Proof Complexity for Quantified Boolean Formulas

Judith Clymo

School of Computing

University of Leeds

A thesis submitted for the degree of

Doctor of Philosophy

30th November 2020

Declaration

The candidate confirms that the work submitted is her own, except where work which has formed part of a jointly authored publication has been included. The contribution of the candidate and the other authors to this work has been explicitly indicated below. The candidate confirms that appropriate credit has been given within the thesis where reference has been made to the work of others.

Some parts of the work presented in this thesis have been published in the following articles:

Publications in Journals

1. Olaf Beyersdorff and Judith Clymo (2018), Relating size and width in variants of Q-Resolution, Information Processing Letters.

Refereed Contributions in Conference Proceedings

2. Olaf Beyersdorff, Leroy Chew, Judith Clymo and Meena Mahajan (2019), Short proofs in QBF expansion, Theory and Applications of Satisfiability Testing.
3. Leroy Chew and Judith Clymo (2019), The equivalences of refutational QRAT, Theory and Applications of Satisfiability Testing.
4. Leroy Chew and Judith Clymo (2020), How QBF expansion makes strategy extraction hard, Automated Reasoning.

The candidate confirms that the role of the authors in above jointly-authored publications are as follows:

- Chapters 4 and 5 contain work from 2. I was the main author. The proofs in this paper were a result of discussions between the authors.
- Chapters 6 and 7 contain work from 3. Chew and I were the main authors. The proofs in this paper were a result of discussions between the authors.
- Chapter 6 contains work from 4. Chew and I were the main authors. The proofs in this paper were a result of discussions between the authors.
- Chapter 8 contains work from 1. I was the main author. The proofs in this paper were a result of discussions between the authors.

Acknowledgements

First of all I would like to thank my excellent supervisors Olaf Beyersdorff and Brandon Bennett. Their direction and encouragement has been hugely valuable throughout my time as a PhD student. I am also grateful for discussions with collaborators and others in the research group, including Leroy Chew, Meena Mahajan, Barnaby Martin, Stefan Dantchev, Joshua Blinkhorn, Luke Hinde and Sarah Sigley.

The School of Computing at Leeds is a vibrant and supportive environment, I have greatly enjoyed the opportunity to attend lots of interesting talks, assist in teaching undergraduate classes, and be surrounded by so many impressive researchers.

Several generous people have proof-read various parts of this thesis – thanks to Noleen, Sam, Petar, Jake, Markus and Luke.

2020 has been a challenging year for everyone. Quite apart from writing a thesis, I would not have got through the year without the love and kindness of family and friends. So thank you to my husband, parents and sisters, and to Sarah, the best friend that anyone could wish for.

Abstract

Quantified Boolean formulas (QBF) extend the propositional satisfiability problem by allowing variables to be universally as well as existentially quantified. Deciding whether a QBF is true or false is PSPACE-complete and a wide range of mathematical and industrial problems can be expressed as QBFs. QBF proof complexity is the theoretical analysis of algorithmic techniques for solving QBFs.

We make a detailed comparison of the proof systems Q-Res, QU-Res, and $\forall\mathsf{Exp}+\mathsf{Res}$ which extend propositional Resolution with different rules for reasoning about universally quantified variables. We give new simulation and separation results between these proof systems under two natural restrictions, when the proofs are tree-like, and when the QBFs have bounded quantifier complexity.

We consider a strong QBF proof system, QRAT, proposed as a universal proof checking format. We show that, unless P = PSPACE, QRAT does not admit strategy extraction. This is proved by constructing a family of QBFs that have short QRAT proofs but whose strategies are hard to compute in general. We also explore why strategy extraction fails for QRAT, including presenting a restricted version of QRAT which does admit strategy extraction.

We study two results from propositional proof complexity and their analogues in QBF proof complexity, showing in both cases how the additional complexity of QBF solving compared to refuting propositional formulas causes these results to fail in the QBF setting.

Contents

List of Figures

Abbreviations

SAT	The propositional satisfiability problem
QBF	Quantified Boolean formula
CNF	Conjunctive normal form
PCNF	Prenex conjunctive normal form
DNF	Disjunctive normal form
DAG	Directed acyclic graph
DPLL	The algorithm of Davis, Putney, Logemann and Loveland
CDCL	Conflict driven clause learning
QDPLL	QBF version of the DPLL algorithm
QCDCL	QBF version of the CDCL algorithm
CEGAR	Counterexample guided abstraction refinement
$\forall$-Red	Universal reduction
Res	Resolution proof system
Frege	Frege proof systems
Q-Res	Q-Resolution proof system
QU-Res	QU-Resolution proof system
$\forall$Exp + Res	Expansion and resolution proof system
IR-calc	Instantiation and resolution proof system
IRM-calc	Instantiation, resolution and merging proof system
LD-Q-Res	Long-distance Q-Resolution proof system
LQU$^+$-Res	QU-Resolution with long distance resolution
QRAT	The QRAT proof system
QRAT(UR)	The QRAT proof system with universal reduction
QRAT+	The QRAT+ proof system
f + $\forall$red	Propositional proof system f augmented with the universal reduction rule
f + $\forall$Exp	Propositional proof system f augmented with the universal expansion rule

Notation

$\mathrm{vars}(\Phi)$	The set of variables in a formula Φ
$\exists_\Psi$	The set of existentially quantified variables in QBF Ψ
$\forall_\Psi$	The set of universally quantified variables in QBF Ψ
$\mathrm{lv}(x)$	The level of variable x in the quantifier prefix of a QBF
$\lvert\Phi\rvert$	The size of a formula Φ
$\Phi[f/x]$	The result of substituting Boolean function f for x throughout Φ
$\Phi[\alpha]$	The result of substituting variables in Φ according to the assignment α
$\Phi\vert_\alpha$	The result of applying $\Phi[\alpha]$ followed by simplification
$\alpha(\Phi)$	Φ evaluated under assignment α
$\alpha \circ \tau$	The extension of assignment α by τ
u^*	A merged literal
$\lfloor\alpha\rfloor_l$	The restriction of assignment α to variables with level less that $\mathrm{lv}(l)$
$\mathsf{subexp}_\pi(\Psi)$	The part of the expansion of Ψ used by a proof π
$\Phi \vdash_\mathsf{f} C$	Proof system f can derive C from Φ
$\lvert\pi\rvert$	The size of a proof π
$S_\mathsf{f}(\Phi)$	The minimum size of an f-proof of Φ

Chapter 1

Introduction

Proof complexity seeks to understand and analyse the computational resources required to prove or refute logical statements. Proof complexity is applicable to many different logics but the majority of research has focused on propositional logic, specifically on the size of refutations of unsatisfiable propositional formulas.

The Boolean satisfiability problem (SAT) is the problem of deciding whether a given formula in propositional logic can be satisfied (made to evaluate to true) by any assignment to the variables. Formulas for which the answer is 'yes' have a short and easily verifiable proof of this – the assignment that causes the formula to evaluate to true. Therefore, SAT belongs to the complexity class NP, which contains all decision problems for which 'yes' instances can be verified in deterministic polynomial time in the size of the input. In fact, it was shown by Cook (1971) that SAT is NP-complete, meaning that any problem in NP can be efficiently expressed as an instance of the SAT problem.

For propositional formulas with no satisfying assignment to the variables it is not known whether there always exists a proof of this which can be verified in polynomial time in the size of the formula. The problem of deciding whether a formula is unsatisfiable is coNP-complete, where coNP is the complement of NP and contains all decision problems for which 'no' instances can be verified in deterministic polynomial time.

Instead of supplying an assignment to the variables, a proof that a propositional formula is unsatisfiable is typically given as a sequence of applications of logical deductions. A proof system specifies which deductions can be applied and defines a method for checking the validity of a claimed proof. The proof must always be checkable in polynomial time in the size of the proof, but the proof may be super-polynomial in the size of the input.

Cook & Reckhow (1979) gave a precise formal definition of a proof system and showed an important connection between propositional proof complexity and computational complexity. A proof system is polynomially bounded if the size of the smallest proof (the number of symbols needed to write it) of any input formula is bounded above by a polynomial in the size of the input. If there is such a proof system for unsatisfiable propositional formulas then we are able to provide proofs of unsatisfiability which can be verified in polynomial time the size of the formula, meaning that this problem belongs to NP. It would follow from this that $\mathsf{coNP} = \mathsf{NP}$.

A famous open problem in computational complexity is to determine whether NP contains exactly the same decision problems as P, the class of problems that can be decided in deterministic polynomial time. Although problems in NP have proofs that can be verified efficiently, it is not known whether the proof can be found in polynomial time in the size of the problem instance. It is known that if $\mathsf{coNP} \neq \mathsf{NP}$ then $\mathsf{P} \neq \mathsf{NP}$. Therefore, a possible approach to gaining insight into this famous open problem of computational complexity is to study propositional proof systems and find formulas for which increasingly strong systems require super-polynomial sized proofs.

Quantified Boolean formulas (QBF) extend propositional logic by introducing universal and existential quantifiers for the Boolean variables. PSPACE is the complexity class containing all problems that can be decided by a deterministic machine using only a polynomial amount of memory space in the size of the input. Determining whether a QBF is true or false is the canonical PSPACE-complete problem. It is easy to show that $\mathsf{NP} \subseteq \mathsf{PSPACE}$, but it is not known whether the inclusion is proper. If there is a polynomially bounded proof system for QBFs then it follows that $\mathsf{NP} = \mathsf{PSPACE}$.

Aside from the possibility of gaining insight into fundamental open problems in computational complexity, there is another important reason for studying proof complexity. A considerable variety of important problems from mathematics, theoretical computer science, and industrial settings can be naturally expressed as instances of SAT. Despite the widely-held expectation that SAT is intractable in the worst case (i.e. that $\mathsf{P} \neq \mathsf{NP}$), research in SAT-solving algorithms has shown impressive progress in recent years and modern SAT solvers such as Kissat (Biere *et al.*, 2020), CryptoMiniSat (Soos *et al.*, 2009), MapleSAT (Liang *et al.*, 2016) and Glucose (Audemard & Simon, 2018) are able to solve instances with hundreds of thousands of variables.

In industry, SAT solvers are used to solve planning problems in artificial intelligence (Kautz & Selman, 1992) testing and verification (Larrabee, 1992) and resolving software package dependencies. On the theoretical side, SAT solvers have been applied to cryptanalysis (Mironov & Zhang, 2006) and have been used to solve longstanding problems in mathematics (Heule *et al.*, 2016). At the same time, it remains possible to construct small instances which cannot be solved by any current algorithm in a reasonable amount of time. A full understanding of what makes an easy or hard SAT instance currently seems out of reach.

The reasoning steps in SAT-solving algorithms can be modelled by propositional proof systems. In particular, solvers based on the DPLL algorithm (Davis *et al.*, 1962) and conflict driven clause learning (CDCL) (Marques Silva & Sakallah, 1996) produce proofs in the Resolution proof system (Res). Proof complexity results can help us to understand the strengths and limitations of solving algorithms. Lower bounds on the size of proofs in a system imply lower bounds on the running time of an algorithm which produces proofs in that system. Similarly, we can study the space requirements of a proof system, with lower bounds implying lower bounds on memory use of a related algorithm. It may also be possible to show that a short proof in one proof system implies the existence of a short proof in another system, or conversely that one system can prove certain formulas in polynomial-size proofs while proofs of the same formula are always of exponential size in another system. Results of this kind may indicate which algorithms are likely to be fastest in general or for some class of formulas. For example, although SAT solvers based on the DPLL and CDCL algorithms both produce proofs in Resolution, the DPLL algorithm is only capable of generating tree-like Resolution proofs. Tree-like Resolution is weaker than general (DAG-like) Resolution, and this is reflected in the (typically) faster speed of CDCL-based solvers.

Proof complexity research generally ignores any implementation details and the question of how an algorithm constructs a proof. This is a significant limitation. Upper bounds on complexity measures for proofs can be understood as indicating what might be achieved in principle by a given reasoning method. Generally, however, upper bounds are less useful than lower bounds for understanding algorithms because the ability of an algorithm to find a proof depends on heuristics, which are not modelled by proof systems. For the same reason, it is hard for proof complexity to offer any insights into the effectiveness of SAT-solving algorithms on satisfiable instances. In principle, any satisfiable instance can be solved in linear time if the heuristics happen to try a satisfying assignment first (although no heuristic could achieve this for all true formulas unless $\mathsf{P} = \mathsf{NP}$). As a result, propositional proof complexity focuses almost exclusively on unsatisfiable formulas.

Proving lower bounds in any system is difficult. In 1985 Haken proved a super-polynomoial lower bound on the size of Resolution refutations of the Pigeonhole Principle (Haken, 1985). Other lower bounds followed in the late 1980s (Chvátal & Szemerédi, 1988; Urquhart, 1987) using similar methods and Haken's result was also later improved to an exponential lower bound (Pitassi *et al.*, 1993). Some general techniques have now been developed for proving lower bounds on proof size for Resolution and other propositional proof systems.

Some propositional proof systems, including Resolution (Pudlák, 1997), have a property called feasible interpolation which gives a technique for proving proof size lower bounds from lower bounds in circuit complexity. Pudlák & Impagliazzo (2000) introduced a description of Resolution proofs which allows optimal strategies in a two-player game to be turned into Resolution proof size lower bounds. A connection between the 'width' of a proof and its overall size was shown in Ben-Sasson & Wigderson (2001). This allows lower bounds on proof size to be inferred from lower bounds on proof width, which can be easier to reason about in some instances. This paper began a program of research on the interplay between proof size, width and space (Beame *et al.*, 2012; Ben-Sasson & Nordström, 2011; Beyersdorff & Kullmann, 2014; Bonet & Galesi, 2001) in Resolution proofs. Despite these techniques and significant efforts towards finding lower bounds there remain strong propositional proof systems for which no exponential lower bounds on proof size are known.

The applications for QBF solvers are similar to those for SAT solvers and include verification (Benedetti & Mangassarian, 2008), model checking (Zhang, 2014), and planning (Egly *et al.*, 2014). Due to the additional expressiveness given by quantification, QBF can encode some problems more succinctly than propositional logic. The success of modern SAT solvers has encouraged the development of solving algorithms for extensions of SAT, including QBF. This in turn motivates the definition and study of QBF proof systems, which model the reasoning used by such algorithms. Since SAT is a sub-problem of QBF (a QBF in which all variables are existentially quantified is a SAT instance) every QBF proof system implicitly defines a propositional proof system. QBF solvers also take inspiration from SAT solvers and therefore many of them produce proofs in systems that extend Resolution.

Some QBF solvers, such as QUBE++ (Giunchiglia *et al.*, 2004), GhostQ (Klieber *et al.*, 2010) and DepQBF (Lonsing & Biere, 2010), are based on extensions of the DPLL and CDCL algorithms for SAT. Others, such as CAQE (Rabe & Tentrup, 2015), decompose the QBF into propositional formulas which can be handed to a SAT solver. RAReQS (Janota *et al.*, 2012) also uses a SAT solver as an oracle to guide a procedure that incrementally eliminates variables from a QBF.

Creating a QBF proof system from a propositional proof system such as Resolution requires the addition of rules for reasoning about universally quantified variables. Although all QBF proof systems based on Resolution coincide in the case when all variables are existentially quantified they do not all have the same strength in general. Several QBF proof systems that extend propositional Resolution for QBFs are introduced in Chapter 3.

One theme in proof complexity research for QBFs is comparing the strengths of these closely related systems and finding formulas which have polynomial-size proofs in one such system and require exponential-size proofs in another.

Some methods for proving lower bounds in propositions Resolution may be lifted to QBF proof systems. Beyersdorff *et al.* (2017b) modified the game technique used in propositional proof complexity and developed a game to be played on a QBF between a 'Prover' and a 'Delayer'. A false QBF is chosen. The Prover tries to refute the QBF and so end the game, the Delayer scores points by delaying this. The logarithm of the score gives the size of a proof in tree-like Q-Res. Similarly, feasible interpolation can be extended to the QBF setting (Beyersdorff *et al.*, 2017a), allowing for proof size lower bounds to be derived from circuit complexity lower bounds. In some QBF systems, lower bounds can be proved by showing that a large number of assignments to the universally quantified variables need to be considered.

