. These proof

systems are all refutational in nature, i.e. they are used to refute QBFs, that is prove them to be false.

The Q-Res, QU-Res and LDQ-Res proof systems

The first QBF proof system to be formalised was known as Q-Res (Q-Resolution). Being the first as well as one of the simplest, it is one of the most well studied systems. It was first introduced and defined as a proof system by Kleine Büning et al [KKF95] . It works on QBFs in PCNF. A derivation in the Q-Res proof system is made up of lines, where each line is a clause. A Q-Res derivation from clauses of the given formula (we refer to these clauses as axioms) derives clauses by using the rules of the system described below. A derivation is called a refutation if it derives the empty clause.

1. Axiom Rule:	$\dfrac{}{C}$	A clause C from the initial set of axioms ϕ
2. Resolution Rule:	$\dfrac{A \vee x \quad \bar{x} \vee B}{A \vee B}$	Where x is an existential variable, and $A \vee B$ is non-tautological
3. Reduction Rule: (∀-red)	$\dfrac{A \vee \ell}{A}$	Where var(ℓ) is a universal variable u, and u is to the right of all existential variables in A

Figure 2.1: Rules of Q-Res proof system

Trying to further modify the rules of Q-Res to create "better" systems for QBFs gave rise to two new systems, the first being QU-Res. The system QU-Res modifies the resolution rule of Q-Res slightly by allowing the resolution pivot variable to also be a universal variable. The remaining rules remain the same. Figure 2.2 presents the complete rules of the QU-Res system [Gel12].

The other extension of Q-Res was made to create the LDQ-Res (long distance Q-Resolution) proof system. It was first presented by Zhang and Malik in [ZM02] and

1. Axiom Rule:	$\dfrac{}{C}$	A clause C from the initial set of axioms ϕ
2. Resolution Rule:	$\dfrac{A \vee x \quad \bar{x} \vee B}{A \vee B}$	Where x is an existential or universal variable, and $A \vee B$ is non-tautological
3. Reduction Rule: (∀-red)	$\dfrac{A \vee \ell}{A}$	Where $\mathtt{var}(\ell)$ is a universal variable u, and u is to the right of all existential variables in A

Figure 2.2: Rules of QU-Res proof system

formalized as a proof system for QBFs by Balabanov and Jiang [BJ12]. The LDQ-Res system also differs from Q-Res with respect to the resolution rule, by introducing the concept of *long-distance* resolution, which allows the formation of tautological clauses under certain conditions. Long-distance resolution says that resolution can be performed on two clauses if they do not contain any existential variables (except the pivot) in opposite polarities, and if they contain a universal variable in opposite polarities, this variable must be to the right of the pivot variable in the quantifier prefix. The rules of the proof system are described below:

1. Axiom Rule:	$\dfrac{}{C}$	A clause C from the initial set of axioms ϕ
2. Long-Distance Resolution Rule:	$\dfrac{A \vee x \quad \bar{x} \vee B}{A \vee B}$	Where x is an existential variable, and $A \vee B$ has no tautology on existential variables. If universal $u, \bar{u} \in A \vee B$ then $(x, u) \in R_\Phi$
3. Reduction Rule: (∀-red)	$\dfrac{A \vee \ell}{A}$	Where $\mathtt{var}(\ell)$ is a universal variable u, and u is to the right of all existential variables in A

Figure 2.3: Rules of LDQ-Res proof system

Completeness and Soundness of Q-Res, QU-Res and LDQ-Res

Now, by definition of the rules of the proof systems above we can see that anything that can be done in Q-Res is also allowed by the rules of QU-Res and LDQ-Res.

Therefore, any derivation in Q-Res is a derivation in the other two systems.

Therefore, to show completeness of the three systems it is enough to show that Q-Res is complete.

To show completeness of Q-Res, the idea is to use a proof by induction on the number of variables in the formula [KKF95]. Establishing the base case for zero variables is trivial. From the induction hypothesis that says Q-Res is complete for less than n variables, we want to show it is complete for a formula with n variables. For such a formula, we show that depending on the quantification of the outermost variable, we can construct a Q-Res refutation for it using the Q-Res refutation(s) of the inner formula instantiated by the outermost variable.

To show soundness of Q-Res, strategy extraction is used [KKF95], where it can be shown that from a given Q-Res refutation of a formula, we can extract a countermodel for the formula, hence certifying that the formula must be false. The proof proceeds by constructing a strategy to assign the leftmost universal variable in the quantifier prefix, and then iteratively working inwards. This approach is referred to as round-based strategy extraction.

Soundness of QU-Res and LDQ-Res was shown in [Gel12] and [BJ12] respectively by extending the argument for Q-Res.

2.4 Term Resolution based QBF proof systems

The previous section introduced refutational proof systems. In this section we look at two verification proof systems Q-TermRes and LDQ-TermRes which are used to prove the truth of QBFs. These are also resolution based and extensions of TermRes in propositional formulas. From a QBF point of view these systems are a kind of dual of their refutational counterparts (Q-Res and LDQ-Res) respectively.

The Q-TermRes and LDQ-TermRes proof systems

Q-TermRes is the simplest proof system for true QBFs, and was introduced in [GNT06]. A derivation in the Q-TermRes system is a sequence of terms which are derived from the clauses of the formula using the three rules of the system, and such a sequence of terms is a proof if the final term derived in the sequence is the empty term. The rules of the Q-TermRes proof system are described below:

1. Model Generation Rule:	$\dfrac{}{T}$	A cube T such that $T \cap C \neq \emptyset$ for every axiom clause $C \in \phi$
2. Term Resolution Rule:	$\dfrac{A \wedge u \quad \bar{u} \wedge B}{A \wedge B}$	Where u is a universal variable, and $A \wedge B$ is non-contradictory
3. Existential Reduction Rule: (∃-red)	$\dfrac{A \wedge \ell}{A}$	Where $\mathtt{var}(\ell) = x$ is an existential variable, and x is to the right of all universal variables in A

Figure 2.4: Rules of Q-TermRes proof system

Similar to LDQ-Res being the extension of Q-Res, LDQ-TermRes is the exact same extension of Q-TermRes. Where LDQ-Res allowed formation of tautological clauses, LDQ-TermRes allows the formation of contradictory terms. The resolution rule gets modified to allow existential literals $x, \bar{x}$ to appear the in the resolvent if x appears to the right of the universal pivot variable u in the quantifier prefix. The rules of the system are in the figure below

1. Model Generation Rule:	$\dfrac{}{T}$	A cube T such that $T \cap C \neq \emptyset$ for every axiom clause $C \in \phi$
2. Long-Distance Term Resolution Rule:	$\dfrac{A \wedge u \quad \bar{u} \wedge B}{A \wedge B}$	Where u is a universal variable, and $A \wedge B$ is non-contradictory on universal variables. If existential $x, \bar{x} \in A \wedge B$ then $(u, x) \in R_\Phi$
3. Existential Reduction Rule: (∃-red)	$\dfrac{A \wedge \ell}{A}$	Where $\mathtt{var}(\ell) = x$ is an existential variable, and x is to the right of all universal variables in A

Figure 2.5: Rules of LDQ-TermRes proof system

Completeness and Soundness of Q-TermRes and LDQ-TermRes

Analogous to the previous section we see that anything that can be done by the rules of Q-TermRes can also be done in LDQ-TermRes. Therefore any derivation in Q-TermRes is also a valid LDQ-TermRes derivation.

Therefore, showing that Q-TermRes is complete on true formulas is enough to establish the completeness of both systems. Now, the proof of completeness for Q-TermRes is the exact dual to the one for Q-Res [GNT06]. Here, again using induction on the number of variables we show that if there exists a Q-TermRes proof for a formula with less than n variables, for a formula with n variables, we can use the Q-TermRes proof(s) of the instantiated subformula(s) to create a proof of the bigger formula.

To show soundness of Q-TermRes, the idea is to prove the contrapositive that if the formula is false, then it cannot have a Q-TermRes proof. The core idea behind this is that Q-TermRes preserves countermodels [GNT06]. What this means is that if there is a false PCNF formula with a countermodel, then any term derived by Q-TermRes is is also falsifiable by the same countermodel. If there exists a Q-TermRes proof, then the empty term is derivable from the formula. The empty term is always true; it is not falsifiable by any countermodel. Therefore, if there is a Q-TermRes proof, the formula cannot have a countermodel, and hence must be true.

To see that Q-TermRes preserves countermodels, let Φ be a false PCNF formula and f a countermodel of Φ. Then let σ be any assignment to the existential variables, then $\sigma \cup f(\sigma)$ is a complete assignment falsifying Φ. It can easily be verified via induction on the length of the derivation, that the same assignment will falsify any term that can be derived from the formula using the rules of Q-TermRes. For the base case, consider any term T derived by the model generation rule from Φ. By definition of the rule, T must satisfy all axiom clauses. Since f is a countermodel for Φ, T must be necessarily falsified by $\sigma \cup f(\sigma)$. The inductive cases involving resolution and $\exists$-reduction rule can be showed in a similar straightforward manner.

Soundness of LDQ-TermRes can be shown in the similar manner by extending the argument for Q-TermRes [LE17].

2.5 The QCDCL proof system

In this section, we develop the formal framework of the QCDCL proof system, originally introduced by Beyersdorff and Böhm in [BB23a]. Rather than diving directly into the full definition, we begin by gradually building up the essential components that constitute the QCDCL methodology. These include how variable assignments are represented using trails, the role of decision policies in guiding the solver, and the mechanisms for unit propagation, learning, and deriving new clauses. By breaking down these elements and examining how they interact during a solver's execution, we gain a clearer understanding of how the QCDCL proof system models the behavior of QBF solvers.

Trails: A trail T for Φ is a sequence of literals (or $\square$) of variables from Φ with some specific properties. There are two types of literals in T : decision literals, that can be both existential and universal, and propagated literals, that are either existential or $\square$. A trail T is denoted as:

$$T := (p_{(0,1)}, \cdots, p_{(0,g_0)}; \mathbf{d_1}, p_{(1,1)}, \cdots p_{(1,g_1)}; \mathbf{d_2}, \cdots\cdots\cdots; \mathbf{d_r}, p_{(r,1)}, \cdots p_{(r,g_r)})$$

where decision literals are denoted by d_i and are in boldface, while propagated literals are $p_{(i,j)}$ where it is the j^{th} literal in the i^{th} decision level. A semicolon is put at the end of each decision level. A trail can also be viewed as a set of literals or as a partial assignment, and we use the notation $\ell \in T$ if the literal ℓ appears in the trail T. The notation $T[s,t]$ denotes the subtrail of T till the position $p_{(s,t)}$. For a trail to be valid, the decisions must be non-tautological and non-repeating, i.e. $\mathtt{var}(d_i) \neq \mathtt{var}(d_j)$ for all $i, j \in \{1, 2, \cdots, r\}$. If $\square$ appears in a trail it must be

the rightmost literal, and such a trail is said to reach a *conflict*. There are 2 aspects that control the construction of a trail:

1. The decision policy: it determines how to choose the next variable to branch on and decide. The most basic and restrictive decision policy is called `LEV-ORD`. In this policy, decisions must respect the quantifier prefix level order, i.e. at any point in the trail the decision can be made on a variable x only if there is no unassigned variable y in the trail with $\texttt{lev}(y) < \texttt{lev}(x)$. The other extreme of complete freedom in the decision policy is `ANY-ORD` where variables can be decided in any order. Other decision policies which lie in the middle of these two extremes include `UNI-ANY` and `EXI-ANY`.

2. The unit propagation policy: It determines that for a partial assignment α to some variables, when does a clause C propagate a literal? There are two policies to consider. First, the No-Reduction (`NO-RED`) policy. In this policy, a clause C is unit at a point in the trail if, after restricting it by the partial assignment generated by the trail so far, there exists exactly one existential literal $\ell \in C$. The other policy is the Reduction (`RED`) policy. In this policy, a clause C becomes unit at a point in the trail if, when $\forall$-`red` is performed on C restricted by the partial assignment of the trail so far, there exists exactly one existential literal $\ell \in C$.

   If a clause C becomes unit then its literal is propagated so as to satisfy it.

The set of learnable clauses: When a trail ends in a conflict, a clause can be learnt from it. The potential set of learnable clauses from a trail is defined as follows.

Definition 2.1 (learnable clauses, Def 3.3 [BB23a])**.** *From a trail*

$$T := (p_{(0,1)}, \cdots, p_{(0,g_0)}; \mathbf{d_1}, p_{(1,1)}, \cdots p_{(1,g_1)}; \mathbf{d_2}, \cdots\cdots\cdots; \mathbf{d_r}, p_{(r,1)}, \cdots p_{(r,g_r)})$$

ending in a conflict $p_{(r,g_r)} = \square$*, the sequence* L_T *of learnable clauses has a clause*

associated with each propagation in the trail, and one more clause, described by tracing the conflict backwards through the trail as follows. ($\texttt{ante}(\ell)$ *denotes the clause that causes literal* ℓ *to be propagated; i.e. the antecedent., and* $\texttt{RED}(C)$ *denotes the clause resulting by performing* $\forall$-$\texttt{red}$ *on a clause* C*).*

- $C_{(r,g_r)} = \texttt{red}(\texttt{ante}(p_{(r,g_r)}))$.
- *For* $i \in \{0, 1, \cdots, r\}$ *and* $j \in [g_i - 1]$,

$$C_{(i,j)} = \begin{cases} \texttt{red}[\texttt{res}(C_{(i,j+1)}, \texttt{red}(\texttt{ante}(p_{(i,j)})), p_{(i,j)})] & \text{if } \bar{p}_{(i,j)} \in C_{(i,j+1)} \\ C_{(i,j+1)} & \text{otherwise} \end{cases}$$

- *For* $i \in \{0, 1, \cdots, r-1\}$.

$$C_{(i,g_i)} = \begin{cases} \texttt{red}[\texttt{res}(C_{(i+1,1)}, \texttt{red}(\texttt{ante}(p_{(i,g_i)})), p_{(i,g_i)})] & \text{if } \bar{p}_{(i,g_i)} \in C_{(i+1,1)} \\ C_{(i+1,1)} & \text{otherwise} \end{cases}$$

Using the definition of trails, propagation policy and learnable clauses from a trail, the QCDCL proof system is defined as follows.

Definition 2.2 (QCDCL proof system, Def 3.5 [BB23a])**.** *Let a QBF* $\Phi = \mathcal{Q} \cdot \phi$ *be a PCNF formula in* n *variables. We call a triple of sequences*

$$\iota = (T_1, C_1, \pi_1), \cdots, (T_m, C_m, \pi_m)$$

a $\texttt{QCDCL}_R^P$ *derivation of* C_m *from* Φ*, where* $R \in \{\texttt{NO-RED}, \texttt{RED}\}$ *and* $P \in \{\texttt{LEV-ORD}, \texttt{ANY-ORD}, \texttt{EXI-ANY}, \texttt{UNI-ANY}\}$*, if for all* $i \in [m]$*,* T_i *is a trail following* P *and* R *from* $\phi \cup \{C_1, \cdots, C_{i-1}\}$ *, each* C_i *is a learnable clause from* T_i*, and* π_i *is the derivation of* C_i *from* $\phi \cup \{C_1, \cdots, C_{i-1}\}$*. If* $C_m = \square$ *then* ι *is a refutation of* Φ*.*

The simplest system underlying QCDCL based solvers is the $\texttt{QCDCL}_{\texttt{RED}}^{\texttt{LEV-ORD}}$ system and is referred to as the QCDCL proof system.

Soundness and Completeness of QCDCL proof system

For $R \in \{\texttt{NO-RED}, \texttt{RED}\}$ and $P \in \{\texttt{LEV-ORD}, \texttt{ANY-ORD}, \texttt{EXI-ANY}, \texttt{UNI-ANY}\}$ the $\texttt{QCDCL}_R^P$ proof system was shown to be sound and complete in [BB23a]. A gist of the idea behind the proofs is provided below.

The soundness of the QCDCL proof system is established by demonstrating that for every clause C derived within the system, the accompanied corresponding derivation π is valid in the LDQ-Res proof system. Since LDQ-Res known to be a sound system, this directly implies the soundness of QCDCL. The key to this argument lies in verifying that each individual step in the construction of the sequence of learnable clauses adheres to the rules of LDQ-Res. Consequently, every clause learned during a QCDCL derivation has an associated valid LDQ-Res derivation, thereby ensuring the overall soundness of the system.

The refutational completeness is shown by establishing the following three steps:

1. Since all decision policies are generalizations of $P = \texttt{LEV-ORD}$ if completeness can be shown for LEV-ORD it would imply completeness for all decision policies.

2. From every LEV-ORD trail hitting a conflict, we can learn a "new clause". Since there are finitely many possible clauses, for a given formula on n variables, eventually we must learn the empty clause.

3. We can always construct a LEV-ORD trail that hits a conflict. This is because, since the given formula is false, we have a winning strategy for the universal player. The trails are constructed by ensuring all decisions on the universal variables are made using this winning strategy (this can be done since decision making is in LEV-ORD). Therefore, since it is a winning strategy, at the end of the trail some clause must be falsified, causing the trail to end in a conflict.

Thus the $\texttt{QCDCL}_R^P$ proof system is a sound and complete refutational proof system.

The $\texttt{QCDCL}^{\texttt{cube}}$ proof system

An important aspect of QCDCL algorithms is that they are designed not only for refuting false QBFs but also for establishing the truth of satisfiable ones. To this end, QCDCL incorporates both clause learning (used in refutations) and cube learning—the process of learning terms that represent satisfying assignments. Cube learning is essential for ensuring that QCDCL algorithms are complete for all QBFs, regardless of their truth value. The QCDCL proof system defined in [BB23a] was a purely refutational proof system. The concept of cube learning within QCDCL was formally integrated into the QCDCL proof system in [BPB24], giving the proof system $\texttt{QCDCL}^{\texttt{cube}}$.

The $\texttt{QCDCL}^{\texttt{cube}}$ system modifies some aspects of the QCDCL proof system to account for cube-learning. A brief overview of the modifications is highlighted below.

- The decision policy of the proof system remains unchanged; it remains as it was without cube-learning.
- Unit propagation in trails now changes to also allow the propagation of universal literals from cubes. Unit clauses and their corresponding propagation of existential literals remains the same. For cubes, the notion of unit cubes is defined analogously: a cube C is called a unit cube at a point in the trail (s, t), if $\texttt{red}_\exists(C|_{T[s,t]})$, i.e. $\exists$-red of the cube C restricted to that point (s, t) of the trail, is a single universal literal. The trail propagates this literal so as to falsify the antecedent cube it arises from.
- Learning can now be done in three different ways depending on the different types of trails:
  - Learnable clauses from a conflict clause: For trails ending in a conflict of a clause, clauses are learnt in the same manner as before, the only

modification being that while constructing the learnable clause sequence by backtracking, propagated universals and their antecedents are jumped over and ignored.

- A trail can end in the satisfaction of a cube; this is considered a cube conflict. From a cube conflict a cube is learnt from the set of learnable cubes. The set of learnable cubes is created in a manner exactly dual to the clauses, by performing `TermRes` and ∃-`red` while backtracking over the propagated universals. Propagated existentials are ignore.
- A trail can also end by satisfying all clauses without satisfying any cube. From such a trail a cube can be learnt. The learnt cube is the ∃-`red` of the cube corresponding to the trail (i.e. the conjunction of all literals in the trail).

In $\texttt{QCDCL}^{\texttt{cube}}$ the notion of refutation stays the same as `QCDCL`, i.e. the derivation of the empty clause. A $\texttt{QCDCL}^{\texttt{cube}}$ verification of a formula is the derivation of the empty term.

The system $\texttt{QCDCL}^{\texttt{cube}}$ is refutationally complete, because every `QCDCL` refutation is a $\texttt{QCDCL}^{\texttt{cube}}$ refutation as well. The system is also complete for true QBFs [BPB24] because from every trail ending in either a conflict cube or by satisfying all clauses, a new cube can be learnt. Since, there are only finitely many new cubes, therefore eventually the empty term must be learnt.

The soundness of $\texttt{QCDCL}^{\texttt{cube}}$ was shown in [BPB24], by showing that from a refutation (verification) in this system, we can extract a `LDQ-Res`(`LDQ-TermRes`) derivation of the empty clause (term).

2.6 Dependency Schemes

In QBFs, the order of variables in the quantifier prefix plays a critical role in determining the truth or falsity of the formula. Simply reordering variables in the prefix can alter the logical meaning of the formula and, consequently, its truth value. As a result, QBF proof systems must incorporate rules that respect this quantifier structure. However, not all dependencies implied by the prefix are logically necessary for evaluating the formula. Some of these dependencies are spurious—they are enforced by the ordering of the variables, but are not necessary for preserving the truth value of the formula. Dependency schemes provide a formal mechanism to detect and eliminate such spurious dependencies. When incorporated into a proof system, a dependency scheme refines the interpretation of the quantifier prefix by specifying which variable dependencies must be preserved and which can be safely ignored. This allows the proof system to relax certain restrictions imposed by the original variable order, potentially enabling more efficient reasoning without compromising correctness.

Formally, dependency schemes are mappings that associate every PCNF formula with a binary relation on its variable in a manner that encodes constraints on the order of pairs of variables. The most basic dependency scheme defined is what is called the trivial dependency scheme:

Definition 2.3 (Definition 1, [SS16])**.** *The trivial dependency scheme* $\mathtt{D}^{\mathtt{trv}}$ *associates each PCNF formula* Φ *with the relation* $\mathtt{D}^{\mathtt{trv}}(\Phi) = \{(x, y) \in R_\Phi : q(x) \neq q(y)\}$. $\mathtt{D}^{\mathtt{trv}}(\Phi)$ *is called the trivial dependency relation of* Φ.

The trivial dependency scheme basically states that given a pair of variables x, y in Φ with different quantification, y depends on x if y is to the right of x in the quantifier prefix.

A scheme which for every formula Φ maps it to a subset of $\mathtt{D}^{\mathtt{trv}}(\Phi)$ is a potential

dependency scheme and is said to refine $\mathtt{D}^{\mathtt{trv}}$. Such schemes are referred to as *proto dependency schemes.*

Definition 2.4 (Definition 2, [SS16])**.** *Consider a mapping* $\mathtt{D}$ *that associates each PCNF formula* Φ *with a relation* $\mathtt{D}(\Phi) \subseteq \mathtt{D}^{\mathtt{trv}}(\Phi)$*. Then* $\mathtt{D}(\Phi)$ *is a dependency relation of* Φ *with respect to* $\mathtt{D}$*. Such a mapping* $\mathtt{D}$ *is called a proto dependency scheme.*
A proto dependency scheme $\mathtt{D}$ *is said to be tractable if there exists a polynomial time algorithm that, given a PCNF* Φ*, computes the dependency relation* $\mathtt{D}(\Phi)$*.*

The dependency schemes that are discussed in this thesis are all proto-dependency schemes. The first significant refinement of the trivial dependency scheme that we describe is called the *standard dependency scheme* [SS09]. The standard dependency scheme for a formula is defined in terms of the primal graph of the formula.

Definition 2.5 (Primal Graph (Defintion 7, [SS16]))**.** *Let* $\Phi = Q.\phi$ *be a PCNF formula. The primal graph* G *of the formula* Φ *is an undirected graph whose vertex set is* $V = var(\Phi)$ *and whose edge set is* $E = \{(x, y) : x, y \in var(\Phi),\ x \neq y,\ \exists C \in \phi$ *such that* $x, y \in var(C)\}$*.*

Using the definition of a primal graph, a standard dependency pair is defined as:

Definition 2.6 (Standard Dependency Pair (Defintion 8, [SS16]))**.** *Let* Φ *be a PCNF formula, and* $x, y \in var(\Phi)$*. We say* $\{x, y\}$ *is a standard dependency pair of* Φ *with respect to* $X \subseteq var_{\exists}(\Phi)$ *if there exists a path between* x *and* y *in* $G[X \cup \{x, y\}]$*, where* G *denotes the primal graph of* Φ*, and* $G[X \cup \{x, y\}]$ *denotes the sub-graph of* G *induced by* $X \cup \{x, y\}$*.*

In layman terms, any two nodes in the primal graph connected via a path only passing through vertices in X are a standard dependency pair with respect to X. These standard dependency pairs are then used to define the standard dependency scheme:

Definition 2.7 (Standard Dependency Scheme (Defintion 9, [SS16])). *The standard dependency scheme is a mapping* $\mathtt{D}^{\mathtt{std}}$ *that maps every PCNF formula* $\Phi = Q.\phi$ *to the relation* $\mathtt{D}^{\mathtt{std}}(\Phi)$ *defined by:*

$$\mathtt{D}^{\mathtt{std}}(\Phi) = \{(x, y) \in \mathtt{D}^{\mathtt{trv}}(\Phi) : \{x, y\} \text{ is a standard path dependency pair in } \Phi \text{ with respect to } (R_\Phi(x) \cap var_\exists(\Phi))\}.$$

Apart from the standard dependency schemes, there are two further dependency schemes that are of major interest. These two schemes are refinements of the standard dependency scheme, and hence also of the trivial dependency scheme. The schemes are the *resolution path dependency scheme* [SS12], and a slight variant of it, the *reflexive resolution path dependency scheme.* [SS16] Both of these schemes are based on the concept of a *resolution path.* Therefore to understand these schemes, first a resolution path must be understood.

Definition 2.8 (Resolution Path (Defintion 3, [SS16])). *Consider the PCNF formula* $\Phi = Q.\phi$*, and let* $X \subseteq var_\exists(\Phi)$*. A resolution path from a literal* l_1 *to a literal* l_{2k} *via* X *in* Φ *is a sequence of literals* $l_1, l_2, ..., l_{2k-1}, l_{2k}$ *such that:*

1. $\forall i \in [k]$*, there exists a clause* $C_i \in \phi$*, such that* $l_{2i-1}, l_{2i} \in C_i$
2. $\forall i \in [k]$, $var(l_{2i-1}) \neq var(l_{2i})$
3. $\forall i \in [k-1]$, $\{l_{2i}, l_{2i+1}\} \subseteq X \cup \bar{X}$
4. $\forall i \in [k-1]$, $\bar{l}_{2i} = l_{2i+1}$

From the definition of a resolution path, two properties of resolution paths that immediately pop out are: firstly, the reverse of a resolution path is also a resolution path, and secondly, if there are two resolution paths whose first and last literals respectively are different polarities of the same variable, then these resolution paths

can be concatenated together.

The definition of resolution path is used to define resolution path dependency pairs, which define a relation between pairs of variables in the formula, with respect to a subset of existential variables of the formula.

Definition 2.9 (Resolution Path dependency pair(Defintion 4, [SS16]))**.** *Let* Φ *be a PCNF formula, and* $x, y \subseteq var(\Phi)$*.* $\{x, y\}$ *is a resolution path dependency pair of* Φ *with respect to* $X \subseteq var_\exists(\Phi)$ *if at least one of the following is true:*

- *There exists a resolution path between* x *and* y*, and a resolution path between* $\neg x$ *and* $\neg y$ *in* Φ *via* X*.*
- *There exists a resolution path between* x *and* $\neg y$*, and a resolution path between* $\neg x$ *and* y *in* Φ *via* X*.*

Resolution path dependency pairs which have been defined above form the crux of the definition of the two resolution path based dependency schemes that are of interest to us. These schemes are defined below:

Definition 2.10 (Resolution Path Dependency Scheme (Defintion 5, [SS16]))**.** *The resolution path dependency scheme is the mapping* $\mathtt{D}^{\mathtt{res}}$*, which maps every PCNF formula* $\Phi = Q.\phi$ *to the relation* $\mathtt{D}^{\mathtt{res}}(\Phi)$*, defined as*

$$\mathtt{D}^{\mathtt{res}}(\Phi) = \{(x, y) \in \mathtt{D}^{\mathtt{trv}}(\Phi) : \{x, y\} \textit{ is a resolution path dependency pair in } \Phi \textit{ with respect to } (R_\Phi(x) \cap var_\exists(\Phi)) \setminus \{y\}\}.$$

Definition 2.11 (Reflexive Resolution Path Dependency Scheme (Defintion 6, [SS16]))**.** *The resolution path dependency scheme is the mapping* $\mathtt{D}^{\mathtt{rrs}}$*, which maps*

every PCNF formula $\Phi = Q.\phi$ *to the relation* $\mathtt{D}^{\mathtt{rrs}}(\Phi)$*, defined as*

$$\mathtt{D}^{\mathtt{rrs}}(\Phi) = \{(x, y) \in \mathtt{D}^{\mathtt{trv}}(\Phi) : \{x, y\} \textit{ is a resolution path dependency pair in } \Phi \textit{ with respect to } (R_\Phi(x) \cap var_\exists(\Phi))\}.$$

As can be seen from the definitions, the resolution path and the reflexive resolution path dependency scheme are nearly identical in definition. The only difference between them is the set of variables that the resolution paths involved are allowed to traverse via. In the reflexive resolution path dependency scheme, the resolution path is allowed to use the variable corresponding to the last literal in the middle of the path, but in the resolution path dependency scheme it is not so.

From the definition of the standard, resolution, and reflexive resolution path dependency schemes, it can be concluded that they are all proto-dependency schemes. One can define a relation between a pair of proto-dependency schemes by saying a proto-dependency scheme $\mathtt{D}$ is *at least as general* as a proto-dependency scheme $\mathtt{D}'$ if $\mathtt{D}(\Phi) \subseteq \mathtt{D}'(\Phi)$ for every PCNF formula Φ. Further, if the inclusion is strict for some Φ, then it is said that $\mathtt{D}$ is *strictly more general* than $\mathtt{D}'$.

Having defined the notion of *generality* between dependency schemes, the goal is to try to establish the relation between the three dependency schemes that have been defined so far in this chapter.

Proposition 2.12 (Proposition 1, [SS16] .)**.** $\mathtt{D}^{\mathtt{res}}$ *is strictly more general than* $\mathtt{D}^{\mathtt{rrs}}$*, and* $\mathtt{D}^{\mathtt{rrs}}$ *is strictly more general than* $\mathtt{D}^{\mathtt{std}}$*, and* $\mathtt{D}^{\mathtt{std}}$ *is strictly more general than* $\mathtt{D}^{\mathtt{trv}}$*.*

To see why this holds, note that by definition, all three schemes are at least as general as $\mathtt{D}^{\mathtt{trv}}$. Further, $\mathtt{D}^{\mathtt{res}}$ is at least as general as $\mathtt{D}^{\mathtt{rrs}}$. To see that this is strict, consider the formula $\Phi = Q.\phi$, where: $Q = \forall x \exists z \forall u \exists y$, and
$\phi = (x \vee u \vee \neg y) \wedge (\neg x \vee \neg u \vee \neg y) \wedge (z \vee u \vee y) \wedge (\neg z \vee u \vee \neg y) \wedge (\neg z \vee \neg u \vee y) \wedge (z \vee \neg u \vee \neg y)$.

One can see that $(x, y) \in \mathtt{D}^{\mathtt{rrs}}$ and $(x, y) \notin \mathtt{D}^{\mathtt{res}}$.