Another important method for proving lower bounds in QBF proof systems is called strategy extraction (Balabanov & Jiang, 2012; Beyersdorff *et al.*, 2015). A QBF can be conceptualised as a game between two players, one who controls the existentially quantified variables and another who controls the universally quantified variables. The universal player wins the game if the assignments cause the formula to evaluate to false. The decision about how to make assignments is called a strategy, a winning strategy for the universal player is one that ensures this player wins every game played on this QBF. Many QBF proof systems have the property that such a winning strategy can be efficiently extracted from a proof. In some cases the extracted strategy also has a restricted form. We may construct a family of QBFs so that the winning strategies cannot be decided in polynomial time, or cannot be efficiently expressed in the form required by a certain proof system. It then follows that the proof system cannot have polynomial-sized proofs for that family of QBFs. Strategy extraction also allows proof size lower bounds to be derived from circuit complexity lower bounds.

1.1 Structure and Contributions

We begin in Chapters 2 and 3 with some preliminaries, introducing concepts and notation that will be used throughout this thesis. Chapter 2 defines propositional logic and quantified Boolean logic, the SAT and QBF problems, and describes solving algorithms for both. Chapter 3 gives rigorous definitions of a proof system and how proof systems are compared. We introduce Resolution and several QBF proof systems, we also briefly review the relationship between proof systems and solving algorithms.

Chapters 4 and 5 are mainly concerned with the QBF proof systems $\forall$Exp + Res and Q-Res. These model two different solving paradigms and the systems are known to be incomparable in general (Beyersdorff *et al.*, 2015; Janota & Marques-Silva, 2015), meaning that there are QBFs that are provably hard to refute in one system but easy to refute in the other. We consider two restricted but natural situations in which $\forall$Exp + Res is strictly stronger than Q-Res. This occurs when the proofs are required to have a tree-like structure, or when the input QBFs have bounded quantifier complexity (this will be defined in Chapter 4). The method we use can also be extended to another proof system, QU-Res. In general, QU-Res is strictly stronger than Q-Res and incomparable with $\forall$Exp + Res. The main results of these chapters can be summarised as follows:

Theorem 1.1.1. *$\forall$Exp+ Res p-simulates Q-Res on QBFs with bounded quantifier complexity.*

Theorem 1.1.2. *Q-Res p-simulates QU-Res on QBFs with bounded quantifier complexity.*

Theorem 1.1.3. *Tree-like Q-Res p-simulates tree-like QU-Res.*

Theorem 1.1.4. *Tree-like Q-Res* cannot *p-simulate tree-like $\forall$Exp+ Res.*

Although these three systems are identical to propositional Resolution when restricted to inputs with only existentially quantified variables, it was previously thought that they demonstrate very different strengths. Showing that the situation in which QU-Res is stronger than Q-Res, or Q-Res is stronger than $\forall$Exp + Res, is so tightly defined challenges this view. The results demonstrate the weakness of universal reduction and universal resolution as ways to reason about universally quantified variables.

Chapters 6 and 7 are concerned with a proof system that is very different from those explored so far in the thesis. QRAT is a QBF proof system that does not correspond with any particular solving algorithm but instead is able to model the reasoning of all current QBF solvers and pre-processing techniques. We study whether QRAT admits polynomial-time strategy extraction. We find that a version of QRAT which uses the universal reduction rule from Q-Res (though the system as a whole remains significantly stronger than Q-Res) does admit strategy extraction. However, when QRAT uses a stronger rule for reasoning about universally quantified variables it does not admit strategy extraction (unless $\mathsf{P} = \mathsf{PSPACE}$). We observe that the weaker universal reduction rule is not able to simulate the universal expansion rule from $\forall$Exp + Res, whereas the stronger rule can, and demonstrate a connection between strategy extraction and universal expansion. In Chapter 7 we consider an extension of QRAT known as QRAT+ and show that the two systems have the same strength. The main results of these chapters are summarised as follows:

Theorem 1.1.5. *QRAT(UR) has polynomial-time strategy extraction on false QBFs.*

Theorem 1.1.6. *QRAT does not have polynomial-time strategy extraction on false QBFs unless $\mathsf{P} = \mathsf{PSPACE}$.*

Theorem 1.1.7. *Given any refutational propositional proof system f, if the refutational QBF proof system $\mathsf{f} + \forall$Exp has polynomial-time strategy extraction then f must have feasible interpolation.*

Theorem 1.1.8. *QRAT is p-equivalent to QRAT+ on false QBFs.*

Finally, Chapters 8 and 9 examine two results of propositional proof complexity and whether these can be lifted to the QBF situation. In both cases fully lifting the results to Q-Res is not possible. Chapter 8 considers the famous result of Ben-Sasson & Wigderson (2001) which relates the maximum width of a clause in a Resolution proof to the size of the proof. This result is known to fail in Q-Res. We show that the argument can be partially applied to level-ordered Q-Res, yielding a slightly different result.

Theorem 1.1.9. *If the minimum size of a tree-like refutation of QBF Ψ in level-ordered Q-Res is S, then the minimum width of a tree-like Q-Res refutation is bounded above by $\lceil \log(S) + w(\Psi) \rceil$, where $w(\Psi)$ is the maximum width of a clause in Ψ.*

In Chapter 9 we study an interesting result from Riis (2001). This states that under a natural translation from first-order formulas to propositional logic, the size of Resolution proofs depends on whether the first-order formulas have any models. We give a translation from first-order formulas to QBF and show that the result applies in $\forall$Exp + Res, but provide a counter-example for Q-Res. For ϕ a first-order formula and Φ_n a QBF representing the (false) statement that there is a model for ϕ of size n, we have:

Theorem 1.1.10. *If ϕ has an infinite model but no finite model, then any tree-like $\forall$Exp + Res or Q-Res refutations of the QBFs $\{\Phi_n\}_{n\in\mathbb{N}}$ have size $2^{\Omega(n)}$.*

Theorem 1.1.11. *If ϕ has no models then the QBFs $\{\Phi_n\}_{n\in\mathbb{N}}$ have tree-like refutations in $\forall$Exp + Res of size $n^{O(1)}$.*

Theorem 1.1.12. *There is a first-order formula ϕ with no models but for which tree-like Q-Res refutations of the QBFs $\{\Phi_n\}_{n\in\mathbb{N}}$ have size $2^{\Omega(n)}$.*

Some of the work in this thesis has appeared in previous publications. Chapters 4 and 5 contain results from Beyersdorff *et al.* (2019). Chapters 6 and 7 contain results from Chew & Clymo (2019) and Chew & Clymo (2020). Chapter 8 contains work from Beyersdorff & Clymo (2018). The simulation of QU-Res by Q-Res for QBFs with bounded quantifier complexity (Chapter 4) has now been independently proved by Beyersdorff *et al.* (2020).

Chapter 2

Logic and Complexity

This chapter introduces propositional logic and quantified Boolean logic and their associated decision problems, explaining their importance as NP-complete and PSPACE-complete problems respectively. This is followed by an overview of algorithms used to solve these problems.

2.1 Computational Complexity

Formal Languages Let Σ be a set of symbols. A word w over Σ is a tuple $(w_1, \ldots, w_n)$ with $w_i \in \Sigma$. The length of a word is the number of symbols it contains and we denote the set of all finite length words over Σ by Σ^*. A language L over Σ is a subset of Σ^*. Every language has an associated decision problem: is a given word a member of the language?

Turing Machines A Turing machine is an abstract model of computation. It consists of a strip of tape divided into cells on which input, output and any intermediate values are written, a pointer that moves along the tape and can read or update the contents of any cell, and a set of states including at least one state to indicate that computation has been completed. A transition function δ determines, given the current state and a tape symbol, what the machine should do next. If the transition function allows for multiple possibilities for this next step then the machine is non-deterministic, otherwise it is deterministic.

Time and Space Complexity The time and space requirements of a computation are expressed relative to the size of its input. If the possible inputs for the computation are specified by a formal language then the input will be a word in this language, so the size of the input is the length of the word. More generally, the size of an input to a computation is the number of symbols used to represent it.

The time to complete a computation on a given input is the number of steps which are carried out before the computation terminates. If this is bounded above by a polynomial function of the size of the input then we say this is a polynomial time computation. A computation is generally considered to be 'feasible', 'efficient', 'tractable' if and only if it can be carried out in polynomial time or less. The space needed to complete a computation is the amount of tape required during the computation, again expressed as a function of the size of the input.

Decision problems can be classified according how much time or space a Turing machine requires to solve them. The class of all decision problems that can be solved in polynomial time in the size of the input by a deterministic Turing machine is denoted P. The class of all decision problems that can be solved in polynomial time in the size of the input by a non-deterministic Turing machine is NP. The class of decision problems that can be solved using at most polynomially many cells on the tape of a deterministic Turing machine is PSPACE. For a complexity class C the complexity class coC is the class of all decision problems whose complement (the same problem with the 'yes' and 'no' answers switched) is in C. A decision problem D is C-hard every decision problem in C can be converted into an instance of D in polynomial time. The conversion procedure is called a reduction. If a decision problem is both C-hard and in C then it is C-complete. In practice, we do not usually consider Turing machines explicitly but rather algorithms consisting of basic operations which are known to be efficiently computable on a Turing machine.

2.2 Propositional Logic

A logic may be considered syntactically, in terms of proofs and a proof system, or semantically, through the concepts of truth and satisfiability. We begin here with a semantic approach in which a logic consists of

- a set $\mathcal{F}$ of well-formed formulas,
- a class $\mathcal{V}$ of possible valuations,
- a satisfiability relation $\models$ over $\mathcal{V}$ and $\mathcal{F}$.

For a valuation $v \in \mathcal{V}$ and formula $f \in \mathcal{F}$, if $v \models f$ we say v satisfies f, or v is a model of f.

Formulas of Propositional Logic The Boolean constants 1 and 0 stand for true and false respectively. Both 1 and 0 are formulas in propositional logic. We also have a set of propositional variables, $\{x_i \mid i \in \mathbb{N}\}$ (though in practice we may use other letters to stand for propositional variables). Every propositional variable is a propositional formula.

Further formulas are defined inductively using the connectives $\neg$, $\wedge$, $\vee$ to stand for negation, conjunction and disjunction respectively. Let Φ_1 and Φ_2 be propositional formulas, then the following are also propositional formulas:

- $\neg\Phi_1$,
- $(\Phi_1 \wedge \Phi_2)$,
- $(\Phi_1 \vee \Phi_2)$.

We will focus mainly on formulas using the connectives $\wedge$, $\vee$ and $\neg$. Other common connectives can be defined in terms of these, for example $(\Phi_1 \rightarrow \Phi_2)$ is equivalent to $(\neg\Phi_1 \vee \Phi_2)$ and $(\Phi_1 \leftrightarrow \Phi_2)$ is equivalent to $((\Phi_1 \rightarrow \Phi_2) \wedge (\Phi_2 \rightarrow \Phi_1))$.

For a propositional formula Φ, let $\mathrm{vars}(\Phi)$ denote the variables that appear in Φ. A literal is a propositional variable or a negation of a propositional variable.

The size of a formula $|\Phi|$ is the number of symbols it contains.

Valuations Let Φ be a propositional formula and $x \in \mathrm{vars}(\Phi)$. Then $\Phi[f/x]$ denotes the formula that results from substituting every occurrence of x in Φ by the propositional formula f. In particular, f may be a Boolean constant.

A Boolean assignment to a set X of propositional variables is a mapping from X to the Boolean constants, $\alpha : X \rightarrow \{1, 0\}$. $\Phi[\alpha]$ indicates that we apply the substitution $\alpha(x)/x$ in Φ for every variable $x \in X \cap \mathrm{vars}(\Phi)$.

A propositional formula is evaluated by inductively applying the following definitions to sub-formulas:

$$\neg x = \begin{cases} 1 & \text{if } x = 0 \\ 0 & \text{if } x = 1 \end{cases}$$

$$(x \wedge y) = \begin{cases} 1 & \text{if } x = 1 \text{ and } y = 1 \\ 0 & \text{if } x = 0 \text{ or } y = 0 \end{cases}$$

$$(x \vee y) = \begin{cases} 1 & \text{if } x = 1 \text{ or } y = 1 \\ 0 & \text{if } x = 0 \text{ and } y = 0 \end{cases}$$

If all variables in Φ are substituted with Boolean constants then repeated application of these definitions will result in the whole formula evaluating to 1 or 0. It may be that $\Phi[\alpha]$ evaluates to 1 or 0 even if some of $\mathrm{vars}(\Phi)$ are not assigned a value in α.

Two propositional formulas are equivalent if they are defined over the same set of variables and evaluate to the same truth value under every assignment to those variables. In particular, the definition of $\neg$ means that $\neg\neg\Phi$ is equivalent to Φ, so we will assume that propositional formulas do not contain any double negation. Also, $\wedge$ and $\vee$ are both associative so we can drop some parenthesis from propositional formulas without changing the meaning, for example writing $(x_1 \vee \cdots \vee x_n)$ instead of $((\ldots(x_1 \vee x_2) \vee \cdots \vee x_{n-1}) \vee x_n)$. Now we can consider $\wedge$ and $\vee$ to be arity k operators for any $k \geq 0$. In the case that $k = 0$ the empty conjunction evaluates to 1 and the empty disjunction to 0.

If α is not a complete assignment to vars(Φ) then sub-formulas in $\Phi[\alpha]$ that do not evaluate to 1 or 0 can be simplified by applying the following rules, where a is a propositional formula:

- Replace $(a \vee 0)$ or $(0 \vee a)$ by a,
- Replace $(a \wedge 1)$ or $(1 \wedge a)$ by a,
- Replace $\neg\neg a$ by a.

If we make substitutions according to α in Φ and simplify the result we say Φ is restricted by α, denoted $\Phi|_\alpha$. If the result of restricting by α is a Boolean constant b we can write $\alpha(\Phi) = b$ and say that Φ evaluates to b under assignment α. It is often convenient to denote an assignment α by the set of literals which evaluate to 1 under α.

An extension of $\alpha : X \to \{1, 0\}$ to a set of variables Y with $X \subseteq Y$ is an assignment $\alpha' : Y \to \{1, 0\}$ such that for every $x \in X$, $\alpha'(x) = \alpha(x)$. Let α and τ be two assignments. Then $\sigma = \alpha \circ \tau$ is an assignment such that $\sigma(x) = \alpha(x)$ if $\alpha(x)$ is defined, and otherwise $\sigma(x) = \tau(x)$ (if this is defined). We say α has been extended by τ.

Satisfiability An assignment α is a model of Φ, written $\alpha \models \Phi$, if $\alpha(\Phi) = 1$. Alternatively we can say that α satisfies Φ. If α is an assignment to a set of variables $X \subseteq$ vars(Φ) and α satisfies Φ then also every extension of α to vars(Φ) satisfies Φ. A formula Φ is satisfiable if there exists a Boolean assignment α on (any subset of) vars(Φ) such that α satisfies Φ. A formula is a tautology if every Boolean assignment to vars(Φ) satisfies Φ.

Conjunctive Normal Form Let l be a literal and x a variable. If $l = x$ or $l = \neg x$ the we write var$(l) = x$. If $l = x$ then $\bar{l} = \neg x$ and if $l = \neg x$ then $\bar{l} = x$. A clause is a disjunction of literals. A cube is a conjunction of literals.

A formula is in conjunctive normal form (CNF) if it is a conjunction of clauses. Similarly a formula in disjunctive normal form (DNF) is a disjunction of cubes. The values of $\Phi \star \Phi$ and Φ are equal for $\star \in \{\vee, \wedge\}$. Together with the commutativity of $\wedge$ and $\vee$ this allows us to consider a clause as a set of literals and a CNF as a set of clauses, which is sometimes notationally convenient.

Every propositional formula is equivalent to (i.e. has the same models as) a formula in CNF and to a formula in DNF. Further, every propositional formula can be converted into a propositional formula in CNF in polynomial time and with at most polynomial increase in size (Tseitin, 1968). The CNF may contain new variables which were not in the original formula but it is satisfiable if and only if the original formula is.

Boolean Circuits A formula in propositional logic can be represented by a tree in which each leaf is associated with a variable or constant and all other nodes with one of $\neg$, $\vee$ or $\wedge$. This idea can be generalised to a directed acyclic graph, which allows subformulas to be used as input for more than one subsequent node. Such a graph defines a Boolean circuit. The size of a circuit is the number of nodes it contains, and its depth is the maximum number of edges in a path from a leaf to the root. The leaf nodes are the inputs to the circuit, non-leaf nodes are called gates, the root is the output. The function represented by the output is the function that the circuit computes.

Circuit complexity studies which Boolean functions can be efficiently expressed as Boolean circuits with a certain structure. For example, AC^i contains functions that can be represented by circuits with size $n^{O(1)}$ and depth $O(\log^i(n))$, where each node can have an unbounded number of inputs. Thus AC^0 contains all constant depth circuits of polynomial size.

2.2.1 The Boolean Satisfiability Problem

The Boolean satisfiability problem (SAT) asks whether there exists any model for a given propositional formula. It is the decision problem associated with the language of satisfiable propositional formulas and belongs to the class NP. Given assignment α and formula Φ it is straightforward to check in polynomial time in the size of Φ whether α is a model of Φ by evaluating $\Phi[\alpha]$ according to the definitions given above. However, since the number of possible assignments to a formula may be exponential in the size of the formula, it is not efficient to exhaustively check all assignments in order to decide whether a formula is satisfiable.

As well as belonging to NP, SAT is the canonical NP-complete problem. Because all other problems in NP can be be reduced to an instance of SAT, finding an algorithm to efficiently solve SAT would show that all problems in NP can be solved efficiently. In practice, also, it means that efforts put in to building SAT-solving algorithms pay off in many different domains. Deciding whether a propositional formula is unsatisfiable (UNSAT) is coNP-complete. It is conjectured that $\mathsf{coNP} \neq \mathsf{NP}$.

2.3 SAT-Solving Algorithms

DPLL The Davis Putnam Logemann Loveland algorithm (DPLL) (Davis & Putnam, 1960; Davis *et al.*, 1962) is a simple algorithm for deciding formulas in propositional logic. The input formula Φ must be in CNF. Each clause can be represented as a set of literals and the formula itself as a set of clauses. The algorithm performs a backtracking search through possible assignments to the variables, as shown in Algorithm 2.3. The depth-first search begins by selecting a literal which is assigned 1 and the formula

Algorithm 1 DPLL Algorithm

```
function ASSIGN($\Phi$, $l$)
    return $\{C \setminus \{\bar{l}\} \mid C \in \Phi, l \notin C\}$
function UNITPROPAGATE($\Phi$)
    while $\Phi$ contains unit clause $\{l\}$ do
        $\Phi \leftarrow$ ASSIGN($\Phi$, $l$)
    return $\Phi$
function PURELITERAL($\Phi$)
    while $\Phi$ contains pure literal $l$ do
        $\Phi \leftarrow$ ASSIGN($\Phi$, $l$)
    return $\Phi$
function DPLL($\Phi$)
    $\Phi \leftarrow$UNITPROPAGATE($\Phi$)
    $\Phi \leftarrow$PURELITERAL($\Phi$)
    if $\Phi = \{\}$ then return 1
    if $\{\} \in \Phi$ then return 0
    Select some literal $l$ in $\Phi$
    $\Phi \leftarrow$ ASSIGN($\Phi$, $l$)
    if DPLL($\Phi$) $= 1$ then return 1
    $\Phi \leftarrow$ ASSIGN($\Phi$, $\bar{l}$)
    if DPLL($\Phi$) $= 1$ then return 1
    return 0
```

simplified accordingly. If the formula evaluates to true then we have identified a satisfying assignment. If the formula does not evaluate to either true or false then another variable is selected and assigned a value. If the formula evaluates to false under the current assignment then the algorithm backtracks to try an alternative value for one of the previously assigned variables. The search creates a tree of possible assignments. If all assignments are searched without finding a satisfying assignment then the formula is unsatisfiable.

The search is augmented with two procedures which identify literals that can be assigned without branching. These are called unit propagation and pure literal elimination. A unit clause is a clause containing a single literal. If a literal l appears in a unit clause then unit propagation extends the current assignment with $\{l\}$. Literal l is pure in Φ if l appears in Φ but $\bar{l}$ does not. If l is pure in Φ then pure literal elimination extends the current assignment with $\{l\}$.

CDCL Conflict driven clause learning (CDCL) was introduced by Marques Silva & Sakallah (1996) and has been an important advancement in SAT-solving algorithms. It uses the same principle as DPLL, first selecting a literal to assign and then using unit propagation to make further implied assignments. While making these assignments, CDCL maintains an implication graph which records for every literal assigned by unit propagation which previous assignments were responsible for this (i.e. which other literals were in the clause that has become unit). When unit propagation forces some variable to be assigned both 0 and 1 this creates a conflict. A new clause is learnt from the implication graph by taking the negation of the assignments that led to x and $\neg x$ becoming unit. The new clause records that this combination of assignments must never occur simultaneously in a satisfying assignment. The algorithm then backtracks and the new clause is added to the formula. Unlike in DPLL, the backtracking is non-chronological, so several decisions may be undone. The choice of which variable assignment to switch is determined by which decisions caused the conflict.

By learning from failed assignments, CDCL is able to reduce the number of branches which must be explored in the assignment tree. Since a large number of new clauses may be generated in this way it is usually not possible to keep all of them due to memory limitations, so solvers implementing this algorithm employ heuristics to score learnt clauses, with only those judged most valuable being retained.

2.4 Quantified Boolean Logic

Formulas of Quantified Boolean Logic All propositional formulas are also quantified Boolean formulas (QBF). In addition, the quantifiers $\forall$ and $\exists$ are available, and whenever Ψ is a QBF so are $\forall x \Psi$ and $\exists x \Psi$, for x a Boolean variable.

Valuations and Satisfiability $\forall x \Psi$ evaluates to true under the same assignments as $\Psi[0/x] \wedge \Psi[1/x]$, and $\exists x \Psi$ evaluates to true under the same assignments as $\Psi[0/x] \vee \Psi[1/x]$. A QBF can be transformed in to a propositional formula by repeatedly applying these identities from the innermost sub-formulas until no quantifiers remain. A QBF is satisfied by an assignment if the propositional formula generated by this method is satisfied by the assignment. A QBF is true if the propositional formula generated in this way is a tautology. If x does not appear in Ψ then $\forall x \Psi$ and $\exists x \Psi$ are both equivalent to Ψ, so we can assume that only the variables that appear in Ψ are quantified over.

Bound and Free Variables For QBF $\mathcal{Q} x \Psi$ with $\mathcal{Q} \in \{\forall, \exists\}$, we say that $\mathcal{Q}$ is a quantifier binding x and Ψ is the scope of $\mathcal{Q} x$. If $\mathcal{Q} = \exists$ then x is existentially quantified, if $\mathcal{Q} = \forall$ then x is universally quantified. $\exists_\Psi$ is the set of all existentially quantified variables in Ψ, $\forall_\Psi$ is the set of universally quantified variables in Ψ. It is convenient also to say that a literal l is existentially (resp. universally) quantified if $\text{var}(l) = x$ and x is existentially (resp. universally) quantified.

If a variable x appears in QBF Ψ without being in the scope of any quantifier binding x then we say this occurrence of x is free, or that x appears free in Ψ. A QBF is closed if it has no free variables.

Prenex Conjunctive Normal Form In a prenex QBF all quantification is done outside of the propositional connectives. As such, it consists of two parts and we write $\Psi = \Pi\Phi$. Π is called the quantifier prefix, with $\Pi = \mathcal{Q}_1 x_1 \mathcal{Q}_2 x_2 \ldots \mathcal{Q}_n x_n$ with $\mathcal{Q}_i \in \{\forall, \exists\}$ and $x_i \in \text{vars}(\Phi)$ for each i. Φ is called the matrix and is a propositional formula with no quantifiers. If Φ is also a CNF then Ψ is in prenex conjunctive normal form (PCNF). Given a PCNF $\Pi\Phi$, a sub-sequence Π' of Π that is also in the form of a quantifier prefix is called a sub-prefix of Π. In other words, Π' consists of a subset of the variables that appear in Π, all quantified as they are in Π and in the same order.

An arbitrary QBF can be transformed into a PCNF with at most polynomial increase in the size of the formula by first moving the quantifiers to the beginning using the identities that $\neg\exists\Psi = \forall\neg\Psi$ and $\neg\forall\Psi = \exists\neg\Psi$, and then applying the Tseitin transformation to the matrix. When moving the quantifiers to the beginning of the QBF some variables may need to be renamed to avoid conflicts. A new variable that is introduced by this transformation must be placed carefully in the prefix so that it does not appear earlier than (to the left of) any of the variables used to define it. The new variables are all existentially quantified. Example 2.4.1 demonstrates how the transformation works for a simple QBF. The formula generated by this transformation is true if and only if the original formula is true.

Example 2.4.1. *The QBF*

$$\exists x \forall u (x \vee u) \wedge \forall u \exists x (x \vee \exists y (u \wedge y))$$

is not in prenex conjunctive normal form. The variables must be renamed according to the scope of the quantifiers. Variable x in $(x \vee u)$ is not the same variable as the x which appears in $(x \vee \exists y (u \wedge y))$ because they occur in the scope of two different quantifiers. First we rename all variables to remove this ambiguity:

$$\exists x_1 \forall u_1 (x_1 \vee u_1) \wedge \forall u_2 \exists x_2 (x_2 \vee \exists y_1 (u_2 \wedge y_1)).$$

Now the quantifiers can all be moved to the front of the formula:

$$\exists x_1 \forall u_1 \forall u_2 \exists x_2 \exists y_1 (x_1 \vee u_1) \wedge (x_2 \vee (u_2 \wedge y_1)).$$

Finally a Tseitin variable $t_1 = (u_2 \wedge y_1)$ *is introduced to transform the matrix to CNF, any new variables must be existentially quantified and are added to the prefix after any variables that appear in their definition. We can also modify the notation to indicate the blocks of variables in the quantifier prefix:*

$$\exists x_1 \forall u_1, u_2 \exists x_2, y_1, t_1 (x_1 \vee u_1) \wedge (x_2 \vee t_1) \wedge (t_1 \vee \neg u_2 \vee \neg y_1) \wedge (\neg t_1 \vee u_2) \wedge (\neg t_1 \vee y_1).$$

The order in which variables are assigned in the quantifier prefix is important, and in general changing the order changes the meaning of the QBF and whether it is true or false. However, it is sound to switch the order of two variables that appear adjacent in the quantifier prefix and with the same quantifier type. Because of this it is natural to consider the quantifier prefix as consisting of blocks of adjacent variables that have the same quantifier type. Then we write the prefix as $\Pi = \mathcal{Q}_1 X_1 \dots \mathcal{Q}_k X_k$ where $\mathcal{Q}_i \in \{\forall, \exists\}$, $\mathcal{Q}_i \neq \mathcal{Q}_{i+1}$, and X_i are disjoint sets of variables. For a variable $x \in X_i$ we say x belongs to level i of the prefix, and write $\mathrm{lv}(x) = i$. If l is a literal and $\mathrm{var}(l) = x \in X_i$ then we can also write $\mathrm{lv}(l) = i$. We use the standard comparison operators ($<$, $\leq$ etc.) to compare the levels of variables in a prefix. If it is not clear from the context then the quantifier prefix being considered is added as a subscript, for example $\leq_\Pi$.

2.4.1 Extending the Satisfiability Problem to QBF

For a QBF that has free variables we could choose to ask whether the formula is satisfiable (effectively considering the free variables as existentially quantified), or whether the formula is true (treating the free variables as universally quantified). We will restrict our attention to closed QBFs, so every variable is explicitly quantified, and ask whether the QBF is true or false.

The set of true QBFs is a language whose associated decision problem is PSPACE-complete, i.e. all other decision problems in PSPACE can be reduced to deciding the truth of some QBF. This decision problem also has SAT as a special instance since deciding whether a propositional formula Φ is satisfiable is identical to deciding whether $\exists x_1 \exists x_2 \dots \exists x_n \Phi$ is true, where $x_1 \dots x_n$ are all the variables in Φ. Similarly, the language of false QBFs defines a PSPACE-complete decision problem with UNSAT as a special instance. More generally, for any constant k the problem of deciding the truth of a PCNF with (at most) k blocks in the quantifier prefix is complete for the kth level of the Polynomial Hierarchy. A PCNF whose first block of variables is existentially (resp. universally) quantified and which has k quantifier blocks in total is in Σ_k^b (resp. Π_k^b), and deciding such a QBF is $\mathsf{\Sigma_k^p}$-complete (resp. $\mathsf{\Pi_k^p}$-complete). The Polynomial Hierarchy is the union of the complexity classes $\mathsf{\Sigma_k^p}$ and $\mathsf{\Pi_k^p}$ for all k.

2.4.2 Game Semantics for QBF

A natural way of interpreting a QBF is as a game between one player who assigns values to the universally quantified variables and another player who assigns values to the existentially quantified variables. The two players take turns making assignments to variables according to the order of variables in the quantifier prefix. Playing over QBF $\mathcal{Q}_1 X_1 \dots \mathcal{Q}_n X_n\ \Phi$ in the ith round of the game the player responsible for the $\mathcal{Q}_i$ quantified variables assigns values to variables in X_i.

The universal player wins the game if Φ evaluates to false after all the assignments have been made, the existential player wins if Φ evaluates to true. A QBF is false if and only if the universal player is able to play in such a way that the formula is always made to evaluate to false, regardless of the assignments made by the existential player. Similarly, the QBF is true if and only if the existential player can ensure that they are able to win every game regardless of the behaviour of the universal player. For a closed QBF one or other of the players must be able to guarantee a win.

A strategy for the universal player on QBF $\Pi\Phi$ is a set of Boolean functions (or circuits), one for each universally quantified variable u. The function for u specifies how u will be assigned during the game and must depend only on variables earlier than (to the left of) u in Π, respecting the idea that when u is being decided the universal player cannot know what choices will be made in future turns. Similarly, a strategy for the existential player is a set containing a Boolean function for each existentially quantified variable, with each function depending only on variables that appear earlier in the quantifier prefix.

It is sufficient to consider functions whose input is restricted to variables with the opposite quantifier type to the variable whose strategy is being computed. For universal variable u in QBF $\Pi\Phi$ a Herbrand function σ_u is a function from assignments to the existential variables prior to u in Π to a Boolean value. If σ is a collection of Herbrand functions σ_u for all universal variables u then σ is a strategy for $\Pi\Phi$ and in general a strategy for the universal player may be expressed as a collection of Herbrand functions. Skolem functions are defined analogously for the existentially quantified variables, and a set of Skolem functions is an existential strategy. If a strategy for one player ensures that they always wins games on $\Pi\Phi$, however the other player makes assignments, then it is called a winning strategy. A closed QBF is true if and only if there is a winning strategy for the existential player, it is false if and only if there is a winning strategy for the universal player.

Substituting the Skolem functions from a winning existential strategy into the original QBF yields a tautology. Similarly, substituting Herbrand functions from a winning universal strategy into the QBF results in an unsatisfiable propositional formula.

2.5 QBF-Solving Algorithms

QDPLL and QCDCL Both the DPLL and CDCL algorithms can be extended to act on QBF $\Pi\Phi$ in prenex conjunctive normal form, the QDPLL algorithm is outlined in Algorithm 2.

In both cases, the algorithm is modified to enable reasoning about universally quantified variables as well as existentially quantified variables. When a decision literal l is chosen to be assigned it must be the case that $\mathrm{var}(l)$ is in the outermost block of the quantifier prefix of all the variables that are currently unassigned. Then, if l is universally quantified, the algorithm returns true if *both* $\Phi|_{\{l\}}$ and $\Phi|_{\{\bar{l}\}}$ are found to be true in the recursive call. For l existentially quantified only one of the restricted formulas must be true.

The definition of a unit clause is modified to any clause containing a single existentially quantified literal l and any number of universally quantified literals, provided that $\mathrm{lv}(l) < \mathrm{lv}(u)$ for all universally quantified u in the clause. Pure literal elimination assigns a literal l to 1 if l is existentially quantified and pure in Φ, or assigns l to 0 if l is universally quantified and pure in Φ. Clause learning can be applied as in the propositional case to improve the efficiency of the search and, symmetrically, the algorithm also learns cubes (conjuncts of literals) to record assignments that cause the formula to be satisfied.

The requirement that the decision variable must be in the outermost quantifier block can sometimes be relaxed. The reason for this requirement can be understood in terms of the two-player game model of a QBF: a player should not know what their opponent will do in future turns when making a decision about how to play in this turn. The strategy for a variable can only depend on earlier variables for the same reason. However, the strategy for a variable need not depend on *every* variable that is earlier in the prefix. If we can show that a variable does not depend on the assignments to any earlier unassigned variables then it can be used as the next decision variable. The QBF solver DepQBF (Lonsing & Biere, 2010) uses dependency schemes (Samer & Szeider, 2009; Slivovsky & Szeider, 2015), a method for calculating variable dependencies and, as a result, can sometimes make sound assignments to variables that would not be allowed by the basic QDPLL algorithm. The QBF solver QUTE uses a technique called dependency learning (Peitl *et al.*, 2019) to achieve a similar effect. This approach begins by assuming that variables can be assigned in any order. A conflict is resolved according to the QCDCL algorithm, but because the variables have been assigned out of order it may not be possible to derive a new learned clause. In this case, the algorithm instead learns about some variable dependency and so gradually restricts the order in which variables can be assigned.