To see $\mathtt{D}^{\mathtt{std}}$ is strictly more general that $\mathtt{D}^{\mathtt{trv}}$, consider the QBF $\Phi = Q.\phi$, where: $Q = \forall u \exists x \exists y$ and $\phi = (u \vee x) \wedge (y)$. One can see that $(u, y) \in \mathtt{D}^{\mathtt{trv}}$ and $(u, y) \notin \mathtt{D}^{\mathtt{std}}$.

Thus, it remains to show $\mathtt{D}^{\mathtt{rrs}}$ is strictly more general than $\mathtt{D}^{\mathtt{std}}$. By definition, for any ordered pair of literals in $\mathtt{D}^{\mathtt{rrs}}$, there is a resolution path between them. From the definition of resolution paths, one can see that if there is a resolution path between two literals, then there is a path between their corresponding variables in the primal graph using the same variables of the resolution path. Therefore, if $(x, y) \in \mathtt{D}^{\mathtt{rrs}}$, then $(x, y) \in \mathtt{D}^{\mathtt{std}}$. Thus $\mathtt{D}^{\mathtt{rrs}}$ is at least as general as $\mathtt{D}^{\mathtt{std}}$. Now, to see that this is strict, consider the QBF $\Psi = \forall x \exists y (x \vee y)$. One can see that $\{x, y\} \in \mathtt{D}^{\mathtt{std}}(\Psi)$, but $\{x, y\} \notin \mathtt{D}^{\mathtt{rrs}}(\Psi)$.

Adding dependency schemes to QBF proof systems

When dependency schemes interact with a proof system, they modify its existing rules by parameterizing them with the dependency scheme. Which rules get modified and how depends on the proof system in question.

When a dependency scheme `D` gets added to the `Q-Res` proof system, its axiom and resolution rules do not change, but the $\forall$-`red` rule is modified and is replaced by the $\forall$(`D`)-`red` rule which states that instead of being allowed to drop only the trailing universal literals from a clause C, any universal variable u such that $(u, x) \notin \mathtt{D}$ for all existential variables $x \in C$ can be reduced. This new proof system is called the `Q`(`D`)-`Res` proof system and its rules are presented below.

However, for the `LDQ-Res` proof system, when a dependency scheme `D` is added to it, both the resolution and reduction rule get modified. The modification of the reduction rule is the same as in the case of `Q-Res`. For the resolution rule, the modification is such that instead of a tautology being allowed only on universal

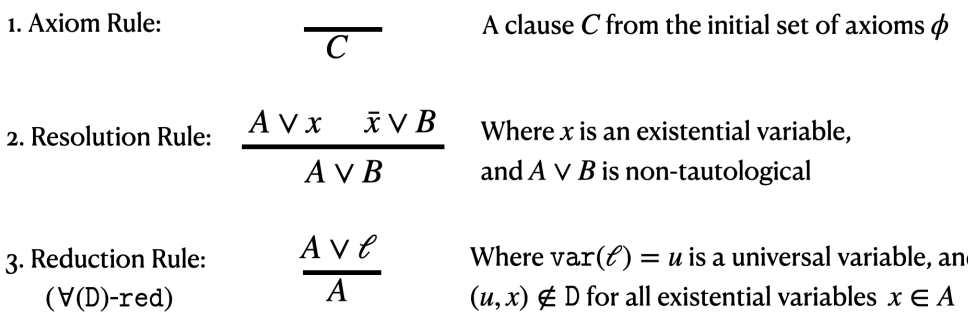

Figure 2.6: Rules of Q(D)-Res proof system

variables u to the right of the resolution pivot variable x, formation of a tautology across a resolution step with pivot x is now allowed for any universal variable u with $(u, x) \notin \mathtt{D}$. The system LDQ(D)-Res is summarised in the figure below.

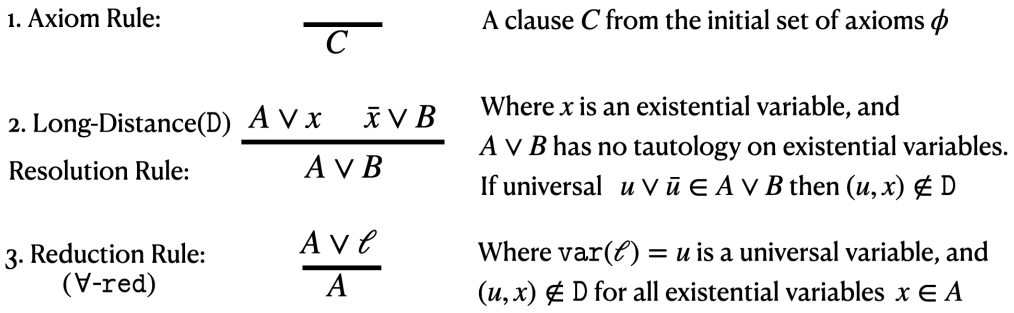

Figure 2.7: Rules of LDQ(D)-Res proof system

Similar to the aforementioned refutational systems Q-Res and LDQ-Res, dependency schemes can also be added to the term resolution proof systems Q-TermRes and LDQ-TermRes. When a scheme D is added to them, the new systems are called Q(D)-TermRes and LDQ(D)-TermRes respectively. Their rules modify in a similar dual manner as to their refutational counterparts and are compiled in the figures below.

Completeness and soundness of these systems

We first look at completeness of these dependency scheme parameterized systems.

1. Model Generation Rule: $\dfrac{}{T}$ — A cube T such that $T \cap C \neq \emptyset$ for all the initial set of axiom clauses $C \in \phi$

2. Term Resolution Rule: $\dfrac{A \wedge u \quad \bar{u} \wedge B}{A \wedge B}$ — Where u is a universal variable, and $A \wedge B$ is non-contradictory

3. Existential Reduction Rule (∃(D)-red): $\dfrac{A \wedge \ell}{A}$ — Where $\mathtt{var}(\ell) = x$ is an existential variable, and $(x, u) \notin \mathtt{D}$ for all universal variables $u \in A$

Figure 2.8: Rules of Q(D)-TermRes proof system

1. Model Generation Rule: $\dfrac{}{T}$ — A cube T such that $T \cap C \neq \emptyset$ for all the initial set of axiom clauses $C \in \phi$

2. Long-Distance(D) Term Resolution Rule: $\dfrac{A \wedge u \quad \bar{u} \wedge B}{A \wedge B}$ — Where u is an universal variable, and $A \wedge B$ is non-contradictory on universal variables. If existential $x \wedge \bar{x} \in A \wedge B$ then $(x, u) \notin \mathtt{D}$

3. Existential Reduction Rule (∃-red): $\dfrac{A \wedge \ell}{A}$ — Where $\mathtt{var}(\ell) = x$ is an existential variable, and $(x, u) \notin \mathtt{D}$ for all universal variables $u \in A$

Figure 2.9: Rules of LDQ(D)-TermRes proof system

From the rules of the systems one can see that by definition, for any proto dependency scheme D, anything that can be done in the respective systems without a dependency scheme can also be done when dependency schemes are involved. Therefore, any derivation in the system without a dependency scheme is a derivation in the system with the dependency scheme. Thus since Q-Res, LDQ-Res, Q-TermRes, and LDQ-TermRes are complete so are Q(D)-Res, LDQ(D)-Res, Q(D)-TermRes, and LDQ(D)-TermRes for a proto dependency scheme D.

When talking about soundness, let us first concentrate of Q(D)-Res and LDQ(D)-Res. By definition anything Q(D)-Res can do is also doable in LDQ(D)-Res. Therefore, soundness for LDQ(D)-Res would imply soundness for Q(D)-Res. Now, to show soundness of LDQ(D)-Res an additional constraint property is demanded of the dependency scheme D called *"normalcy"*, which is defined below:

Definition 2.13 (normal dependency scheme, (Definition 7,[PSS19b]))**.** *A dependency scheme* $\mathtt{D}$ *is said to be a monotone dependency scheme if* $\mathtt{D}(\phi|_\tau]) \subseteq \mathtt{D}(\phi)$ *for every PCNF formula* ϕ *and assignment* τ *to subset* $var(\phi)$*. The scheme* $\mathtt{D}$ *is said to be a simple dependency scheme if for every PCNF formula* $\Phi = \forall XQ.\phi$*, every* $LDQ(D)$ *derivation* P *from* Φ*, and for every* $u \in X$*, either* u *or* $\bar{u}$ *does not appear in* P*.*

A dependency scheme $\mathtt{D}$ *is said to be a normal dependency scheme if it is monotone and simple.*

In [PSS19b] it was shown that $\mathtt{LDQ(D)\text{-}Res}$ is sound for normal dependency schemes $\mathtt{D}$. The proof of soundness and arguments is similar to that of $\mathtt{LDQ\text{-}Res}$. Also it was shown that $\mathtt{D}^{\mathtt{rrs}}$ is a normal dependency scheme and as a corollary so is $\mathtt{D}^{\mathtt{std}}$. Therefore $\mathtt{Q(D}^{\mathtt{rrs}}\mathtt{)\text{-}Res}$, $\mathtt{Q(D}^{\mathtt{std}}\mathtt{)\text{-}Res}$ $\mathtt{LDQ(D}^{\mathtt{rrs}}\mathtt{)\text{-}Res}$ and $\mathtt{LDQ(D}^{\mathtt{std}}\mathtt{)\text{-}Res}$ are all sound proof systems.

Now, for $\mathtt{Q(D)\text{-}TermRes}$ the proof of soundness follows a different approach. Using the fact that $\mathtt{Q\text{-}TermRes}$ preserves countermodels, [SS16] showed that $\mathtt{Q(D}^{\mathtt{res}}\mathtt{)\text{-}TermRes}$ is sound by showing that it also preserves countermodels. As a corollary, since $\mathtt{D}^{\mathtt{res}}$ is strictly more general that $\mathtt{D}^{\mathtt{rrs}}$ and $\mathtt{D}^{\mathtt{std}}$, we can conclude $\mathtt{Q(D}^{\mathtt{rrs}}\mathtt{)\text{-}TermRes}$ and $\mathtt{Q(D}^{\mathtt{std}}\mathtt{)\text{-}TermRes}$ are sound proof systems as well.

However for $\mathtt{LDQ(D)\text{-}TermRes}$ as of now we do not know soundness for any dependency scheme other than $\mathtt{D}^{\mathtt{trv}}$. This is an outstanding open problem.

Chapter 3

Adding Dependency Schemes to QCDCL based Proof Systems

In the previous chapter, we were introduced to a plethora of QBF proof systems and to the notion of dependency schemes. We also saw how dependency schemes could be added to a few of the resolution based proof systems. In this chapter we will focus on the QCDCL proof system and how dependency schemes can be added to it.

3.1 Ways of adding Dependency Schemes to QCDCL based systems

From our understanding of the QCDCL proof system, we have seen that there are three main aspects to the system, the decision policy, unit propagation and learnable clauses, and whether or not cube-learning is used. The idea now is to see how these aspects get modified under the addition of dependency schemes.

Given the decision policy LEV-ORD, adding dependency schemes does not affect the

decision order at all. For dependency schemes to affect the decision order, a specific dependency-scheme-based decision policy must be defined. We define such policy, which we call D-ORD, below.

Definition 3.1 (D-ORD)**.** *For a dependency scheme* D*, the decision policy* D-ORD *permits a decision on a variable x at some point in a trail if all variables y on which x depends according to* D *(i.e. $(y, x) \in$ D), have already been assigned.*

For the propagation policy, the notion of unit clauses depends on the universal reductions allowed, and this in turn is affected by the dependency scheme. In the case of $R =$ NO-RED, no universal reductions are allowed anyway, so adding a dependency scheme to the proof system does not affect the policy. In the case of $R =$ RED, the definition of a unit propagation changes. A clause C propagates a literal ℓ at a position in the trail if the $\forall$(D)-reduction of C restricted to the trail so far is a unit clause. That is, the partial assignment α specified by the trail so far does not satisfy C, and after restricting C by α, applying all universal reductions allowed by D leaves the single literal ℓ; red-D$(C|_\alpha) = \{\ell\}$. In this case ℓ is propagated, to satisfy the clause C. Analogously, we can define unit propagation for cubes under the RED policy with dependency. In this case a cube C will propagate a universal literal ℓ at a point in the trail, if the $\exists$(D)-red of C restricted to the trail at that point is a unit cube, i.e. a single universal literal ℓ. In this case, $\bar{\ell}$ is propagated to falsify the cube. We denote this propagation policy as RED + D.

The dependency scheme modifies the reduction rule, which modifies the set of learnable clauses. The set of learnable clauses is now defined in a similar way as in Definition 2.1, but replacing red everywhere with red-D, the $\forall$(D) rule for universal reduction with respect to the dependency. Analogously the set of learnable cubes gets modified by replacing the red$_\exists$ rule by red-D$_\exists$, the $\exists$(D)-red rule for performing existential reduction on cubes with respect to the dependency scheme D.

A completely different way in which a dependency scheme D can be added to QCDCL

proof systems is by adding it as a preprocessing step, by applying the red-D rule on the axioms of the given formula. That is, produce QCDCL refutations of red-D(Φ) instead of Φ. Such a step does not change anything in the QCDCL system internally but doing so potentially changes the initial formula the proof system runs on.

3.2 Formalising the QCDCL proof system with Dependency Schemes

In the previous section, we saw that dependency schemes can primarily affect a QCDCL proof system in three ways: (i) Preprocessing, (ii) Decision Order, and (iii) Propagation and Learning. To account for all these posibilities, we introduce a new notation for QCDCL proof systems such that they can accomodate dependency schemes as well.

Definition 3.2. *The system* $\mathtt{QCDCL}^{\mathtt{ORD}}(\mathtt{ClausePol}, \mathtt{CubePol})$ *is the* QCDCL *proof system where*

1. ORD *denotes the decision policy; e.g.* LEV-ORD, D′-ORD, ANY-ORD.
   In D′-ORD, D′ *denotes the dependency scheme, such as* $\mathtt{D}^{\mathtt{rrs}}$-ORD, $\mathtt{D}^{\mathtt{std}}$-ORD.

2. ClausePol *is the dependency scheme* D *used in reduction, propagation, and learning for clauses. Note that this scheme need not be the same as the* D′ *in* D′-ORD.

3. CubePol $\in \{\mathtt{No\text{-}Cube}, \mathtt{Cube\text{-}LD}, \mathtt{Cube\text{-}D}\}$ *denotes the type of usage of cube learning;*

   (a) No-Cube*: No cube learning.*

   (b) Cube-LD*: Cube Learning used, but no dependency (only* $\mathtt{D}^{\mathtt{trv}}$*) in propagation, and cube learning using* LDQ-TermRes.

(c) `Cube-D`*: Cube Learning used, dependency scheme* `D` *(same as* `ClausePol`*) used in propagation, and cube learning is done using* `Q(D)-TermRes`*.*

Note that in this notation, clause learning always uses `LDQ(D)-Res`, where `D` might well be $\mathtt{D}^{\mathtt{trv}}$. However, for cube learning, the propagation and learning can use either long-distance term resolution or dependency schemes in `Q(D)-TermRes` without long-distance, not both. We impose this condition because the soundness of `LDQ(D)-TermRes` in general is not known. Therefore, in the case of cube learning, one must make a choice of whether to use dependency schemes or the power of long-distance term resolution.

In the notation of Definition 3.2, the standard vanilla `QCDCL` system denoted $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}_{\mathtt{RED}}$ in [BB23a] would be $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{No\text{-}Cube})$, whereas the $\mathtt{QCDCL}^{\mathtt{cube}}$ system from [BPB24] would be $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{Cube\text{-}LD})$.

In order to define what a derivation in these `QCDCL` systems must look like, we must first define trails and the learnable clauses and cubes from trails in this system

Definition 3.3 (Trails)**.** *Let* $\Phi := \mathtt{Q} \cdot \phi$ *be a QCNF in* n *variables. A trail* $\mathcal{T}$ *for* Φ *is a sequence of literals (or* $\Box, \top$*) of variables from* Φ *with some specific properties. We distinguish two types of literals in* $\mathcal{T}$ *: decision literals,, and propagated literals. Most of the time we write a trail* $\mathcal{T}$ *as*

$$\mathcal{T} = (p_{(0,1)}, \cdots, p_{(0,g_0)}; \mathbf{d_1}, p_{(1,1)}, \cdots p_{(1,g_1)}; \mathbf{d_2}, \cdots\cdots\cdots; \mathbf{d_r}, p_{(r,1)}, \cdots p_{(r,g_r)})$$

where the literals $\mathbf{d_i}$ *are decision literals, and the literals* $p_{i,j}$ *are propagated literals.*

To emphasize decisions, we will set decision literals in the trail in boldface and put a semicolon at the end of each decision level. The literal $p_{(i,j)}$ *represents the* j^{th} *propagated literal in the* i^{th} *decision level, determined by the corresponding decision* d_i*. The decision level 0 is the only level where we do not have a decision literal. We can view* $\mathcal{T}$ *as a set of literals or as an assignment and use the notation* $x \in \mathcal{T}$ *if*

the literal x is contained in $\mathcal{T}$. Let $s \in \{0, ..., r\}$ and $t \in \{0, ..., g_s\}$. The subtrail of $\mathcal{T}$ at the time (s, t) is the trail consisting of all literals from the leftmost literal in $\mathcal{T}$ up to (including) $p_{(s,t)}$, if $t \neq 0$, or d_s otherwise. We denote this subtrail by $T[s, t]$. The subtrail $T[0, 0]$ is defined as the empty trail.

Now there are further requirements for $\mathcal{T}$ to be a trail. The decisions have to be non-tautological and non-repeating, i.e., we require $\texttt{var}(d_i) \neq \texttt{var}(d_k)$ for each $i \neq k \in \{0, ..., r\}$. If $\square, \top \in \mathcal{T}$, then this must be the last (rightmost) literal in $\mathcal{T}$. In this case we will say that $\mathcal{T}$ has run into a conflict.

Definition 3.4 (Unit Clauses and Unit Propagation)**.** *The unit propagation policy used is $\texttt{RED} + \texttt{D}$. In this policy, at a point $\mathcal{T}[s, t]$ of a trail, a clause C is unit if $\texttt{red-D}(C|_{T[s,t]}) = (x)$ where x is an existential variable, and a cube C is unit if $\texttt{red-D}_\exists(C|_{T[s,t]}) = (u)$ where u is a universal variable.*

In trails we necessarily propagate both clauses and cubes if they are unit, and the clause/cube C is known as the antecedent clause/cube.

Having defined trails and the rules of propagation for trails in this setting, we next define the learnable constraints from a trail as follows:

Definition 3.5 (learning from conflict)**.** *From a trail*

$$\mathcal{T} = (p_{(0,1)}, \cdots, p_{(0,g_0)}; \mathbf{d_1}, p_{(1,1)}, \cdots p_{(1,g_1)}; \mathbf{d_2}, \cdots\cdots\cdots; \mathbf{d_r}, p_{(r,1)}, \cdots p_{(r,g_r)})$$

ending in a conflict $p_{(r,g_r)} \in \{\square, \top\}$, the sequence L_T of learnable constraints has a clause associated with each existential propagation in the trail if $p_{(r,g_r)} = \square$, and a cube associated with each universal propagation in the trail if $p_{(r,g_r)} = \top$. These associated clauses/cubes are constructed by tracing the conflict backwards through the trail as follows. ($\texttt{ante}(\ell)$ denotes the clause/cube that causes literal ℓ to be propagated; i.e. the antecedent.) Starting with $\texttt{ante}(\square)$ (respectively $\texttt{ante}(\top)$), we resolve in reverse over the antecedent clauses (cubes) that propagated the existential (uni-

versal) variables as described below. All such resulting clauses (cubes) are learnable constraints. The set of learnable constraints is denoted L_T.

In particular, if $p_{(r,g_r)} = \square$, *then*

- $C_{(r,g_r)} = \texttt{red-D}(\texttt{ante}(p_{(r,g_r)}))$.
- *For* $i \in \{0, 1, \cdots, r\}$ *and* $j \in [g_i - 1]$,

$$C_{(i,j)} = \begin{cases} \texttt{red-D}[\texttt{res}(C_{(i,j+1)}, \texttt{red-D}(\texttt{ante}(p_{(i,j)})), p_{(i,j)})] \\ \qquad\qquad \textit{if } p_{(i,j)} \textit{is existential and } \bar{p}_{(i,j)} \in C_{(i,j+1)} \\ C_{(i,j+1)} \qquad\qquad \textit{otherwise} \end{cases}$$

- *For* $i \in \{0, 1, \cdots, r-1\}$.

$$C_{(i,g_i)} = \begin{cases} \texttt{red-D}[\texttt{res}(C_{(i+1,1)}, \texttt{red-D}(\texttt{ante}(p_{(i,g_i)})), p_{(i,g_i)})] \\ \qquad\qquad \textit{if } p_{(i,g_i)} \textit{is existential and } \bar{p}_{(i,g_i)} \in C_{(i+1,1)} \\ C_{(i+1,1)} \qquad\qquad \textit{otherwise} \end{cases}$$

If $p_{(r,g_r)} = \top$, *then non-trivial cube-resolution is performed when* $p_{(i,j)}$ *is universal, not existential. The set* L_T *depends on* `CubePol`. *If* `CubePol` = `Cube-LD`, *then* `red-D` *is replaced by* $\texttt{red}_\exists$. *If* `CubePol` = `Cube-D`, *then* `red-D` *is replaced by* $\texttt{red-D}_\exists$, *but the* `res` *steps are performed only if the resolution is a valid* `Q(D)-TermRes` *resolution step.*

Definition 3.6 (learning from satisfaction)**.** *From a trail* $\mathcal{T}$

$$\mathcal{T} = (p_{(0,1)}, \cdots, p_{(0,g_0)}; \mathbf{d_1}, p_{(1,1)}, \cdots p_{(1,g_1)}; \mathbf{d_2}, \cdots\cdots\cdots; \mathbf{d_r}, p_{(r,1)}, \cdots p_{(r,g_r)})$$

that assigns all variables, satisfies all clauses, and does not satisfy any cube, the set

of learnable constraints is the set of cubes

$$L_T = \begin{cases} \{\texttt{red}_\exists(T') \mid T' \subset T; T' \textit{ satisfies all axioms and learnt clauses}\} & \textit{if } \texttt{CubePol} = \texttt{Cube-LD} \\ \{\texttt{red-D}_\exists(T') \mid T' \subset T; T' \textit{ satisfies all axioms and learnt clauses}\} & \textit{if } \texttt{CubePol} = \texttt{Cube-D} \end{cases}$$

Having set up all the necessary notations and definitions required to completely describe the $\texttt{QCDCL}^{\texttt{ORD}}(\texttt{ClausePol}, \texttt{CubePol})$ proof system, we will now define how a derivation in this system looks like.

Definition 3.7. *For a particular choice of* ORD, ClausePol, CubePol, *let* P *be the proof system* $\texttt{QCDCL}^{\texttt{ORD}}(\texttt{ClausePol}, \texttt{CubePol})$. *A* P*-derivation* ι *from a PCNF QBF* $\Phi = \mathcal{Q}\vec{x} \cdot \varphi$ *of a clause or cube* C *is a sequence of triples of the form:*

$$\iota = (T_1, C_1, \pi_1), \cdots, (T_m, C_m, \pi_m)$$

where each T_i *is a trail, each* C_i *is a clause/cube, and* $C_m = C$. *The objects* T_i, C_i, π_i *are as defined below.*

For each $i \in [m]$, φ_i *is a propositional formula of the form*

$$\varphi_i = \left(\bigwedge_{C \in \mathcal{C}_i} C\right) \vee \left(\bigvee_{T \in \mathcal{T}_i} T\right),$$

where $\mathcal{C}_i$ *is a set of clauses and* $\mathcal{T}_i$ *is a set of cubes. These formulas are defined iteratively:*

$$\begin{aligned} & & \mathcal{C}_1 &= \{C \mid C \in \varphi\} & \mathcal{T}_1 &= \emptyset \\ \textit{If } C_i \textit{ is a clause:} & & \mathcal{C}_{i+1} &= \mathcal{C}_i \cup \{C_i\}, & \mathcal{T}_{i+1} &= \mathcal{T}_i. \\ \textit{If } C_i \textit{ is a cube:} & & \mathcal{C}_{i+1} &= \mathcal{C}_i, & \mathcal{T}_{i+1} &= \mathcal{T}_i \cup \{C_i\}. \end{aligned}$$

For each $i \in [m]$, Φ_i *is the QBF with the same quantifier prefix as* Φ, *and inner formula* φ_i. *For each* $i \in [m]$, T_i *is a trail in the formula* Φ_i, C_i *is a learnable clause*

or cube from T_i, and π_i is the derivation of C_i as per `ClausePol` *or* `CubePol`*.*

A refutation in these systems is a derivation of the empty clause $\square$, and a verification in this system is a derivation of the empty term $\top$.

When preprocessing is involved we denote the `QCDCL` proof system as $\mathtt{D} + \mathtt{QCDCL}^{\mathtt{ORD}}(\mathtt{ClausePol}, \mathtt{CubePol})$ where `D` is the dependency scheme used to preprocess the given formula.

Definition 3.8. *For a particular choice of* `D`*,* `ORD`*,* `ClausePol`*,* `CubePol`*, let* `P` *be the proof system* $\mathtt{D} + \mathtt{QCDCL}^{\mathtt{ORD}}(\mathtt{ClausePol}, \mathtt{CubePol})$*. A* `P`*-derivation ι from a PCNF QBF $\Phi = \mathcal{Q}\vec{x} \cdot \varphi$ of a clause or cube C is the* $\mathtt{QCDCL}^{\mathtt{ORD}}(\mathtt{ClausePol}, \mathtt{CubePol})$ *of C form the PCNF $\Psi = \mathtt{D}(\Phi)$.*

Next we highlight some observations about the aforementioned family of systems that can be deduced with a simple understanding of the various parameters involved.

By definition, `D-ORD` generalises `LEV-ORD`, and `ANY-ORD` generalises `D-ORD`. Therefore this leads to the following observation.

Observation 3.9. *For dependency schemes* `D`*,* $\mathtt{D}'$*, and a* `CubePol` $\in \{\mathtt{No\text{-}Cube}, \mathtt{Cube\text{-}LD}, \mathtt{Cube\text{-}D}\}$*,*

- *Every derivation in* $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}, \mathtt{CubePol})$ *is a derivation in* $\mathtt{QCDCL}^{\mathtt{D'\text{-}ORD}}(\mathtt{D}, \mathtt{CubePol})$*.*
- *Every derivation in* $\mathtt{QCDCL}^{\mathtt{D'\text{-}ORD}}(\mathtt{D}, \mathtt{CubePol})$ *is a derivation in* $\mathtt{QCDCL}^{\mathtt{ANY\text{-}ORD}}(\mathtt{D}, \mathtt{CubePol})$*.*

Trails in a system without cube learning are also trails in the corresponding system with cube learning. Therefore all derivations without cube learning are valid derivations in the system with cube-learning as well. Hence:

Observation 3.10. *For* $\texttt{CubePol} \in \{\texttt{Cube-LD}, \texttt{Cube-D}\}$*, and for any decision policy* ORD*, any derivation in* $\texttt{QCDCL}^{\texttt{ORD}}(\texttt{D}, \texttt{No-Cube})$ *is also a derivation in* $\texttt{QCDCL}^{\texttt{ORD}}(\texttt{D}, \texttt{CubePol})$.

If the matrix of the given PCNF formula is *unsatisfiable*, then no satisfying trail can ever be constructed, no matter what policy is used, so no "cube learning" can ever happen. Hence:

Observation 3.11. *For* $\texttt{CubePol} \in \{\texttt{Cube-LD}, \texttt{Cube-D}\}$*, and for any decision policy* ORD*, if a PCNF formula* Φ *has an unsatisfiable matrix, then any derivation from* Φ *in* $\texttt{QCDCL}^{\texttt{ORD}}(\texttt{D}, \texttt{CubePol})$ *is also a derivation in* $\texttt{QCDCL}^{\texttt{ORD}}(\texttt{D}, \texttt{No-Cube})$.

Before proceeding further, the following propositions are noteworthy to keep in mind under special cases of the properties of dependency schemes.

Proposition 3.12. *For a given* ORD*,* D *and any* $\texttt{CubePol} \in \{\texttt{No-Cube}, \texttt{Cube-LD}, \texttt{Cube-D}\}$ *and a QBF* Φ*, if* $\texttt{D}(\Phi) = \texttt{D}^{\texttt{trv}}(\Phi)$*, then the systems* $\texttt{QCDCL}^{\texttt{ORD}}(\texttt{D}^{\texttt{trv}}, \texttt{CubePol})$*,* $\texttt{QCDCL}^{\texttt{ORD}}(\texttt{D}, \texttt{CubePol})$*,* $\texttt{D} + \texttt{QCDCL}^{\texttt{ORD}}(\texttt{D}^{\texttt{trv}}, \texttt{CubePol})$ *and* $\texttt{D} + \texttt{QCDCL}^{\texttt{ORD}}(\texttt{D}, \texttt{CubePol})$ *behave equivalently on* Φ *and produce exactly the same refutations*

This is simply because if for a particular formula $\texttt{D} = \texttt{D}^{\texttt{trv}}$, then adding the dependency scheme gives nothing new to the system as no extra reductions are enabled.

Next,

Proposition 3.13. *For a given QBF* $\Phi = \mathcal{Q} \cdot \varphi$ *in PCNF, and a dependency scheme* D*, if* $\texttt{D}(\Phi) = \emptyset$*, then* $\texttt{red-D}(\Phi)$ *is a propositional formula (no universal variables in any clause).*

Further, if $\texttt{D}(\Phi) = \emptyset$*, then the propositional formula* $\texttt{red-D}(\Phi)$ *is easy to refute in Resolution if and only if* Φ *is easy to refute in* $\texttt{D} + \texttt{QCDCL}^{\texttt{ORD}}(\texttt{ClausePol}, \texttt{CubePol})$.

This is because, if $\mathtt{D}(\Phi)$ is empty, then all universals can be dropped from the formula on the application of $\forall(\mathtt{D})$-$\mathtt{red}$ on it. Preprocessing the formula achieves this reduction on it. Therefore, in this case $\mathtt{QCDCL}^{\mathtt{ORD}}(\mathtt{ClausePol}, \mathtt{CubePol})$ acts on a purely propositional formula and thus is equivalent to CDCL, which is known to be equivalent to the Resolution proof system [PD11].

These observations and propositions prove to be useful when trying to understand the relative hardness of various formulas in this family of systems, as we will see in Chapter 4.

3.3 Completeness and Soundness of QCDCL proof systems with Dependency Schemes

Before we can work with these families of QCDCL proof systems with dependency schemes, we must show that they are sound and complete. Since this thesis focuses on false formulas, we only consider soundness of refutations, showing that these systems cannot refute true formulas.