Algorithm 2 QDPLL Algorithm

```
function ASSIGN($\Phi$, $l$)
    return $\{D \mid D = C \setminus \{\bar{l}\}, C \in \Phi, l \notin C\}$
function UNITPROPAGATE($\Pi\Phi$)
        ▷ $\{l \vee C\}$ is unit if all $u \in C$ are universally quantified and $\mathrm{lv}(u) >_\Pi \mathrm{lv}(l)$
    while $\Phi$ contains unit clause $\{l \vee C\}$ do
        $\Phi \leftarrow$ ASSIGN($\Phi$, $l$)
    return $\Phi$
function PURELITERAL($\Pi\Phi$)
    while $\Phi$ contains pure literal $l$ do
        if $l$ is existentially quantified in $\Pi$ then
            $\Phi \leftarrow$ ASSIGN($\Phi$, $l$)
        else if $l$ is universally quantified in $\Pi$ then
            $\Phi \leftarrow$ ASSIGN($\Phi$, $\bar{l}$)
    return $\Phi$
function QDPLL($\Pi\Phi$)
    $\Phi \leftarrow$UNITPROPAGATE($\Pi\Phi$)
    $\Phi \leftarrow$PURELITERAL($\Pi\Phi$)
    if $\Phi = \{\}$ then return 1
    if $\{\} \in \Phi$ then return 0
    Select some literal $l$ in $\Phi$ with $\mathrm{lv}(l) \leq_\Pi \mathrm{lv}(x)$ for all $x \in \Phi$
    $\Phi_1 \leftarrow$ ASSIGN($\Phi$, $l$)
    $\Phi_0 \leftarrow$ ASSIGN($\Phi$, $\bar{l}$)
    if $l$ is existentially quantified then
        return (QDPLL($\Pi\Phi_1$) = 1) $\vee$ (QDPLL($\Pi\Phi_0$) = 1)
    else if $l$ is universally quantified then
        return (QDPLL($\Pi\Phi_1$) = 1) $\wedge$ (QDPLL($\Pi\Phi_0$) = 1)
```

Algorithm 3 CEGAR Algorithm

```
function SOLVE($\mathcal{Q}X\Psi$)
    if $\Psi$ is a propositional formula then
        if $\mathcal{Q} = \exists$ then return SAT($\Psi$)
        else return SAT($\neg\Psi$)
    $\Sigma \leftarrow \{\}$
    while True do
        if $\mathcal{Q} = \exists$ then $\Theta \leftarrow$PRENEX($\bigwedge_{\tau\in\Sigma} \Psi|_\tau$)
        else $\Theta \leftarrow$PRENEX($\bigvee_{\tau\in\Sigma} \Psi|_\tau$)
        $\alpha \leftarrow$ SOLVE($\mathcal{Q}X\Theta$)
        if $\alpha$ = NULL then return NULL
        $\alpha \leftarrow \{l \mid l \in \alpha \wedge \text{var}(l) \in X\}$
        $\tau \leftarrow$SOLVE($\Psi|_\alpha$)
        if $\tau$ = NULL then return $\alpha$
        $\Sigma \leftarrow \Sigma \cup \{\tau\}$
```

CEGAR Solving An alternative approach to QBF solving is based on a paradigm called Counter-Example Guided Abstraction Refinement (CEGAR). This recursive algorithm attempts to find assignments to each block of the quantifier prefix in order, it is described in Algorithm 3. The function Prenex(Ψ) converts Ψ into prenex form. For QBF Ψ the algorithm suggests a candidate assignment α to the outermost block then checks whether $\Psi|_\alpha$ is true or false by a recursive call. The base case is handled by a SAT solver that is able to return a satisfying assignment, if one exists. If the current block of variables is existentially (resp. universally) quantified and $\Psi|_\alpha$ returns false (resp. true) with assignment τ to the next block of variables then τ is a counter-example to α. The algorithm then seeks a new assignment to the current block of variables which satisfies (resp. falsifies) $\Phi|_\tau$. The process continues until some assignment to the outermost block is found for which no counter-example can be generated, or the set of counter-examples means that no new candidate assignment to the outermost block exists.

Chapter 3

Proof Systems

Definition 3.0.1. *(Cook & Reckhow, 1979) Let Σ and Σ_0 be alphabets and Γ a language over Σ. A proof system for Γ is a polynomial-time computable function $\mathsf{f} : \Sigma_0^* \to \Sigma^*$ such that the range of f is Γ.*

The function that defines a proof system can be thought of as a proof checking algorithm. If the input to the algorithm is a valid proof according to this proof system then the output is the member of Γ which is proved, so π is an f-proof of $\gamma \in \Gamma$ if and only if $\mathsf{f}(\pi) = \gamma$. The requirement that f must be computable in polynomial time is required to ensure that it is feasible to check that a reputed proof is indeed valid.

Let $rng(\mathsf{f})$ denote the range of function f. A proof system according to the definition above is complete since $\Gamma \subseteq rng(\mathsf{f})$, thus every member of Γ must have at least one valid f-proof. We also require that only members of Γ can have valid proofs according to the system f, which is guaranteed by $\Gamma \supseteq rng(\mathsf{f})$.

Line Based Proof Systems We will define propositional and QBF proof systems using more intuitive definitions that specify inference rules which may be used in a proof. It is straightforward to define a proof checking algorithm from the set of permitted rules and we will informally refer to the rules themselves as the proof system. In a line-based proof system the proof π is a sequence of lines $L_1 \dots L_m$. For every $i \leq m$, either L_i is derived from $L_1 \wedge \dots \wedge L_{i-1}$ using an inference rule, or L_i is introduced using an axiom rule.

Such a proof can also be viewed as a directed acyclic graph (DAG) with a vertex for each of $L_1, \dots, L_m$ and an edge from L_i to L_j if and only if L_i is used as a premise in the rule that derives L_j. When thinking of a proof as a graph it is natural to refer to the premises used in a derivation step as parents and the derived line as their child. There is a path in π from L_i to L_j if there exists a sub-sequence $L_{a_1}, \dots, L_{a_n}$ with $L_i = L_{a_1}$ and $L_j = L_{a_n}$ such that there is an edge in π from L_{a_i} to $L_{a_{i+1}}$ for all $i = 1, \dots n-1$.

In some cases we will require the graph induced by a proof to be tree-like, and refer to tree-like proof systems if this restriction is imposed. If there is no such requirement then the proof or proof system may be called DAG-like.

According to the formal definition of a proof system, the size of a proof π is the number of symbols required to write it, however in line based proof systems it is more convenient to consider the number of lines to be the size of a proof: for $\pi = L_1, \ldots, L_m$, $|\pi| = m$. The implicit assumption is that the number of lines in a proof is always asymptotically greater than the length of individual lines.

3.1 Proof Systems for Propositional Tautologies

A proof system is implicationally complete if any semantic implication of a set of statements Θ can be derived by rules in that system. A proof system is refutationally complete if it is able to derive 0 or an immediate contradiction (typically the empty clause, denoted $\bot$) from an inconsistent set of statements, that is a set of statements whose conjunction is not satisfiable. In practice, refutational completeness is sufficient to show that an implication holds because in propositional logic if $\Theta \wedge \neg C$ is inconsistent then Θ implies C. We write $\Theta \vdash_{\mathsf{f}} C$ to indicate that proof system f is able to derive C from Θ. The subscript is omitted when the proof system under consideration is clear from the context.

Proving that a formula belongs to the language of propositional tautologies is equivalent to showing that the negation of that formula is unsatisfiable and therefore we focus on refutational proof systems. In this context, we will use the words 'proof' and 'refutation' interchangeably.

Given a satisfiable propositional formula Φ there is an assignment α to the variables of Φ such that $\alpha(\Phi) = 1$. The assignment is a witness that the formula is satisfiable, and it is possible to confirm in polynomial time (in the size of Φ) that it does in fact satisfy Φ, though it may not be possible to *find* the witness in polynomial time. Given an unsatisfiable propositional formula we would like to find a proof, which will act as a witness of unsatisfiability.

The Resolution Proof System for Propositional Logic A famous and well-studied proof system for propositional logic is Resolution (Res) (Davis & Putnam, 1960; Robinson, 1965). Resolution is refutationally complete, so that from any unsatisfiable CNF formula there is a Resolution derivation of the empty clause.

Resolution is a line-based system with one axiom rule and one inference rule. Every line L_i is either a clause from the formula being refuted, or is derived by the resolution rule from two other lines $L_j, L_k \in \pi$ where $j, k < i$.

Axioms
$$\frac{}{x_1 \rightarrow (x_2 \rightarrow x_1)} \qquad \frac{}{(\neg(\neg x_1)) \rightarrow x_1}$$

$$\frac{}{(x_1 \rightarrow (x_2 \rightarrow x_3)) \rightarrow ((x_1 \rightarrow x_2) \rightarrow (x_1 \rightarrow x_3))}$$

Modus Ponens
$$\frac{x_1 \qquad (x_1 \rightarrow x_2)}{x_2}$$

Figure 3.1: An example set of rules for a Frege Proof System.

The Resolution proof system uses only the resolution rule and an axiom rule. The axiom rule allows any clause from the formula being refuted to be introduced into the proof. The resolution rule allows the following inference, where x is called the pivot and $(C \vee D)$ the resolvent.

$$\frac{C \vee x \qquad D \vee \neg x}{C \vee D}$$

Frege Systems A powerful class of propositional proof systems are known as Frege systems. The exact rules can vary without affecting the power of the system (Cook & Reckhow, 1979; Reckhow, 1976), provided that they are sound and implicationally complete, so it is common to take Modus Ponens (see Figure 3.1) as the only inference rule. Frege systems are line based proof systems so a Frege proof of Φ is a sequence of lines $L_1, \ldots, L_m$ such that $L_m = \Phi$ and every line is a propositional formula which was added according to an axiom rule or inference rule. For example, the axiom rule

$$\frac{}{x_1 \rightarrow (x_2 \rightarrow x_1)}$$

allows the line $(\Phi_1 \rightarrow (\Phi_2 \rightarrow \Phi_1))$ for any propositional formulas Φ_1 and Φ_2. Modus Ponens allows inferences of the following form

$$\frac{x_1 \qquad (x_1 \rightarrow x_2)}{x_2}$$

so that if we already have lines in the proof $L_i = \Phi_1$ and $L_j = (\Phi_1 \rightarrow \Phi_2)$, where Φ_1 and Φ_2 are propositional formulas, then we can add the line Φ_2 to the proof. In order to make derivations from a set of assumptions A we also allow that any member of A can be added as a line in the proof, and we have a refutational proof system by deriving $\perp$ from a set of assumptions.

Extension Variables Both Resolution and Frege proof systems can be made more powerful by allowing extension variables, that is the introduction of new variables not yet present in the proof. Specifically, in Extended Frege it is permitted to introduce a line $(v \leftrightarrow p_1)$ for v a new variable not in the proof or in p_1, and p_1 a propositional formula. In Extended Res we can introduce a new variable v defined by $v \leftrightarrow \neg(x_1 \wedge x_2)$, which adds three new clauses, $(\neg v \vee \neg x_1 \vee \neg x_2)$, $(v \vee x_1)$, $(v \vee x_2)$. In both cases, the extension variables allow the system to work with abbreviations of formulas.

Circuit Frege Extended Frege can be thought of as a Frege system in which lines are Boolean circuits rather than formulas. This idea can be extended by defining the set of proof systems C-Frege, which are Frege systems in which each line is a member of the circuit class C.

3.2 QBF Proof Systems

A propositional proof system such as Resolution is sound on QBFs but not complete. To extend a propositional proof system so that it is a complete system for QBFs additional rules are introduced for reasoning about the universally quantified variables. Two ways of doing this are universal reduction ($\forall$-Red) and universal expansion. We begin by introducing Q-Res, which lifts propositional Resolution to the QBF setting by the addition of universal reduction, and then describing some extensions and restrictions of it. Next, the proof systems $\forall$Exp + Res and IR-calc will be introduced. $\forall$Exp + Res uses universal expansion to reason about universally quantified variables, IR-calc generalises $\forall$Exp + Res and Q-Res. These proof systems are all refutational systems and assume that the QBF to be refuted is in prenex conjunctive normal form.

Q-Resolution and Universal Reduction The universal reduction rule was first introduced for the proof system Q-Res (Kleine Büning *et al.*, 1995) and is based on the observation that if a clause C in QBF $\Psi = \Pi\Phi$ contains a universal literal u with $\text{lv}(u) > \text{lv}(x)$ for all existential literals $x \in C$ then u can be removed from C without changing the truth value of the formula.

Although Buning et al. stated the inference rules so that universal reduction was incorporated into the resolution rule, more commonly they are stated separately as in Figure 3.2. The systems given by the two definitions are equivalent (formally, they are p-equivalent. See Section 3.3).

In propositional Resolution it is permitted, though unhelpful, to introduce tautologous clauses either as an axiom or as the result of a resolution step. In Q-Res the introduction of tautologies is explicitly forbidden, either by the Axiom rule or by the resolution rule. This is because tautologous clauses could make the proof system unsound, as demonstrated in Figure 3.3 where the empty clause is derived from a true QBF.

Axiom
$C \in \Psi$ is not a tautology.

$$\frac{}{C}$$

Resolution
$(C \vee D)$ is not a tautology, $x \in \exists_\Psi$.

$$\frac{C \vee x \qquad D \vee \neg x}{C \vee D}$$

$\forall$-Reduction
$u \in \forall_\Psi$. For all $x \in \exists_\Psi \cap C$, $\text{lv}(x) <_\Pi \text{lv}(u)$.

$$\frac{C \vee u}{C}$$

Figure 3.2: Rules in the Q-Res Proof System acting on QBF $\Psi = \Pi\Phi$.

$$\cfrac{\cfrac{\cfrac{x \vee u \qquad \neg x \vee \neg u}{u \vee \neg u}}{u}}{\bot}$$

Figure 3.3: An unsound refutation of $\forall u \exists x (x \vee u) \wedge (\neg x \vee \neg u)$ due to the introduction of a tautology.

The universal reduction rule is most easily understood via the game semantics of QBF. Suppose that u is quantified at level i in the prefix. During round i the universal player must decide how to assign u. In clause $(C \vee u)$ every existential variable has been assigned before round i, as well as universal variables at earlier levels. In order to win the game, the universal player only needs to ensure that one clause is falsified. If at round i the clause $(C \vee u)$ is not already satisfied then the universal player can certainly make it evaluate to false by assigning the remaining literals (which are all universal) to be false. Therefore, if the formula $\Pi\Phi \wedge (C \vee u)$ is to be made to evaluate to true then the clause $(C \vee u)$ must be satisfied before round i, regardless of the value which is eventually assigned to u, which is to say that C must be made to evaluate to true.

It is worth noting here that the meaning of universal reduction is within the context of a whole QBF, not just the single clause. The rule cannot be correctly stated without reference to the quantifier prefix. Additionally, if we considered a QBF with just a single (non-tautologous) clause including universally quantified literal u, $\Pi(C \vee u)$, it would always be valid to infer ΠC regardless of where u appears in Π. This is because $\bar{u}$ does not appear in the formula so it cannot be beneficial to the universal player to ever play so that u is made true. This reasoning is called pure literal elimination and is used in QBF solvers as a pre-processing and in-processing step. However, it is not sound in general to remove an arbitrary universal literal from any clause.

While a Q-Res proof is generally expressed as a sequence of clauses introduced as an axiom or derived by one of the inference rules, it can alternatively be represented as a sequence of QBFs, which justifies the understanding of universal reduction in the context of a whole formula. The refutation $L_1 \dots L_m$ of $\Pi\Phi$ in the usual formulation is equivalent to $\Psi_1 \dots \Psi_m$ where $\Psi_1 = \Pi\Phi$ (or, strictly, $\Pi\Phi$ with any tautologous clauses removed) and $\Psi_i = \Psi_{i-1} \cup L_i$. Then $\Psi_{i-1} \models \Psi_i$ (since the proof system is sound) and $\bot \in \Psi_m$, so clearly Ψ_m is false.

Universal reduction can alternatively be stated as allowing the substitution of 0 or 1 for universal literal u in a clause $(C \vee u)$ provided that $\mathrm{lv}(u) > \mathrm{lv}(x)$ for all existentially quantified $x \in C$. This definition can be further generalised to allow universal reduction to be applied in expressions which are not clauses.