First we see that:

Theorem 3.14. *For any normal dependency scheme* $\mathtt{D}$*, the proof system* $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}, \mathtt{No\text{-}Cube})$ *is refutationally complete.*

Proof. In Theorem 3.16 of [BB23a], $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{No\text{-}Cube})$ (denoted there as $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}_{\mathtt{RED}}$) is shown to be complete. Exactly the same proof, which is actually quite intricate, works also to show the completeness here. The idea is as follows: for a false formula Φ, in the 2-player evaluation game, the universal player has a winning strategy on Φ. Now, we can construct trails in level order that perform propagations whenever applicable, decide the polarity of existential variables arbitrarily, and decide the polarities of universal variables following this winning strategy. (This is

possible because decisions are level-ordered.) The winning strategy guarantees that each such trail runs into a conflict. The set of learnable clauses either contains the empty clause, or is shown to contain an asserting clause – one which after backtracking becomes unit at some point in the trail – and an asserting clause is shown to be new. Thus each trail that does not terminate the refutation learns a new clause, and there are only finitely many clauses that can be added. All the arguments in this outline work also in the presence of a normal dependency scheme (D) that is used in both propagation and learning. □

Next,

Theorem 3.15. *For any normal dependency scheme* D*, the proof systems* $\mathtt{QCDCL}^{\mathtt{ANY\text{-}ORD}}(\mathtt{D}, \mathtt{Cube\text{-}LD})$ *and* $\mathtt{QCDCL}^{\mathtt{ANY\text{-}ORD}}(\mathtt{D}, \mathtt{Cube\text{-}D})$ *are sound.*

Proof. To show that both these systems are sound, it is enough to show the following three statements: (1) The derivation of any learnt clause is a valid LDQ(D)-Res derivation. (2) If CubePol = Cube-LD, the derivation of any learnt cube is a valid LDQ-TermRes derivation, and the addition of cubes when learning from satisfaction is sound. (3) If CubePol = Cube-D, the derivation of any learnt cube is a valld Q(D)-TermRes derivation. From these three, it follows that sticking together the derivations of the learnt clauses/cubes gives a derivation of the empty clause/cube in the corresponding systems (LDQ(D)-Res, LDQ-TermRes, Q(D)-TermRes), and all these systems are known to be sound.

Statement (3) is true by definition: we perform term resolution in learning only if the resolution is valid in Q(D)-TermRes. For Statement (2), the addition of cubes is shown to be sound in [BPB24, Theorem 3.8].

For statement (1), we need to show that the resolution steps performed while learning respect the side-conditions of LDRes(D). The analogous statement when $\mathtt{D} = \mathtt{D}^{\mathtt{trv}}$ is proved in [BB23a, Lemma 3.7, Proposition 3.8, Theorem 3.9], but the same proof

works with any D. For details, see Lemma 3.16 below. □

The following lemma completes the proof of Theorem 3.15. Its proof replicates the proof of Lemma 3.7 and Proposition 3.8 of [BB23a].

Lemma 3.16. *For any normal dependency scheme* D*, the derivation of a clause learnt from a trail in the proof systems* $\mathtt{QCDCL}^{\mathtt{ANY\text{-}ORD}}(\mathtt{D}, \mathtt{Cube\text{-}LD})$ *and* $\mathtt{QCDCL}^{\mathtt{ANY\text{-}ORD}}(\mathtt{D}, \mathtt{Cube\text{-}D})$ *are valid* LDQ(D)-Res *derivations.*

Proof. We need to show that every clause learnt from a trail :

$$\mathcal{T} = (p_{(0,1)}, \cdots, p_{(0,g_0)}; \mathbf{d_1}, p_{(1,1)}, \cdots p_{(1,g_1)}; \mathbf{d_2}, \cdots\cdots\cdots; \mathbf{d_r}, p_{(r,1)}, \cdots p_{(r,g_r)})$$

has a valid LDQ(D)-Res derivation. Let $C_{i,j}$ denote the clause learnt corresponding to propagated literal $p_{(i,j)}$. We do not need to consider any cubes learnt along the way as they do not effect the extracted LDQ(D)-Res derivation in any manner.

Step 1: No $C_{i,j}$ contains an existential tautology.
Suppose there exists a variable x such that $x \neq \mathtt{var}(p_{(i,j)})$ and $x \in C_{i,j+1}$ and $\bar{x} \in \mathtt{red\text{-}D}(\mathtt{ante}(p_{i,j}))$. Let $A = \mathtt{ante}(p_{(i,j)})$, then since x is existential variable, and $\bar{x} \in A$, therefore x must be assigned in the trail prior to the propagation of $p_{(i,j)}$.
On the other hand, we have $x \in C_{i,j+1}$, which is the learnable clause 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 $x \in \mathtt{ante}(p_{(k,m)})$. But because x appears in the trail to the left of $p_{(i,j)}$, this gives a contradiction since $\mathtt{ante}(p_{(k,m)})$ must not become true before propagating $p_{(k,m)}$.

Step 2: Derivation of clauses with universal tautologies obeys the rules of LDQ(D)-Res.
Proof goes via contradiction. Suppose the rules of LDQ(D)-Res are violated along the learnable sequence. This means that there is a propagated literal $p_{(i,j)}$, such that

the formation of learnt clause $C_{i,j}$ violates the long-distance condition. Without loss of generality, let u be the universal variable causing the violation. This implies that both $(u, \texttt{var}(p_{(i,j)})) \in \texttt{D}$, and one of the following conditions must hold (let ℓ_u denote a literal of u).

1. $\ell_u \in C_{i,j+1}$ and $\bar{\ell_u} \in \texttt{ante}(p_{i,j})$
2. $u \vee \bar{u} \in C_{i,j+1}$ and $\ell_u \in \texttt{ante}(p_{i,j})$
3. $\ell_u \in C_{i,j+1}$ and $u \vee \bar{u} \in \texttt{ante}(p_{i,j})$
4. $u \vee \bar{u} \in C_{i,j+1}$ and $u \vee \bar{u} \in \texttt{ante}(p_{i,j})$

Consider the first case:
Since, $\ell_u \in C_{i,j+1}$ therefore there exists a propagated literal $p_{k,m}$ right of $p_{i,j}$ in the trail such that $\ell_u \in \texttt{ante}(p_{k,m})$. Now, for $p_{k,m}$ to have been propagated one of the following two cases must hold:

- $\bar{\ell_u}$ was assigned in the trail before $p_{k,m}$ was propagated. This has two subcases: (i) $\bar{\ell_u}$ was assigned before $p_{i,j}$ was propagated. This is not possible as $\bar{\ell_u} \in \texttt{ante}(p_{i,j})$, therefore its appearance in the trail would mean $p_{i,j}$ could not have been propagated. (ii) $\bar{\ell_u}$ was assigned after $p_{i,j}$ was propagated. In this case it would mean that the variable u had not been assigned in the trail before $p_{i,j}$, and had gone away due to $\forall$(D)-red during the propagation of $p_{i,j}$. This violates the fact $(u, \texttt{var}(p_{(i,j)})) \in \texttt{D}$.
- ℓ_u was reduced from $\texttt{ante}(p_{k,m})$ by the $\forall$(D)-red at the point of propagation of $p_{k,m}$. In this case it would mean that the variable u had not been assigned in the trail upto this point. Hence, during the propagation of $p_{i,j}$ it had gone away due to $\forall$(D)-red . This violates the fact $(u, \texttt{var}(p_{(i,j)})) \in \texttt{D}$.

This shows that $(u, \mathtt{var}(p_{(i,j)})) \in \mathtt{D}$ and first case cannot hold together. A similar argument holds for $(u, \mathtt{var}(p_{(i,j)})) \in \mathtt{D}$ with all other cases. Therefore, rules of $\mathtt{LDQ(D)\text{-}Res}$ could not have been violated during creating the learnable sequence. □

Now we can conclude that the new systems using dependency schemes in decision order are sound and complete.

Theorem 3.17. *For any dependency schemes* $\mathtt{D}'$, $\mathtt{D}$ *where* $\mathtt{D}$ *is normal, and for each* $\mathtt{CubePol} \in \{\mathtt{No\text{-}Cube}, \mathtt{Cube\text{-}LD}, \mathtt{Cube\text{-}D}\}$, *the proof system* $\mathtt{QCDCL}^{\mathtt{D}'\text{-}\mathtt{ORD}}(\mathtt{D}, \mathtt{CubePol})$ *is sound and refutationally complete.*

Proof. By Theorem 3.14, $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}, \mathtt{No\text{-}Cube})$ is refutationally complete.

By Observation 3.9, $\mathtt{QCDCL}^{\mathtt{D}'\text{-}\mathtt{ORD}}(\mathtt{D}, \mathtt{No\text{-}Cube})$ simulates $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}, \mathtt{No\text{-}Cube})$ and hence is refutationally complete.

By Observation 3.10 both $\mathtt{QCDCL}^{\mathtt{D}'\text{-}\mathtt{ORD}}(\mathtt{D}, \mathtt{Cube\text{-}LD})$ and $\mathtt{QCDCL}^{\mathtt{D}'\text{-}\mathtt{ORD}}(\mathtt{D}, \mathtt{Cube\text{-}D})$ simulate $\mathtt{QCDCL}^{\mathtt{D}'\text{-}\mathtt{ORD}}(\mathtt{D}, \mathtt{No\text{-}Cube})$ and hence are refutationally complete.

By Observation 3.9, $\mathtt{QCDCL}^{\mathtt{ANY\text{-}ORD}}(\mathtt{D}, \mathtt{CubePol})$ simulates $\mathtt{QCDCL}^{\mathtt{D}'\text{-}\mathtt{ORD}}(\mathtt{D}, \mathtt{CubePol})$. By Theorem 3.15, $\mathtt{QCDCL}^{\mathtt{ANY\text{-}ORD}}(\mathtt{D}, \mathtt{CubePol})$ is sound. Hence all the systems it simulates are also sound.

□

This concludes the proof that the entire family of QCDCL systems with normal dependency schemes are sound and complete.

Chapter 4

Some Relevant Formulae of Interest

In this chapter, we revisit some known formulas as well as introduce some new formulas and study their strength in various QBF proof systems.

4.1 The PHP formula

The Pigeonhole principle states that if you have more pigeons than pigeonholes and each pigeon must go into one hole, then at least one hole must contain more than one pigeon.

The PHP formula the encoding of this principle as an unsatisfiable propositional formula which tries to map $n+1$ pigeons to n holes in a manner such that: (1) each pigeon must be assigned to atleast one hole, and (2) no two pigeons are mapped to the same hole . Formally, they can be encoded as a CNF formula below.

Formula 4.1 (PHP formula)**.** *Let $P_{i,j}$ be a propositional variable that is true iff*

pigeon i is assigned to hole j.

(1) Each pigeon goes into at least one hole:

$$\bigwedge_{i=1}^{n+1} \left(\bigvee_{j=1}^{n} P_{i,j} \right)$$

(2) No two pigeons share the same hole:

$$\bigwedge_{1 \le i < k \le n+1} \bigwedge_{j=1}^{n} (\neg P_{i,j} \vee \neg P_{k,j})$$

These formulas are of interest to us because of their unsatisfiability and the fact that they require resolution proofs of exponential size [Hak85]. They will appear frequently in many of the newly constructed QBFs in this chapter, and the fact that they are hard to refute in resolution will be used as a crucial aspect of many of the lower bounds.

4.2 The QParity Formula

The first family of formulas that we study are the QParity formulas, first defined in [BCJ19]

Formula 4.2. *[$\texttt{QParity}_n$, [BCJ19]] The $\texttt{QParity}_n$ formula has the prefix $\exists x_1, \cdots, x_n \forall z \exists t_2, \cdots, t_n$ and the matrix*

$$\begin{array}{rcccc} & x_1 \vee x_2 \vee \bar{t}_2 & \bar{x}_1 \vee \bar{x}_2 \vee \bar{t}_2 & x_1 \vee \bar{x}_2 \vee t_2 & \bar{x}_1 \vee x_2 \vee t_2 \\ \textit{for } i = 2, \ldots, n: & x_i \vee t_{i-1} \vee \bar{t}_i & x_i \vee \bar{t}_{i-1} \vee t_i & \bar{x}_i \vee t_{i-1} \vee \bar{t}_i & \bar{x}_i \vee \bar{t}_{i-1} \vee \bar{t}_i \\ & & t_n \vee z & & \bar{t}_n \vee \bar{z} \end{array}$$

Let us look at the dependency schemes for these formulas

Proposition 4.3. *For the QParity formulas, $\mathtt{D}^{\mathtt{trv}}$, $\mathtt{D}^{\mathtt{std}}$, and $\mathtt{D}^{\mathtt{rrs}}$ are equivalent.*

i.e.

$$\mathtt{D}^{\mathtt{trv}}(\mathtt{QParity}) = \mathtt{D}^{\mathtt{std}}(\mathtt{QParity}) = \mathtt{D}^{\mathtt{rrs}}(\mathtt{QParity})$$

Proof. Since, for each t_i, there is a path $(z, t_n, \bar{t}_n, t_{n-1}, \bar{t}_{n-1}, \cdots, t_i)$ and a path $(\bar{z}, \bar{t}_n, t_n, \bar{t}_{n-1}, t_{n-1}, \cdots, \bar{t}_i)$ therefore $(z, t_i) \in \mathtt{D}^{\mathtt{rrs}}$ for all i. Thus $\mathtt{D}^{\mathtt{rrs}}$ is equivalent to $\mathtt{D}^{\mathtt{trv}}$. Since, we know $\mathtt{D}^{\mathtt{rrs}} \subseteq \mathtt{D}^{\mathtt{std}}$, therefore they are all equivalent. □

Since, $\mathtt{D}^{\mathtt{trv}}$, $\mathtt{D}^{\mathtt{std}}$ and $\mathtt{D}^{\mathtt{rrs}}$ are equivalent for this family of formulas our expectation is clearly that adding these dependency schemes to the QCDCL system when acting on the QParity formulas make no difference.

Lemma 4.4. *The* QParity *formulas require exponential size refutations in* Q-Res*,* QU-Res*,* Q($\mathtt{D}^{\mathtt{rrs}}$)-Res*, but they have polynomial size refutations in* $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{No\text{-}Cube})$.

Proof. The fact that QParity requires exponential size refutations in QU-Res, Q-Res was shown in [BCJ19], the result for Q($\mathtt{D}^{\mathtt{rrs}}$)-Res follows from the fact that $\mathtt{D}^{\mathtt{trv}} = \mathtt{D}^{\mathtt{rrs}}$.

QParity formulas having polynomial size refutations in $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{No\text{-}Cube})$ was shown in [BB23a]. □

Lemma 4.5. *For* $\mathtt{ClausePol} \in \{\mathtt{D}^{\mathtt{trv}}, \mathtt{D}^{\mathtt{std}}, \mathtt{D}^{\mathtt{rrs}}\}$ *and* $\mathtt{CubePol} \in \{\mathtt{No\text{-}Cube}, \mathtt{Cube\text{-}LD}, \mathtt{Cube\text{-}D}^{\mathtt{rrs}}\}$ *the* QParity *formulas have polynomial size refutations in* $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{ClausePol}, \mathtt{CubePol})$.

Proof. Since, $\mathtt{D}^{\mathtt{trv}} = \mathtt{D}^{\mathtt{std}} = \mathtt{D}^{\mathtt{rrs}}$, and by Observation 3.10, and that the QParity formulas having polynomial size refutations in $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{No\text{-}Cube})$ [BB23a], the QParity formulas have polynomial size refutations in $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{ClausePol}, \mathtt{CubePol})$. □

4.3 The Equality Formula

The second formula we look at is among the most well known family of QBF formulas, the Equality formula first introduced in [BBH19].

Formula 4.6. *[$\texttt{Equality}_n$, [BBH19]] The $\texttt{Equality}_n$ formula has the prefix $\exists x_1 \cdots x_n \forall u_1 \cdots u_n \exists t_1 \cdots t_n$ and the PCNF matrix*

$$\underbrace{(\bar{t}_1 \vee \cdots \vee \bar{t}_n)}_{T_n} \wedge \bigwedge_{i=1}^{n} \left[\underbrace{(x_i \vee u_i \vee t_i)}_{A_i} \wedge \underbrace{(\bar{x}_i \vee \bar{u}_i \vee t_i)}_{B_i} \right]$$

(In [BBH19], the long clause T_n has positive t literals and the short clauses A_i, B_i have negated t literals; it is straightforward to see that the formulations are equivalent upto renaming of variables. The formulation we define above is as used in [BB23a], Definition 5.6.)

Proposition 4.7. *For the* Equality *formulas* $\texttt{D}^{\texttt{trv}}$ *and* $\texttt{D}^{\texttt{std}}$ *are equivalent, but for* $\texttt{D}^{\texttt{rrs}}$*, the t variables are independent of the u variables. i.e.*

$$\texttt{D}^{\texttt{trv}}(\texttt{Equality}) = \texttt{D}^{\texttt{std}}(\texttt{Equality})$$

$$\texttt{D}^{\texttt{rrs}}(\texttt{Equality}) = \emptyset$$

Proof. In the case for $\texttt{D}^{\texttt{std}}$, for each (u_i, t_j) pair if $i = j$ they are in the same clause, while if $i \neq j$, there is a path u_i, t_i, t_j between them, thus $(u_i, t_j) \in \texttt{D}^{\texttt{std}}$ for all i, j.

Now in the case for $\texttt{D}^{\texttt{rrs}}$, there are paths between $u_i, \bar{t}_j$ and $\bar{u}_i, \bar{t}_j$, but no paths from u_i, t_j or $\bar{u}_i, t_j$ therefore $\texttt{D}^{\texttt{rrs}} = \emptyset$. □

As before, we first highlight some known existing results about the hardness of these formula

Lemma 4.8. *The* Equality *formulas*

- *require exponential size refutations in* $\mathtt{QU\text{-}Res}$, $\mathtt{Q\text{-}Res}$, $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{No\text{-}Cube})$.
- *and for* $\mathtt{CubePol} \in \{\mathtt{Cube\text{-}LD}, \mathtt{Cube\text{-}D}^{\mathtt{trv}}\}$ *have polynomial size refutations in* $\mathtt{Q}(\mathtt{D}^{\mathtt{rrs}})\mathtt{\text{-}Res}$ *and* $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{CubePol})$.

Proof. In [BBH19] it was shown that the $\mathtt{Equality}$ formulas require exponential size refutations in $\mathtt{QU\text{-}Res}$, and hence they also require exponential size refutations in $\mathtt{Q\text{-}Res}$. In [BB23a], it was shown that they require exponential size refutations in $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{No\text{-}Cube})$.

That $\mathtt{Equality}$ formulas have polynomial size refutations in $\mathtt{Q}(\mathtt{D}^{\mathtt{rrs}})\mathtt{\text{-}Res}$ and $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{CubePol})$ where $\mathtt{CubePol} \in \{\mathtt{Cube\text{-}LD}, \mathtt{Cube\text{-}D}^{\mathtt{trv}}\}$ were shown in [BB20] and [BPB24] respectively. □

Now the question is what happens to the hardness of these formulas when dependency schemes get involved? Our known results indicate that these formulas are hard in standard $\mathtt{QCDCL}$ without cube-learning but become easy to refute when cube-learning is involved. Similarly, they are hard in $\mathtt{Q\text{-}Res}$ but have short refutations in the presence of $\mathtt{D}^{\mathtt{rrs}}$.

It is easy to observe that since $\mathtt{D}^{\mathtt{rrs}}$ and $\mathtt{D}^{\mathtt{std}}$ are equivalent for these formulas the known results for $\mathtt{D}^{\mathtt{trv}}$ translate to $\mathtt{D}^{\mathtt{std}}$ due to Proposition 3.12 i.e.

Observation 4.9. *The* $\mathtt{Equality}$ *formulas require exponential size refutations in* $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{std}}, \mathtt{No\text{-}Cube})$*, but for* $\mathtt{CubePol} \in \{\mathtt{No\text{-}Cube}, \mathtt{Cube\text{-}LD}, \mathtt{Cube\text{-}D}^{\mathtt{std}}\}$*, they have polynomial size refutations in* $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{std}}, \mathtt{CubePol})$.

Lemma 4.10. *For* $\mathtt{CubePol} \in \{\mathtt{No\text{-}Cube}, \mathtt{Cube\text{-}LD}, \mathtt{Cube\text{-}D}^{\mathtt{rrs}}, \mathtt{Cube\text{-}D}^{\mathtt{std}}\}$*, the* $\mathtt{Equality}_n$ *formulas have* $O(n^2)$ *refutations in* $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{CubePol})$.

Proof. Due to Observation 3.10, it is enough to show that the $\mathtt{Equality}_n$ formulas have $O(n^2)$ refutations in $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{No\text{-}Cube})$.

Since $\mathtt{D}_{\forall}^{\mathtt{rrs}}(\mathtt{Equality}_n) = \emptyset$, the propagation policy and clause learning always reduce the universal u variables from the corresponding clauses.

We will construct a polynomial size refutation for the $\mathtt{Equality}_n$ formulas containing $2(n-1)$ trails all of which end in conflict. Hence, cube learning is never used and the refutation is independent of the `CubePol`.

Define the following clauses: For $i \in [n]$, $T_i = \bigvee_{j \leq i} \bar{t}_j$; for $i \in [n] \setminus \{1\}$, $L_i = \bar{x}_i \vee T_{i-1}$ and $R_i = x_i \vee T_{i-1}$.

We will construct the trails $\mathcal{U}_{n-1}, \mathcal{V}_{n-1}, \mathcal{U}_{n-2}, \mathcal{V}_{n-2}, \cdots, \mathcal{U}_1, \mathcal{V}_1$, and learn clauses $L_{n-1}, R_{n-1}, \cdots L_1, R_1$ corresponding to these trails. The $\mathcal{U}$ trails decide x variables (as many as is possible until conflict) positively; the $\mathcal{V}$ trails decide them negatively. Due to the `RED` policy, each decision propagates at least one t literal.

The initial trail is

$$\mathcal{U}_{n-1} := (\mathbf{x_1}, t_1; \mathbf{x_2}, t_2, ..., \mathbf{x_{n-1}}, t_{n-1}, \bar{t}_n, x_n, \square)$$

and the antecedent clauses are $\mathtt{ante}(t_j) = B_j$ for $j \in [n-1]$, $\mathtt{ante}(\bar{t}_n) = T_n$, $\mathtt{ante}(x_n) = A_n$, and $\mathtt{ante}(\square) = B_n$. From these set of clauses we learn the clause $L_{n-1} = \bar{x}_{n-1} \vee T_{n-2}$.

Restarting, create a symmetric trail to $\mathcal{U}_{n-1}$ flipping each decision:

$$\mathcal{V}_{n-1} := (\bar{\mathbf{x}}_\mathbf{1}, t_1; \bar{\mathbf{x}}_\mathbf{2}, t_2, ..., \bar{\mathbf{x}}_\mathbf{n-1}, t_{n-1}, \bar{t}_n, x_n, \square)$$

where the antecedent clauses are $\mathtt{ante}(t_j) = A_j$ for $j \in [n-1]$, $\mathtt{ante}(\bar{t}_n) = T_n$, $\mathtt{ante}(x_n) = A_n$, and $\mathtt{ante}(\square) = B_n$. From these set of clauses we learn the clause $R_{n-1} = x_{n-1} \vee T_{n-2}$.

We now go down now from $i = n-2$ down to $i = 2$. At stage i, we first construct trail $\mathcal{U}_i$ by deciding x variables positively; we reach a conflict after deciding x_i. The

trail and antecedent clauses are as follows:

$$\mathcal{U}_i := (\mathbf{x_1}, t_1; \mathbf{x_2}, t_2, ..., \mathbf{x_i}, t_i, x_{i+1}, \square)$$

with $\texttt{ante}(t_j) = B_j$ for $j \in [i]$, $\texttt{ante}(x_{i+1}) = L_{i+1}$, $\texttt{ante}(\square) = R_{i+1}$. From this we learn the clause L_i.

Next, we create the symmetrical trail by deciding the x variables negatively

$$\mathcal{V}_i := (\bar{\mathbf{x}}_\mathbf{1}, t_1; \bar{\mathbf{x}}_\mathbf{2}, t_2, ..., \bar{\mathbf{x}}_\mathbf{i}, t_i, x_{i+1}, \square)$$

with antecedent clauses $\texttt{ante}(t_j) = A_j$ for $j \in [i]$, $\texttt{ante}(x_{i+1}) = L_{i+1}$, $\texttt{ante}(\square) = R_{i+1}$. From this we learn the clause R_i.

The proof ends with the two trails

$$\mathcal{U}_1 = (\mathbf{x_1}, t_1, x_2, \square)$$

with antecedents $\texttt{ante}(t_1) = B_1$, $\texttt{ante}(x_2) = L_2$, $\texttt{ante}(\square) = R_2$, from which we learn $L_1 = \bar{x}_1$, and finally the last trail

$$\mathcal{V}_1 = (\bar{x_1}, t_1, x_2, \square)$$

with antecedents $\texttt{ante}(\bar{x}_1) = L_1$, $\texttt{ante}(t_1) = B_1$, $\texttt{ante}(x_2) = L_2$, $\texttt{ante}(\square) = R_2$, from which we learn the empty clause $\square$, completing the refutation.
The $\texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{rrs}}, \texttt{No-Cube})$ refutation we have created has $O(n)$ trails and hence overall size $O(n^2)$. $\square$

Thus, we have seen that when $\texttt{D}^{\texttt{rrs}}$ is used in propagation and learning the `Equality` formulas have polynomial size refutations irrespective of cube-learning. What happens when $\texttt{D}^{\texttt{rrs}}$ is used in preprocessing ? We see that in that case irrespective of

cube-learning or even if $\mathtt{D}^{\mathtt{rrs}}$ is used in propagation, the `Equality` formulas have polynomial size refutations.

Lemma 4.11. *For* $\mathtt{CubePol} \in \{\mathtt{No\text{-}Cube}, \mathtt{Cube\text{-}LD}, \mathtt{Cube\text{-}D}^{\mathtt{rrs}}, \mathtt{Cube\text{-}D}^{\mathtt{std}}\}$, *and* $\mathtt{ClausePol} \in \{\mathtt{D}^{\mathtt{trv}}, \mathtt{D}^{\mathtt{std}}.\mathtt{D}^{\mathtt{rrs}}\}$ *the* `Equality` *formulas have polynomial size refutations in* $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{ClausePol}, \mathtt{CubePol})$.

Proof. Since $\mathtt{D}^{\mathtt{rrs}}(\mathtt{Equality}) = \emptyset$, preprocessing the `Equality` formulas gives us $\mathtt{red\text{-}D}^{\mathtt{rrs}}(\mathtt{Equality})$ in which all universals are reduced away leaving us with a purely propositional formula with matrix

$$\underbrace{(\bar{t}_1 \vee \cdots \vee \bar{t}_n)}_{T_n} \wedge \bigwedge_{i=1}^{n} \left[\underbrace{(x_i \vee t_i)}_{A_i} \wedge \underbrace{(\bar{x}_i \vee t_i)}_{B_i} \right]$$

which has a linear size resolution refutation. This refutation is easy to construct: For all $i \in [n]$ perform $\mathtt{res}(A_i, B_i, x_i)$. This will yield the n clauses t_i for $i \in [n]$, all of which can then be reolved with the clause T_n to yield the empty clause.

Therefore, Proposition 3.13 tells us that the `Equality` formula will have polynomial size refutations in $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{ClausePol}, \mathtt{CubePol})$. □

4.4 The `Trapdoor` Formula

The `Trapdoor` formulas were introduced in [BB23a], in order to compare `QCDCL` with the variant with the `NO-RED` policy. The idea is to juxtapose two propositional formulas, one hard for `Res` and one easy for `Res`, and judiciously interject universal and existential variables tying the two together. The tying is done in such a way that QCDCL trails with the `NO-RED` policy can quickly get to the easy part, whereas with `RED` and the ensuing forced propagations, `QCDCL` is trapped into refuting the hard part. Thus for QCDCL proof systems, allowing reductions, which force more

unit propagations in a trail, is not necessarily a good thing.

Formula 4.12. *[*$\texttt{Trapdoor}_n$*, [BB23a]] The* $\texttt{Trapdoor}_n$ *QBF has the prefix* $\exists y_1, \cdots, y_{s_n} \forall w \exists t \exists x_1, \cdots, x_{s_n} \forall u$*, where* s_n *is the number or variables in the propositional pigeonhole principle* $\texttt{PHP}_n^{n+1}$*, and the following matrix:*

$$\begin{aligned} & & \texttt{PHP}_n^{n+1}(x_1, \cdots, x_{s_n}) \\ \textit{for } \ i \in [s_n]: & & \bar{y}_i \vee x_i \vee u \ , \ y_i \vee \bar{x}_i \vee u \\ \textit{for } \ i \in [s_n]: & \ y_i \vee w \vee t \ , \ y_i \vee w \vee \bar{t} \ , \ \bar{y}_i \vee w \vee t \ , \ \bar{y}_i \vee w \vee \bar{t} \end{aligned}$$

Let us note the dependency schemes for this formula

Proposition 4.13. *For the* $\texttt{Trapdoor}$ *formulas* $\texttt{D}^{\texttt{std}}$ *is a non-empty proper subset of* $\texttt{D}^{\texttt{trv}}$*, while* $\texttt{D}^{\texttt{rrs}}$ *is empty. Specifically,*

$$\texttt{D}^{\texttt{std}}(\texttt{Trapdoor}) = \{(w, t)\}$$

$$\texttt{D}^{\texttt{rrs}}(\texttt{Trapdoor}) = \emptyset$$

Proof. For $\texttt{D}^{\texttt{std}}$ clearly (w, t) are connected by virtue of being in the same clause. but none of the x_i's can be connected to w since the only possible connecting variable t does not appear in any clause with x. While $\texttt{D}^{\texttt{rrs}} = \emptyset$ obviously since no universal varaible appears in both polarities in the formula. □

First let us revisit some known and some easy to see results about the hardness of these formulas when QCDCL and dependency schemes do not mix.

Lemma 4.14. *The* $\texttt{Trapdoor}$ *formula have*

- *polynomial size* Q-Res QU-Res, LDQ-Res $\texttt{Q}(\texttt{D}^{\texttt{rrs}})$-Res *refutations.*
- *but require exponential size refutations in* $\texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{trv}}, \texttt{No-Cube})$.

Proof. These results were shown explicitly in [BB23a].

The polysize resolution refutations come from resolving for any chosen i, the 4 y_i, w, t clauses. The exponetial size refutation in QCDCL comes from the fact that any LEV-ORD trail of the refutation would start with deciding y_i's, and each decision would propagate a corresponding x_i in the same polarity. Therefore, every trail would propagate all x_i's before w could be decided. Now, notice that assigning values to all x_i's gives us a conflict in PHP clauses since it is unsatisfiable, therefore each trail raises a conflict in the PHP clauses, and refuting Trapdoor reduces to refuting PHP which is known to require exponential size. □

This style or technique where reduction forces propagations in the trail comes in handy and is an important tool in the future where these reductions force the trail down a "trap" which it could otherwise have avoided.

Now, the question arises, what happens when dependency schemes are introduced?

In the case of preprocessing it is easy to see that it reduces to an easy propositional formula and hence,

Lemma 4.15. *For* $\texttt{CubePol} \in \{\texttt{No-Cube}, \texttt{Cube-LD}, \texttt{Cube-D}^{\texttt{rrs}}, \texttt{Cube-D}^{\texttt{std}}\}$, *and* $\texttt{ClausePol} \in \{\texttt{D}^{\texttt{trv}}, \texttt{D}^{\texttt{std}}.\texttt{D}^{\texttt{rrs}}\}$ *the* Trapdoor *formulas have polynomial size refutations in* $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{ClausePol}, \texttt{CubePol})$.

Proof. Since $\texttt{D}^{\texttt{rrs}}(\texttt{Trapdoor}) = \emptyset$, preprocessing the formulas gives us $\texttt{red-D}^{\texttt{rrs}}(\texttt{Trapdoor})$ in which all universals are reduced away leaving us with a purely propositional formula with matrix

$$\begin{aligned} & \texttt{PHP}_n^{n+1}(x_1, \cdots, x_{s_n}) \\ \text{for } i \in [s_n]: \quad & \bar{y}_i \vee x_i \ , \ y_i \vee \bar{x}_i \\ \text{for } i \in [s_n]: \quad & y_i \vee t \ , \ y_i \vee \bar{t} \ , \ \bar{y}_i \vee t \ , \ \bar{y}_i \vee \bar{t} \end{aligned}$$

which has a polynomial size resolution refutation, which can be constructed by resolving for any $i \in [s_n]$, the 4 y_i, t clauses corresponding to it. Therefore, Proposition 3.13 tells us that the $\texttt{Trapdoor}$ formula will have polynomial size refutations in $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{ClausePol}, \texttt{CubePol})$. □

When $\texttt{D}^{\texttt{rrs}}$ is not used in preprocessing, but used exclusively in propagation and learning, the $\texttt{Trapdoor}$ formulas have a constant size refutations.

Lemma 4.16. *For* $\texttt{CubePol} \in \{\texttt{No-Cube}, \texttt{Cube-LD}, \texttt{Cube-D}^{\texttt{rrs}}, \texttt{Cube-D}^{\texttt{std}}\}$, *the* $\texttt{Trapdoor}_n$ *formulas have* $O(1)$*-size refutation in* $\texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{rrs}}, \texttt{CubePol})$.

Proof. Due to Observation 3.10, it is enough to show that the $\texttt{Trapdoor}_n$ formulas have $O(1)$ refutations in $\texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{rrs}}, \texttt{No-Cube})$.

As seen before, $\texttt{D}^{\texttt{rrs}}(\texttt{Trapdoor}_n) = \emptyset$. Using this fact we can construct a $\texttt{QCDCL}(\texttt{D}^{\texttt{rrs}})$ refutation consisting of two trails T_1 and T_2. The first trail decides y_1 and learns $\bar{y}_1$, the second trail has no decisions. More precisely, the first trail is

$$T_1 := (\mathbf{y_1}, t, \square)$$

with $\texttt{red-D}^{\texttt{rrs}}(\texttt{ante}(\square)) = \texttt{red-D}^{\texttt{rrs}}(\bar{y}_1 \vee w \vee \bar{t}) = (\bar{y}_1 \vee \bar{t})$, and $\texttt{red-D}^{\texttt{rrs}}(\texttt{ante}(t)) = \texttt{red-D}^{\texttt{rrs}}(\bar{y}_1 \vee w \vee t) = (\bar{y}_1 \vee t)$. This allows us to learn the clause $(\bar{y}_1)$. The second trail begins by propagating $\bar{y}_1$.

$$T_2 := (\bar{y}_1, t, \square)$$

Here, $\texttt{red-D}^{\texttt{rrs}}(\texttt{ante}(\square)) = \texttt{red-D}^{\texttt{rrs}}(y_1 \vee w \vee \bar{t}) = y_1 \vee \bar{t}$, $\texttt{red-D}^{\texttt{rrs}}(\texttt{ante}(t)) = \texttt{red-D}^{\texttt{rrs}}(y_1 \vee w \vee t) = y_1 \vee t$ and $\texttt{ante}(\bar{y}_1) = \bar{y}_1$ Therefore, we can learn the empty clause $(\square)$, completing the refutation. □

The final observation we make is that although $\texttt{D}^{\texttt{rrs}}$ "helps" in reducing the hardness

of Trapdoor formulas in QCDCL the $\mathtt{D}^{\mathtt{std}}$ scheme is of no help and behaves equivalent to the $\mathtt{D}^{\mathtt{trv}}$ scheme.

Lemma 4.17. *For* $\mathtt{CubePol} \in \{\mathtt{No\text{-}Cube}, \mathtt{Cube\text{-}LD}, \mathtt{Cube\text{-}D}^{\mathtt{rrs}}, \mathtt{Cube\text{-}D}^{\mathtt{std}}\}$, *the* $\mathtt{Trapdoor}_n$ *formulas require exponential size refutation in* $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{std}}, \mathtt{CubePol})$.

Proof. Now, since the Trapdoor formula have an unsatisfiable matrix, by Observation 3.11 it is sufficient to show hardness for $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{std}}, \mathtt{No\text{-}Cube})$.

This proof is exactly the same as the proof of hardness for Trapdoor for $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{No\text{-}Cube})$, this comes from the fact that again during propagation the u is reduced when y_i is decided propagating x_i, but since $(w, t) \in \mathtt{D}^{\mathtt{std}}(\mathtt{Trapdoor})$, the w is not reduced.

Thus, each trail in a refutation must decide all y_i variables first due to LEV-ORD propagating all x_i variables and thus falling into the "trap" of refuting PHP before the w, t can even play a role. □

4.5 The TwinEq Formula

The TwinEq formulas were introduced in [BPB24] to show hardness in the system $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{No\text{-}Cube})$. They are formally defined as follows:

Formula 4.18. *[*$\mathtt{TwinEq}_n$*, [BPB24]] The* $\mathtt{TwinEq}_n$ *formula has the prefix* $\exists x_1 \cdots x_n \forall u_1 \cdots u_n, w_1, \cdots, w_n \exists t_1 \cdots t_n$ *and the PCNF matrix*

$$\underbrace{(\bar{t}_1 \vee \cdots \vee \bar{t}_n)}_{T_n} \wedge \bigwedge_{i=1}^{n} \left[\underbrace{(x_i \vee u_i \vee t_i)}_{A_i} \wedge \underbrace{(\bar{x}_i \vee \bar{u}_i \vee t_i)}_{B_i} \right] \wedge \bigwedge_{i=1}^{n} \left[\underbrace{(x_i \vee w_i \vee t_i)}_{C_i} \wedge \underbrace{(\bar{x}_i \vee \bar{w}_i \vee t_i)}_{D_i} \right]$$

Proposition 4.19. *For the* TwinEq *formulas* $\mathtt{D}^{\mathtt{trv}}$ *and* $\mathtt{D}^{\mathtt{std}}$ *are equivalent, but for*

$\mathtt{D}^{\mathtt{rrs}}$, *the* t *variables are independent of the* u *variables. i.e.*

$$\mathtt{D}^{\mathtt{trv}}(\mathtt{TwinEq}) = \mathtt{D}^{\mathtt{std}}(\mathtt{TwinEq})$$

$$\mathtt{D}^{\mathtt{rrs}}(\mathtt{TwinEq}) = \emptyset$$

Proof. The arguments are exactly the same as for the `Equality` formulas as the 3-clauses here are just a duplicate of the ones in `Equality`.

In the case for $\mathtt{D}^{\mathtt{std}}$, for each (u_i, t_j) ,(w_i, t_j) pair if $i = j$ they are in the same clause, while if $i \neq j$, there is a path (u_i, t_i, t_j) , (w_i, t_i, t_j)between them, thus $(u_i, t_j), (w_i, t_j) \in \mathtt{D}^{\mathtt{std}}$ for all i, j.

Now in the case for $\mathtt{D}^{\mathtt{rrs}}$, there are paths between $u_i, \bar{t}_j$ and $\bar{u}_i, \bar{t}_j$, but no paths from u_i, t_j or $\bar{u}_i, t_j$ and there are paths between $w_i, \bar{t}_j$ and $\bar{w}_i, \bar{t}_j$, but no paths from w_i, t_j or $\bar{w}_i, t_j$therefore $\mathtt{D}^{\mathtt{rrs}} = \emptyset$. □

The already known results for this formula can be summarised as follows

Lemma 4.20. *The* `TwinEq` *formulas require exponential size refutations in* `Q-Res`*,* `QU-Res`*, but have polynomial size refutations in* $\mathtt{Q}(\mathtt{D}^{\mathtt{rrs}})$-`Res`*.*

Proof. For the same reason as for `Equality` (the size-cost-capacity theorem from [BBH19]), they are hard for `QU-Res` and `Q-Res`. Wheras $\mathtt{D}^{\mathtt{rrs}}$ is empty giving us a short $\mathtt{Q}(\mathtt{D}^{\mathtt{rrs}})$-`Res` refutation. □

When either $\mathtt{D}^{\mathtt{trv}}$ or $\mathtt{D}^{\mathtt{std}}$ are the dependency schemes these formulas are hard for `QCDCL` systems even with cube-learning.

Lemma 4.21. *For* `ClausePol` $\in \{\mathtt{D}^{\mathtt{trv}}, \mathtt{D}^{\mathtt{std}}\}$ *and* `CubePol` $\in \{\mathtt{No\text{-}Cube}, \mathtt{Cube\text{-}LD}, \mathtt{Cube\text{-}D}^{\mathtt{std}}\}$*, the* `TwinEq` *formulas require exponential size refutations in* $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{ClausePol}, \mathtt{CubePol})$*.*

Proof. These formulas were shown to require exponential size refutations in $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{Cube\text{-}LD})$ in [BPB24]. Therefore, using the fact that $\mathtt{D}^{\mathtt{std}} = \mathtt{D}^{\mathtt{trv}}$ for these formulas and Observation 3.10, these formulas require exponential refutation in all the aforementioned systems. □

However, like the Equality formulas, under the usage of $\mathtt{D}^{\mathtt{rrs}}$ they become easy to refute.

Lemma 4.22. *For* $\mathtt{CubePol} \in \{\mathtt{No\text{-}Cube}, \mathtt{Cube\text{-}LD}, \mathtt{Cube\text{-}D}^{\mathtt{rrs}}\}$*, the* TwinEq *fomulas have polynomial size refutations in* $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{CubePol})$.

Proof. Since, $\mathtt{D}^{\mathtt{rrs}}(\mathtt{TwinEq}) = \emptyset$ and $\mathtt{red\text{-}D}^{\mathtt{rrs}}(\mathtt{TwinEq}) = \mathtt{red\text{-}D}^{\mathtt{rrs}}(\mathtt{Equality})$, the refutation for the Equality formulas work exactly the same as a refutation for the TwinEq formulas. □

Lemma 4.23. *For* $\mathtt{ClausePol} \in \{\mathtt{D}^{\mathtt{trv}}, \mathtt{D}^{\mathtt{std}}, \mathtt{D}^{\mathtt{rrs}}\}$ *and* $\mathtt{CubePol} \in \{\mathtt{No\text{-}Cube}, \mathtt{Cube\text{-}LD}, \mathtt{Cube\text{-}D}^{\mathtt{std}}, \mathtt{Cube\text{-}D}^{\mathtt{rrs}}\}$*, the* TwinEq *formulas have polynomial size refutations in* $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{ClausePol}, \mathtt{CubePol})$.

Proof. $\mathtt{D}^{\mathtt{rrs}}(\mathtt{TwinEq}) = \emptyset$. Therefore, $\mathtt{red\text{-}D}^{\mathtt{rrs}}(\mathtt{TwinEq})$ is a propositional formula, which is easy to refute in resolution. Hence, it has polynomial size refutations in all the aforementioned systems. □

4.6 The Dep-Trap Formula

From the previous introduced formulas it may appear that adding $\mathtt{D}^{\mathtt{rrs}}$ to QCDCL systems only adds to its strength and suggests that the addition of the dependency scheme gives us strictly stronger proof system as with Q-Res. However, this is not the case; QCDCL's are tricky proof systems where the extra power in reductions added due to the dependency scheme could prove a detriment. This motivates the idea of

designing a formula where adding the dependency scheme enables reductions that send the refutation down a trap into which the seemingly weaker systems do not fall. Based on this idea, we introduce the family Dep-Trap which is a slight modification of the Trapdoor family, and is defined as follows:

Formula 4.24. *[Dep-Trap$_n$] The* Dep-Trap$_n$ *formula has the prefix* $\exists y_1, \cdots, y_{s_n} \forall w \exists t \forall u \exists x_1, \cdots, x_{s_n}$*, and the matrix is as given below.*

$$\begin{array}{lll} & & \mathtt{PHP}_n^{n+1}(x_1, \cdots, x_{s_n}) \\ \textit{for } \ i \in [s_n]: & & \bar{y}_i \vee u \vee x_i \ , \ y_i \vee u \vee \bar{x}_i \\ \textit{for } \ i \in [s_n]: & y_i \vee w \vee t \ , \ y_i \vee w \vee \bar{t} \ , \ \bar{y}_i \vee w \vee t \ , \ \bar{y}_i \vee w \vee \bar{t} & \\ & & \bar{w} \vee \bar{t} \end{array}$$

Note that there are two differences from the Trapdoor formulas. Firstly, the universal variable u which was earlier quantified at the end is now quantified just before the existential variables $x_1, \cdots, x_{s_n}$. Secondly, there is an additional clause $\bar{w} \vee \bar{t}$. We will see that these serve a dual purpose: the shifting of the position of u stops the standard QCDCL i.e. $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{No\text{-}Cube})$ from falling into the trap, making the formulas easy to refute in the system, while the additional clause prevents the $\mathtt{D}^{\mathtt{rrs}}$ scheme from bypassing the trap as it used to in the Trapdoor formulas, since now the $\mathtt{D}^{\mathtt{rrs}}$ set is non-empty.

The dependency scheme for these formulas are as follows

Proposition 4.25. *For the* Dep-Trap *formulae,* $\mathtt{D}^{\mathtt{std}}$ *and* $\mathtt{D}^{\mathtt{rrs}}$ *are a non-empty subset of* $\mathtt{D}^{\mathtt{trv}}$ *as follows.*

$$\mathtt{D}^{\mathtt{std}}(\mathtt{Trapdoor}) = \{(u, x_i) | i \in [s_n]\} \cup \{(w, t)\}$$

$$\mathtt{D}^{\mathtt{rrs}}(\mathtt{Trapdoor}) = \{(w, t)\}$$

Proof. For $\mathtt{D}^{\mathtt{std}}$, the paths exist trivially between u and x_i's and between w and t.

Additionally there is no way to connect the w to the x's via t.

For $\mathtt{D}^{\mathtt{trv}}$, u appears only in one polarity, whereas (w, t) is in $\mathtt{D}^{\mathtt{rrs}}$ via two trivial paths. □

Now, to understand the hardness of these formulas in various systems

Lemma 4.26. *The* Dep-Trap *formulas have constant-size refutations in* Q-Res*,* QU-Res*,* LDQ-Res *and* $\mathtt{Q}(\mathtt{D}^{\mathtt{rrs}})$-Res*.*

Proof. As all the other systems simulate Q-Res it is enough to show that the formulas have constant size Q-Res refutations. This can be seen in the same manner as for the Trapdoor formulas. For any fixed i, take the 4 clauses corresponding to y_i, w, t and resolve them to yield the empty clause. □

Now, in the case where dependency schemes become involved we see that if pre-processing is not used, then using either $\mathtt{D}^{\mathtt{trv}}$ or $\mathtt{D}^{\mathtt{std}}$ we can obtain linear-sized refutations irrespective of cube policy.

Lemma 4.27. *For* $\mathtt{CubePol} \in \{\mathtt{No\text{-}Cube}, \mathtt{Cube\text{-}LD}, \mathtt{Cube\text{-}D}^{\mathtt{std}}, \mathtt{Cube\text{-}D}^{\mathtt{rrs}}\}$ *and* $\mathtt{ClausePol} \in \{\mathtt{D}^{\mathtt{trv}}, \mathtt{D}^{\mathtt{std}}\}$*, the* Dep-Trap *formulas have polynomial-size refutations in* $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{ClausePol}, \mathtt{CubePol})$*.*

Proof. By Observation 3.10, it is enough to prove the statement for $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{ClausePol}, \mathtt{No\text{-}Cube})$ where $\mathtt{ClausePol} \in \{\mathtt{D}^{\mathtt{trv}}, \mathtt{D}^{\mathtt{std}}\}$.

We will do so by constructing a complete refutation for both cases parallely and observe that exactly the same refutation holds for both scenarios. The first trail decides all y variables positively, decides w negatively, and then propagates t and a conflict. Unlike Trapdoor we don't propagate an x after an y decision, since the u cannot be reduced either in $\mathtt{D}^{\mathtt{trv}}$ or $\mathtt{D}^{\mathtt{std}}$ and so blocks unit propagation.

$$T_1 := (\mathbf{y_1}; \mathbf{y_2}; \cdots; \mathbf{y_{s_n}}; \bar{\mathbf{w}}, t, \square)$$

where $\texttt{ante}(\square) = (\bar{y}_1 \vee w \vee \bar{t})$ and $\texttt{ante}(t) = (\bar{y}_1 \vee w \vee t)$. Hence the set of learnable clauses for this trail is $L_{T_1} = ((\bar{y}_1 \vee w \vee \bar{t}), (\bar{y}_1))$; allowing us to learn the clause $(\bar{y}_1)$. Now the second trail propagates $(\bar{y}_1)$ followed by remaining decisions as before.

$$T_2 := (\bar{y}_1; \mathbf{y_2}; \cdots; \mathbf{y_{s_n}}; \bar{\mathbf{w}}, t, \square)$$

where $\texttt{ante}(\square) = (y_1 \vee w \vee \bar{t})$, $\texttt{ante}(t) = (y_1 \vee w \vee t)$, and $\texttt{ante}(\bar{y}_1) = \bar{y}_1$. Therefore, the set of learnable clauses for this trail is $L_{T_2} = ((y_1 \vee w \vee \bar{t}), (y_1), (\square))$ which allows us to learn the empty clause $(\square)$, completing the refutation. The refutation size is $O(s_n)$, which is linear in the formula size. □

Next we show that these formulas are hard to refute in QCDCL variants that use $\texttt{D}^{\texttt{rrs}}$ in any manner.

Lemma 4.28. *For* $\texttt{CubePol} \in \{\texttt{No-Cube}, \texttt{Cube-LD}, \texttt{Cube-D}^{\texttt{std}}, \texttt{Cube-D}^{\texttt{rrs}}\}$ *refutations of the* Dep-Trap *formulas in* $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{trv}}, \texttt{CubePol})$, $\texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{rrs}}, \texttt{CubePol})$, *and* $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{rrs}}, \texttt{CubePol})$ *require exponential size.*

Proof. Since, the formula Dep-Trap contains PHP embedded in it, it has an unsatisfiable matrix. Therefore, by Observation 3.11 it is enugh to show the above statement for $\texttt{CubePol} = \texttt{No-Cube}$.

First observe, $\texttt{D}^{\texttt{rrs}}_{\forall}(\texttt{Dep-Trap}_n) = \{(w, t)\}$. This means that the universal variable w cannot be reduced from any axiom clause it appears in as they all also contain the variable t, whereas the universal variable u can be reduced from the axiom clauses as no existential variable depends on it.

Let us first see hardness for $\texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{rrs}}, \texttt{No-Cube})$. Since u can be reduced but not w, the proof of hardness of Trapdoor in QCDCL from [BB23a] carries over as is to hardness of Dep-Trap in this system; the decisions on the y variables propagate x

literals, sending the trails down the PHP trap. The additional clause cannot change the course of such trails, because its variables appear after the y part in the prefix and decisions are required to be level-ordered.

Next consider the other two variants. Preprocessing yields the matrix –

$$\begin{array}{lll} & & \mathtt{PHP}_n^{n+1}(x_1, \cdots, x_{s_n}) \\ \text{for } \; i \in [s_n] : & & \bar{y}_i \vee x_i \;,\; y_i \vee \bar{x}_i \\ \text{for } \; i \in [s_n] : & y_i \vee w \vee t \;,\; y_i \vee w \vee \bar{t} \;,\; \bar{y}_i \vee w \vee t \;,\; \bar{y}_i \vee w \vee \bar{t} & \\ & & \bar{w} \vee \bar{t} \end{array}$$

Now all trails in a refutation of this formula must start with deciding the y_i's; these decisions propagate the x_i's; and w cannot be reduced from the clauses it exists in (irrespective of whether ClausePol is $\mathtt{D}^{\mathtt{trv}}$ or $\mathtt{D}^{\mathtt{rrs}}$) nor decided before all the y_i's are decided. Again, the proof of hardness of Trapdoor in QCDCL from [BB23a] lifts to hardness of Dep-Trap in $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{No\text{-}Cube})$ and $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{No\text{-}Cube})$. □

Thus, we see that these formulas are easy for QCDCL with $\mathtt{D}^{\mathtt{trv}}, \mathtt{D}^{\mathtt{std}}$, but become hard to refute when $\mathtt{D}^{\mathtt{rrs}}$ is added to the system, demonstrating that allowing more reductions and removing spurious dependencies does not necessarily help for the QCDCL system. In fact a more powerful dependency scheme allowing more reductions in some cases can be detrimental to the system.

4.7 The TwoPHPandCT Formula

The formulas introduced so far seem to suggest that Proposition 3.13, could also extend to include $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{CubePol})$. Which would say that if for a given QBF Φ, $\mathtt{red\text{-}D}^{\mathtt{rrs}}(\Phi)$ is an easy-to-refute (in resolution) propositional formula, Φ is

also easy to refute in $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{CubePol})$. We show now that this is not the case.

The motivation for defining the following formula also comes from the Trapdoor formulas, using the propositional hardness of PHP and the "easiness" of the (negation of) complete tautology on two variables. The added new element is the use of two disjoint copies of the hard part.

Formula 4.29. *[$\mathtt{TwoPHPandCT}_n$] The $\mathtt{TwoPHPandCT}_n$ formulas has the prefix, $\mathcal{Q} = \forall u \exists x_1 \cdots x_{s_n}\ \exists y_1 \cdots y_{s_n}\ \forall v \exists z_1, z_2$ and the matrix*

$$u \vee \mathtt{PHP}(x_1, \cdots, x_{s_n})$$
$$\bar{u} \vee \mathtt{PHP}(y_1, \cdots, y_{s_n})$$
$$v \vee z_1 \vee z_2\ ,\ v \vee \bar{z}_1 \vee z_2\ ,\ v \vee z_1 \vee \bar{z}_2\ ,\ v \vee \bar{z}_1 \vee \bar{z}_2$$

Now, if we look at the dependency scheme for the formula:

Proposition 4.30. *For the $\mathtt{TwoPHPandCT}$ formulae, $\mathtt{D}^{\mathtt{std}}$ is a proper subset of $\mathtt{D}^{\mathtt{trv}}$, while $\mathtt{D}^{\mathtt{rrs}}$ is empty. Specifically,*

$$\mathtt{D}^{\mathtt{std}}(\mathtt{TwoPHPandCT}) = \{(u, x_i), (u, y_i) | \forall i \in [s_n]\} \cup \{(v, z_1), (v, z_2)\}$$

$$\mathtt{D}^{\mathtt{rrs}}(\mathtt{TwoPHPandCT}) = \emptyset$$

Proof. In the case for $\mathtt{D}^{\mathtt{std}}$, the (u, x), (u, y) pairs and $(v, z_1), (v, z_2)$ are trivially connected. Whereas, there is no way to connect u to the z variables.

For $\mathtt{D}^{\mathtt{rrs}}$, the v appears in only one polarity while u and $\bar{u}$ appear in clauses with disjoint set of existential variables, hence there cannot be a resolution path from $\bar{v}$ to any literal, nor can there be a resolution path from both u and $\bar{u}$ to opposing literals of the same variable. Thus, $\mathtt{D}^{\mathtt{rrs}}$ is empty. □

First we consider these formulas in resolution based systems.

Lemma 4.31. *The* TwoPHPandCT *formulas have constant-size refutations in* Q-Res*,* QU-Res*,* LDQ-Res *and* Q($\mathtt{D}^{\mathtt{rrs}}$)-Res*.*

Proof. As all the other systems simulate Q-Res it is enough to show that the formulas have constant size Q-Res refutations. Observe that these formulas are easy to refute in Q-Res, using the four z_1, z_2 clauses. □

In the case of dependency based QCDCL systems, it is easy to see that these formulas have short refutations whenever we have a system preprocessed by $\mathtt{D}^{\mathtt{rrs}}$.

Lemma 4.32. *For* $\mathtt{CubePol} \in \{\mathtt{No\text{-}Cube}, \mathtt{Cube\text{-}LD}, \mathtt{Cube\text{-}D}^{\mathtt{rrs}}, \mathtt{Cube\text{-}D}^{\mathtt{std}}\}$*, and* $\mathtt{ClausePol} \in \{\mathtt{D}^{\mathtt{trv}}, \mathtt{D}^{\mathtt{std}}.\mathtt{D}^{\mathtt{rrs}}\}$ *the* TwoPHPandCT *formulas have polynomial size refutations in* $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{ClausePol}, \mathtt{CubePol})$.

Proof. $\mathtt{D}^{\mathtt{rrs}}(\mathtt{TwoPHPandCT}) = \emptyset$ and $\mathtt{red\text{-}D}^{\mathtt{rrs}}(\mathtt{TwoPHPandCT})$ is the propositional formula

$$\mathtt{PHP}(x_1, \cdots, x_{s_n})$$
$$\mathtt{PHP}(y_1, \cdots, y_{s_n})$$
$$z_1 \vee z_2 \ , \ \bar{z}_1 \vee z_2 \ , \ z_1 \vee \bar{z}_2 \ , \ \bar{z}_1 \vee \bar{z}_2$$

This formula is easy to refute in Res using the z_1, z_2 clauses; hence, the original QBFs are easy to refute in $\mathtt{D}^{\mathtt{rrs}}$ preprocessed QCDCL systems. □

But if preprocessing is not present these formulas require exponential size refutations whatever (if any) be the dependency scheme used for propagation and learning irrespective of cube learning.

Lemma 4.33. *For* $\mathtt{CubePol} \in \{\mathtt{No\text{-}Cube}, \mathtt{Cube\text{-}LD}, \mathtt{Cube\text{-}D}^{\mathtt{rrs}}, \mathtt{Cube\text{-}D}^{\mathtt{std}}\}$*, and* $\mathtt{ClausePol} \in \{\mathtt{D}^{\mathtt{trv}}, \mathtt{D}^{\mathtt{std}}.\mathtt{D}^{\mathtt{rrs}}\}$*, the QBF formulas* $\mathtt{TwoPHPandCT}_n$ *require exponential size refutations in* $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{ClausePol}, \mathtt{CubePol})$.

Proof. Observe that the matrix of the formula is unsatisfiable: since exactly one of u or $\bar{u}$ can be true, and since all PHP clauses cannot be satisfied, one set of them

must be falsified. Therefore, by Observation 3.11 it is enough to prove the case for $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{ClausePol}, \mathtt{No\text{-}Cube})$.

Now, we consider refutations in these systems, firstly, none of the systems allow for any preprocessing. Hence the first decision in each of these three systems must be on u, which allows for no propagations in any case. And depending on the choice of u, the next set of decisions are either all on x variables or all on y variables. The variable v could be dropped during unit propagation in the $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{No\text{-}Cube})$ system, but neither z_1 or z_2 could be decided or propagated before all the y or x variables are decided/propagated. Therefore, all conflicts these trails hit come directly from the PHP clauses. Thus refuting TwoPHPandCT in any of these systems is equivalent to refuting PHP in CDCL, requiring exponential size. □

This formula importantly highlights that even if the $\mathtt{D}^{\mathtt{rrs}}$ reduction of a formula is an easy-to-refute propositional formula, a QCDCL system using $\mathtt{D}^{\mathtt{rrs}}$ may only be able to produce an exponential size refutation.

4.8 The RRSTrapEq Formula

The next family of formulas are obtained by making a slight modification to the Equality formulas. The motivation to define such formulas comes from trying to ascertain whether after preprocessing with $\mathtt{D}^{\mathtt{rrs}}$, does allowing reductions using $\mathtt{D}^{\mathtt{rrs}}$ for unit propagation add any power over $\mathtt{D}^{\mathtt{trv}}$ universal reductions.

Formula 4.34. *[$\mathtt{RRSTrapEq}_n$] The $\mathtt{RRSTrapEq}_n$ formula has the prefix $\exists a \exists x_1 \cdots x_n \forall u_1 \cdots u_n \exists t_1 \cdots t_n \exists b$ and the PCNF matrix as below:*

$$\underbrace{(\bar{t}_1 \vee \cdots \vee \bar{t}_n)}_{T_n} \wedge \bigwedge_{i=1}^{n} \left[\underbrace{(x_i \vee u_i \vee t_i \vee b)}_{A_i} \wedge \underbrace{(\bar{x}_i \vee \bar{u}_i \vee t_i \vee b)}_{B_i} \right] \wedge \bigwedge_{i=1}^{n} \underbrace{(u_i \vee \bar{b})}_{C_i} \wedge (a \vee \bar{b}) \wedge (\bar{a} \vee \bar{b})$$

The prefix has the variables of $\texttt{Equality}$ sandwiched between a and b, b is added to each x, u, t clause and there are a few extra clauses. The dependency schemes for this formula are as follows:

Proposition 4.35.

$$\texttt{D}^{\texttt{std}}(\texttt{RRSTrapEq}) = \texttt{D}^{\texttt{trv}}(\texttt{RRSTrapEq})$$

$$\texttt{D}^{\texttt{rrs}}(\texttt{RRSTrapEq}) = \{(u_i, b) | \forall i \in [n]\}$$

Proof. That $\texttt{D}^{\texttt{std}}$ equals $\texttt{D}^{\texttt{trv}}$ is easy to see via trivial connection from each u_i to t_i and b and a connection form each u_i to t_j $(j \neq i)$ via t_i.

For $\texttt{D}^{\texttt{rrs}}$, $(u_i, t_j) \notin \texttt{D}^{\texttt{rrs}}$ due to the same reason as $\texttt{Equality}$ formulas and $(u_i, b) \in \texttt{D}^{\texttt{rrs}}$ via trivial paths in clauses B_i and C_i. □

The underlying idea in the formulation is that unlike $\texttt{D}^{\texttt{rrs}}(\texttt{Equality})$ which is empty, the additional C_i clauses act in a manner so as to establish that $\texttt{red-D}^{\texttt{rrs}}(\texttt{RRSTrapEq}) = \texttt{RRSTrapEq}$. and hence, preprocessing by $\texttt{D}^{\texttt{rrs}}$ has no effect on the formula.