QU-Resolution The simplest extension of Q-Resolution is QU-Resolution (QU-Res), introduced by Van Gelder (2012), which allows the pivot of a resolution step to be universally quantified. The other rules are unchanged.

Long Distance Q-Resolution As shown in Figure 3.3, allowing a resolution step to derive a clause that contains opposing universal literals is not sound in general. However, it is sound to use the resolution rule with two clauses that contain opposing universal literals u and $\neg u$ provided that the pivot variable x occurs earlier in the prefix than u.

Informally, the reason this is sound can be understood again in terms of the game semantics of QBF. Suppose we have two clauses, $(C \vee u \vee x)$ and $(D \vee \neg u \vee \neg x)$ with $\mathrm{lv}(x) < \mathrm{lv}(u)$. Because it is impossible to satisfy both x and $\neg x$ simultaneously, we know that either $(C \vee u)$ or $(D \vee \neg u)$ must be satisfied. Equally, it is certain that one of x or $\neg x$ must be true. Since the universal player can make their assignment to u after seeing how x is assigned, they could choose to assign u so that the literal made true belongs to the same clause as the literal of x which was made true. Whether this is the best thing for the universal player to do depends on the rest of the formula and the rest of the game, but instead of forbidding the resolution step we can allow the derivation of $(C \vee D \vee u^*)$, where u^* stands in for the formula $((x \vee u) \wedge (\neg x \vee \neg u))$.

As the proof progresses, the function being represented by u^* will change, but the idea is that u could be chosen in a way that causes this hidden function to evaluate to false. If during the proof we derive a clause $(C \vee u^*)$ where $\mathrm{lv}(u) > \mathrm{lv}(x)$ for all $x \in \exists_\Psi$ then we know the formula Ψ is true only if either C is made true or the function hidden in u^* is made true. The rules of the proof system ensure that it is always possible for the universal player to make u^* false, and all of the other variables u^* implicitly depends on are assigned earlier in the game than u. Therefore if C does not evaluate to true by the time u is being assigned then the universal player should assign u to make u^* evaluate to false.

The rules of Long Distance Q-Resolution (LD-Q-Res) are shown in Figure 3.4. Notice that it is not possible to perform a resolution step on pivot x if the two clauses both

Axiom

$C \in \Psi$ is not a tautology.

$$\frac{}{C}$$

Resolution

$(C \vee D)$ is not a tautology, $x \in \exists_\Psi$.
$U = \{u^* | \operatorname{var}(u) \in \operatorname{vars}(U_1)\}$

$$\frac{C \vee U_1 \vee x \qquad D \vee U_2 \vee \neg x}{C \vee D \vee U}$$

$\operatorname{vars}(U_1) = \operatorname{vars}(U_2)$. If $\operatorname{var}(u) \in \operatorname{vars}(U_1)$ then $u \in \forall_\Psi$ and $\operatorname{lv}(x) <_\Pi \operatorname{lv}(u)$.
$u \in U_1$ if and only if $\bar{u} \in U_2$.

∀-Reduction

$u \in \forall_\Psi$. For all $x \in \exists_\Psi \cap C$, $\operatorname{lv}(x) < \operatorname{lv}(u)$.

$$\frac{C \vee u}{C} \qquad \frac{C \vee u^*}{C}$$

Figure 3.4: Rules in the LD-Q-Res Proof System acting on QBF $\Psi = \Pi\Phi$.

contain u^* where $\operatorname{lv}(u) < \operatorname{lv}(x)$. Without this restriction the system would be unsound, for example it would be possible to refute the true QBF $\exists x \forall u \exists y \ (x \vee u \vee y) \wedge (\neg x \vee \neg u \vee y) \wedge (\neg x \vee u \vee \neg y) \wedge (x \vee \neg u \vee \neg y)$. LD-Q-Res was introduced in Zhang & Malik (2002).

LQU$^+$-Res further extends LD-Q-Res by allowing resolution pivots to be universal. However, the pivot cannot be a merged literal u^*.

Adding Universal Reduction to Other Propositional Systems The universal reduction rule (defined as substitution of a Boolean value to a universally quantified literal) can be added to a line-based propositional proof system to create a QBF proof system. For a propositional proof system f the corresponding QBF proof system is denoted f + ∀red. For example, Res + ∀red is identical to QU-Res.

Universal Expansion with Resolution Instead of using universal reduction to reason about universally quantified variables we can augment Resolution with universal expansion. Recall that $\forall u \Psi$ is semantically equivalent to $\Psi[0/u] \wedge \Psi[1/u]$. This conjunct can be returned to PCNF by renaming variables in each part and moving the quantifiers to the front. Expansion of universal variables decreases the number of quantifiers and keeps the formula in PCNF but at the cost of introducing more variables and increasing the size of the formula. In fact, the size of the formula can grow exponentially in the number of universally quantified variables.

Once all universal variables have been expanded the resulting propositional formula can be refuted using Resolution. The full expansion may not be required in this refutation. Therefore, the problem of exponential growth due to expanding the universal variables may be partially mitigated by expanding the formula more carefully.

For a QBF $\Psi = \forall X_1 \exists X_2 \dots \exists X_n \Phi$ where each X_i is a set of variables, expanding the block X_1 with assignments $\alpha_1, \dots, \alpha_m$ (which need not be all possible assignments to X_1) gives a QBF

$$\exists X_2 \dots \exists X_n \ \Phi|_{\alpha_1} \wedge \dots \wedge \exists X_2 \dots \exists X_n \ \Phi|_{\alpha_n}$$

which is implied by Ψ. We could prenex this formula completely, but instead only the copies of X_2 are brought to the front:

$$\exists X_2^{\alpha_1} \dots X_2^{\alpha_m} \ \forall X_3 \dots \exists X_n \ \Phi|_{\alpha_1}[X_2^{\alpha_1}/X_2] \wedge \dots \wedge \forall X_3 \dots \exists X_n \ \Phi|_{\alpha_m}[X_2^{\alpha_m}/X_2].$$

Then the universal variables in each of the sub-QBFs can also be expanded in the same way. Delaying the prenexing in this way means that each sub-QBF can have different assignments used in the expansion of later blocks of universals, which reduces the total size of the expanded formula. At the end of this expansion process we have a propositional formula which is a sub-formula of the full expansion of Ψ. If there is a Resolution refutation of this propositional formula then there is clearly also a Resolution refutation of the fully expanded formula, which has the same truth value as the original QBF. Therefore, we have a QBF proof system which consists of an expansion stage followed by a resolution stage.

Let $\mathcal{A}$ be the number of distinct complete assignments to the universal variables that were used to generate the partially expanded formula. Then the size of the partially expanded formula is $\mathcal{A} \times |\Psi|$. $\mathcal{A}$ is bounded above by the size of the Resolution refutation. Therefore, if the Resolution refutation is polynomial-sized in the size Ψ then the whole proof, including the expansion phase, is polynomial-sized.

The rules of the proof system $\forall$Exp + Res (Janota & Marques-Silva, 2015) are shown in Figure 3.5. The expansion phase is not shown explicitly but is reflected in the new variables that are introduced. When an axiom is used in the refutation the existential literals must be annotated with some universal assignment α that does not satisfy the universal literals in that clause, and the universal literals are removed. This shows that the clause was taken from the part of the expanded formula that corresponds to the assignment α. An existential literal l is annotated with the part of α that relates to variables earlier than l in the quantifier prefix of Ψ. The relevant part of the assignment is denoted $\lfloor\alpha\rfloor_l$, so $\lfloor\alpha\rfloor_l = \{u \mid u \in \alpha, \ \mathrm{lv}(u) < \mathrm{lv}(l)\}$ for l an existential literal and α a universal assignment. Variables with different annotations are different propositional variables and cannot be resolved together.

IR-calc A strengthening of $\forall$Exp + Res, IR-calc (Beyersdorff *et al.*, 2014) also uses universal annotations on the existentially quantified literals, but these can be added lazily through the proof. When an axiom is introduced, the literals are annotated with the partial universal assignment that exactly negates the universal literals in that clause. Resolution steps are still limited to pivots with exactly matching annotations, and annotations can be extended by the instantiation rule. Given a clause of annotated literals

Axiom

$C \in \Phi$. α a full assignment to variables in $\forall_\Psi$. $\alpha(C) \neq 1$. $\lfloor \alpha \rfloor_l = \{u \mid u \in \alpha,\ \mathrm{lv}(u) <_\Pi \mathrm{lv}(l)\}$

$$\frac{}{\{l^{\lfloor \alpha \rfloor_l} | l \in \exists_\Psi \cap C\}}$$

Resolution

$$\frac{C \vee x^\alpha \qquad D \vee \neg x^\alpha}{C \vee D}$$

Figure 3.5: Rules in the $\forall$Exp + Res Proof System acting on QBF $\Psi = \Pi\Phi$.

C already in the proof (the annotations need not all be the same), the instantiation rule allows to introduce $\{l^{\alpha \circ \lfloor \tau \rfloor_l} \mid l^\alpha \in C\}$ for any (partial) universal assignment τ.

IRM-calc IR-calc can be further generalised by allowing annotations which are not strictly assignments but allow universal variable u to take the special value $*$. This is exactly analogous to the special literals u^* in LD-Q-Res, and the special annotations are introduced by merging literals with opposing annotations during a resolution step.

Adding Universal Expansion to Other Propositional Systems As with $\forall$-Red, universal expansion can be added to a line-based propositional proof system to create a QBF proof system. We start with a propositional proof system f and prenex QBF $\Psi = \Pi\Phi$ that we wish to refute. Let α be a full assignment to all universally quantified variables and let l be an existentially quantified literal.

Definition 3.2.1. *(Beyersdorff et al., 2016a) The refutational QBF proof system* f + $\forall$Exp *allows the introduction of axiom* $\{l^{\lfloor \alpha \rfloor_l} \mid l \in \exists_\Psi \cap C\}$ *for any* $C \in \Phi$ *such that* C *does not evaluate to* 1 *under* α. *The inference rules are those of* f, *with any variables with non-matching annotations treated as different variables by* f.

An f + $\forall$Exp refutation π of $\Pi\Phi$ therefore consists of a propositional f proof of a sub-conjunction of the full expansion. The part of the full expansion of Ψ used by π is denoted $\mathsf{subexp}_\pi(\Psi)$.

Level Ordered QBF Proofs All of the QBF proof systems introduced so far may additionally be required to produce level-ordered proofs. The idea applies whenever derivation steps can be associated with a level of the quantifier prefix of the input QBF. In the case of a resolution step, the associated level is the level of the pivot variable.

Definition 3.2.2. *Let* f *be a QBF proof system based on resolution and* π *an* f*-refutation of* $\Pi\Phi$. *Then* π *is level-ordered if and only if every resolution step in* π *obeys the following rule: if the derivation derives* C *and has pivot* x *then* $\mathrm{lv}(x) \geq_\Pi \mathrm{lv}(y)$ *for all* $y \in C$.

Restricting a Proof In a line-based propositional or QBF proof system each line L_i is a propositional formula or QBF. Therefore we can extend the notion of restricting a formula by an assignment and define the restriction of a proof. For a proof $\pi = L_1 \dots L_m$ the restriction of π by an assignment α is denoted $\pi|_\alpha$ and is equal to $L_1|_\alpha \dots L_m|_\alpha$.

3.3 Proof Complexity

Given a proof system f for language Γ and some $\gamma \in \Gamma$, $S_\mathsf{f}(\gamma)$ is the minimum size of a valid f-proof of γ. Formally, $S_\mathsf{f}(\gamma) = min\{|\pi| \mid \mathsf{f}(\pi) = \gamma\}$. A proof system f is polynomially bounded if there exists a polynomial $p(n)$ such that for any $\gamma \in \Gamma$ it holds that $S_\mathsf{f}(\gamma) \leq p(|\gamma|)$.

The claim that there exists a polynomially bounded proof system that recognises unsatisfiable propositional formulas is equivalent to the claim that $\mathsf{coNP} = \mathsf{NP}$. Thus, proving that there is no such system would imply that $\mathsf{P} \neq \mathsf{NP}$. To show that some proof system is not polynomially bounded we find an infinite sequence γ_i of members of Γ such that there is no polynomial $p(n)$ with $S_f(\gamma_i) \leq p(|\gamma_i|)$.

In addition to considering the sizes of proofs in one system we can compare the strength of two proof systems for the same language.

Definition 3.3.1. *For two proof systems* f *and* g, f *simulates* g *if there exists a polynomial* $p(n)$ *such that for every* $\gamma \in \Gamma$ *and every* g*-proof* π_g *of* γ *there is an* f*-proof* π_f *of* γ *such that* $|\pi_\mathsf{f}| \leq p(|\pi_\mathsf{g}|)$. *If, in addition,* π_f *can be constructed from* π_g *in polynomial time, then* f *p-simulates* g. *If* f *p-simulates* g *and* g *p-simulates* f *then the two systems are said to be p-equivalent.*

Alternatively, we may show a separation between two systems by constructing a sequence γ_i of members of Γ and a sequence of f-proofs π^i_f of γ_i for which there is no polynomial $p(n)$ with $S_\mathsf{g}(\gamma_i) \leq p(|\pi^i_\mathsf{f}|)$. If we have separating examples to show that f is not simulated by g and g is also not simulated by f then we say the two systems are incomparable.

Several simulation and separation results are known between the QBF proof systems described above. Some of these results are summarised in Figure 3.6. Solid lines indicate that the upper system p-simulates the lower system. Dotted lines indicate that the systems are incomparable.

3.3.1 Strategy Extraction

Recall that if a closed QBF is false then there is a winning strategy for the universally quantified variables. The strategy is a Boolean function for each universally quantified

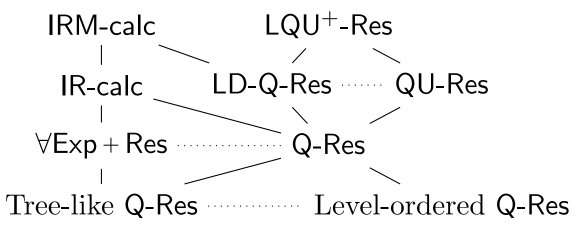

Figure 3.6: Known p-simulations between QBF proof systems based on Resolution.

variable u that takes as input the assignments made to variables earlier than u in the prefix and outputs a Boolean value. Together the functions represents the universal response to each existential assignment which ensures that the formula evaluates to false.

In order to show that a QBF is false it is sufficient to provide a proof in a sound proof system. However, it may also be desirable to find the winning universal strategy explicitly. A proof system f is said to admit strategy extraction if it is possible to construct a winning universal strategy σ for Ψ from an f-proof π of Ψ in polynomial time in the size of π.

For some proof systems it is possible to show not only that a strategy can be extracted but that they can always be expressed as circuits from some class C of polynomial size in $|\pi|$. In particular, given a proof in the system C-$\mathsf{Frege} + \forall \mathsf{red}$ it is possible to extract universal strategies in C (Beyersdorff *et al.*, 2016a). Q-Res and QU-Res admit strategy extraction and the strategies can be expressed as circuits in AC^0 (Balabanov & Jiang, 2012; Beyersdorff *et al.*, 2015).

If a proof system has strategy extraction this also provides a method for creating lower bounds for that system. Given a proof system from which it is possible to extract circuits in C we construct a QBF such that a universal variable has a unique winning strategy that cannot be expressed by polynomial-size circuits in C. Then this formula cannot have polynomial-size proofs in the given proof system.

3.3.2 Using Proof Systems to Understand Algorithms

From the trace of a SAT solver implementing the DPLL algorithm it is possible to construct in polynomial time a tree-like Resolution refutation which corresponds to the tree of assignments made during the search (Beek, 2006). In this sense, tree-like Resolution can be said to p-simulate the DPLL algorithm. Similarly, the trace of a solver implementing the CDCL algorithm for an unsatisfiable formula can be used to efficiently generate a (not necessarily tree-like) Resolution refutation (Beame *et al.*, 2004). Further, it has been shown that CDCL-based SAT solvers can p-simulate Resolution under some assumptions (Pipatsrisawat & Darwiche, 2011). For example, the solver is required to restart often and must retain learnt clauses correctly.