First let us look at refuting this formula without preprocessing, we see that it is easy to do so when we propagate using $\texttt{D}^{\texttt{rrs}}$, but requires exponential size when propagating using $\texttt{D}^{\texttt{std}}$ or $\texttt{D}^{\texttt{trv}}$.

Lemma 4.36. *For* $\texttt{ClausePol} \in \{\texttt{D}^{\texttt{std}}, \texttt{D}^{\texttt{trv}}\}$*, the* $\texttt{RRSTrapEq}$ *formulas require exponential size refutations in* $\texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{ClausePol}, \texttt{No-Cube})$.

Proof. Since $\texttt{D}^{\texttt{std}} = \texttt{D}^{\texttt{trv}}$, therefore for $\texttt{ClausePol} \in \{\texttt{D}^{\texttt{trv}}, \texttt{D}^{\texttt{std}}\}$ the refutational hardness and in fact the refutation for these two formulas will be exactly the same for the two systems.

Now, if you consider any trail $\mathcal{T}$ of a refutation, the first decision must be on the variable a, and irrespective of the manner of that decision we propagate $\bar{b}$, which satisfies all the C_i clauses and removes the b literal from the A_i and B_i clauses. The remaining formula at this point is exactly the `Equality` formula, i.e. $\texttt{RRSTrapEq}|_{a=*,b=0} = \texttt{Equality}$, and hence refuting the `RRSTrapEq` formulas is equivalent to refuting the `Equality` formulas in $\texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{trv}}, \texttt{No-Cube})$ which we know requires exponential size (Lemma 4.8). □

Lemma 4.37. *For* $\texttt{CubePol} \in \{\texttt{No-Cube}, \texttt{Cube-LD}, \texttt{Cube-D}^{\texttt{trv}}, \texttt{Cube-D}^{\texttt{std}}, \texttt{Cube-D}^{\texttt{rrs}}\}$ *the* `RRSTrapEq` *have polynomial size refutations in* $\texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{rrs}}, \texttt{CubePol})$.

Proof. By Observation 3.10 it is enough to show the statement for $\texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{rrs}}, \texttt{No-Cube})$.

Now, given this system if you consider any trail $\mathcal{T}$ of a refutation, the first decision must be on the variable a, and irrespective of the manner of that decision we propagate $\bar{b}$, which satisfies all the C_i clauses and removes the b literal from the A_i and B_i clauses. The remaining formula at this point is exactly the `Equality` formula, i.e. $\texttt{RRSTrapEq}|_{a=*,b=0} = \texttt{Equality}$, and hence refuting the `RRSTrapEq` formulas is equivalent to refuting the `Equality` formulas in it which has shown to be easy in Lemma 4.10.

Follwing exactly the same process in that proof, prefix every trail in that refuation by $\bar{\mathbf{a}}, \bar{\mathbf{b}}$ and exactly the same argument follows. Here at the end of those trails instead of learning □ we learn a, and then repeating the whole process by starting with the propagating of a and $\bar{b}$ would allow us to learn the empty clause. Hence we would require twice as many trails.

The complete refutation is provided below for the sake of completeness.

Define the following clauses: For $i \in [n]$, $T_i = \bigvee_{j \leq i} \bar{t}_j$; for $i \in [n] \setminus \{1\}$, $L_i = \bar{x}_i \vee T_{i-1} \vee b$ and $R_i = x_i \vee T_{i-1} \vee b$.

We will construct the trails $\mathcal{U}_{n-1}, \mathcal{V}_{n-1}, \mathcal{U}_{n-2}, \mathcal{V}_{n-2}, \cdots, \mathcal{U}_1, \mathcal{V}_1$, and learn clauses $L_{n-1}, R_{n-1}, \cdots L_1, R_1$ corresponding to these trails. The $\mathcal{U}$ trails decide x variables (as many as is possible until conflict) positively; the $\mathcal{V}$ trails decide them negatively.

The initial trail is

$$\mathcal{U}_{n-1} := (\bar{\mathbf{a}}, \bar{b}, \mathbf{x_1}, t_1; \mathbf{x_2}, t_2, ..., \mathbf{x_{n-1}}, t_{n-1}, \bar{t}_n, x_n, \square)$$

and the antecedent clauses are $\texttt{ante}(\bar{b}) = a \vee \bar{b}$, $\texttt{ante}(t_j) = B_j$ for $j \in [n-1]$, $\texttt{ante}(\bar{t}_n) = T_n$, $\texttt{ante}(x_n) = A_n$, and $\texttt{ante}(\square) = B_n$. From these set of clauses we learn the clause $L_{n-1} = \bar{x}_{n-1} \vee T_{n-2} \vee b$.

Restarting, create a symmetric trail to $\mathcal{U}_{n-1}$ flipping each decision:

$$\mathcal{V}_{n-1} := (\bar{\mathbf{x}}_\mathbf{1}, t_1; \bar{\mathbf{x}}_\mathbf{2}, t_2, ..., \bar{\mathbf{x}}_\mathbf{n-1}, t_{n-1}, \bar{t}_n, x_n, \square)$$

where the antecedent clauses are $\texttt{ante}(\bar{b}) = a \vee \bar{b}$, $\texttt{ante}(t_j) = A_j$ for $j \in [n-1]$, $\texttt{ante}(\bar{t}_n) = T_n$, $\texttt{ante}(x_n) = A_n$, and $\texttt{ante}(\square) = B_n$. From these set of clauses we learn the clause $R_{n-1} = x_{n-1} \vee T_{n-2} \vee b$.

We now go down now from $i = n-2$ down to $i = 2$. At stage i, we first construct trail $\mathcal{U}_i$ by deciding x variables positively; we reach a conflict after deciding x_i. The trail and antecedent clauses are as follows:

$$\mathcal{U}_i := (\bar{\mathbf{a}}, \bar{b}, \mathbf{x_1}, t_1; \mathbf{x_2}, t_2, ..., \mathbf{x_i}, t_i, x_{i+1}, \square)$$

with $\texttt{ante}(\bar{b}) = a \vee \bar{b}$, $\texttt{ante}(t_j) = B_j$ for $j \in [i]$, $\texttt{ante}(x_{i+1}) = L_{i+1}$, $\texttt{ante}(\square) = R_{i+1}$. From this we learn the clause L_i.

Next, we create the symmetrical trail by deciding the x variables negatively

$$\mathcal{V}_i := (\bar{\mathbf{a}}, \bar{b}, \bar{\mathbf{x}}_\mathbf{1}, t_1; \bar{\mathbf{x}}_\mathbf{2}, t_2, ..., \bar{\mathbf{x}}_\mathbf{i}, t_i, x_{i+1}, \square)$$

with antecedent clauses $\texttt{ante}(\bar{b}) = a \vee \bar{b}$,$\texttt{ante}(t_j) = A_j$ for $j \in [i]$, $\texttt{ante}(x_{i+1}) = L_{i+1}$, $\texttt{ante}(\square) = R_{i+1}$. From this we learn the clause R_i.

The proof ends with the trails

$$\mathcal{U}_1 = (\bar{\mathbf{a}}, \bar{b}, \mathbf{x_1}, t_1, x_2, \square)$$

with antecedents $\texttt{ante}(\bar{b}) = a \vee \bar{b}$, $\texttt{ante}(t_1) = B_1$, $\texttt{ante}(x_2) = L_2$, $\texttt{ante}(\square) = R_2$, from which we learn $L_1 = \bar{x}_1$, and and

$$\mathcal{V}_1 = (\bar{\mathbf{a}}, \bar{b}, \bar{x_1}, t_1, x_2, \square)$$

with antecedents $\texttt{ante}(\bar{b}) = a \vee \bar{b}$, $\texttt{ante}(\bar{x}_1) = L_1$, $\texttt{ante}(t_1) = B_1$, $\texttt{ante}(x_2) = L_2$, $\texttt{ante}(\square) = R_2$ From which we learn a.

Now if repeat the same process from the start again this time each trail starts with the propagations $a, \bar{b}$. Following exactly the same steps would give us the final trail as

$$\mathcal{V'}_1 = (a, \bar{b}, \bar{x_1}, t_1, x_2, \square)$$

with antecedents $\texttt{ante}(a) = a\texttt{ante}(\bar{b}) = \bar{a} \vee \bar{b}$, $\texttt{ante}(\bar{x}_1) = L_1$, $\texttt{ante}(t_1) = B_1$, $\texttt{ante}(x_2) = L_2$, $\texttt{ante}(\square) = R_2$ From which we can learn $\square$ completing the refutation.

□

Now, since preprocessing by $\texttt{D}^{\texttt{rrs}}$ does not change the formula at all and the fact that $\texttt{D}^{\texttt{std}} = \texttt{D}^{\texttt{trv}}$, therefore for $\texttt{ClausePol} \in \{\texttt{D}^{\texttt{trv}}, \texttt{D}^{\texttt{std}}\}$ the refutational hardness and in fact the refutation for these two formulas will be exactly the same for the systems $\texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{ClausePol}, \texttt{No-Cube})$ and $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{ClausePol}, \texttt{No-Cube})$, and similarly for the pair $\texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{rrs}}, \texttt{No-Cube})$ and $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{rrs}}, \texttt{No-Cube})$.

Lemma 4.38. *The* RRSTrapEq *formulas*

- *for* $\texttt{ClausePol} \in \{\texttt{D}^{\texttt{trv}}, \texttt{D}^{\texttt{std}}\}$ *require exponential size refutations in* $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{ClausePol}, \texttt{No-Cube})$

- *for* $\texttt{CubePol} \in \{\texttt{Cube-LD}, \texttt{Cube-D}^{\texttt{trv}}, \texttt{Cube-D}^{\texttt{std}}, \texttt{Cube-D}^{\texttt{rrs}}\}$ *have polynomial size refutations in* $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{rrs}}, \texttt{CubePol})$.

Proof. Since, $\texttt{red-D}^{\texttt{rrs}}(\texttt{RRSTrapEq}) = \texttt{RRSTrapEq}$, there is no effect of preprocessing, therefore by Lemmas 4.36 and 4.37 the statement holds. □

4.9 The PreDepTrap Formula

The previous section underlined that preprocessing by $\texttt{D}^{\texttt{rrs}}$ may not necessarily make $\texttt{D}^{\texttt{rrs}}$ during propagation obsolete and/or give an advantage. It is reasonable to believe that at least it won't make things worse. But the following formula shows that this is not the case: preprocessing before allowing a QCDCL system to refute could in fact make the refutation harder.

The construction of the formula is pretty straightforward. It is the disjoint union of the two formulas Dep-Trap and Equality, connected by a universal quantified right at the beginning, appearing in the two sub-formulas in opposite polarities.

Formula 4.39. *[*$\texttt{PreDepTrap}_n$*]* *The* $\texttt{PreDepTrap}_n$ *formula has the prefix* $\forall a\ \exists y_1 \cdots y_{s_n} \forall w \exists t \forall u\ \exists x_1 \cdots x_{s_n}\ \exists p_1 \cdots p_n \forall q_1 \cdots q_n \exists r_1 \cdots r_n$, *and the matrix*

$$a \vee \texttt{Dep-Trap}(y_1, \cdots, y_{s_n}, w, t, u, x_1, \cdots, x_{s_n})$$
$$\bar{a} \vee \texttt{Equality}(p_1, \cdots, p_n, q_1, \cdots, q_n, r_1, \cdots, r_n)$$

(Recall that $a \vee \texttt{Dep-Trap}$ *is the disjunction of* a *with every clause of* Dep-Trap*, and similarly for* $\bar{a} \vee \texttt{Equality}$*.)*

First let us look at the dependency schemes for the formulas

Proposition 4.40. *For the* PreDepTrap *formulas*

$$\mathtt{D}^{\mathtt{std}}(\mathtt{PreDepTrap}) = \mathtt{D}^{\mathtt{std}}(\mathtt{Dep\text{-}Trap}) \cup \mathtt{D}^{\mathtt{std}}(\mathtt{Equality}) \cup \{(a, z) |\ \textit{for any}\ \ z \in \mathtt{var}_{\exists}(\mathtt{PreDepTrap})\}$$

$$\mathtt{D}^{\mathtt{rrs}}(\mathtt{PreDepTrap}) = \mathtt{D}^{\mathtt{rrs}}(\mathtt{Dep\text{-}Trap}) \cup \mathtt{D}^{\mathtt{rrs}}(\mathtt{Equality})$$

Proof. For $\mathtt{D}^{\mathtt{rrs}}$ and $\mathtt{D}^{\mathtt{std}}$ trivially the dependencies in the sub formulas carry over by definition of the new formula. Now in the case for $\mathtt{D}^{\mathtt{rrs}}$, no extra dependency is added because a and $\bar{a}$ appear in clauses with disjoint existential variables. In the case of $\mathtt{D}^{\mathtt{std}}$ however since a appears in every clause in some polarity it is connected to every existential variable of the formula trivially. □

Firstly observe that it is easy to refute in resolution based proof systems. i.e.

Lemma 4.41. *The* PreDepTrap *formulas have constant-size refutations in* Q-Res*,* QU-Res*,* Q($\mathtt{D}^{\mathtt{rrs}}$)-Res*, and* LDQ-Res*.*

Proof. Since, Q-Res is simulated by all the other systems it is enough to show the statemenrt for Q-Res. The short Q-Res refutation is easy to see: from Lemma 4.26 we know that Dep-Trap has a constant-size refutation. Repeating the same process on this formula would let us derive the universal a which can be dropped to complete the refutation for PreDepTrap. □

We now show that these formulas have short refutations in QCDCL systems with and without dependency as long as there is no preprocessing.

Lemma 4.42. *For* $\mathtt{CubePol} \in \{\mathtt{No\text{-}Cube}, \mathtt{Cube\text{-}LD}, \mathtt{Cube\text{-}D}^{\mathtt{std}}, \mathtt{Cube\text{-}D}^{\mathtt{rrs}}\}$ *and* $\mathtt{ClausePol} \in \{\mathtt{D}^{\mathtt{trv}}, \mathtt{D}^{\mathtt{std}}, \mathtt{D}^{\mathtt{rrs}}\}$ *the* PreDepTrap *formulas have a polynomial size refutation in* $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{ClausePol}, \mathtt{CubePol})$*.*

Proof. By Observation 3.10, it is enough to show the above statement for $\texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{ClausePol}, \texttt{No-Cube})$.

Since there is no preprocessing, the first decision of every trail must set the variable a. For a trail that starts with the decision $\bar{a}$, the PreDepTrap formula after the decision $\bar{a}$ reduces exactly to the Dep-Trap formula which by Lemma 4.27 we know is easy to refute in $\texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{trv}}, \texttt{No-Cube})$ and $\texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{std}}, \texttt{No-Cube})$. Therefore, the PreDepTrap formulas are easy to refute in these systems as well. Since, exactly the same derivation would follow through with a leading a in every learnt clause, and can be dropped at the end.

Whereas in the case of the system $\texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{rrs}}, \texttt{No-Cube})$, let every trail start with the decision a, the PreDepTrap formula reduces after this decision exactly to the Equality formulas, which are easy to refute in this system by Lemma 4.10. Again exactly the same derivation would follow through with a leading $\bar{a}$ in every learnt clause, and can be dropped at the end.

This completes the proof. □

Finally, now consider the case when the formula is preprocessed. We show that this makes refutations exponentially long in this system irrespective of whether cube learning is involved.

Lemma 4.43. *For* $\texttt{CubePol} \in \{\texttt{No-Cube}, \texttt{Cube-LD}, \texttt{Cube-D}^{\texttt{std}}, \texttt{Cube-D}^{\texttt{rrs}}\}$ *and* $\texttt{ClausePol} \in \{\texttt{D}^{\texttt{trv}}, \texttt{D}^{\texttt{std}}, \texttt{D}^{\texttt{rrs}}\}$ *the* PreDepTrap *formulas require exponential size refutations in* $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{ClausePol}, \texttt{CubePol})$.

Proof. Let us first consider the system $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{ClausePol}, \texttt{No-Cube})$.

On preprocessing by $\mathtt{D}^{\mathtt{rrs}}$, the formula $\mathtt{red\text{-}D}^{\mathtt{rrs}}(\mathtt{PreDepTrap})$ has the matrix

$$\begin{array}{lll} & & \mathtt{PHP}_n^{n+1}(x_1, \cdots, x_{s_n}) \\ \text{for} \;\; i \in [s_n] : & & \bar{y}_i \vee x_i \;,\; y_i \vee \bar{x}_i \\ \text{for} \;\; i \in [s_n] : & y_i \vee w \vee t \;,\; y_i \vee w \vee \bar{t} \;,\; \bar{y}_i \vee w \vee t \;,\; \bar{y}_i \vee w \vee \bar{t} & \\ & & \bar{w} \vee \bar{t} \\ & & (\bar{r}_1 \vee \cdots \vee \bar{r}_n) \\ \text{for} \;\; i \in [n] & & (p_i \vee r_i) \\ \text{for} \;\; i \in [n] & & (\bar{p}_i \vee r_i) \end{array}$$

At this point, therefore, all trails must start with deciding the y_i's, propagating the x_i's. The only other universal variable w is blocked by t in all clauses where it occurs irrespective of the `ClausePol`. Since we use `LEV-ORD`, it cannot be decided before all the y_i's are decided. Nor can t be propagated before w is decided. Since the p_i's and r_i's also cannot be decided before the y_i's, the trails are led into the trap of refuting `PHP`. The proof of hardness of `Trapdoor` in $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{No\text{-}Cube})$ from [BB23a] carries over exactly as it is to show hardness for $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{ClausePol}, \mathtt{No\text{-}Cube})$.

Now, what happens if cube-learning is allowed ? the formula still requires exponential size refutations since once `PreDepTrap` is preprocessed by $\mathtt{D}^{\mathtt{rrs}}$ to yield $\mathtt{red\text{-}D}^{\mathtt{rrs}}(\mathtt{PreDepTrap})$ the matrix of this formula becomes unsaatisfiable, and hence by Observation 3.11, the hardness in the absence of cube-learning lifts to hardness in the presence of cube-learning. □

4.10 The `PropDep-Trap` Formula

The previous section illustrated an example where having $\mathtt{D}^{\mathtt{rrs}}$ as a preprocessing technique was a detriment to refuting it and in fact it was better to use $\mathtt{D}^{\mathtt{rrs}}$ only in

unit propagation, and not also for preprocessing. This leads to the question — could there be a formula where having $\mathtt{D}^{\mathtt{rrs}}$ only for preprocessing was strictly better than having it for both preprocessing and propagation? Addressing this led to the birth of the following formula which is a slight modification of the Dep-Trap formulas, and which witnesses that the answer is yes. This new formula is constructed in a manner which nullifies preprocessing and like Dep-Trap , use of $\mathtt{D}^{\mathtt{rrs}}$ in propagation and learning is detrimental.

Formula 4.44. *[*$\mathtt{PropDep\text{-}Trap}_n$*,] The* $\mathtt{PropDep\text{-}Trap}_n$ *formulas have the prefix* $\exists s \;\; \exists y_1 \cdots y_{s_n} \forall w \exists t \forall b_1, b_2 \; \exists x_1 \cdots x_{s_n} \; \exists z_1, z_2$ *and the matrix as given below.*

$$\begin{aligned} & & \mathtt{PHP}_n^{n+1}(x_1, \cdots, x_{s_n}) \\ \textit{for} \;\; i \in [s_n] : & & \bar{y}_i \vee b_1 \vee x_i \vee z_1 \;\; , \;\; y_i \vee b_2 \vee \bar{x}_i \vee z_2 \\ & & s \vee w \vee t \;\; , \;\; s \vee w \vee \bar{t} \;\; , \;\; \bar{s} \vee w \vee t \;\; , \;\; \bar{s} \vee w \vee \bar{t} \\ & & \bar{w} \vee \bar{t} \\ & & \bar{b}_1 \vee \bar{z}_1 \;\; , \;\; \bar{b}_2 \vee \bar{z}_2 \;\; , \;\; \bar{z}_1 \;\; , \;\; \bar{z}_2 \end{aligned}$$

First we describe the dependencies for this formula

Proposition 4.45. *For the* PropDep-Trap *formulas*

$$\mathtt{D}^{\mathtt{std}}(\mathtt{PropDep\text{-}Trap}) = \{(w,t) \cup \{(b_i, z_j), (b_i, x_k) | i \in [2], j \in [2], k \in [n]\}\}$$

$$\mathtt{D}^{\mathtt{rrs}}(\mathtt{PropDep\text{-}Trap}) = \{(w,t) \cup \{(b_1, z_1), (b_2, z_2)\}\}$$

Proof. For $\mathtt{D}^{\mathtt{std}}$, there is no way to connect w to the x or z variables, whereas the b variables are connected to them trivially.

For $\mathtt{D}^{\mathtt{rrs}}$, the three pairs in $\mathtt{D}^{\mathtt{rrs}}$ have trivial path-pairs connecting them, whereas there can be no path from the variable b_1 to the literal $\bar{x}_i$ nor from the variable b_2 to the literal x_i. □

Lemma 4.46. *The* PropDep-Trap *formulas are easy to refute in* Q-Res, QU-Res, Q($\mathtt{D}^{\mathtt{rrs}}$)-Res, LDQ-Res.

Proof. First observe that the presence of the four s, w, t clauses make this formula very easy to refute in Q-Res and hence in Q($\mathtt{D}^{\mathtt{rrs}}$)-Res, QU-Resand LDQ-Res.

Next, notice that the formulas are also easy to refute in $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}_{\mathtt{NO\text{-}RED}}$, QCDCL, and $\mathtt{QCDCL}^{\mathtt{cube}}$ because, after the initial propagation of the unit clauses $\bar{z}_1, \bar{z}_2$ in level 0, there are no propagations possible before a w decision; the decisions on s and all the y_i's cause no propagations. A trail that decides s and $\bar{w}$ quickly reaches a conflict and learns $\bar{s}$; a next trail that propagates $\bar{s}$ and decides $\bar{w}$ then learns the empty clause. □

Lemma 4.47. *For* $\mathtt{CubePol} \in \{\mathtt{No\text{-}Cube}, \mathtt{Cube\text{-}LD}, \mathtt{Cube\text{-}D}^{\mathtt{trv}}, \mathtt{Cube\text{-}D}^{\mathtt{std}}, \mathtt{Cube\text{-}D}^{\mathtt{rrs}}\}$ *and* $\mathtt{ClausePol} \in \{\mathtt{D}^{\mathtt{trv}}, \mathtt{D}^{\mathtt{std}}\}$*, the* PropDep-Trap *formulas have a linear size refutation in* $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{ClausePol}, \mathtt{CubePol})$.

Proof. By Observation 3.10, it is enough to show for $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{ClausePol}, \mathtt{No\text{-}Cube})$

After the initial propagation of the unit clauses $\bar{z}_1, \bar{z}_2$ in level 0, there are no propagations possible before a w decision; the decisions on s and all the y_i's cause no propagations. A trail that decides s and $\bar{w}$ quickly reaches a conflict and learns $\bar{s}$; a next trail that propagates $\bar{s}$ and decides $\bar{w}$ then learns the empty clause.

For the sake of completeness here is the complete refutation below.

The first trail would look like

$$\mathcal{T}_\infty := (\bar{z}_1, \bar{z}_2, \mathbf{s}, \mathbf{y_1}, \mathbf{y_2}, \cdots, \mathbf{y_{s_n}}, \bar{w}, t, \square)$$

Here, $\mathtt{ante}(\square) = (\bar{s} \vee w \vee \bar{t})$, $\mathtt{ante}(t) = (\bar{s} \vee w \vee t)$, $\mathtt{ante}(\bar{z}_2) = \bar{z}_2$, $\mathtt{ante}(\bar{z}_1) = z_1$. This allows us to learnt the clause $\bar{s}$. The next trail will look like

$$\mathcal{T}_{\in} := (\bar{z}_1, \bar{z}_2, \bar{s}, \mathbf{y_1}, \mathbf{y_2}, \cdots, \mathbf{y_{s_n}}, \bar{w}, t, \square)$$

Here, $\texttt{ante}(\square) = (s \vee w \vee \bar{t})$, $\texttt{ante}(t) = (s \vee w \vee t)$, $\texttt{ante}(\bar{s}) = \bar{s}$, $\texttt{ante}(\bar{z}_2) = \bar{z}_2$, $\texttt{ante}(\bar{z}_1) = z_1$. From these we can learn the empty clause $\square$ completing the refutation. $\square$

But instead if $\texttt{D}^{\texttt{rrs}}$ is used as the `ClausePol` we require exponential size

Lemma 4.48. *For* $\texttt{CubePol} \in \{\texttt{No-Cube}, \texttt{Cube-LD}, \texttt{Cube-D}^{\texttt{trv}}, \texttt{Cube-D}^{\texttt{std}}, \texttt{Cube-D}^{\texttt{rrs}}\}$, *the* `PropDep-Trap` *formulas require exponential size refutations in* $\texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{rrs}}, \texttt{CubePol})$.

Proof. Since due to the presence of `PHP`, the matrix is unsatisfiable, by Observation 3.11, it is enough to show the result for $\texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{rrs}}, \texttt{No-Cube})$

Let us observe how such a refutation looks. The two unit clauses $\bar{z}_1, \bar{z}_2$ get propagated initially and then s and the y_i's have to be decided. A decision on s, irrespective of the polarity, causes no further propagations until w is also decided, as every clause containing the variable s also contains the variables w and t, and the reduction of w is blocked by t. Since neither (b_1, x_i) nor (b_2, x_i) is in $\texttt{D}^{\texttt{rrs}}$, a decision on variable y_i propagates either x_i (due to the clause containing $\bar{y}_i$, where b_1 can now be reduced), or $\bar{x}_i$ (due to the clause containing y_i, where b_2 can now be reduced). The propagating of x_i due to y_i sends therefutation down the same "trap" as the `Trapdoor` formulas for $\texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{trv}}, \texttt{No-Cube})$ [BB23a](as highlighted in Lemma 4.14), and thus $\texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{rrs}}, \texttt{No-Cube})$ requires exponential size refutations to refute these formulas. $\square$

Now, since preprocessing by $\texttt{D}^{\texttt{rrs}}$ does not change the formula at all, therefore for $\texttt{ClausePol} \in \{\texttt{D}^{\texttt{trv}}, \texttt{D}^{\texttt{std}}\}$ the refutational hardness and in fact the refutation for these two formulas will be exactly the same for the systems

$\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{ClausePol}, \mathtt{No\text{-}Cube})$ and $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{ClausePol}, \mathtt{No\text{-}Cube})$, and similarly for the pair $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{No\text{-}Cube})$ and $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{No\text{-}Cube})$.

Lemma 4.49. *For* $\mathtt{CubePol} \in \{\mathtt{No\text{-}Cube}, \mathtt{Cube\text{-}LD}, \mathtt{Cube\text{-}D}^{\mathtt{trv}}, \mathtt{Cube\text{-}D}^{\mathtt{std}}, \mathtt{Cube\text{-}D}^{\mathtt{rrs}}\}$*,* *the* $\mathtt{PropDep\text{-}Trap}$*formulas*

- *for* $\mathtt{ClausePol} \in \{\mathtt{D}^{\mathtt{trv}}, \mathtt{D}^{\mathtt{std}}\}$ *have polynomial size refutations in* $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{ClausePol}, \mathtt{CubePol})$.
- *require exponential size refutations in* $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{No\text{-}Cube})$

Proof. Since, $\mathtt{red\text{-}D}^{\mathtt{rrs}}(\mathtt{PropDep\text{-}Trap}) = \mathtt{PropDep\text{-}Trap}$, there is no effect of preprocessing, therefore by Lemmas 4.47 and 4.48 the statement holds. □

4.11 The DoubleLongEq Formula

The Equality formulas witness that $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{Cube\text{-}LD})$ is stronger than $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{No\text{-}Cube})$ [BPB24]. We wish to show a similar separation for QCDCL-like systems with $\mathtt{D}^{\mathtt{rrs}}$; namely, that cube-learning provides an advantage. The Equality formulas themselves cannot show such a separation because even without cube-learning, once $\mathtt{D}^{\mathtt{rrs}}$ is included in any form, they are easy to refute. We therefore look for formulas that separate $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{Cube\text{-}LD})$ from $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{No\text{-}Cube})$ and additionally have $\mathtt{D}^{\mathtt{rrs}}$ coinciding with $\mathtt{D}^{\mathtt{trv}}$; these would give us the desired separation. To obtain such a formula, we modify the Equality formula by adding two clauses to the matrix. These clauses enforce that $\mathtt{D}^{\mathtt{rrs}}$ and $\mathtt{D}^{\mathtt{trv}}$ are identical, but do not affect the hardness of Equality in $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{No\text{-}Cube})$ or its ease in $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{Cube\text{-}LD})$. The new formulas, called DoubleLongEq, are defined below.

Formula 4.50. *[*$\texttt{DoubleLongEq}_n$*]* *The* $\texttt{DoubleLongEq}_n$ *formula has the prefix* $\exists x_1 \cdots x_n \; \forall u_1 \cdots u_n \; \exists t_1 \cdots t_n$ *and the PCNF matrix*

$$\underbrace{(\bar{t}_1 \vee \cdots \vee \bar{t}_n)}_{T_n} \wedge \bigwedge_{i=1}^{n} \left[\underbrace{(x_i \vee u_i \vee t_i)}_{A_i} \wedge \underbrace{(\bar{x}_i \vee \bar{u}_i \vee t_i)}_{B_i} \right] \wedge$$
$$\underbrace{(\bar{u}_1 \vee \cdots \bar{u}_n \vee \bar{t}_1 \vee \cdots \bar{t}_n)}_{UT_n} \wedge \underbrace{(\bar{u}_1 \vee \cdots \bar{u}_n \vee t_1 \vee \cdots t_n)}_{UT'_n}$$

(Note: deleting the clauses UT_n and UT'_n gives the `Equality` formulas.)

Now let us look at the dependency scheme for these formulas.

Proposition 4.51. *For the* `DoubleLongEq` *formulas,* $\mathtt{D}^{\mathtt{rrs}}, \mathtt{D}^{\mathtt{std}}, \mathtt{D}^{\mathtt{trv}}$ *are all equivalent. i.e.*

$$\mathtt{D}^{\mathtt{trv}}(\mathtt{DoubleLongEq}) = \mathtt{D}^{\mathtt{std}}(\mathtt{DoubleLongEq}) = \mathtt{D}^{\mathtt{rrs}}(\mathtt{DoubleLongEq})$$

Proof. Let us look at $\mathtt{D}^{\mathtt{rrs}}$ for any (u_i, t_j) pair, if $i = j$ then trivially there is a (u_i, t_i) and $(\bar{u}_i, \bar{t}_i)$ path by virtue of being in the same clause. Now, if $i \neq j$, then there is a (u_i, t_j) path via t_i as well as a trivial $(\bar{u}_i, \bar{t}_j)$ path. Therefore $\mathtt{D}^{\mathtt{rrs}}$ and $\mathtt{D}^{\mathtt{trv}}$ are equivalent and hence $\mathtt{D}^{\mathtt{trv}} = \mathtt{D}^{\mathtt{std}} = \mathtt{D}^{\mathtt{rrs}}$. □

Hence, preprocessing with any of these `D`, or using $\mathtt{D}^{\mathtt{rrs}}$ or $\mathtt{D}^{\mathtt{std}}$ in propagation/learning, makes no difference. We first show these formulas are hard in the absence of cube learning, and then observe that they are easy to refute when cube learning is switched on. Both these results closely mirror the corresponding results for the `Equality` formulas shown in [BB23a] and [BPB24] respectively.