For QBFs the situation is not so simple. Solvers based on QDPLL can generate tree-like Q-Res proofs (Giunchiglia *et al.*, 2006), but QCDCL does not correspond to Q-Res as neatly as CDCL does to propositional Resolution. Two issues arise: firstly, variable assignment decisions must be made according to the order of the quantifier prefix. In the worst case this could mean that the generated proofs are level-ordered, which is a significant limitation on Q-Res. Janota (2016) showed that QCDCL-based solvers cannot p-simulate even tree-like Q-Res as a result of this restriction. In the other direction, the unit propagation procedure does allow some assignments to be made 'out of order' compared to the quantifier prefix. Clauses are considered unit which contain universally quantified literals at a higher level than the single existentially quantified literal in the clause. As a result, the proof which is produced from analysing a conflict clause may contain tautologous clauses due to resolution steps corresponding to unit propagation having opposing universal literals in the two parent clauses. This is not permitted in Q-Res but is possible in LD-Q-Res. The trace of a QCDCL solver can be expressed as an LD-Q-Res proof but QCDCL solvers are clearly much weaker than full LD-Q-Res. As such, we can say QCDCL is "related to" Q-Res, but the exact relationship between the algorithm and proof systems is not clearly understood. To further complicate matters, it is common to use pre-processing tools, such as Bloqqer (Biere *et al.*, 2011) and HQSpre (Wimmer *et al.*, 2017), which include a wide range of reasoning techniques that cannot all be easily expressed in a Q-Res or even LD-Q-Res proof. The proof system $\forall\mathsf{Exp}+\mathsf{Res}$ was introduced to model the solver RAReQS (Janota & Marques-Silva, 2015) which is based on an algorithm similar to that shown in Algorithm 3 after it was observed that traces from this algorithm could not easily be converted to Q-Res proofs.

Chapter 4

QBFs with Bounded Quantifier Complexity

The proof systems $\forall$Exp + Res, Q-Res, QU-Res and LD-Q-Res model different ways to lift propositional Resolution to QBF. They all act on QBFs in prenex conjunctive normal form (PCNF) and, when used to refute formulas containing only existentially quantified variables, all four systems are equivalent to propositional Resolution. However they employ different techniques for reasoning about universally quantified variables.

An $\forall$Exp + Res proof is a propositional Resolution proof acting on (a part of) the formula that is obtained when the universally quantified variables have been eliminated through universal expansion. Universally quantified variables appear in the proof only as labels for the new existentially quantified variables that are introduced in the expansion phase.

The other proof systems considered in this chapter act on clauses from the original QBF containing both existentially and universally quantified literals. Resolution is augmented with rules for reasoning about universally quantified variables that can be applied throughout the proof. Most importantly, all of Q-Res, QU-Res and LD-Q-Res use universal reduction to remove universally quantified variables from a clause. QU-Res generalises Q-Res by allowing resolution steps on universally as well as existentially quantified variables. LD-Q-Res introduces the concept of long-distance resolution steps, in which the clauses being resolved may contain conflicting universally quantified literals. The precise rules of each of these systems are given in Section 3.2.

The proof systems Q-Res and $\forall$Exp + Res are known to be incomparable. There are families of QBFs that have polynomial-size refutations in one system, but require exponential-size refutations in the other (Beyersdorff *et al.*, 2015; Janota & Marques-Silva, 2015). As such we would not expect QCDCL-based solver to consistently outperform those based on quantifier expansion, or vice versa, but would instead anticipate

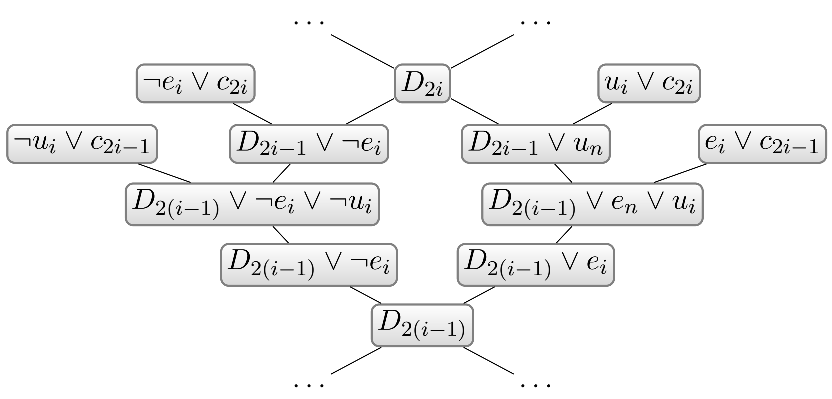

Figure 4.1: A section of the Q-Res refutation for Theorem 4.0.1.

that solvers implementing the two paradigms should show complementary strengths. QU-Res and LD-Q-Res are strictly stronger than Q-Res. Both systems p-simulate Q-Res and there are families of QBF with polynomial-size proofs in either QU-Res or LD-Q-Res but which require exponential-size proofs in Q-Res. The two systems are incomparable to each other and to $\forall$Exp + Res. We begin by recalling the QBFs that separate Q-Res from $\forall$Exp + Res.

Theorem 4.0.1. *(Janota & Marques-Silva, 2015) There exists a family of QBFs with polynomial-size Q-Res refutations but requiring exponential-size refutations in $\forall$Exp + Res.*

Proof Sketch. Consider the following QBF.

$$\Psi_n = \exists e_1 \forall u_1 \exists c_1 c_2 \dots \exists e_i \forall u_i \exists c_{2i-1} c_{2i} \dots \exists e_n \forall u_n \exists c_{2n-1} c_{2n}$$
$$D_{2n} \wedge \bigwedge_{i=1}^{n} (e_i \vee c_{2i-1}) \wedge (\neg e_i \vee c_{2i}) \wedge (\neg u_i \vee c_{2i-1}) \wedge (u_i \vee c_{2i})$$

where $D_i = (\neg c_1 \vee \dots \vee \neg c_i)$.

The size of Ψ_n is linear in n. There are n existentially quantified variables e_i, n universally quantified variables u_i, and $2n$ existentially quantified variables c_i. To see why Ψ_n is false, consider each pair of variables e_i and u_i. The winning strategy for the universal player is to assign $u_i = e_i$ (note that for each i, e_i appears before u_i in the quantifier prefix). When $e_i = 1$ it is necessary to make $c_{2i} = 1$ in order to satisfy the clause $(\neg e_i \vee c_{2i})$. According to the strategy, $u_i = 1$ also, and so we require $c_{2i-1} = 1$ to satisfy the clause $(\neg u_i \vee c_{2i-1})$. Similarly, when $e_i = 0$ it is necessary to make $c_{2i-1} = 1$ to satisfy the clause $(e_i \vee c_{2i-1})$. According to the strategy, $u_i = 0$ and so we require $c_{2i} = 1$ in order to satisfy the clause $(u_i \vee c_{2i})$. Playing according to this strategy forces all $c_{2i} = 1$ and $c_{2i-1} = 1$ for all i. Clearly this creates a contradiction with the clause D_{2n}, which states that at least one of the variables $c_1, \dots, c_{2n}$ must equal 0.

A part of the Q-Res refutation of Ψ_n is shown in Figure 4.1. This demonstrates how $D_{2(i-1)}$ is derived from D_{2i} in a constant number of proof steps. We begin with the long clauses D_{2n} and derive $D_{2(i-1)}$ for each $i = n, \ldots, 1$. D_0 is the empty clause, which has been derived in linearly many steps in n.

Any refutation in $\forall$Exp + Res must have exponential size in n because there are exponentially many possible assignments to the universally quantified variables and all of these assignments are relevant and must appear somewhere in the proof.

First consider the expansion on u_1. Ψ_n is equivalent to $\exists e_1 \Phi_{u_1} \wedge \Phi_{\neg u_1}$, where

$$\begin{aligned}\Phi_{u_1} =& \exists c_1 c_2 \exists e_2 \forall u_2 \exists c_3 c_4 \ldots \exists e_n \forall u_n \exists c_{2n-1} c_{2n} \\ & D_{2n} \wedge (e_1 \vee c_1) \wedge (\neg e_1 \vee c_2) \wedge (c_1) \\ & \bigwedge_{i=2}^{n} (e_i \vee c_{2i-1}) \wedge (\neg e_i \vee c_{2i}) \wedge (\neg u_i \vee c_{2i-1}) \wedge (u_i \vee c_{2i})\end{aligned}$$

$$\begin{aligned}\Phi_{\neg u_1} =& \exists c_1 c_2 \exists e_2 \forall u_2 \exists c_3 c_4 \ldots \exists e_n \forall u_n \exists c_{2n-1} c_{2n} \\ & D_{2n} \wedge (e_1 \vee c_1) \wedge (\neg e_1 \vee c_2) \wedge (c_2) \\ & \bigwedge_{i=2}^{n} (e_i \vee c_{2i-1}) \wedge (\neg e_i \vee c_{2i}) \wedge (\neg u_i \vee c_{2i-1}) \wedge (u_i \vee c_{2i})\end{aligned}$$

Since Φ_{u_1} is satisfiable when $e_1 = 0$, and $\Phi_{\neg u_1}$ is satisfiable when $e_1 = 1$, we know that both parts of the conjunction are necessary to find a contradiction. Φ_{u_1} is unsatisfiable when $e_1 = 1$ since both $c_1 = 1$ and $c_2 = 1$ are forced by unit clauses. After renaming of variables, $\Phi_{u_1}[1/c_1, 1/c_2]$ is equivalent to the formula Ψ_{n-1}. It therefore follows that both assignments to u_2 are necessary to refute it, using an identical argument. The same reasoning applies to $\Phi_{\neg u_1}$, which must be unsatisfiable when $e_1 = 0$ but requires both assignments to u_2 in order to prove this.

Continuing to expand Ψ_n in this way we see that all assignments to the universal variables must be considered in any $\forall$Exp + Res refutation of Ψ_n. $\square$

The QBFs used to separate $\forall$Exp + Res from Q-Res express a contradiction about a circuit for calculating the parity of n input bits. They are discussed in Section 4.3.1 and Chapter 5.

Despite knowing that $\forall$Exp + Res and Q-Res are incomparable in general, we can still discover restricted situations in which one system simulates the other. The restriction may be on the type of QBFs considered or on the proof systems themselves. For example, Janota and Marques-Silva also showed that $\forall$Exp + Res p-simulates tree-like Q-Res, and (general, DAG-like) Q-Res p-simulates level-ordered $\forall$Exp + Res. It does not hold that $\forall$Exp + Res can p-simulate level-ordered Q-Res, since the Q-Res refutation of formulas Ψ_n in Theorem 4.0.1 is level-ordered. We show in Chapter 5 that Q-Res cannot p-simulate tree-like $\forall$Exp + Res.

In this chapter we consider families of QBFs of *bounded quantifier complexity*, which express exactly all problems from the Polynomial Hierarchy and so cover a wide range of application scenarios. In this case, we show that the relationship between these proof systems is significantly simplified: Q-Res and QU-Res are p-equivalent, and they are p-simulated by $\forall$Exp + Res. The proof is constructive, building an $\forall$Exp + Res refutation from a Q-Res or QU-Res refutation. The construction increases in complexity as the number of blocks in the quantifier prefix increases, finally reaching an exponential separation for QBFs with an unbounded number of quantifier blocks (as expected). We also discuss how this result may offer a partial explanation for the observation that "the performance of solvers based on different solving paradigms substantially varies on classes of PCNFs defined by their numbers of alternations" (Lonsing & Egly, 2018a). Our construction provides an alternative proof that tree-like Q-Res is p-simulated by (tree-like) $\forall$Exp + Res and newly demonstrates that tree-like Q-Res and tree-like QU-Res are p-equivalent. LD-Q-Res remains incomparable to $\forall$Exp + Res for QBFs with bounded quantifier complexity, which we demonstrate with a separating example.

4.1 Simulating Q-Resolution by Expansion and Resolution

In this section we consider only $\forall$Exp + Res and Q-Res. Section 4.2 broadens the simulation to QU-Res.

Recall that a PCNF consists of a quantifier prefix and a quantifier free formula in CNF. When two variables appear next to each other in the quantifier prefix and have the same quantifier type then their relative order does not affect whether the QBF is true or false. Therefore it is convenient to think of the quantifier prefix as sets or blocks of variables which appear consecutively and are quantified in the same way. We are interested in PCNFs for which the number of these blocks is bounded above by a constant, although the number of variables may not be.

Definition 4.1.1 (Bounded Quantifier Complexity). *A family of PCNFs has bounded quantifier complexity if there exists some constant k such that every member has at most k blocks in its quantifier prefix.*

We begin by considering a naive approach to constructing an $\forall$Exp + Res refutation from a Q-Res refutation. We will show that, in general, this may lead to an exponential increase in the size of the refutation. We then go on to show how the blow-up can be avoided by a more careful, but conceptually similar, method.

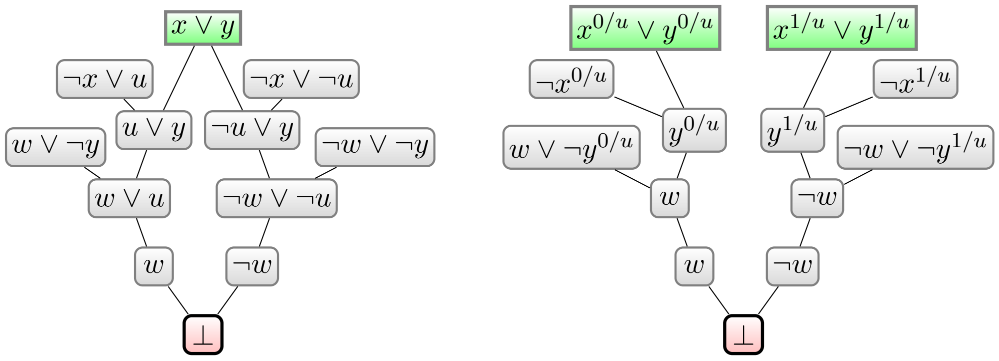

Figure 4.2: Duplicating clauses to create an expansion refutation of QBF with prefix $\exists w \forall u \exists xy$.

Suppose we have a Q-Res refutation π of an arbitrary PCNF Ψ. The natural way to transform a clause in the Q-Res refutation into a clause in an $\forall$Exp + Res refutation is similar to how axioms are instantiated in $\forall$Exp + Res. For a clause C in the Q-Res refutation, define a complete assignment α to the universally quantified variables of Ψ that does not satisfy C. All universal literals are removed from C since they are falsified under α. Each existentially quantified literal $x \in C$ is replaced by $x^{\lfloor \alpha \rfloor_x}$ (recall that $\lfloor \alpha \rfloor_x$ indicates the restriction of α to those variables that appear before x in the quantifier prefix of Ψ). By applying this transformation to every clause in π we might hope to produce a refutation in $\forall$Exp + Res.

The problem with this approach is that the resolution steps in the refutation which results from this modification are not valid in $\forall$Exp + Res. The rules of $\forall$Exp + Res require that in every resolution step the pivot literals have exactly the same annotation. It may be impossible to find suitable annotations for each clause in the given Q-Res refutation that respect this restriction. This can be demonstrated with a simple example.

Consider the Q-Res refutation shown in Figure 4.2. The clause $(x \vee y)$ is resolved once with a clause containing universal literal u, and separately with another clause containing $\neg u$. It is not possible to define a single annotation for x in $(x \vee y)$ so that both of these steps are valid in $\forall$Exp + Res. The clause $(\neg x \vee u)$ must become $(\neg x^{0/u})$ because the annotation to x must falsify the universal literal u in the clause. Similarly, $(\neg x \vee \neg u)$ must become $(\neg x^{1/u})$. In order to annotate $(x \vee y)$ with both $0/u$ and $1/u$ we could duplicate the clause, as shown in the right-hand part of Figure 4.2. After this clauses has been duplicated it is straightforward to create a valid $\forall$Exp + Res refutation from the Q-Res refutation.

If $(x \vee y)$ had occurred part way through a Q-Res proof then its entire derivation would need to be copied, not only the clause itself, once with $0/u$ in the annotation, and again with $1/u$. Every time a clause in the proof is used in multiple resolution steps that require conflicting annotations, a similar duplication of the sub-proof deriving that clause would be necessary. Since this may apply to many clauses in the refutation, the overall effect could be an exponential increase in the proof size. For example, applying this procedure to the formulas Ψ_n from Theorem 4.0.1 does result in a valid $\forall\mathsf{Exp}+\mathsf{Res}$ proof but with an exponential increase in size.

Notice that the duplication of clauses is only necessary in proofs that re-use clauses in separate resolution steps, i.e. that have a DAG-like structure. This intuitively demonstrates why we have a simulation in the tree-like systems. More significantly, we will now show that if attention is restricted to QBFs that have a constant number of quantifier blocks, then we can construct an $\forall\mathsf{Exp}+\mathsf{Res}$ refutation from a Q-Res refutation without an exponential increase in size.