In [BB23b], the authors consider Σ^3 formulas with a specific structure, called XUT-formulas with the XT-property. They introduce a semantic measure called gauge for Σ^3 QBFs, and show that for an XUT-formula with the XT-property, refutation size in $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{No\text{-}Cube})$ is at least exponential in its gauge (Theorem 6,

[BB23b]). We use this argument to prove our lower bound for `DoubleLongEq`. First we state the relevant definitions.

Definition 4.52 (XT-property, (Definition 5, [BB23b])). *Let Φ be a PCNF QBF of the form $\exists X \forall U \exists T \cdot \phi$, where X, U, T are non-empty sets of variables. Then Φ is an XUT-formula. We call a clause C an*

- *X-clause: if it is non-empty and contains only X variables,*
- *T-clause: if it is non-empty and contains only T variables,*
- *XT-clause: if it contains no U variable and at least one X and one T variable,*
- *XUT-clause: if it contains atleast one each of X,U, and T variables.*

Φ is said to fulfill the XT-property if ϕ contains no XT-clauses or unit T clause, and if no two T clauses in ϕ are resolvable (the resolvent of any two T clauses, if defined, is tautological).

It is easy to see that `DoubleLongEq` is an XUT formula with the XT property.

The gauge measure is defined as follows:

Definition 4.53 (gauge, (Definition 6, [BB23b])). *Let Φ be an XUT formula. The gauge of Φ is the size of the narrowest X-clause derivable using only reductions and resolutions over variables in T.*

We now show that our formula has a linear size gauge

Proposition 4.54. *The* `DoubleLongEq` *formulas have a gauge of size n.*

Proof. First observe that none of the axioms are X-clauses. Therefore to derive an X-clause, there has to be some T-resolutions. A first T-resolution must involve either T_n or UT_n, since only these clauses have T variables negated. However, both

these clauses have all n T-variables. Thus to eventually derive an X-clause, there must be a resolution on every t_i variable. Each such resolution introduces an x_i variable. Therefore by the time all T variables are removed, all the X variables are introduced. Therefore, the gauge of $\texttt{DoubleLongEq}_n$ is n. □

This helps us show that these formula require exponential size refutations in the absence of cube-learning

Lemma 4.55. *For* $\texttt{D}_1, \texttt{D}_2 \in \{\texttt{D}^{\texttt{trv}}, \texttt{D}^{\texttt{std}}, \texttt{D}^{\texttt{rrs}}\}$ *the* DoubleLongEq *formulas require exponential size refutations in* $\texttt{D}_1 + \texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}_2, \texttt{No-Cube})$.

Proof. Since $\texttt{D}^{\texttt{rrs}}_\forall$, $\texttt{D}^{\texttt{std}}_\forall$ and $\texttt{D}^{\texttt{trv}}_\forall$ coincide for the DoubleLongEq formula, therefore refutations in $\texttt{D}_1 + \texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}_2, \texttt{No-Cube})$ where $\texttt{D}_1, \texttt{D}_2 \in \{\texttt{D}^{\texttt{trv}}, \texttt{D}^{\texttt{std}}, \texttt{D}^{\texttt{rrs}}\}$ are all equivalent. Therefore by the discussion above, it suffices to show that the formula requires exponential size refutations in $\texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{trv}}, \texttt{No-Cube})$.

The DoubleLongEq formulas are easily seen to be XUT-formulas with the XT-property. It is shown in [BB23b] that if an XUT formula with the XT property has linear gauge, then it requires exponential size to refute in QCDCL. By Proposition 4.54, it follows that they require exponential size $\texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{trv}}, \texttt{No-Cube})$ refutations. □

We now show that when cube learning is allowed these formulas have a polynomial size refutation.

Proposition 4.56. *The* DoubleLongEq *formulas have polynomial size refutations in the proof system* $\texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{trv}}, \texttt{Cube-LD})$.

Proof. The polynomial size refutation for these formulas in $\texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{trv}}, \texttt{Cube-LD})$ is exactly the same as the refutation in $\texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{trv}}, \texttt{Cube-LD})$ for the Equality formulas, as described in [BPB24]. By

constructing trails in exactly the same manner, we first learn $2n-2$ cubes of the form $(x_i \wedge \bar{u}_i)$ and $(\bar{x}_i \wedge u_i)$ for $i = 1...n-1$ and then start clause learning by constructing trails ending in a conflict. The two new clauses UT_n and UT'_n play no role whatsoever. For completeness, we reproduce the entire refutation below; a reader familiar with the construction from [BPB24] can completely skip these details.

The proof goes in two stages. The first stage involves learning the cubes $x_i \wedge \bar{u}_i$ and $\bar{x}_i \wedge u_i$ for $i \in [n-1]$. The first trail is the following.

$$\mathcal{T}_1 = \mathbf{x_1}; \cdots; \mathbf{x_n}; \bar{\mathbf{u}}_\mathbf{1}; \cdots; \bar{\mathbf{u}}_\mathbf{n}; \bar{\mathbf{t}}_\mathbf{1}; \mathbf{t_2}; \cdots; \mathbf{t_n}$$

It assigns all variables without conflict and satisfies the matrix. The partial assignment $x_1 \wedge \bar{u}_1 \wedge \bar{t}_1 \wedge t_2 \wedge \cdots \wedge t_n$ contained in it is also a satisfying assignment, and reducing it with $\texttt{red}_\exists$ we can learn the cube $x_1 \wedge \bar{u}_1$ from this trail.

Analogously, creating a complementary trail $\mathcal{T}'_1$ where each decision is the complement of the decision in $\mathcal{T}_1$, we can learn the cube $\bar{x}_1 \wedge u_1$.

Suppose we have learn $2i$ cubes in the same manner; $x_j \wedge \bar{u_j}$ and $\bar{x_j} \wedge u_j$ for $j = 1, \cdots, i$. For $i+1$, create the following trail.

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

In this trail, for $j \leq i$, $\texttt{ante}(u_j) = x_j \wedge \bar{u}_j$ and $\texttt{ante}(t_j) = \bar{x}_j \vee \bar{u}_j \vee t_j$. As earlier, the trail satisfies all clauses without conflict. Extracting the partial assignment $x_{i+1} \wedge \bar{u}_{i+1} \wedge t_1 \wedge \cdots t_i \wedge \bar{t}_{i+1} \wedge t_{i+2} \wedge \cdots \wedge t_n$ which also satisfies the matrix, and reducing it, we can learn the cube $x_{i+1} \wedge u_{i+1}$. Analogously through a trail $\mathcal{T}'_{i+1}$ we learn $\bar{x}_{i+1} \wedge u_{i+1}$.

Having learnt the $2n-2$ cubes in this manner, we start with clause learning, where

we proceed by constructing the trails $\mathcal{U}_{n-1}, \mathcal{V}_{n-1}, \mathcal{U}_{n-2}, \mathcal{V}_{n-2}, \cdots, \mathcal{U}_1, \mathcal{V}_1$ described below, and learn clauses $L_{n-1}, R_{n-1}, \cdots L_1, R_1$ corresponding to these trails. We use T_j to denote the subclause of T_n with literals $\bar{t}_i$ for $i\ ej$.

The initial trail is

$$\mathcal{U}_{n-1} = (\mathbf{x_1}, u_1, t_1; \mathbf{x_2}, u_2, t_2; \cdots; \mathbf{x_{n-1}}, u_{n-1}, t_{n-1}, \bar{t}_n, x_n, \square)$$

The antecedent clauses are as follows:

$$\begin{aligned}
\texttt{ante}(u_j) &= x_j \wedge \bar{u}_j \\
\texttt{ante}(t_j) &= \bar{x}_j \vee \bar{u}_j \vee t_j \\
\texttt{ante}(\bar{t}_n) &= T_n \\
\texttt{ante}(x_n) &= x_n \vee u_n \vee t_n \\
\texttt{ante}(\square) &= \bar{x}_n \vee \bar{u}_n \vee t_n
\end{aligned}$$

From these clauses we learn the clause $L_{n-1} = \bar{x}_{n-1} \vee \bar{u}_{n-1} \vee (u_n \vee \bar{u}_n) \vee T_{n-2}$.

Then we restart and create a symmetric trail to $\mathcal{U}_{n-1}$:

$$\mathcal{V}_{n-1} = (\bar{\mathbf{x}}_\mathbf{1}, \bar{u}_1, t_1; \bar{\mathbf{x}}_\mathbf{2}, \bar{u}_2, t_2; \cdots; \bar{\mathbf{x}}_\mathbf{n-1}, \bar{u}_{n-1}, t_{n-1}, \bar{t}_n, x_n, \square)$$

where the antecedent clauses are

$$\begin{aligned}
\texttt{ante}(\bar{u}_j) &= \bar{x}_j \wedge u_j \\
\texttt{ante}(t_j) &= x_j \vee u_j \vee t_j \\
\texttt{ante}(\bar{t}_n) &= T_n \\
\texttt{ante}(x_n) &= x_n \vee u_n \vee t_n \\
\texttt{ante}(\square) &= \bar{x}_n \vee \bar{u}_n \vee t_n.
\end{aligned}$$

From this trail we can learn the clause $R_{n-1} = x_{n-1} \vee u_{n-1} \vee (u_n \vee \bar{u}_n) \vee T_{n-2}$.

For i in the range of 2 to $n-1$, we define the following clauses:

$$L_i = \bar{x}_i \vee \bar{u}_i \vee \bigvee_{j=i+1}^{n} (u_j \vee \bar{u}_j) \vee T_{i-1}$$
$$R_i = x_i \vee u_i \vee \bigvee_{j=i+1}^{n} (u_j \vee \bar{u}_j) \vee T_{i-1}$$

We claim that from the trail $\mathcal{U}_i$ we learn the clause L_i and from the trail $\mathcal{V}_i$ we learn the clause R_i. We have already established this for $i = n-1$. Suppose we have already learnt $L_{n-1}, R_{n-1}, ..., L_{i+1}, R_{j+1}$ for $1 \leq i < n-1$. Continuing, we consider the next trail,

$$\mathcal{U}_i = (\mathbf{x_1}, u_1, t_1; \mathbf{x_2}, u_2, t_2; \cdots ; \mathbf{x_i}, u_i, t_i, x_{i+1}, \square)$$

where the antecedent clauses are as follows.

$$\begin{aligned} \texttt{ante}(u_j) &= x_j \wedge \bar{u}_j \\ \texttt{ante}(t_j) &= \bar{x}_j \vee \bar{u}_j \vee t_j \\ ante(x_{i+1}) &= L_{i+1} \\ ante(\square) &= R_{i+1} \end{aligned}$$

From this we learn the clause $L_i = \bar{x}_i \bar{u}_i \vee \bigvee_{j=i+1}^{n} (u_j \vee \bar{u}_j) \vee T_{i-1}$.
Next we create the symmetrical trail,

$$\mathcal{V}_i = (\bar{\mathbf{x}}_\mathbf{1}, \bar{u}_1, t_1; \bar{\mathbf{x}}_\mathbf{2}, \bar{u}_2, t_2; \cdots ; \bar{\mathbf{x}}_\mathbf{i}, \bar{u}_i, t_i, x_{i+1}, \square)$$

and the antecedent clauses are as follows:

$$\begin{aligned} \texttt{ante}(\bar{u}_j) &= \bar{x}_j \wedge u_j \\ \texttt{ante}(t_j) &= x_j \vee u_j \vee t_j \\ \texttt{ante}(x_{i+1}) &= L_{i+1} \\ \texttt{ante}(\square) &= R_{i+1} \end{aligned}$$

From this we can learn the clause $R_i = x_i \vee u_i \vee \bigvee_{j=i+1}^{n}(u_j \vee \bar{u}_j) \vee T_{i-1}$.

The proof ends with the two trails

$$\mathcal{U}_1 = (\mathbf{x_1}, u_1, t_1, x_2, \square)$$

with antecedents clauses

$$\begin{aligned} \texttt{ante}(u_1) &= x_1 \wedge \bar{u}_1 \\ \texttt{ante}(t_1) &= \bar{x}_1 \vee t_1 \\ \texttt{ante}(x_2) &= L_2 \\ \texttt{ante}(\square) &= R_2 \end{aligned}$$

allowing us to learn the clause $L_1 = \bar{x}_1$, and finally the last trail

$$\mathcal{V}_1 = (\bar{x}_1, \bar{u}_1, t_1, x_2, \square)$$

with antecedent clauses

$$\texttt{ante}(\bar{x}_1) = \bar{x}_1$$

$$\texttt{ante}(\bar{u}_1) = \bar{x}_1 \wedge u_1$$

$$\texttt{ante}(t_1) = \bar{x}_1 \vee u_1 \vee t_1$$

$$\texttt{ante}(x_2) = L_2$$

$$\texttt{ante}(\square) = R_2$$

Resolving over all propagations in this trail, we learn the empty clause, completing the refutation. □

Lemma 4.57. *For* $\texttt{D}_1, \texttt{D}_2 \in \{\texttt{D}^{\texttt{trv}}, \texttt{D}^{\texttt{std}}, \texttt{D}^{\texttt{rrs}}\}$ *and* $\texttt{CubePol} \in \{\texttt{Cube-LD}, \texttt{Cube-D}_2\}$*, the* DoubleLongEq *formulas have polynomial size refutations in* $\texttt{D}_1 + \texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}_2, \texttt{CubePol})$.

Proof. It can be seen that the cubes learnt in the refutation described in Proposition 4.56 require no cube learning via resolution steps; they are all learnt from trails ending in satisfaction, using the term axiom rule and the $\texttt{red}_\exists$ rule. Therefore for this particular refutation, every cube learning step in the $\texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{trv}}, \texttt{Cube-LD})$ refutation is also a valid step in a $\texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{trv}}, \texttt{Cube-D}_2)$ refutation, for $\texttt{D}_2 \in \{\texttt{D}^{\texttt{trv}}, \texttt{D}^{\texttt{std}}, \texttt{D}^{\texttt{rrs}}\}$.

Since, for these formulas $\texttt{D}^{\texttt{trv}} = \texttt{D}^{\texttt{std}} = \texttt{D}^{\texttt{rrs}}$, therefore neither preprocessing nor the ClausePol have any effect. Therefore the refutation described in Proposition 4.56 is also a valid refutation in $\texttt{D}_1 + \texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}_2, \texttt{Cube-D}_2)$ where $\texttt{D}_1, \texttt{D}_2 \in \{\texttt{D}^{\texttt{trv}}, \texttt{D}^{\texttt{std}}, \texttt{D}^{\texttt{rrs}}\}$. □

This shows us that the DoubleLongEq formula are hard to refute without cube learning, but have polynomial size refutations with cube learning even when dependency schemes $\texttt{D}^{\texttt{rrs}}, \texttt{D}^{\texttt{std}}$ are used and irrespective of preprocessing.

4.12 The PreRRSTrapdoor Formula

The underlying motivation to construct the next formula is to understand how the different ways of adding $\mathtt{D}^{\mathtt{rrs}}$ have different effects. The goal is to build a formula which sends the QCDCL trails down a "trap" (in this case, of refuting the purely existential PigeonHole Principle formula PHP, known to be hard for propositional resolution) if $\mathtt{D}^{\mathtt{rrs}}$ is not allowed in propagation, but allows a quick jump into a propositionally easy-to-refute section (in this case, the negation of the complete tautology on two variables) when $\mathtt{D}^{\mathtt{rrs}}$ is available in propagation. This leads us to define the following formula motivated by the Trapdoor and Dep-Trap formulas

Formula 4.58. *[$\mathtt{PreRRSTrapdoor}_n$]*

The $\mathtt{PreRRSTrapdoor}_n$ *formula has the prefix*
$\exists a\ \forall p\ \exists y_1, \cdots, y_{s_n}\ \forall w\ \forall v\ \exists t\ \exists x_1, \cdots, x_{s_n}\ \forall u\ \exists b\ \exists q\ \exists r\ \exists s$, *and the matrix is as given below.*

$$\mathtt{PHP}_n^{n+1}(x_1, \cdots, x_{s_n})$$

$$\textit{for}\ \ i \in [s_n]: \quad (\bar{y}_i \vee x_i \vee u \vee b)\ ,\ (y_i \vee \bar{x}_i \vee u \vee b)$$

$$\textit{for}\ \ i \in [s_n]: \quad (y_i \vee w \vee v \vee t \vee b)\ ,\ (y_i \vee w \vee v \vee \bar{t} \vee b)$$

$$\textit{for}\ \ i \in [s_n]: \quad (\bar{y}_i \vee w \vee v \vee t \vee b)\ ,\ (\bar{y}_i \vee w \vee v \vee \bar{t} \vee b)$$

$$(\bar{u} \vee \bar{b})\ ,\ (v \vee \bar{b} \vee \bar{r})\ ,\ (\bar{v} \vee b \vee s)$$

$$(a \vee \bar{b})\ ,\ (\bar{a} \vee \bar{b})\ ,\ (p \vee q)\ ,\ (\bar{p} \vee \bar{q})$$

Note 1. *The variable "*w*" is not necessary for the lower or upper bounds proved in this section. Initialising* $\mathtt{PreRRSTrapdoor}|_{w=0}$ *or removing the variable "*w*" entirely affects neither the bounds nor their proofs.*

However we keep it in because the PreRRSTrapdoor *formulas are defined to extend the* Trapdoor *formula (defined in [BB23a], see section 4.4), which has the "*w*" variable. Also, it shows that even if the preprocessing step (by* $\mathtt{D}^{\mathtt{rrs}}$*) is non-trivial and changes the formula, addition of* $\mathtt{D}^{\mathtt{rrs}}$ *in propagation can still make a difference.*

Clearly, this formula has an unsatisfiable matrix (due to the presence of PHP).

First let us look at the dependency schemes for this formula

Proposition 4.59. *For the* PreDepTrap *formulas:*

$$\mathtt{D}^{\mathtt{std}}(\mathtt{PreDepTrap}) = \{(p,q)\} \cup \{((\alpha,\beta)) | \alpha \in \{w,v\}, \beta \in \{t,b,r,s,x_i\}\} \cup \{(u,b),(u,r),(u,s)\}$$

$$\mathtt{D}^{\mathtt{rrs}}(\mathtt{PreDepTrap}) = \{(p,q),(v,b),(u,b)\}$$

Now, we first see that when $\mathtt{D}^{\mathtt{rrs}}$ is used in both preprocessing and in propagation the formulas have polynomial size refutation.

Lemma 4.60. *For* $\mathtt{CubePol} \in \{\mathtt{No\text{-}Cube}, \mathtt{Cube\text{-}LD}, \mathtt{Cube\text{-}D}^{\mathtt{rrs}}\}$ *the* PreRRSTrapdoor *formulas have polynomial size refutations in* $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{CubePol})$.

Proof. By Observation 3.10, a refutation in a system without cube learning is a refutation in a system with any type of cube-learning given other parameters stay the same. Therefore, in this case it suffices to show polynomial size refutations in $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{No\text{-}Cube})$.

Any trail must start with a decision on a. A decision in either polarity propagates $\bar{b}$, and with $\mathtt{D}^{\mathtt{rrs}}$ used in propagation, further propagates s since s does not depend on v. Next, the variable p must be decided; any polarity propagates a q literal. At this point y_1 must be decided. Since t also does not depend on v, this decision in either polarity propagates a t literal and then a conflict. An example trail is as follows: $\mathcal{T} = \mathbf{a}, \bar{b}, s; \mathbf{p}, \bar{q}; \mathbf{y_1}, t, \square$. The conflict reached is due to the complete tautology on y_1 and t. Thus in 4 such trails the empty clause can be learnt, completing the refutation.

The refutation is described in detail as follows: We construct a polynomial time $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{No\text{-}Cube})$ refutation of PreRRSTrapdoor. Since $\mathtt{D}^{\mathtt{rrs}}$ for the

formula is $\{(u,b),(v,b),(p,q)\}$, preprocessing using $\mathtt{D}^{\mathtt{rrs}}$ reduces the formula to

$$\exists a \forall p \exists y_1, \cdots, y_{s_n} \forall v \exists t \exists x_1, \cdots, x_{s_n} \forall u \exists b \exists q \exists r \exists s$$

$$\mathtt{PHP}_n^{n+1}(x_1, \cdots, x_{s_n})$$

$$\text{for } \; i \in [s_n]: \qquad \bar{y}_i \vee x_i \vee u \vee b \;,\; y_i \vee \bar{x}_i \vee u \vee b$$

$$\text{for } \; i \in [s_n]: \quad y_i \vee v \vee t \vee b \;,\; y_i \vee v \vee \bar{t} \vee b \;,\; \bar{y}_i \vee v \vee t \vee b \;,\; \bar{y}_i \vee v \vee \bar{t} \vee b$$

$$\bar{u} \vee \bar{b} \;,\; v \vee \bar{b} \vee \bar{r} \;,\; \bar{v} \vee b \vee s$$

$$a \vee \bar{b} \;,\; \bar{a} \vee \bar{b} \;,\; p \vee q \;,\; \bar{p} \vee \bar{q}$$

Now consider the following trail. Due to $\mathtt{D}^{\mathtt{rrs}}$ being used in propagation as well, the literal t will be propagated even before v is decided, producing a conflict in the clauses involving t.

$$T_1 = (\mathbf{a}, \bar{b}, s; \mathbf{p}, \bar{q}; \mathbf{y_1}, t, \Box)$$

where the antecedent clauses are as follows:

$$\begin{aligned}
\mathtt{ante}(\bar{b}) &= \bar{a} \vee \bar{b} \\
\mathtt{ante}(s) &= \bar{v} \vee b \vee s \\
\mathtt{ante}(\bar{q}) &= \bar{p} \vee \bar{q} \\
\mathtt{ante}(t) &= \bar{y}_1 \vee v \vee t \vee b \\
\mathtt{ante}(\Box) &= \bar{y}_1 \vee v \vee \bar{t} \vee b
\end{aligned}$$

From this trail we learn the clause $L_1 = \bar{a} \vee \bar{y}_1$.

Next construct the trail:

$$T_2 = (\bar{\mathbf{a}}, \bar{b}, s; \mathbf{p}, \bar{q}; \bar{\mathbf{y}}_\mathbf{1}, t, \Box)$$

where the antecedent clauses are as follows:

$$\begin{aligned}
\texttt{ante}(\bar{b}) &= a \vee \bar{b} \\
\texttt{ante}(s) &= \bar{v} \vee b \vee s \\
\texttt{ante}(\bar{q}) &= \bar{p} \vee \bar{q} \\
\texttt{ante}(t) &= y_1 \vee v \vee t \vee b \\
\texttt{ante}(\square) &= y_1 \vee v \vee \bar{t} \vee b
\end{aligned}$$

From this trail, we learn the clause $L_2 = a \vee y_1$.

Now consider the following third trail:

$$T_3 = (\mathbf{a}, \bar{b}, \bar{y}_1, t, \square)$$

with antecedent clauses as follows:

$$\begin{aligned}
\texttt{ante}(\bar{b}) &= \bar{a} \vee \bar{b} \\
\texttt{ante}(\bar{y}_1) &= L_1 = \bar{a} \vee \bar{y}_1 \\
\texttt{ante}(t) &= y_1 \vee v \vee t \vee b \\
\texttt{ante}(\square) &= y_1 \vee v \vee \bar{t} \vee b
\end{aligned}$$

From here we learn the unit clause $L_3 = \bar{a}$.

Finally we have the fourth trail which is fully propagated and has no decisions.

$$T_4 = (\bar{a}, \bar{b}, y_1, t, \square)$$

where,

$$\begin{aligned}
\texttt{ante}(\bar{a}) &= \bar{a} \\
\texttt{ante}(\bar{b}) &= \bar{a} \vee \bar{b} \\
\texttt{ante}(y_1) &= L_1 = a \vee y_1 \\
\texttt{ante}(t) &= y_1 \vee v \vee t \vee b \\
\texttt{ante}(\square) &= y_1 \vee v \vee \bar{t} \vee b
\end{aligned}$$

From this trail we learn the empty clause ($\square$), thus completing the refutation. $\square$

However if we use $\texttt{D}^{\texttt{rrs}}$ only in preprocessing then formula requires exponential size refutations due to falling into the PHP trap.

Lemma 4.61. *For* $\texttt{CubePol} \in \{\texttt{No-Cube}, \texttt{Cube-LD}, \texttt{Cube-D}^{\texttt{rrs}}\}$ *the* PreRRSTrapdoor *formulas require exponential size refutations in* $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{trv}}, \texttt{CubePol})$.

Proof. From Observation 3.11 since the PreRRSTrapdoor formulas have an unsatisfiable matrix, therefore, it suffices to show hardness in $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{trv}}, \texttt{No-Cube})$.

Observe that any trail must start with a decision on a, which always propagates the literal $\bar{b}$, followed by a decision on p, propagating a q literal: $\mathcal{T} = \mathbf{a}/\bar{\mathbf{a}}, \bar{b}; \mathbf{p}/\bar{\mathbf{p}}, \bar{q}/q$. At this point in the trail, the formula matrix has reduced to

$$\begin{array}{lll}
 & & \texttt{PHP}_n^{n+1}(x_1, \cdots, x_{s_n}) \\
\text{for} \;\; i \in [s_n]: & & (\bar{y}_i \vee x_i \vee u) \;,\; (y_i \vee \bar{x}_i \vee u) \\
\text{for} \;\; i \in [s_n]: & (y_i \vee v \vee t) \;,\; (y_i \vee v \vee \bar{t}) \;,\; (\bar{y}_i \vee v \vee t) \;,\; (\bar{y}_i \vee v \vee \bar{t}) & \\
 & & (\bar{v} \vee s)
\end{array}$$

This is effectively the matrix of the Trapdoor formulas from [BB23a], with one extra clause $\bar{v} \vee s$. After this point, all decisions are made on y variables propagating a

corresponding x variable. By the time all y variables are decided, a conflict in PHP on the x variables is achieved. Thus, exactly like the Trapdoor formulas, refuting PreRRSTrapdoor boils down to refuting PHP, which is known to require exponential size. □

But if $\mathtt{D}^{\mathtt{rrs}}$ is used only in propagation, then the formula has polynomial size refutations.

Lemma 4.62. *For* $\mathtt{CubePol} \in \{\mathtt{No\text{-}Cube}, \mathtt{Cube\text{-}LD}, \mathtt{Cube\text{-}D}^{\mathtt{rrs}}\}$ *the* PreRRSTrapdoor *formulas have polynomial size refutations in* $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{CubePol})$.

Proof. This can be seen by exactly reconstructing the same trails seen above and recreating the same proof as in Lemma 4.60, the only difference being that the w variables instead of going away during preprocessing will now be reduced parallely with the v variables during propagation, with the trails and learnt clauses staying exactly the same. □

4.13 The StdDepTrapFormula

The StdDepTrap formula is introduced with the motivation of showing that having the power of $\mathtt{D}^{\mathtt{std}}$ in propagation and learning can be an advantage. To do this we must build a set of clauses which allow us to learn "something" in the presence of $\mathtt{D}^{\mathtt{std}}$, which is impossible (or at least hard) to learn in its absence. Further, we must be able to use this learnt clause to leapfrog into a short level-ordered QCDCL refutation in the presence of $\mathtt{D}^{\mathtt{std}}$, but in its absence we are stuck in a trap of refuting something hard. This leads to the definition of the following formula.

Formula 4.63. *[*$\mathtt{StdDepTrap}_n$*]*

We define the fomula StdDepTrap *having prefix*

$$\exists b \forall w_1 \exists z_1, \cdots, z_{s_n} \forall w_2 \exists a, d, c \forall u \exists x, y \exists p \exists e_1 \exists e_2$$

and the PCNF matrix

$$\bar{b} \vee \mathtt{PHP}_n^{n+1}(z_1, \cdots, z_{s_n})$$
$$(y \vee p) \ , \ (y \vee \bar{p})$$
$$(w_1 \vee e_1) \ , \ (w_2 \vee e_2)$$
$$(b \vee y) \ , \ (a \vee \bar{y}) \ , \ (\bar{a} \vee x) \ , \ (\bar{c} \vee u \vee \bar{x})$$
$$(d \vee c \vee \bar{y}) \ , \ (\bar{d} \vee c \vee \bar{y})$$

The variables w_1, w_2, u are essentially "separators", putting existential variables into different levels. The variables e_1, e_2, x are blockers, ensuring that the separators do not get reduced too early in the trail.

First as usual we look at what the dependency schemes are for this formula.

Proposition 4.64. *For the* `StdDepTrap` *formula*

$$\mathtt{D}^{\mathtt{std}}(\mathtt{StdDepTrap}) = \{(w_1, e_1), (w_2, e_2), (u, x)\}$$

$$\mathtt{D}^{\mathtt{rrs}}(\mathtt{StdDepTrap}) = \emptyset$$

Proof. Since no universal variable appears in negated polarity $\mathtt{D}^{\mathtt{rrs}}$ is empty. For $\mathtt{D}^{\mathtt{std}}$, the pairs $(w_1, e_1), (w_2, e_2), (u, x)$ are trivially present, no other pair is present as both the e's and x appear in exactly one clause. □

In the first lemma, we show that the `StdDepTrap` formulas are easy for resolution based QBF proof systems irrespective of dependency schemes.

Lemma 4.65. *The* `StdDepTrap` *formulas have constant-size refutations in* `Q-Res`*,* `QU-Res`*,* `LDQ-Res`*,* `Q`$(\mathtt{D}^{\mathtt{rrs}})$`-Res`*.*

Proof. By definition, it is enough to show constant-size refutation in Q-Res.

Consider the clauses $(y \vee p)$, $(y \vee \bar{p})$ they can be resolved to obtain y. Similarly the clauses $(d \vee c \vee \bar{y})$, $(\bar{d} \vee c \vee \bar{y})$, $(\bar{c} \vee u \vee \bar{x})$, $(\bar{a} \vee x)$, $(a \vee \bar{y})$ can be resolved in a sequence to obtain $u \vee \bar{y}$. Resolving y and $u \vee \bar{y}$ followed by u-reduction gives us the empty clause, completing the refutation. □

The next two lemmas show that these formulas have polynomial size refutations in LEV-ORD based QCDCL systems irrespective of cube-learning if the ClausePol = $\mathtt{D}^{\mathtt{std}}, \mathtt{D}^{\mathtt{rrs}}$ respectively.

Lemma 4.66. *For* CubePol $\in$ {No-Cube, Cube-LD, Cube-$\mathtt{D}^{\mathtt{std}}$, Cube-$\mathtt{D}^{\mathtt{rrs}}$} *the* StdDepTrap *formulas have polynomial size refutations in* $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{std}}, \mathtt{CubePol})$.

Proof. By Observation 3.10, a refutation in a system without cube learning is a refutation in a system with any type of cube-learning given other parameters stay the same. Therefore, in this case it suffices to show polynomial size refutations in $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{std}}, \mathtt{No\text{-}Cube})$.

For this purpose, consider the trail $\mathcal{T}_1 = \bar{\mathbf{b}}, y, a, x, \bar{c}, d, \square$.
Since $(u, y) \notin \mathtt{D}^{\mathtt{std}}(\mathtt{StdDepTrap})$, the learnable sequence for this trail is

$$L_{\mathcal{T}_1} = \{c \vee \bar{y}, u \vee \bar{x} \vee \bar{y}, \bar{a} \vee \bar{y}, \bar{y}, b\}.$$

We choose to learn $\bar{y}$, and then proceed to the next trail $\mathcal{T}_2 = \bar{y}, p, \square$. The learnable clauses for this trail are

$$L_{\mathcal{T}_2} = \{y \vee \bar{p}, y, \square\}.$$

Thus the empty clause □ is learnt, completing the refutation.

Therefore the StdDepTrap formulas have polynomial size refutations in

$\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{std}}, \mathtt{CubePol})$. □

Lemma 4.67. *For* $\mathtt{CubePol} \in \{\mathtt{No\text{-}Cube}, \mathtt{Cube\text{-}LD}, \mathtt{Cube\text{-}D}^{\mathtt{std}}, \mathtt{Cube\text{-}D}^{\mathtt{rrs}}\}$ *the* StdDepTrap *formulas have polynomial size refutations in* $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{CubePol})$.

Proof. By Observation 3.10, a refutation in a system without cube learning is a refutation in a system with any type of cube-learning given other parameters stay the same. Therefore, in this case it suffices to show polynomial size refutations in $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{No\text{-}Cube})$.

This refutation proceeds in exactly the same manner as the previous proof of Lemma 4.66, constructing the same trails, and same learnt clauses, the only difference is a clause in the sequence of learnable clauses which is irrelevant to the refutation. For the sake of completeness it is reproduced below.

$$\mathcal{T}_1 = \bar{\mathbf{b}}, y, a, x, \bar{c}, d, \square$$

.

Since $(u, y), (u, x) \notin \mathtt{D}^{\mathtt{rrs}}(\mathtt{StdDepTrap})$, the learnable sequence for this trail is

$$L_{\mathcal{T}_1} = \{c \vee \bar{y}, \bar{x} \vee \bar{y}, \bar{a} \vee \bar{y}, \bar{y}, b\}.$$

We choose to learn $\bar{y}$, and then proceed to the next trail $\mathcal{T}_2 = \bar{y}, p, \square$. The learnable clauses for this trail are

$$L_{\mathcal{T}_2} = \{y \vee \bar{p}, y, \square\}.$$

Thus the empty clause □ is learnt, completing the refutation.

Therefore the StdDepTrap formulas have polynomial size refutations in $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{CubePol})$. □

Next we see that in LEV-ORD based QCDCL systems without and dependency scheme, the StdDepTrap formulas require exponential size refutations.

Lemma 4.68. *For* $\texttt{CubePol} \in \{\texttt{No-Cube}, \texttt{Cube-LD}, \texttt{Cube-D}^{\texttt{trv}}\}$*, the* StdDepTrap *formulas require exponential size refutations in* $\texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{trv}}, \texttt{CubePol})$.

Proof. We first show that the the StdDepTrap formulas require exponential size without cube learning i.e. in $\texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{trv}}, \texttt{No-Cube})$ and then extend the argument for the case of $\texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{trv}}, \texttt{Cube-LD})$ and $\texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{trv}}, \texttt{Cube-D}^{\texttt{trv}})$.

Every $\texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{trv}}, \texttt{No-Cube})$ trail must start with a decision on the variable b, unless and until a literal on b is learnt; thenceforth a trail must begin by propagating that literal.

Suppose that the $\texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{trv}}, \texttt{No-Cube})$ trail starts with a decision b. In such a case there are no propagations possible. Next a decision must be made on w_1, which may or may not propagate e_1. At this point, no other propagations are possible, and decisions must be made on all the z variables. Since the z variables are only involved in the PHP clauses, this leads to a conflict in the PHP clauses.

Suppose that the$\texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{trv}}, \texttt{No-Cube})$ trail starts with the decision $\bar{b}$. Then there is a unique forced trail leading to a conflict, namely,

$$\mathcal{T}_1 = \bar{\mathbf{b}}, y, a, x, \bar{c}, d, \Box.$$

The learnable sequence for this trail under regular reduction is

$$L_{\mathcal{T}_1} = \{c \vee \bar{y}, u \vee \bar{x} \vee \bar{y}, \bar{a} \vee u \vee \bar{y}, u \vee \bar{y}, b\}.$$

It is easy to see that learning any clause other than b does not affect the decisions in the trail at all. Thus, with trails of this type, in a few stages the clause b will be learnt inevitably.

To summarise, any $\texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{trv}}, \texttt{No-Cube})$ trail must start with a decision on b. If the decision is b, it leads to a conflict in the PHP clauses. If the trails avoid decision b and start with $\bar{b}$, we will eventually be forced to learn the unit clause b, leading to trails starting with propagating b, and again reaching a conflict in the PHP clauses. Therefore, any $\texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{trv}}, \texttt{No-Cube})$ refutation of StdDepTrap reduces to refuting PHP, thus requiring exponential size.

In the case of $\texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{trv}}, \texttt{Cube-LD})$ and $\texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{trv}}, \texttt{Cube-D}^{\texttt{trv}})$, since the PHP clauses are unsatisfiable, $\bar{b}$ must be in any satisfying assignment of the matrix of StdDepTrap. Therefore for cube learning to ever play a role, the $\texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{trv}}, \texttt{Cube-LD})$ or $\texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{trv}}, \texttt{Cube-D}^{\texttt{trv}})$ trail must start with deciding b as $\bar{b}$. However, as discussed above, trails starting with $\bar{b}$ are forced and rapidly hit a conflict; they cannot lead on to a satisfying assignment. Thus, even though the matrix of StdDepTrap is satisfiable (e.g. by the literals $\bar{b}, y, a, x, c, u, w_1, w_2$), such assignments can never be discovered through QCDCL trails. Hence, the $\texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{trv}}, \texttt{No-Cube})$ hardness lifts as cube learning is useless, and the StdDepTrap formulas continue to be hard for $\texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{trv}}, \texttt{Cube-LD})$ and $\texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{D}^{\texttt{trv}}, \texttt{Cube-D}^{\texttt{trv}})$. □

Finally we see that if preprocessed by $\texttt{D}^{\texttt{rrs}}$, the StdDepTrap formulas have constant size refutations in all QCDCL systems.

Lemma 4.69. *For* $\texttt{ClausePol} \in \{\texttt{D}^{\texttt{trv}}, \texttt{D}^{\texttt{rrs}}, \texttt{D}^{\texttt{std}}\}$ *and* $\texttt{CubePol} \in \{\texttt{No-Cube}, \texttt{Cube-LD}, \texttt{Cube-D}^{\texttt{std}}, \texttt{Cube-D}^{\texttt{rrs}}\}$ *the* StdDepTrap *formulas have polynomial size refutations in* $\texttt{D}^{\texttt{rrs}} + \texttt{QCDCL}^{\texttt{LEV-ORD}}(\texttt{ClausePol}, \texttt{CubePol})$.

Proof. Since, $\texttt{D}^{\texttt{rrs}}(\texttt{StdDepTrap}) = \emptyset$, preprocessing it using $\texttt{D}^{\texttt{rrs}}$ gives us a propositional formula which is clearly easy to refute in resolution therefore, by Proposition 3.13 the StdDepTrap formulas become easy to refute in the preprocessed QCDCL proof systems. □

This completes the list of formulas we wish to study. Next using the bound obtained for these formulas in various systems we will analyse the strength (and weaknesses) of the QCDCL systems under dependency schemes.

The table below summarises the results of this chapter

Formula	Matrix	Dependencies	Weakest System where Easy	Strongest System where Hard
QParity	Sat	$\mathtt{D}^{\mathtt{rrs}} = \mathtt{D}^{\mathtt{std}} = \mathtt{D}^{\mathtt{trv}}$	QCDCL with $\mathtt{D}^{\mathtt{rrs}}$ and $\mathtt{D}^{\mathtt{trv}}$	$\mathtt{Q}(\mathtt{D}^{\mathtt{rrs}})$-Res
Equality	Sat	$\mathtt{D}^{\mathtt{rrs}} = \emptyset$ $\mathtt{D}^{\mathtt{std}} = \mathtt{D}^{\mathtt{trv}}$	$\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{Cube\text{-}LD})$ [BPB24] $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{Cube\text{-}D}^{\mathtt{trv}})$ [BPB24] QCDCL with $\mathtt{D}^{\mathtt{rrs}}$ $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{std}}, \mathtt{Cube\text{-}LD})$ $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{Cube\text{-}D}^{\mathtt{std}})$	$\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{No\text{-}Cube})$ [BB23a] $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{std}}, \mathtt{No\text{-}Cube})$
TwinEq	Sat	$\mathtt{D}^{\mathtt{rrs}} = \emptyset$ $\mathtt{D}^{\mathtt{std}} = \mathtt{D}^{\mathtt{trv}}$	QCDCL with $\mathtt{D}^{\mathtt{rrs}}$	$\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{Cube\text{-}LD})$ [BPB24]
Trapdoor	Unsat	$\mathtt{D}^{\mathtt{rrs}} = \emptyset$ $\mathtt{D}^{\mathtt{std}} = \{(w,t)\}$	QCDCL with $\mathtt{D}^{\mathtt{rrs}}$	$\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{Cube\text{-}LD})$ (Obs. 3.11) $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{Cube\text{-}D}^{\mathtt{trv}})$ $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{std}}, \mathtt{Cube\text{-}LD})$ $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{std}}, \mathtt{Cube\text{-}D}^{\mathtt{std}})$
Dep-Trap	Unsat	$\mathtt{D}^{\mathtt{rrs}} = \{(w,t)\}$ $\mathtt{D}^{\mathtt{std}} = \{(w,t)\} \cup \{(u_i,x_i)\}$	$\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{No\text{-}Cube})$ $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{std}}, \mathtt{No\text{-}Cube})$	$\mathtt{QCDCL}^{\mathtt{cube}}$ with $\mathtt{D}^{\mathtt{rrs}}$ (Obs. 3.11)
TwoPHPandCT	Unsat	$\mathtt{D}^{\mathtt{rrs}} = \emptyset$	$\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{No\text{-}Cube})$ $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{No\text{-}Cube})$	$\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{Cube\text{-}LD})$ (Obs. 3.11) $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{Cube\text{-}D}^{\mathtt{trv}})$ (Obs. 3.11) $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{Cube\text{-}LD})$ (Obs. 3.11) $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{Cube\text{-}D}^{\mathtt{rrs}})$ (Obs. 3.11)
PreDepTrap	Sat red-$\mathtt{D}^{\mathtt{rrs}}$: Unsat	$\mathtt{D}^{\mathtt{rrs}} = \{(w,t)\}$	QCDCL $\mathtt{QCDCL}(\mathtt{D}^{\mathtt{rrs}})$	$\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{cube}}$(,Obs. 3.11) $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{cube}}(\mathtt{D}^{\mathtt{rrs}})$(,Obs. 3.11)
PropDep-Trap	Unsat	$\mathtt{D}^{\mathtt{rrs}} = \{(w,t),(b_1,z_1),(b_2,z_2)\}$	$\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{No\text{-}Cube})$ $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{No\text{-}Cube})$	$\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{Cube\text{-}LD})$ (Obs. 3.11) $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{Cube\text{-}D}^{\mathtt{rrs}})$ (Obs. 3.11) $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{Cube\text{-}LD})$ (Obs. 3.11) $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{Cube\text{-}D}^{\mathtt{rrs}})$ (Obs. 3.11)
*DoubleLongEq	Sat	$\mathtt{D}^{\mathtt{rrs}} = \mathtt{D}^{\mathtt{trv}}$	$\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{Cube\text{-}LD})$ (Lemma 4.57) $\mathtt{QCDCL}^{\mathtt{cube}}$ with $\mathtt{D}^{\mathtt{rrs}}$(Lemma 4.57)	QCDCL(Lemma 4.55) QCDCL with $\mathtt{D}^{\mathtt{rrs}}$(Lemma 4.55)
*PreRRSTrapdoor	Unsat	$\mathtt{D}^{\mathtt{rrs}} = \{(u,b),(v,b),(p,q)\}$	$\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{No\text{-}Cube})$ (Lemma 4.60)	$\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{Cube\text{-}LD})$ (Lemma 4.61)
*StdDepTrap	Sat	$\mathtt{D}^{\mathtt{std}} = \{(u,x),(w_1,e_1),(w_2,e_2)\}$	$\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{std}}, \mathtt{No\text{-}Cube})$ (Lemma 4.66)	$\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{Cube\text{-}LD})$ (Lemma 4.68)

Table 4.1: Formulas, their dependencies, and ease/hardness of refutations.
"$\mathtt{QCDCL}$ with $\mathtt{D}^{\mathtt{rrs}}$" includes $\{\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}, \mathtt{No\text{-}Cube})\}$ where $\mathtt{D} \in \{\mathtt{D}^{\mathtt{trv}}, \mathtt{D}^{\mathtt{rrs}}\}$, as well as $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{No\text{-}Cube})$.
"$\mathtt{QCDCL}^{\mathtt{cube}}$ with $\mathtt{D}^{\mathtt{rrs}}$" includes $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}, \mathtt{CubePol})$ as well as $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{CubePol})$,
where $\mathtt{D} \in \{\mathtt{D}^{\mathtt{trv}}, \mathtt{D}^{\mathtt{rrs}}\}$ and $\mathtt{CubePol} \in \{\mathtt{Cube\text{-}LD}, \mathtt{Cube\text{-}D}^{\mathtt{rrs}}\}$.

Chapter 5

Comparing the Relative Strengths of the QCDCL proof systems with Dependency Schemes

In the previous chapter we saw the hardness of formulas in various QCDCL based systems with dependencies. In this chapter we aim to analyse the QCDCL proof systems with dependency schemes in terms of their relative strength with respect to each other and some other systems.

5.1 QCDCL with LEV-ORD decisions and no cube-learning

Since, cube-learning is not an essential tool for the system to be refutationally complete, in this section we choose to ignore its presence and focus on QCDCL systems with dependency schemes but without cube-learning. In this section we restrict ourselves to the $\mathrm{D}^{\mathrm{rrs}}$ scheme and LEV-ORD decisions.

First we observe that the four versions of QCDCL that use or do not use $\mathtt{D}^{\mathtt{rrs}}$ in either of the two ways are all pairwise incomparable.

Theorem 5.1. *For* $\mathtt{D}_1, \mathtt{D}_2 \in \{\mathtt{D}^{\mathtt{trv}}, \mathtt{D}^{\mathtt{rrs}}\}$*, the proof systems* $\mathtt{D}_1 + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}_2, \mathtt{No\text{-}Cube})$ *are pairwise incomparable.*

Proof. Of the four systems under consideration, the Trapdoor formulas are hard only for $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{No\text{-}Cube})$ (Lemmas 4.14 and 4.16) and the Dep-Trap formulas are easy only in the same (Lemmas 4.27 and 4.28). Hence $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{No\text{-}Cube})$ is incomparable with all three systems obtained by adding $\mathtt{D}^{\mathtt{rrs}}$.

Among the three systems using $\mathtt{D}^{\mathtt{rrs}}$ in some manner, the TwoPHPandCT formulas are hard only in $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{No\text{-}Cube})$ (Lemmas 4.32 and 4.33), while the PreDepTrapformulas are easy only in the same (Lemmas 4.42 and 4.43). Hence $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{No\text{-}Cube})$ is incomparable with the systems that use $\mathtt{D}^{\mathtt{rrs}}$ preprocessing.

Finally, the systems $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{No\text{-}Cube})$ and $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{No\text{-}Cube})$ are separated by the formulas RRSTrapEq easy in the latter and hard in the former (Lemmas 4.36 and 4.37), and the formulas PropDep-Trap hard in the latter and easy in the former(Lemma 4.49). □

Next, we see that the three new proof systems generated by adding $\mathtt{D}^{\mathtt{rrs}}$ to QCDCL are incomparable with the resolution-based systems Q-Res, QU-Res, and $\mathtt{Q}(\mathtt{D}^{\mathtt{rrs}})$-Res.

Theorem 5.2. *Any two proof systems* $\mathtt{P}_1 \in \{\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{No\text{-}Cube}), \mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{No\text{-}Cube}), \mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{No\text{-}Cube})\}$*, and* $\mathtt{P}_2 \in \{\mathtt{Q\text{-}Res}, \mathtt{Q}(\mathtt{D}^{\mathtt{rrs}})\text{-}\mathtt{Res}, \mathtt{QU\text{-}Res}\}$*, are incomparable.*

Proof. The QParity formulas require exponential size refutations in $\mathtt{P}_2$ (Lemma 4.4) but have polynomial size refutations in $\mathtt{P}_1$ (Lemma 4.5).

The Dep-Trap formulas have constant size refutations in $\mathtt{P}_2$ (Lemma 4.26) but require exponential size refutations in $\mathtt{P}_1$ (Lemma 4.28). □

Finally, we observe that even when we add cube-learning to standard QCDCL, the system $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{Cube\text{-}LD})$ is still incomparable with all the three versions of QCDCL with dependency scheme $\mathtt{D}^{\mathtt{rrs}}$ added, but without cube learning.

Theorem 5.3. *Every proof system in* $\{\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{No\text{-}Cube}), \mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{No\text{-}Cube}), \mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{No\text{-}Cube})\}$ *is incomparable with* $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{Cube\text{-}LD})$.

Proof. The TwinEq formulas require exponential size refutations in $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{Cube\text{-}LD})$ ([BPB24]) but have poly-size refutations in the others (Lemma 4.22).

The Dep-Trap formulas have short refutations in $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{Cube\text{-}LD})$ (Lemma 4.27) but require exponential size refutations in the others (Lemma 4.28). □

The Figures 5.1 and 5.2 provide an overview of the relative strengths of the QCDCL proof systems with respect to each other that have been established by the theorems in this section.

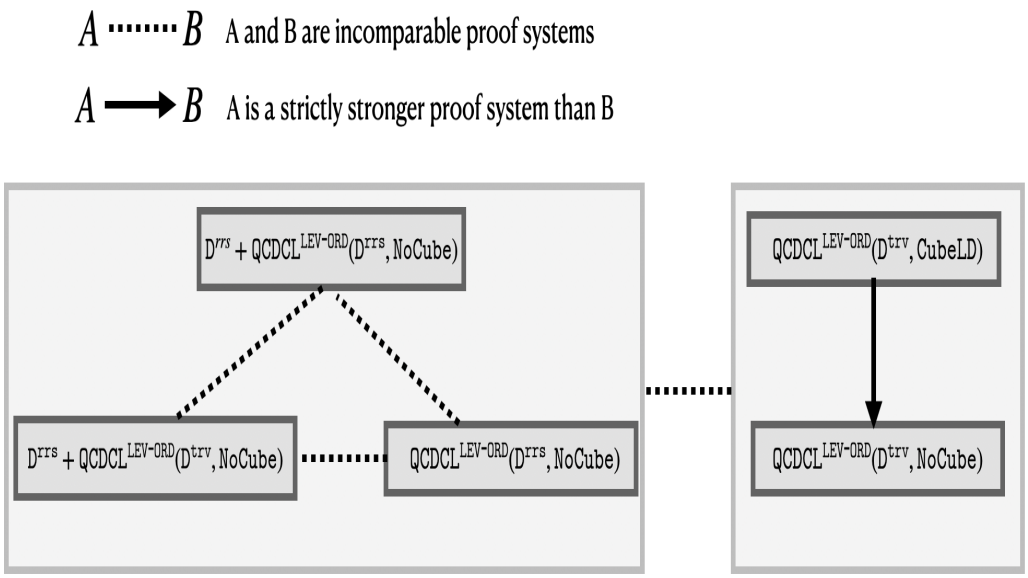

Figure 5.1: Relative strength between QCDCL proof systems with $\mathtt{D}^{\mathtt{rrs}}$ with respect to those without.

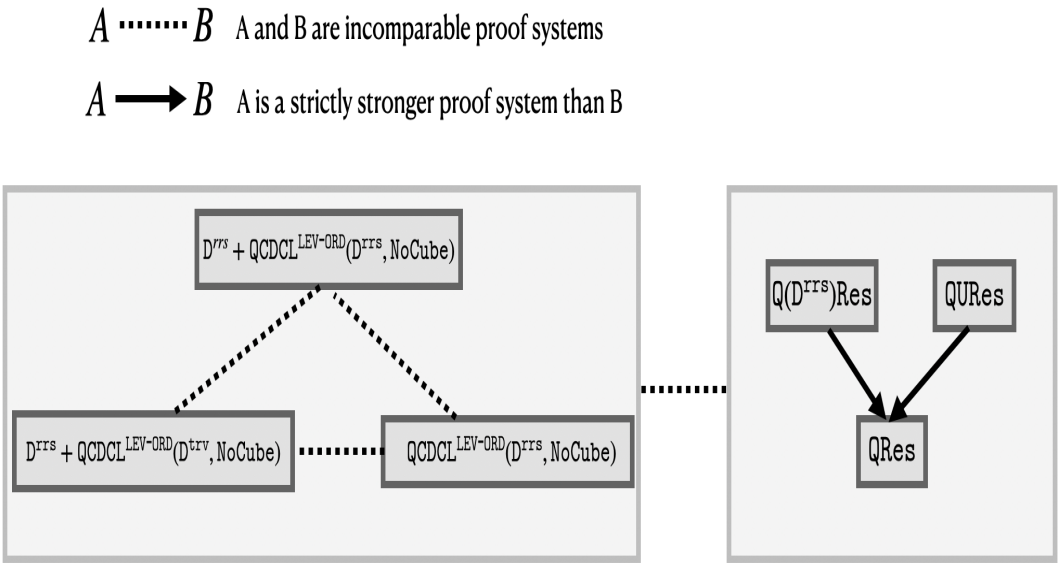

Figure 5.2: Relative strength of QCDCL proof systems with $\mathtt{D}^{\mathtt{rrs}}$ to some resolution-based proof systems.

5.2 QCDCL with LEV-ORD decisions and cube-learning

In the previous section we only saw dependency schemes aided QCDCL systems, where the QCDCL systems did not have cube-learning enabled. However as [BPB24] showed that even for refutations cube-learning can provide an advantage. To see whether that is the case even in the presence of dependency schemes, in this section we study the effect of dependency schemes on QCDCL systems with cube learning.

The first theorem shows that cube-learning in any form provably adds strength to the system even if dependency schemes are being used.

Theorem 5.4. *For* $\mathtt{D}_1, \mathtt{D}_2 \in \{\mathtt{D}^{\mathtt{trv}}, \mathtt{D}^{\mathtt{std}}, \mathtt{D}^{\mathtt{rrs}}\}$ *and* $\mathtt{CubePol} \in \{\mathtt{Cube\text{-}LD}, \mathtt{Cube\text{-}D}_2\}$ *the proof system* $\mathtt{D}_1 + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}_2, \mathtt{CubePol})$ *is strictly stronger than* $\mathtt{D}_1 + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}_2, \mathtt{No\text{-}Cube})$.

Proof. From Observation 3.10, since any refutation without cube-learning is also a refutation in the corresponding system with cube-learning, the systems with cube-learning are at least as strong as the corresponding systems without cube learning.

The DoubleLongEq formulas show that they are in fact strictly stronger; see the bounds in Lemmas 4.55 and 4.57. □

Next, we consider systems with $\mathtt{D}^{\mathtt{rrs}}$ and cube-learning in relation to the base QCDCL proof systems that do not use dependency schemes. These systems are incomparable to $\mathtt{D}_1 + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}_2, \mathtt{No\text{-}Cube})$ where $(\mathtt{D}_1, \mathtt{D}_2) \in \{(\mathtt{D}^{\mathtt{trv}}, \mathtt{D}^{\mathtt{rrs}}), (\mathtt{D}^{\mathtt{rrs}}, \mathtt{D}^{\mathtt{trv}}), (\mathtt{D}^{\mathtt{rrs}}, \mathtt{D}^{\mathtt{rrs}})\}$ (by Theorem 5.1). We show here that the presence of cube learning makes no difference; the systems still remain incomparable.

First we show that, the systems with $\mathtt{D}^{\mathtt{rrs}}$ even in the presence of cube-learning are incomparable to the QCDCL systems without dependency schemes (with and without

cube-learning).

Theorem 5.5. *Any proof systems* $\mathtt{P}_1, \mathtt{P}_2$ *are incomparable, where*

$$\mathtt{P}_1 \in \{\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{CubePol}) \mid \mathtt{CubePol} \in \{\mathtt{No\text{-}Cube}, \mathtt{Cube\text{-}LD}, \mathtt{Cube\text{-}D}^{\mathtt{trv}}\}\} \text{ and}$$

$$\mathtt{P}_2 \in \left\{ \mathtt{D}_1 + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}_2, \mathtt{CubePol}) \mid \begin{array}{r} (\mathtt{D}_1, \mathtt{D}_2) \in \{(\mathtt{D}^{\mathtt{trv}}, \mathtt{D}^{\mathtt{rrs}}), (\mathtt{D}^{\mathtt{rrs}}, \mathtt{D}^{\mathtt{trv}}), (\mathtt{D}^{\mathtt{rrs}}, \mathtt{D}^{\mathtt{rrs}})\}, \\ \mathtt{CubePol} \in \{\mathtt{Cube\text{-}LD}, \mathtt{Cube\text{-}D}_2\} \end{array} \right\}$$

Proof. There are eighteen incomparability claims expressed so thirty-six separations are required! Fortunately, just two formulas establish all the desired separations.

From Lemmas 4.27 and 4.28, the `Dep-Trap` formulas require exponential size refutations in every system in $\mathtt{P}_2$, but have polynomial size refutations in every system in $\mathtt{P}_1$.

The bounds from [BPB24] along with Observation 3.10 imply that the `TwinEq` formulas require exponential size refutations in every system in $\mathtt{P}_1$, but have polynomial size refutations in every system in $\mathtt{P}_2$.

Hence every proof system in $\mathtt{P}_1$ is incomparable with every proof system in $\mathtt{P}_2$ □

Next we show that, the different ways of adding $\mathtt{D}^{\mathtt{rrs}}$ even in the presence of cube-learning yields pairwise incomparable systems.

Theorem 5.6. *For a fixed* $\mathtt{CubePol} \in \{\mathtt{Cube\text{-}LD}, \mathtt{Cube\text{-}D}^{\mathtt{rrs}}\}$*, the three systems* $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{CubePol})$*,* $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{CubePol})$*, and* $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{CubePol})$ *are pairwise incomparable.*

Proof. It can be seen that `TwoPHPandCT` formulas are easy to refute when $\mathtt{D}^{\mathtt{rrs}}$ is used in preprocessing i.e. in $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{CubePol})$ and $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{CubePol})$ (Lemma 4.32), but hard otherwise i.e. $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{CubePol})$(Lemma 4.33) . On the other hand, the `PreDepTrap`

formulas are hard to refute if preprocessed by $\mathtt{D}^{\mathtt{rrs}}$(Lemma 4.43), but easy otherwise(Lemma 4.42). Together, they witness that $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{CubePol})$ is incomparable with $\mathtt{D}^{\mathtt{rrs}}+\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{CubePol})$ and $\mathtt{D}^{\mathtt{rrs}}+\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{CubePol})$.

Further, the PropDep-Trap formulas are easy to refute if the propagations and clause learning do not use $\mathtt{D}^{\mathtt{rrs}}$ Lemma 4.47, but become hard if $\mathtt{D}^{\mathtt{rrs}}$ is used(Lemma 4.48). This is independent of whether preprocessing is used and whether cube-learning is switched on. On the other hand, the PreRRSTrapdoor formulas show that using $\mathtt{D}^{\mathtt{rrs}}$ in propagations can be advantageous, Lemmas 4.60 and 4.61. Together, these two formulas witness $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{CubePol})$ and $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{CubePol})$ are incomparable. □

Finally we show that, adding $\mathtt{D}^{\mathtt{rrs}}$ to QCDCL in one-way in the presence of cube-learning is incomparable to adding $\mathtt{D}^{\mathtt{rrs}}$ in a different way to a QCDCL system without cube-learning.

Theorem 5.7. *For any* $\mathtt{CubePol} \in \{\mathtt{Cube\text{-}LD}, \mathtt{Cube\text{-}D}^{\mathtt{rrs}}\}$*, adding* $\mathtt{D}^{\mathtt{rrs}}$ *to the proof system* $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{CubePol})$ *in one way and to* $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{No\text{-}Cube})$ *in any different way yields incomparable proof systems.*

1. $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{CubePol})$ *is incomparable with* $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{No\text{-}Cube})$ *and* $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{No\text{-}Cube})$.
2. $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{CubePol})$ *is incomparable with* $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{No\text{-}Cube})$ *and* $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{No\text{-}Cube})$.
3. $\mathtt{D}^{\mathtt{rrs}}+\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{CubePol})$ *incomparable with* $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{No\text{-}Cube})$ *and* $\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{No\text{-}Cube})$.

Proof. The TwoPHPandCT formulas are easy to refute if and only if preprocessed by $\mathtt{D}^{\mathtt{rrs}}$, irrespective of whether or not cube-learning is used and whether or not $\mathtt{D}^{\mathtt{rrs}}$ is used in propagation and learning (Lemmas 4.32 and 4.33). The situation is exactly

reversed for the PreDepTrap formulas, which are easy to refute if and only if not preprocessed by $\mathtt{D}^{\mathtt{rrs}}$ (Lemmas 4.42 and 4.43). Together, these show eight of the twelve claimed incomparability relations, namely

$$\begin{aligned}
&\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{CubePol}) \text{ and } && \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{No\text{-}Cube}),\\
&\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{CubePol}) \text{ and } && \mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{No\text{-}Cube}),\\
&\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{CubePol}) \text{ and } && \mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{No\text{-}Cube}),\\
&\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{CubePol}) \text{ and } && \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{No\text{-}Cube}).
\end{aligned}$$

By Lemmas 4.47 to 4.49, the PropDep-Trap formulas are easy to refute if and only if $\mathtt{D}^{\mathtt{rrs}}$ is not used for propagation and learning, irrespective of its use in preprocessing, and irrespective of whether or not cube-learning is used. On the other hand, from Lemmas 4.60 to 4.62 the PreRRSTrapdoor formulas are easy to refute if $\mathtt{D}^{\mathtt{rrs}}$ is used for preprocessing and in propagation and learning, but not if it is used only for preprocessing. Together, these show the remaining four claimed incomparability relations, namely

$$\begin{aligned}
&\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{CubePol}) \text{ and } && \mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{No\text{-}Cube})\\
&\mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{rrs}}, \mathtt{CubePol}) \text{ and } && \mathtt{D}^{\mathtt{rrs}} + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{No\text{-}Cube}).
\end{aligned}$$

□

We now shift our focus to the $\mathtt{D}^{\mathtt{std}}$ scheme. Theorem 5.4 showed that adding cube-learning to $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{std}}, \mathtt{No\text{-}Cube})$ adds strength to the system, but we now see that adding $\mathtt{D}^{\mathtt{std}}$ to $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{No\text{-}Cube})$ is orthogonal to switching on cube-learning in $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{No\text{-}Cube})$.