Recall that a Q-Res proof is a sequence of clauses, not necessarily unique, and induces a directed acyclic graph (DAG) by the inference relationship. Our aim is to construct from a Q-Res proof of Ψ a sequence of new Q-Res proofs of Ψ, the last of which can be readily turned into a valid $\forall\mathsf{Exp}+\mathsf{Res}$ proof of Ψ. We show how the size of the final proof depends on the size of the initial proof and the number of blocks in the quantifier prefix of Ψ.

4.1.1 Expanding a Q-Resolution proof

We start with some observations regarding Q-Res which allow us to make simplifying assumptions about the structure of a Q-Res proof.

First, we assume that universal reduction is carried out as early as possible in any refutation in Q-Res, since no proof step can be prevented by first removing trailing universal literals from the parent clauses. For the same reason we assume that the innermost block of variables in a PCNF is existentially quantified.

If a clause is used in a universal reduction step then it is not used in any resolution step, since this would delay a possible universal reduction on that path. Therefore, all possible universal reduction steps for a clause are carried out consecutively, without branching.

Where a Q-Res refutation contains consecutive universal reduction steps these may be re-ordered arbitrarily without affecting any other part of the proof.

Literals removed in consecutive universal reduction steps cannot be complementary, since they must all appear together in the clause prior to the sequence of universal reductions.

As a result, consecutive universal reduction steps can be treated as a single action. For the remainder of this section let $\pi = L_1 \dots L_m$ be a Q-Res refutation of PCNF $\Psi = \mathcal{Q}_1 X_1 \dots \mathcal{Q}_k X_k\, \Phi$ and let i be some index in $\{1, \dots, k\}$ with $\mathcal{Q}_i = \forall$.

Definition 4.1.2. *Universal reduction at level i is the derivation*

$$\frac{C \cup U}{C}$$

where all $u \in U$ are universally quantified with $i = \min_{u \in U}(\mathrm{lv}(u))$, for all $x \in C$, $\mathrm{lv}(x) < i$. If $y \in C$ is universally quantified then there exists $x \in C$ existentially quantified and with $\mathrm{lv}(x) > \mathrm{lv}(y)$.

Definition 4.1.3. *Let C and P be two clauses in π. C is i-connected to P if there is a path $L_{a_1} \ldots L_{a_n}$ in the DAG induced by π, with $L_{a_1} = C$ and $L_{a_n} = P$, such that no member of $L_{a_1} \ldots L_{a_n}$ is derived by universal reduction at any level $j \leq i$.*

Definition 4.1.4. *The level i derivation of a clause P in π, denoted $\pi(P, i)$, is the subsequence of π ending at P and containing exactly those clauses which are i-connected to P.*

Intuitively, consider an arbitrary clause P in the refutation π and, starting from this clause, walk towards the leaves according to the DAG induced by π. Every node visited is an ancestor of P and all ancestors of P are visited, so this is the section of π required to derive P. Next, we discard any node that is the result of universal reduction at some level $j \leq i$ or that could *only* be reached in the walk via such a node. What remains is the level i derivation of P. For the construction that follows we are interested in the level i derivations of every clause that immediately precedes universal reduction at some level $j \leq i$.

Definition 4.1.5. *$\mathcal{P}^i_\pi$ is the set of clauses that are parents of a universal reduction step at any level $j \leq i$ in π, together with $\bot$ at the root of π.*

Any two clauses that both appear in the same level i derivation will not contain complementary universal literals u and $\neg u$ for any $u \in X_i$. We will use the level i derivation of universal reduction steps at level i to identify and then isolate sections of the proof for which there is an assignment to X_i that does not satisfy any clause in that section. This assignment is used as a partial annotation for these clauses and the proof section converts to $\forall$Exp + Res easily. By applying the same method to each universally quantified block of the quantifier prefix of Ψ we will construct a proof that can be fully annotated in way that is consistent with the rules of $\forall$Exp + Res.

Definition 4.1.6 (Level i Expansion of π)**.** *We begin with a Q-Res refutation π and apply the following construction.*

Let P be a clause in $\mathcal{P}^i_\pi$. Find $\pi(P, i)$, and copy this section of the proof. The original clauses are not discarded until later. Each clause $C \in \pi(P, i)$ generates a new identical clause C'. Suppose clause B is a parent of C in π. If B' exists then B' is a parent of C', otherwise let B be a parent of C'.

P has a unique child in π which will now be derived from P' instead of P.

Repeat this process for each member of $\mathcal{P}^i_\pi$. The clauses are ordered so that for each $C \in \pi$, C is derived before any copies of C. In addition, if clause A is derived before C in π then every copy of A is also derived before C (and all of its copies). This ensures that the parent(s) of a proof step always appear before the clause they are used to derive. Among copies of the same clause, assume an ordering based on the order in which the universal reduction steps occur in π.

Clauses from the original refutation which no longer have any children (because copies of that clause are used for every derivation it was involved in) are removed. The result is the level i expansion of π, written $e_i(\pi)$.

We now make some simple observations about $e_i(\pi)$.

Lemma 4.1.7. *The level i derivations of clauses in $\mathcal{P}^i_{e_i(\pi)}$ are disjoint.*

Proof. The clauses copied for $P \in \mathcal{P}^i_\pi$ are exactly the level i derivation of $P' \in \mathcal{P}^i_{e_i(\pi)}$. □

Lemma 4.1.8. *$e_i(\pi)$ is a Q-Res refutation of Ψ.*

Proof. Every derivation is an exact copy of a derivation in π, and the imposed ordering respects the order of derivations. □

Lemma 4.1.9. *$e_i(\pi)$ and π have the same number of universal reduction steps at all (universal) levels $j \leq i$.*

Proof. A clause which is the result of a universal reduction step at level $j \leq i$ does not belong to the level i derivation of any $P \in \mathcal{P}^i_{e_i(\pi)}$, by definition, so will never be copied. The parent of the reduction step is copied exactly once and the original is discarded. □

Lemma 4.1.10. *Every clause in $e_i(\pi)$ is either the result of universal reduction at some level $j \leq i$ or belongs to the level i derivation of some $P \in \mathcal{P}^i_{e_i(\pi)}$.*

Proof. Suppose for a contradiction that C is the last clause in $e_i(\pi)$ which does not belong to the level i derivation of any $P \in \mathcal{P}^i_{e_i(\pi)}$ and is not derived by universal reduction at any level $j \leq i$.

Let D be a child of C. If D was derived by universal reduction at some level $j \leq i$ then C is in $\mathcal{P}^i_{e_i(\pi)}$ and is therefore in its own level i expansion, contradicting the assumption. Suppose D was derived by resolution or reduction at any level greater than i. If D belongs to the level i derivation of some $P \in \mathcal{P}^i_{e_i(\pi)}$ then by definition so does C, but if it does not then C was not the last such clause in the proof, so we have a contradiction.

The final line of the proof is $\perp$ which belongs to its own level i derivation. □

Lemma 4.1.11. *The parent and child of any proof step in $e_i(\pi)$ cannot belong to level i derivations of different members of $\mathcal{P}^i_{e_i(\pi)}$. The two parents of a resolution step in $e_i(\pi)$ cannot belong to level i derivations of different members of $\mathcal{P}^i_{e_i(\pi)}$.*

Proof. For any $P \in \mathcal{P}^i_{e_i(\pi)}$, the clauses of the level i derivation of P were all copied for the same clause from π. Therefore, other than P itself they cannot be used to derive any clause that does not also belong to the level i derivation of P.

P derives a clause by universal reduction at level $j \leq i$, therefore this clause cannot be in any level i derivation.

If C is in the level i derivation of P for some $P \in \mathcal{P}^i_{e_i(\pi)}$ and its parent B is not, then B is the result of a universal reduction step at some level $j \leq i$ and so cannot be included in the level i derivation of any clause. □

Lemma 4.1.12. *The size of $e_i(\pi)$ is at most $|\pi|^2$.*

Proof. If there are s universal reduction steps at level $j \leq i$ then each clause in π may be copied up to $s+1$ times. Therefore the size of the level i expansion of π is at most $|\pi| \cdot (s+1)$. Clearly $s < |\pi|$. □

Since the level i expansion of a Q-Res refutation is itself a Q-Res refutation, we can apply the process iteratively for different values of i. We will expand the proof for each universal level, starting from the innermost.

Although the level i expansion may square the size of the proof in the worst case, when our proofs are tree-like each clause is copied at most once, and if a clause is copied then the original is deleted, the proof does not grow at all. This holds for all levels so the proof remains the same size and tree-like when expanded for every universal level even without the assumption of bounded alternation.

Definition 4.1.13. *The complete expansion of π is denoted $E(\pi)$ and defined as*

$$E(\pi) = \begin{cases} e_1(e_3 \dots (e_{k-1}(\pi))) & \text{if } \mathcal{Q}_1 = \forall, \\ e_2(e_4 \dots (e_{k-1}(\pi))) & \text{if } \mathcal{Q}_1 = \exists. \end{cases}$$

Intermediate stages are labelled π_j (where $\mathcal{Q}_j = \forall$), so that

$$\pi_j = e_j(\pi_{j+2}) = e_j(e_{j+2} \dots (e_{k-1}(\pi))).$$

The observations we made about the level i expansion of π lift easily to the full expansion of π.

Lemma 4.1.14. *$E(\pi)$ is a Q-Res refutation of Ψ.*

Proof. This follows from repeated application of Lemma 4.1.8. □

Lemma 4.1.15. *The number of universal reduction steps at level $j \leq i$ in π_{i+2} equals the number of universal reduction steps at level $j \leq i$ in π.*

Proof. This follows from repeated application of Lemma 4.1.9. □

Lemma 4.1.16. $|E(\pi)| \leq |\pi|^{(1+k)/2}$.

Proof. The argument proceeds by a simple induction on the number of universal levels that have been expanded, showing that for every level ℓ with $\mathcal{Q}_\ell = \forall$, the size of π_ℓ is no greater than $|\pi|^{(1+(k+1-\ell))/2}$.

Let s be the number of universal reduction steps in π. Then, following Lemma 4.1.12,

$$|\pi_{k-1}| \leq |\pi| \cdot (s+1) \leq |\pi|^2 \leq |\pi|^{1+(k+1-(k-1))/2}.$$

Assume the hypothesis for $k-1, \dots, i+2$. Since π_{i+2} has the same number of universal reduction steps as π at all levels $j \leq i$ (Lemma 4.1.15), we have that

$$|\pi_i| \leq |\pi_{i+2}| \cdot (s+1) \leq |\pi|^{1+(k+1-(i+2))/2} \cdot |\pi| \leq |\pi|^{1+(k+1-i)/2}.$$

Since $E(\pi)$ is either π_1 (if $\mathcal{Q}_1 = \forall$) or π_2 (if $\mathcal{Q}_1 = \exists$), we have that

$$|E(\pi)| \;\leq\; \max\{|\pi|^{1+k/2}, |\pi|^{1+(k-1)/2}\}.$$

□

In summary, $E(\pi)$ is a Q-Res refutation whose size is polynomial in the size of π whenever the refuted formula had bounded quantified complexity. In constructing $E(\pi)$ we have identified sections of the original refutation which do not contain any opposing universally quantified literals. These sections are defined using the position of universal reduction steps in π and may overlap in π. By copying sections of π we have created a Q-Res refutation consisting of sub-derivations which contain no opposing universally quantified literals and do not overlap. These derivations are connected together by the universal reduction steps.

We will proceed to show that $E(\pi)$ can be turned into an $\forall$Exp + Res refutation in a very natural way.

4.1.2 Annotating the Expanded Proof

We introduce a system of labelling clauses in the proofs π_i with partial assignments to the universally quantified variables of Ψ. Eventually, every clause in $E(\pi)$ will be associated with a complete assignment to the universal variables.

Definition 4.1.17. *For a clause $P \in \mathcal{P}^i_{\pi_i}$ the assignment α^P_i sets variables in X_i that are also in P so that P is not satisfied, and sets all other variables in X_i to 0.*

Lemma 4.1.18. *Let $P \in \mathcal{P}^i_{\pi_i}$. Then α^P_i does not satisfy any $C \in \pi_i(P, i)$.*

Proof. $C \in \pi_i(P, i)$, so by construction there is a path between C and P that does not include any clause derived by universal reduction at level $j \leq i$. Therefore any level i literal $u \in C$ also appears in every clause along this path, including in P, so any assignment to variables of X_i that does not satisfy P also will not satisfy C. □

Immediately after generating π_i from π_{i+2} add the following labels: For each $P \in \mathcal{P}^i_{\pi_i}$, label all clauses in $\pi_i(P, i)$ with α^P_i. Any clause in π_i that is not in a level i derivation of some $P \in \mathcal{P}^i_{\pi_i}$ does not contain any literal at level i by Lemma 4.1.10, so is not satisfied by any assignment to level i variables. Label these clauses with the assignment setting all level i variables to 0.

In subsequent expansions, clauses are copied with their labels. This means that all clauses in π_i will be labelled with a complete assignment to all levels greater than or equal to i, and that $E(\pi)$ will have all clauses labelled with a complete assignment to the universal variables in Ψ. No clause is labelled twice (Lemma 4.1.7). Finally we show that these labels on the clauses of $E(\pi)$ can become the annotations of an $\forall\mathsf{Exp}+\mathsf{Res}$ refutation.

Lemma 4.1.19. *In π_i let clause B be a parent of clause C, and let $\ell \geq i$ with $\mathcal{Q}_\ell = \forall$. If both B and C contain any existentially quantified literal belonging to a level greater than ℓ (not necessarily the same literal), then B and C are both labelled with the same assignment for level ℓ. Similarly, if C is derived by resolution and its two parents A and B both contain any existentially quantified literal belonging to a level greater than ℓ, then A and B are labelled with the same assignment for level ℓ.*

Proof. For any universal level $\ell > i$ assume that the result holds for refutation π_{i+2} and recall that every derivation in π_i is an exact copy of a derivation in π_{i+2}. Labels are copied with clauses, so the result also holds for π_i. For the base case where $i = k - 1$, π_{i+2} does not exist but we begin instead with π. There is no universal level $\ell > k - 1$.

For level i consider the labels given to the parent B and child C of a proof step in π_i.

- If either clause does not belong to some $\pi_i(P, i)$ then it is the result of universal reduction at a level $j \leq i$ (Lemma 4.1.10) and so it does not contain any existential literal with level greater than i.
- If both B and C are in $\pi_i(P, i)$ then they are both labelled with α^P_i.
- It is not possible for B and C to belong to level i derivations of different clauses in $\mathcal{P}^i_{\pi_i}$ (Lemma 4.1.11).

We reason identically about the two parents of a resolution step. The two clauses cannot be in different level i derivations (Lemma 4.1.11), and if either is not in a level i derivation then it contains no existential literals at any level greater than i (Lemma 4.1.10). □

4.1.3 The Simulation

We have seen that every clause in $E(\pi)$ can be associated with a complete assignment to the universally quantified variables. To create an $\forall\mathsf{Exp+Res}$ proof from $E(\pi)$, we label each clause with an assignment to the universally quantified variables, as above, and use the clause labels to generate annotations for the existentially quantified literals. By construction, the assignment given in a clause label does not satisfy that clause. The universally quantified literals are removed from clause C and every existentially quantified literal $x \in C$ is given as its annotation the restriction of the clause label to the level of x. Lemma 4.1.19 shows that the annotations are consistent between parent and child, that the annotations of the pivot literals match, and that any existentially quantified literal appearing in both parents of a resolution step will have the same annotation. The main result now follows easily.

Theorem 4.1.20. *Let Ψ be a QBF with k blocks in the quantifier prefix and π a $\mathsf{Q\text{-}Res}$ refutation of Ψ. Then there is an $\forall\mathsf{Exp+Res}$ refutation of Ψ of size at most $|\pi|^{1+k/2}$.*

Proof. From π, generate the $\mathsf{Q\text{-}Res}$ refutation $E(\pi)$ of Ψ and label the clauses of $E(\pi)$ as described in Section 4.1.2. The assignment given in a clause label does not satisfy that clause. Remove all universally quantified literals from clauses of $E(\pi)$ and replace all existentially quantified literals with annotated literals. An existentially quantified literal x in a clause C with label α is replaced by the annotated literal $x^{\lfloor \alpha \rfloor_x}$. Universal reduction steps are now meaningless and can be removed. Resolution steps remain, acting on annotated literals with matching annotations (Lemma 4.1.19). The leaves of $E(\pi)$ were all copies of leaves in π, i.e. clauses from Ψ, so the leaves of the constructed $\forall\mathsf{Exp+Res}$ refutation are annotated versions of those same clauses from Ψ. This gives a valid $\forall\mathsf{Exp+Res}$ refutation of Ψ constructed from the $\mathsf{Q\text{-}Res}$ refutation. By Lemma 4.1.16, the new refutation has size at most $|\pi|^{1+k/2}$. □

The cost of transforming a DAG-like $\mathsf{Q\text{-}Res}$ refutation into an $\forall\mathsf{Exp+Res}$ refutation by this construction depends on the quantifier complexity. We can now demonstrate that $\forall\mathsf{Exp+Res}$ is strictly stronger than $\mathsf{Q\text{-}Res}$ for DAG-like proofs, when restricted to QBFs of bounded quantifier complexity.