Theorem 5.8. *For* $\mathtt{CubePol} \in \{\mathtt{Cube\text{-}LD}, \mathtt{Cube\text{-}D}^{\mathtt{std}}\}$*, the proof systems* $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{std}}, \mathtt{No\text{-}Cube})$ *and* $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{CubePol})$ *are incomparable.*

Proof. For $\mathtt{CubePol} \in \{\mathtt{Cube\text{-}LD}, \mathtt{Cube\text{-}D}^{\mathtt{std}}\}$, the Equality formulas have polynomial size $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{CubePol})$ refutations, but require exponential size $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{std}}, \mathtt{No\text{-}Cube})$ refutations. On the other hand, the newly defined StdDepTrap formulas have polynomial size $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{std}}, \mathtt{No\text{-}Cube})$ refutations

but require exponential size $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{trv}}, \mathtt{CubePol})$ refutations (Lemmas 4.66 and 4.68). □

We have so far studied the addition of two different dependency schemes $\mathtt{D}^{\mathtt{rrs}}$ and $\mathtt{D}^{\mathtt{std}}$ to level-order QCDCL systems with and without cube-learning. A natural question that arises is how do they compare with each other? Can we say that the addition of one dependency scheme is strictly better than the other? The mere fact that $\mathtt{D}^{\mathtt{rrs}}$ is a refinement of $\mathtt{D}^{\mathtt{std}}$ (more general, eliminates more dependencies) does not make it better; for that matter, $\mathtt{D}^{\mathtt{rrs}}$ is a refinement of $\mathtt{D}^{\mathtt{trv}}$, but using it can be a disadvantage for some formulas (see formula Dep-Trap in Section 4.6). Similarly, we prove below that neither of $\mathtt{D}^{\mathtt{rrs}}$ and $\mathtt{D}^{\mathtt{std}}$ has a proof-theoretic advantage over the other, irrespective of the presence or absence of cube-learning.

Theorem 5.9. *For any* $(\mathtt{D}_1, \mathtt{D}_2) \in \{(\mathtt{D}^{\mathtt{trv}}, \mathtt{D}^{\mathtt{rrs}}), (\mathtt{D}^{\mathtt{rrs}}, \mathtt{D}^{\mathtt{trv}}), (\mathtt{D}^{\mathtt{rrs}}, \mathtt{D}^{\mathtt{rrs}})\}$, *and* $\mathtt{CubePol} \in \{\mathtt{No\text{-}Cube}, \mathtt{Cube\text{-}LD}, \mathtt{Cube\text{-}D}^{\mathtt{std}}\}$, *the proof systems* $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{std}}, \mathtt{CubePol})$ *and* $\mathtt{D}_1 + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}_2, \mathtt{CubePol})$ *are incomparable.*

Proof. The Trapdoor and Dep-Trap formulas bear witness; the former are easy in $\mathtt{D}_1 + \mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}_2, \mathtt{CubePol})$ but hard in $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D}^{\mathtt{std}}, \mathtt{CubePol})$, whereas the situation is reversed for the latter. □

The Figures 5.3 to 5.6 summarize the results obtained in this section.

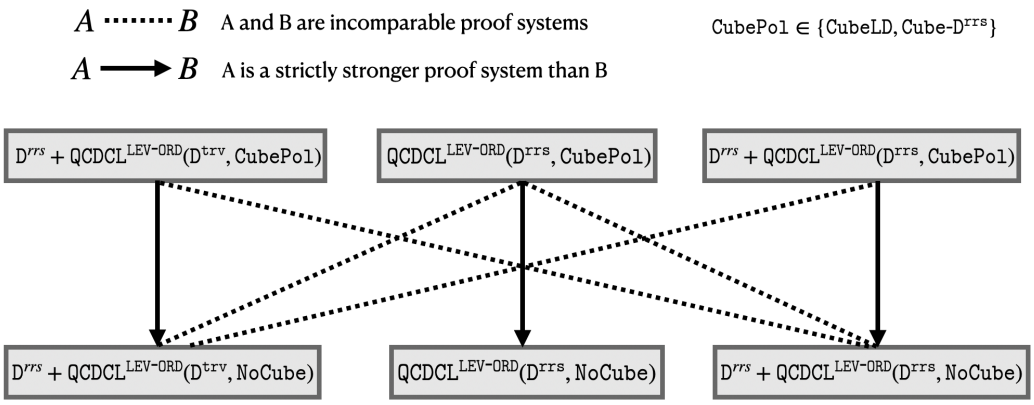

Figure 5.3: Relative strength between QCDCL proof systems with cube-learning and $\mathtt{D}^{\mathtt{rrs}}$.

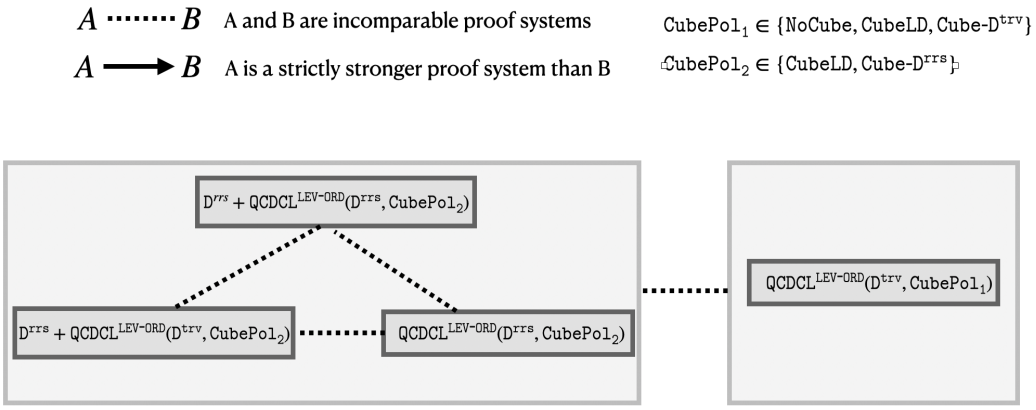

Figure 5.4: Strength of QCDCL proof systems with and without $\mathtt{D}^{\mathtt{rrs}}$ in the presence of cube-learning.

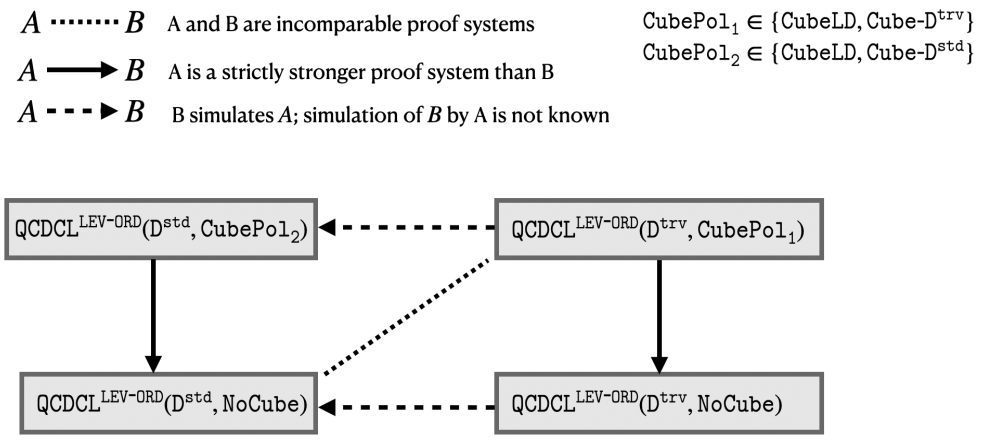

Figure 5.5: Strength of QCDCL proof systems with and without $\mathtt{D}^{\mathtt{std}}$.

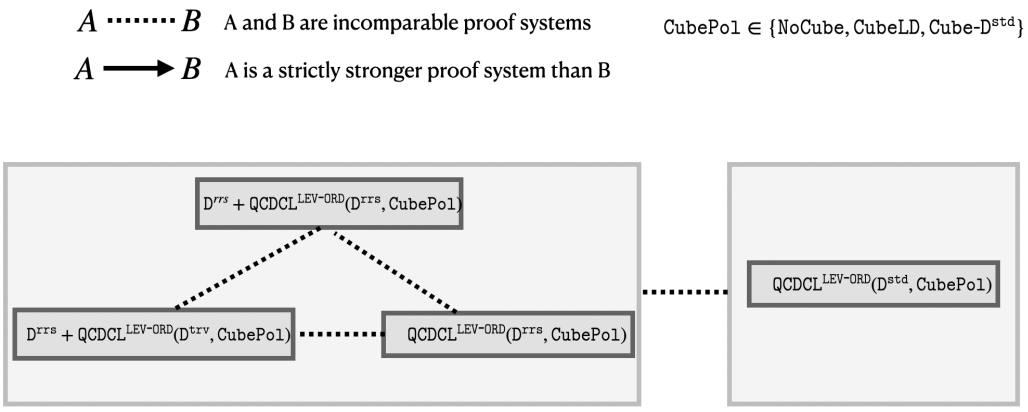

Figure 5.6: Relative stregth between QCDCL proof systems using $\mathtt{D}^{\mathtt{rrs}}$ and $\mathtt{D}^{\mathtt{std}}$.

5.3 QCDCL based proof systems with D-ORD decision policy

So far we have restricted our study to QCDCL systems where the decision policy has been restricted to LEV-ORD. We have seen that, in such LEV-ORD restricted systems, incorporating dependency schemes into propagation and learning processes does not always yield benefits. Certain "trap" motivated formulas can render the addition of dependency schemes disadvantageous when decisions are constrained to the LEV-ORD decision policy.

The critical question that arises is what happens when decisions are not restricted to LEV-ORD. When dependency schemes are allowed to influence the decision policy (i.e., the system adopts the D-ORD decision policy, does provide any advantage? The answer to this question is yes. We show that relaxing the decision policy from LEV-ORD to D-ORD confers a strict advantage. This means that adopting the D-ORD decision policy adds strength to the system when compared to the LEV-ORD policy.

Theorem 5.10. *For dependency schemes* $\mathtt{D}_1 \in \{\mathtt{D}^{\mathtt{rrs}}, \mathtt{D}^{\mathtt{std}}\}$ *and* $\mathtt{D}_2 \in \{\mathtt{D}^{\mathtt{rrs}}, \mathtt{D}^{\mathtt{std}}, \mathtt{D}^{\mathtt{trv}}\}$*, and for policy* $\mathtt{CubePol} \in \{\mathtt{No\text{-}Cube}, \mathtt{Cube\text{-}LD}, \mathtt{Cube\text{-}D}_2\}$*, the proof*

system $\mathtt{QCDCL}^{\mathtt{D_1\text{-}ORD}}(\mathtt{D_2}, \mathtt{CubePol})$ *p-simulates* $\mathtt{QCDCL}^{\mathtt{LEV\text{-}ORD}}(\mathtt{D_2}, \mathtt{CubePol})$ *and is not simulated by it.*

(For $D_1 = \mathtt{D}^{\mathtt{std}}$, an advantage over LEV-ORD was noted already in [Lon12].)

Proof. The p-simulation follows from Observation 3.9. To show that there is no reverse simulation, consider the TwoPHPandCT formulae. These have an unsatisfiable matrix, so by Observation 3.11, it suffices to show lower and upper bounds for $\mathtt{CubePol} = \mathtt{No\text{-}Cube}$. The hardness for the LEV-ORD systems is shown in Lemma 4.33 , and we show that short refutations using $\mathtt{D_1}$-ORD come from allowing to start trails with the v-variables.

To show that they have short refutations in $\mathtt{QCDCL}^{\mathtt{D_1\text{-}ORD}}(\mathtt{D_2}, \mathtt{No\text{-}Cube})$. Note that for $\mathtt{D_1} \in \{\mathtt{D}^{\mathtt{rrs}}, \mathtt{D}^{\mathtt{std}}\}$, the v, z_1, z_2 variables are completely independent from the u, x, y variables; neither (x_i, v) nor (y_i, v) is in D_1 for any i. Therefore using the $\mathtt{D_1}$-ORD decision policy we can start trails with a decision on the v, z variables allowing for a short (constant-sized) refutation sketched below.

$$\mathcal{T}_1 = \bar{\mathbf{v}}; \mathbf{z_1}, z_2, \square$$

$\mathtt{ante}(\square) = v \vee \bar{z}_1 \vee \bar{z}_2$, $\mathtt{ante}(z_2) = v \vee \bar{z}_1 \vee z_2$. From $\mathcal{T}_1$ if $\mathtt{D_2} = \mathtt{D}^{\mathtt{trv}}$ or $\mathtt{D}^{\mathtt{std}}$, then we learn $(v \vee \bar{z}_1)$, and if $\mathtt{D_2} = \mathtt{D}^{\mathtt{rrs}}$, then we learn $\bar{z}_1$.

Now, in the case $\mathtt{D_2} = \mathtt{D}^{\mathtt{trv}}$ or $\mathtt{D}^{\mathtt{rrs}}$,

$$\mathcal{T}_2 = \bar{\mathbf{v}}, \bar{z}_1, z_2, \square$$

$\mathtt{ante}(\square) = v \vee z_1 \vee \bar{z}_2$, $\mathtt{ante}(z_2) = v \vee z_1 \vee z_2$, $\mathtt{ante}(\bar{z}_1) = v \vee \bar{z}_1$, allowing us to learn $\square$.

Similarly for $\mathtt{D_2} = \mathtt{D}^{\mathtt{rrs}}$,

$$\mathcal{T}_2 = \bar{z}_1, z_2, \square$$

$\texttt{ante}(\square) = v \vee z_1 \vee \bar{z}_2$, $\texttt{ante}(z_2) = v \vee z_1 \vee z_2$, $\texttt{ante}(\bar{z}_1) = \bar{z}_1$, allowing us to learn $\square$. $\square$

Next, using the DoubleLongEq formulas we show that the LDQ(D)-Res system are strictly stronger than $\texttt{QCDCL}^{\texttt{D-ORD}}(\texttt{D}, \texttt{No-Cube})$ for $\texttt{D} \in \{\texttt{D}^{\texttt{trv}}, \texttt{D}^{\texttt{std}}, \texttt{D}^{\texttt{rrs}}\}$.

Recall the DoubleLongEq formulas:

Formula 4.50. *[*$\texttt{DoubleLongEq}_n$*]* *The* $\texttt{DoubleLongEq}_n$ *formula has the prefix* $\exists x_1 \cdots x_n \; \forall u_1 \cdots u_n \; \exists t_1 \cdots t_n$ *and the PCNF matrix*

$$\underbrace{(\bar{t}_1 \vee \cdots \vee \bar{t}_n)}_{T_n} \wedge \bigwedge_{i=1}^{n} \left[\underbrace{(x_i \vee u_i \vee t_i)}_{A_i} \wedge \underbrace{(\bar{x}_i \vee \bar{u}_i \vee t_i)}_{B_i} \right] \wedge$$
$$\underbrace{(\bar{u}_1 \vee \cdots \bar{u}_n \vee \bar{t}_1 \vee \cdots \bar{t}_n)}_{UT_n} \wedge \underbrace{(\bar{u}_1 \vee \cdots \bar{u}_n \vee t_1 \vee \cdots t_n)}_{UT'_n}$$

We now prove the theorem

Theorem 5.11. *For* $\texttt{D} \in \{\texttt{D}^{\texttt{trv}}, \texttt{D}^{\texttt{std}}, \texttt{D}^{\texttt{rrs}}\}$*, the proof system* LDQ(D)-Res *p-simulates* $\texttt{QCDCL}^{\texttt{D-ORD}}(\texttt{D}, \texttt{No-Cube})$ *and is not simulated by it.*

Proof. By definition, any refutation extracted from the $\texttt{QCDCL}^{\texttt{D-ORD}}(\texttt{D}, \texttt{No-Cube})$ is a valid LDQ(D)-Res refutation.

To show strictness consider the DoubleLongEq formulas. For these formulas, $\texttt{D}^{\texttt{trv}}$, $\texttt{D}^{\texttt{std}}$ and $\texttt{D}^{\texttt{rrs}}$ are equivalent (Proposition 4.51). Therefore, in this case D-ORD is equivalent to LEV-ORD, and by Lemma 4.55 we know that these formulas are hard for $\texttt{QCDCL}^{\texttt{D-ORD}}(\texttt{D}, \texttt{No-Cube})$.

To see that they are easy to refute in LDQ(D)-Res, it is enough to show a short refutation in LDQ-Res. To see this refutation, consider all A_i and B_i clauses, by the rules of long-distance resolution Figure 2.3, we can resolve these clauses on x_i to get n clauses of the form $u_i \vee \bar{u}_i \vee t_i$. Resolving each of these clauses with the clause T_n

gives us a purely universal clause of the form $\bigvee_{i=1}^{n} u_i \vee \bar{u}_i$, which can be universally reduced to yield the empty clause completing the refutation. □

To extend the separation to systems that allow cube-learning, we slightly modify the DoubleLongEq formula. We add trailing variables to the prefix that encode PHP. This would not only preserve that the formula is false, but also make the formula matrix unsatisfiable. Therefore, cube-learning will never be able to help in any QCDCL refutation. The LDQ-Res refutation of this modified formula would remain the same as for DoubleLongEq since the new clauses are completely disjoint. Finally, note that the hardness result for $\mathtt{QCDCL}^{\mathtt{D\text{-}ORD}}(\mathtt{D}, \mathtt{No\text{-}Cube})$ remains valid even after this modification.

Thus,

Corollary 5.12. *For* $\mathtt{CubePol} \in \{\mathtt{Cube\text{-}LD}, \mathtt{Cube\text{-}D}\}$ *in* $\mathtt{D} \in \{\mathtt{D}^{\mathtt{trv}}, \mathtt{D}^{\mathtt{std}}, \mathtt{D}^{\mathtt{rrs}}\}$*, the proof system* $\mathtt{LDQ}(\mathtt{D})\text{-}\mathtt{Res}$ *p-simulates* $\mathtt{QCDCL}^{\mathtt{D\text{-}ORD}}(\mathtt{D}, \mathtt{CubePol})$ *and is not simulated by it.*

However these separations do not rely on dependency schemes for the separation and do not hinge on the power of the dependency schemes to impact trail-based versus line-based proof systems. These just hinge on trail-based vs line-based proof systems without consideration of dependency schemes.

Chapter 6

Conclusion

QBF solvers that implememnt QCDCL are among the strongest known algorithms to solve QBFs. When studying it from a proof-theoretic view, the fact that QCDCL runs can be simulated by the `LDQ-Res` proof system was established [ZM02; BJ12]. However, the first formalisation of `QCDCL` as a standalone proof system came much later in [BB23a], which was the first to define the `QCDCL` system as the logical proof system underlying QCDCL solvers.

This thesis is the first to study the effects of adding the dependency scheme heuristic to the `QCDCL` proof system. We consider and formalize the following ways that a dependency schemes can potentially affect the proof system :

- As a preprocessing tool
- In the decision order governing trails
- In clause propagation and learning
- in cube propagation and learning

We introduce new notation to denote such `QCDCL` proof systems that use the dependency scheme heuristic, and we establish the soundness and completeness of a vast

family of such proof systems for normal dependency schemes. We show that D-ORD QCDCL systems are provably stronger than LEV-ORD systems.

We construct several formulae families and prove bounds for their hardness in the newly minted dependency scheme aided QCDCL systems. Using these bounds we establish strength relations between the different families of QCDCL systems with dependencies, which show that for the case of $\mathtt{D}^{\mathtt{rrs}}$ when LEV-ORD QCDCL systems are used, adding $\mathtt{D}^{\mathtt{rrs}}$ may not always strengthen the proof system.

Looking ahead, several research directions emerge. While these proof systems abstract many aspects of practical QBF solvers, they do not model solvers entirely. Thus, an interesting avenue would be to design and implement a solver based directly on these proof systems and empirically compare its performance with existing solvers. From a theoretical standpoint, future work could aim to construct a proof system that more precisely captures the behavior of QBF solvers. Additionally, further exploration of D-ORD QCDCL proof systems—particularly whether they can be separated from or shown to be equivalent to the underlying LDQ(D)-Res proof system—presents another promising direction. Finally, one of the outstanding open problems is determining whether LDQ(D)-TermRes is a sound proof system for non-trivial dependency schemes. Since our lower bounds all rely on unsatisfiable matrices, even if LDQ(D)-TermRes is sound our hardness results in this thesis would all carry over.

Bibliography

[BB20] Olaf Beyersdorff and Joshua Blinkhorn. "Dynamic QBF Dependencies in Reduction and Expansion". In: *ACM Trans. Comput. Log.* 21.2 (2020), 8:1–8:27. DOI: 10.1145/3355995.

[BB23a] Olaf Beyersdorff and Benjamin Böhm. "Understanding the Relative Strength of QBF CDCL Solvers and QBF Resolution". In: *Logical Methods in Computer Science* 19.2 (2023). preliminary version in ITCS 2021. DOI: 10.46298/lmcs-19(2:2)2023. URL: https://doi.org/10.46298/lmcs-19(2:2)2023.

[BB23b] Benjamin Böhm and Olaf Beyersdorff. "Lower Bounds for QCDCL via Formula Gauge". In: *J. Autom. Reason.* 67.4 (2023), p. 35. DOI: 10.1007/S10817-023-09683-1. URL: https://doi.org/10.1007/s10817-023-09683-1.

[BBH19] Olaf Beyersdorff, Joshua Blinkhorn, and Luke Hinde. "Size, Cost, and Capacity: A Semantic Technique for Hard Random QBFs". In: *Log. Methods Comput. Sci.* 15.1 (2019). DOI: 10.23638/LMCS-15(1:13)2019. URL: https://doi.org/10.23638/LMCS-15(1:13)2019.

[BCJ19] Olaf Beyersdorff, Leroy Chew, and Mikolás Janota. "New Resolution-Based QBF Calculi and Their Proof Complexity". In: *ACM Trans. Comput. Theory* 11.4 (2019), 26:1–26:42. DOI: 10.1145/3352155.

[BHM09] Armin Biere, Marijn Heule, and Hans van Maaren. *Handbook of satisfiability*. Vol. 185. IOS press, 2009.

[Bie+21] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, eds. *Handbook of Satisfiability - Second Edition*. Vol. 336. Frontiers in Artificial Intelligence and Applications. IOS Press, 2021. ISBN: 978-1-64368-160-3. DOI: 10.3233/FAIA336. URL: https://doi.org/10.3233/FAIA336.

[BJ12] Valeriy Balabanov and Jie-Hong R. Jiang. "Unified QBF certification and its applications". In: *Formal Methods Syst. Des.* 41.1 (2012), pp. 45–65. DOI: 10.1007/s10703-012-0152-6. URL: https://doi.org/10.1007/s10703-012-0152-6.

[BKS04] Paul Beame, Henry A. Kautz, and Ashish Sabharwal. "Towards Understanding and Harnessing the Potential of Clause Learning". In: *J. Artif. Intell. Res.* 22 (2004), pp. 319–351. DOI: 10.1613/JAIR.1410. URL: https://doi.org/10.1613/jair.1410.

[BPB24] Benjamin Böhm, Tomás Peitl, and Olaf Beyersdorff. "QCDCL with cube learning or pure literal elimination - What is best?" In: *Artif. Intell.* 336 (2024). preliminary version on IJCAI 2022, p. 104194. DOI: 10.1016/J.ARTINT.2024.104194. URL: https://doi.org/10.1016/j.artint.2024.104194.

[CCT87] William J. Cook, Collette R. Coullard, and György Turán. "On the complexity of cutting-plane proofs". In: *Discret. Appl. Math.* 18.1 (1987), pp. 25–38. DOI: 10.1016/0166-218X(87)90039-4. URL: https://doi.org/10.1016/0166-218X(87)90039-4.

[CEI96] Matthew Clegg, Jeffery Edmonds, and Russell Impagliazzo. "Using the Groebner basis algorithm to find proofs of unsatisfiability". In: *Proceed-*

ings of the twenty-eighth annual ACM symposium on Theory of computing. 1996, pp. 174–183.

[CM24] Abhimanyu Choudhury and Meena Mahajan. "Dependency Schemes in CDCL-Based QBF Solving: A Proof-Theoretic Study". In: *J. Autom. Reason.* 68.3 (2024). Preliminary version appeared in the 43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2023, p. 16. DOI: 10.1007/S10817-024-09707-4. URL: https://doi.org/10.1007/s10817-024-09707-4.

[CM25] Abhimanyu Choudhury and Meena Mahajan. "On the Interplay of Cube Learning and Dependency Schemes in QCDCL Proof Systems". (manuscript). 2025.

[CR79] Stephen A. Cook and Robert A. Reckhow. "The Relative Efficiency of Propositional Proof Systems". In: *J. Symb. Log.* 44.1 (1979), pp. 36–50. DOI: 10.2307/2273702. URL: https://doi.org/10.2307/2273702.

[DLL62] Martin Davis, George Logemann, and Donald W. Loveland. "A machine program for theorem-proving". In: *Communications of the ACM* 5.7 (1962), pp. 394–397. DOI: 10.1145/368273.368557. URL: https://doi.org/10.1145/368273.368557.

[DP60] Martin Davis and Hilary Putnam. "A computing procedure for quantification theory". In: *Journal of the ACM (JACM)* 7.3 (1960), pp. 201–215.

[Gel12] Allen Van Gelder. "Contributions to the Theory of Practical Quantified Boolean Formula Solving". In: *Principles and Practice of Constraint Programming - 18th International Conference, CP 2012, Québec City, QC, Canada, October 8-12, 2012. Proceedings*. Ed. by Michela Milano. Vol. 7514. Lecture Notes in Computer Science. Springer, 2012, pp. 647–663. DOI: 10.1007/978-3-642-33558-7_47.

[GNT06] Enrico Giunchiglia, Massimo Narizzano, and Armando Tacchella. "Clause/Term Resolution and Learning in the Evaluation of Quantified Boolean Formulas". In: *J. Artif. Intell. Res.* 26 (2006), pp. 371–416. DOI: 10.1613/JAIR.1959. URL: https://doi.org/10.1613/jair.1959.

[Hak85] Armin Haken. "The intractability of resolution". In: *Theoretical Computer Science* 39 (1985), pp. 297–308.

[JM15] Mikoláš Janota and João Marques-Silva. "Expansion-based QBF solving versus Q-resolution". In: 577 (2015), pp. 25–42.

[KKF95] Hans Kleine Büning, Marek Karpinski, and Andreas Flögel. "Resolution for Quantified Boolean Formulas". In: *Inf. Comput.* 117.1 (1995), pp. 12–18. DOI: 10.1006/inco.1995.1025. URL: https://doi.org/10.1006/inco.1995.1025.

[Kra95] Jan Krajícek. *Bounded arithmetic, propositional logic, and complexity theory*. Vol. 60. Encyclopedia of mathematics and its applications. Cambridge University Press, 1995. ISBN: 978-0-521-45205-2.

[LE17] Florian Lonsing and Uwe Egly. "DepQBF 6.0: A Search-Based QBF Solver Beyond Traditional QCDCL". In: *Proceedings of International Conference on Automated Deduction (CADE)*. 2017, pp. 371–384.

[Lon12] Florian Lonsing. "Dependency Schemes and Search-Based QBF Solving: Theory and Practice". Available online. Ph.D. thesis. Johannes Kepler University, Linz, Austria, 2012. URL: https://www.jku.at/en/institute-for-formal-modeling-and-verification/research/publications/.

[PD11] Knot Pipatsrisawat and Adnan Darwiche. "On the power of clause-learning SAT solvers as resolution engines". In: *Artif. Intell.* 175.2 (2011), pp. 512–525. DOI: 10.1016/J.ARTINT.2010.10.002. URL: https://doi.org/10.1016/j.artint.2010.10.002.

[PSS19a] Tomás Peitl, Friedrich Slivovsky, and Stefan Szeider. "Dependency Learning for QBF". In: *J. Artif. Intell. Res.* 65 (2019), pp. 180–208. DOI: 10.1613/JAIR.1.11529. URL: https://doi.org/10.1613/jair.1.11529.

[PSS19b] Tomás Peitl, Friedrich Slivovsky, and Stefan Szeider. "Long-Distance Q-Resolution with Dependency Schemes". In: *J. Autom. Reasoning* 63.1 (2019), pp. 127–155. DOI: 10.1007/s10817-018-9467-3. URL: https://doi.org/10.1007/s10817-018-9467-3.

[SS09] Marko Samer and Stefan Szeider. "Backdoor Sets of Quantified Boolean Formulas". In: *J. Autom. Reasoning* 42.1 (2009), pp. 77–97. DOI: 10.1007/s10817-008-9114-5. URL: https://doi.org/10.1007/s10817-008-9114-5.

[SS12] Friedrich Slivovsky and Stefan Szeider. "Computing Resolution-Path Dependencies in Linear Time ," in: *Theory and Applications of Satisfiability Testing - SAT 2012 - 15th International Conference, Trento, Italy, June 17-20, 2012. Proceedings*. Ed. by Alessandro Cimatti and Roberto Sebastiani. Vol. 7317. Lecture Notes in Computer Science. Springer, 2012, pp. 58–71. DOI: 10.1007/978-3-642-31612-8_6. URL: https://doi.org/10.1007/978-3-642-31612-8_6.

[SS16] Friedrich Slivovsky and Stefan Szeider. "Soundness of Q-resolution with dependency schemes". In: *Theor. Comput. Sci.* 612 (2016), pp. 83–101. DOI: 10.1016/j.tcs.2015.10.020. URL: https://doi.org/10.1016/j.tcs.2015.10.020.

[SS99] João P. Marques Silva and Karem A. Sakallah. "GRASP: A Search Algorithm for Propositional Satisfiability". In: *IEEE Trans. Computers* 48.5 (1999), pp. 506–521. DOI: 10.1109/12.769433. URL: https://doi.org/10.1109/12.769433.

[ZM02] Lintao Zhang and Sharad Malik. “Conflict driven learning in a quantified Boolean Satisfiability solver”. In: *Proceedings of the 2002 IEEE/ACM International Conference on Computer-aided Design, ICCAD 2002, San Jose, California, USA, November 10-14, 2002*. Ed. by Lawrence T. Pileggi and Andreas Kuehlmann. ACM / IEEE Computer Society, 2002, pp. 442–449. DOI: 10.1145/774572.774637. URL: https://doi.org/10.1145/774572.774637.