Theorem 4.1.21. *For each $k \geq 3$, $\forall\mathsf{Exp}+\mathsf{Res}$ p-simulates $\mathsf{Q\text{-}Res}$ on Σ_k^b formulas, but the reverse simulation does not hold, and there are Σ_3^b formulas providing an exponential separation.*

Proof. The simulation is given by Theorem 4.1.20 and the exponential separation is known from Beyersdorff *et al.* (2015) in which is a family of Σ_3^b formulas ($\text{QP}\textsc{arity}_n$) are shown to have polynomial size $\forall\mathsf{Exp}+\mathsf{Res}$ refutations but require exponential size $\mathsf{Q\text{-}Res}$ refutations. $\square$

From Theorem 4.1.20 it also follows that on QBFs with polylogarithmic quantifier blocks, $\forall\mathsf{Exp}+\mathsf{Res}$ simulates $\mathsf{Q\text{-}Res}$ with only a quasi-polynomial blow-up in proof size. Thus the separating examples with polynomial-size $\mathsf{Q\text{-}Res}$ refutations but requiring exponential-size refutations in $\forall\mathsf{Exp}+\mathsf{Res}$ necessarily have super-polylogarithmic quantifier complexity.

We also observe that for QBFs with only one or two quantifier levels $\mathsf{Q\text{-}Res}$ and $\forall\mathsf{Exp}+\mathsf{Res}$ are p-equivalent. Since we always assume the innermost block to be existentially quantified, and both systems are clearly equivalent to propositional Resolution when restricted to propositional formulas, we only need to check the case of Π_2^b formulas.

Lemma 4.1.22. *$\mathsf{Q\text{-}Res}$ and $\forall\mathsf{Exp}+\mathsf{Res}$ are p-equivalent on Π_2^b formulas.*

Proof. Consider a QBF of the form $\forall U \exists X \Phi(U, X)$, where X and U are blocks of variables and Φ is a CNF. We first show that in an $\forall\mathsf{Exp}+\mathsf{Res}$ refutation of $\forall U \exists X \Phi(U, X)$, every existentially quantified literal has the same annotation. This will be used to transform it into a $\mathsf{Q\text{-}Res}$ refutation.

We proceed by induction on the derivation depth.

Induction Hypothesis: For every clause in an $\forall\mathsf{Exp}+\mathsf{Res}$ refutation with depth at most d in the DAG all the literals in that clause have the same annotation.

Base Case: In every axiom clause (depth 0), every literal is in block X and receives the same annotation in the variables of Y.

Inductive Step: In order to perform a resolution step, the annotations need to match exactly. This means, under our induction hypothesis, that all the literals in each of the two parent clauses must have the same annotation. Since the literals which are resolved must have the same annotation it follows that all the literals in the resolvent also have the same annotation.

We therefore conclude that every clause has the same annotation on all its literals; furthermore since child clauses inherit their parents' annotations, all annotations are the same across all the clauses in the proof.

We can now transform this directly into a $\mathsf{Q\text{-}Res}$ proof, keeping the resolution steps the same, however our axioms now contain universal literals. The universal literals will never create a tautology, because they are all falsified by the same assignment. Universally quantified literals which do not actually appear in the axiom clauses can be removed, this is easily propagated through the proof. After performing all of the resolution steps we have a clause containing only universally quantified literals, which are removed by universal reduction.

For the converse, we observe that all universal reduction steps must happen at the end of a Q-Res refutation of $\forall U \exists X \Phi(U, X)$. If any one literal can be removed by universal reduction we must have a clause containing no existentially quantified literals so we can derive $\perp$. The clause P immediately preceding these universal reduction steps is not a tautology, since Q-Res does not permit tautologies. We take an assignment to U that falsifies P. This assignment does not satisfy any clause in the refutation so it can be used throughout a similar $\forall$Exp + Res refutation. $\square$

4.1.4 Empirical Observations Relating Solver Performance and Quantifier Complexity

We have shown that $\forall$Exp + Res strictly p-simulates Q-Res for all QBFs with bounded quantifier complexity, and that the degree of the polynomial increases with the number of quantifier blocks. There are formulas with bounded quantifier complexity for which there exist $\forall$Exp + Res proofs which are exponentially smaller than the smallest Q-Res proof. Although there may also be formulas with bounded quantifier complexity for which have smaller Q-Res proofs than $\forall$Exp + Res proofs, the difference in size is limited according to the p-simulation. We may therefore anticipate that solvers which produce $\forall$Exp + Res proofs could have an advantage (on average) over solvers which produce Q-Res proofs, especially when the number of quantifier alternations is low and therefore the degree of the polynomial is small. If so, we would also anticipate that this advantage decreases as the number of quantifier alternations increases.

Lonsing & Egly (2018a) present an empirical study of the effect of quantifier complexity on the performance of QBF solvers from QBFEVAL17 (Pulina & Seidl, 2017). They sought to highlight the negative impact on solver development of biased benchmark sets, noting a tendency to over-represent problems with low quantifier complexity. Their experiments showed that solvers based on QCDCL often perform most strongly, in comparison to other solvers, on instances with many quantifier blocks. Conversely, the solvers based on universal expansion were generally most effective on instances with few quantifier blocks.

The relationship between proof complexity results and solver performance is not at all straightforward. From a theoretical point of view, if a proof system f p-simulates another system g then we say f is stronger than g. However, when assessing the performance of solving algorithms, it is usually more relevant to directly compare the amount of time required to decide a QBF. Further, proof complexity intentionally ignores implementation details and the question of how easily an algorithm can find a given proof. Despite these difficulties in relating theoretical and empirical results, it is interesting to observe that in this case there is a correlation between the two. QCDCL is the solving paradigm most closely related to Q-Res, and expansion solving to $\forall\mathsf{Exp}+\mathsf{Res}$. As such, the observation that QCDCL solvers perform relatively well on formulas with many quntifier alternations, and expansion solvers perform more strongly on formulas with few quantifier alternations, aligns with the result given here.

4.2 Simulating QU-Resolution by Q-Resolution

In general, Q-Res cannot p-simulate QU-Res. The examples showing the separation are from Kleine Büning *et al.* (1995). Polynomial-size QU-Res refutations were given in Van Gelder (2012). We recall the formulas here and sketch the short QU-Res refutations.

$$\begin{aligned}\mathrm{KBKF}_n = {} & \exists x_0, x_{1,0}, x_{1,1} \forall u_1 \exists x_{2,0}, x_{2,1} \forall u_2 \dots \exists x_{i,0}, x_{i,1} \forall u_i \dots \exists x_{n,0}, x_{n,1} \forall u_n \exists t_1 \dots t_n \\ & (\neg x_0) \wedge (x_0 \vee \neg x_{1,0} \vee \neg x_{1,1}) \\ & \wedge \bigwedge_{i=1}^{n-1} (x_{i,0} \vee u_i \vee \neg x_{i+1,0} \vee \neg x_{i+1,1}) \wedge \bigwedge_{i=1}^{n-1} (x_{i,1} \vee \neg u_i \vee \neg x_{i+1,0} \vee \neg x_{i+1,1}) \\ & \wedge (x_{n,0} \vee u_n \vee \neg t_1 \vee \dots \vee \neg t_n) \wedge (x_{n,1} \vee \neg u_n \vee \neg t_1 \vee \dots \vee \neg t_n) \\ & \wedge \bigwedge_{i=1}^{n} (u_i \vee t_i) \wedge \bigwedge_{i=1}^{n} (\neg u_i \vee t_i).\end{aligned}$$

The formulas have a total of $2n+1$ existentially quantified x variables, n universally quantified u variables and n existentially quantifier t variables. To see that the KBKF formulas are indeed false, consider the universal strategy which sets $u_i = 1$ when either $x_{i,0} = 0$ or $x_{i,1} = 0$. To satisfy the clauses in the first line we require either $x_{1,0} = 0$ or $x_{1,1} = 0$. Now, when playing according to the suggested strategy, the second line ensures that one of $x_{i,0} = 0$ or $x_{i,1} = 0$ for each i, which finally leads to the need to make $t_i = 0$ for some i. This in turn leads to the requirement that $u_i = 0$ and $u_i = 1$, creating a contradiction.

Theorem 4.2.1. *(Van Gelder, 2012) The formulas KBKF$_n$ have polynomial-size* QU-Res *refutations.*

Proof Sketch. For each $i = 1, \dots, n$ derive (t_i) from $(\neg u_i \vee t_i)$ and $(u_i \vee t_i)$ by universal resolution. After this, the formula can be refuted without further universal resolution steps. In linearly many resolution steps we derive $(x_{n,0} \vee u_n)$ and $(x_{n,1} \vee \neg u_n)$, universal reduction then derives $(x_{n,0})$ and $(x_{n,1})$. A similar sequence of steps derives $(x_{i,0})$ and

$(x_{i,1})$ for decreasing values of i until finally we can derive (x_0) which is resolved with $(\neg x_0)$ to complete the proof. $\square$

Kleine Büning *et al.* (1995) proved that KBKF_n require exponential-size refutations in a restricted version of Q-Res, the result was generalised in Beyersdorff *et al.* (2015) to show that they require exponential-size refutations in unrestricted Q-Res and in $\forall$Exp + Res.

In KBKF_n the number of blocks in the quantifier prefix is linear in the size of the formula. We show that this is necessary by demonstrating that the construction from Section 4.1 can be modified to act on a QU-Res refutation. This shows that $\forall$Exp + Res p-simulates QU-Res for QBFs with bounded quantifier complexity. This modified construction transforms the QU-Res refutation into a Q-Res refutation which can then be easily annotated so it also shows that Q-Res p-simulates QU-Res for QBFs with bounded quantifier complexity. Since QU-Res trivially p-simulates Q-Res we now have that the two systems are p-equivalent in this case.

As before, there is no increase in the size of a tree-like refutation undergoing the transformation so we also prove that tree-like QU-Res and tree-like Q-Res are p-equivalent.

During the construction we will introduce some weakening proof steps. These allow extra literals to be added into a clause provided that doing so does not create a tautology. Allowing weakening does not strengthen any of the proof systems we are considering here. We will show that a refutation of a QBF in QU-Res with weakening can be transformed into a refutation in Q-Res with weakening and then into a refutation in $\forall$Exp + Res, and that there is at most a polynomial increase in the proof size when the quantifier complexity of the QBF is bounded above by a constant.

Definition 4.2.2. *Let* f *be a proof system in* $\{\forall$*Exp + Res, Q-Res, QU-Res*$\}$*. Let* W *be a set of literals. The system* f *with weakening allows a proof step of the form*

$$\frac{C}{C \cup W}$$

and if f *allows the derivation of clause* C *by any proof step then* f *with weakening allows clause* $C \cup W$ *to be derived by the same proof step with weakening. In all cases* $C \cup W$ *must not be a tautology.*

It is simple to prove that for a proof system $\mathsf{f} \in \{\forall$Exp + Res, Q-Res, QU-Res$\}$ a proof in f with weakening can be converted into a proof in f (without weakening) with no increase in proof size. Wherever weakening is used we instead make the same derivation without weakening. This may cause subsequent proof steps to include weakening but otherwise the proof is unchanged. Repeating this allows any weakening to be pushed towards the root of the proof until none remains.

Let $\pi = L_1 \dots L_m$ be a refutation of PCNF $\Psi = \mathcal{Q}_1 X_1 \dots \mathcal{Q}_k X_k \Phi$ in QU-Res with weakening and let i be an index with $\mathcal{Q}_i = \forall$.

First, Definitions 4.1.3, 4.1.4 and 4.1.6 are modified. Previously we sought sub-derivations of π ending at clause P and which did not contain any universal reduction at level i. We now also require that these sub-derivations contain no universal resolution on a level i pivot, but instead of excluding the clauses derived by these universal resolution steps entirely we enforce that the sub-derivation will contain exactly one parent of the proof step. The intention is, as before, to define sections of the proof which can be consistently annotated. Then we observe that if the sub-derivation ends with a universal reduction that removes any level i literals we don't need to remove any of these universal literals by resolution in this part of the proof.

Definition 4.2.3. *Let C and P be clauses in π. C is i_α-connected to P if there is a sub-sequence $L_{a_1} \dots L_{a_n}$ of π such that*

- *$L_{a_1} = C$,*
- *$L_{a_n} = P$,*
- *α is a complete assignment to the variables in X_i,*
- *α does not satisfy P,*
- *For all $\ell \in \{2 \dots n\}$*
  - *$L_{a_{\ell-1}}$ is a parent of L_{a_ℓ} in π,*
  - *L_{a_ℓ} is not derived by universal reduction at any level $j \leq i$,*
  - *If L_{a_ℓ} is derived by universal resolution with pivot variable in X_i then α does not satisfy $L_{a_{\ell-1}}$.*

Because α is a complete assignment to X_i there is exactly one parent of any universal resolution step with pivot in level i that is not satisfied by α.

Definition 4.2.4. *Given a level i assignment α, $\pi(P, i_\alpha)$ is the sub-sequence of π ending at clause P and containing exactly those clauses i_α-connected to P.*

By construction, $\pi(P, i_\alpha)$ contains no universal resolution steps with a pivot in X_i, and no clauses in $\pi(P, i_\alpha)$ are satisfied by α. A clause in $\pi(P, i_\alpha)$ that is derived by universal resolution in π has only one parent in $\pi(P, i_\alpha)$. If we attempted to derive the clause only from the parent in $\pi(P, i_\alpha)$ this is not a valid QU-Res proof step but would have the form

$$\frac{C_1 \vee u}{C_1 \vee C_2}$$

where the addition of variables in C_2 is allowed by weakening, but the removal of u from the clause is not permitted in any QU-Res rule. The proof section $\pi(P, i_\alpha)$ is defined so that none of the clauses contain $\bar{u}$. When copying the proof sections, we ensure that every clause copied from $\pi(P, i_\alpha)$ does contain u. As a result, it is possible to remove one parent from each universal resolution step with pivot in X_i, and use weakening instead. The definition of $e_i(\pi)$ is updated to use i_α-connection instead of i-connection. Otherwise the idea is identical to that of Definition 4.1.6.

Definition 4.2.5 (Modified Level i Expansion of π)**.** *For every $P \in \mathcal{P}^i_\pi$ define α so that it does not satisfy P and all variables in X_i that do not appear in P are assigned to* 0*.*

If $C \in \pi(P, i_\alpha)$ then define $C' = C \cup \{l \mid \bar{l} \in \alpha\}$.

Let B be a parent (in π) of $C \in \pi(P, i_\alpha)$. Then B' is a parent of C', if it exists. If B' does not exist and C was not derived by universal resolution on a pivot in X_i then B is a parent of C'.

The clauses of the new proof are ordered so that clause C appears before any of the clauses generated from C. If clause A appears before C in π then A and every clause generated by A must occur before C in the modified proof. Among copies of the same clause assume an ordering based on the order in which the universal reduction steps occurred in π.

The clause D derived from P is instead derived from P' (by universal reduction at level i). P' may contain additional literals that were not in P but these are all from X_i and are removed in the universal reduction step that derives D.

The result of this procedure applied to QU-Res *proof π for universal level i is denoted $e'_i(\pi)$.*

We need to show that the result of this construction is a valid QU-Res proof (with weakening). Then it can be applied for each level of the prefix.

Lemma 4.2.6. *$e'_i(\pi)$ does not contain any tautologies.*

Proof. By definition the clauses in $\pi(P, i_\alpha)$ are not satisfied by α, so adding in the level i literals which evaluate to 0 under α cannot introduce tautologies. Newly generated clauses only differ from clauses in the original proof by the addition of universally quantified literals that are falsified by α. □

We can also observe that these additional universal literals do not block any resolution or universal reduction step so every proof step in $e'_i(\pi)$ is valid in QU-Res with weakening.

Lemma 4.2.7. *If π is a refutation in QU-Res with weakening then so is $e'_i(\pi)$.*

Proof. Every derivation in $e'_i(\pi)$ is based on a derivation in π except for the case of a newly created clause that was previously derived by universal resolution at level i. In this case, the clause is now derived from a single parent, which was also generated for the same clause P from $\mathcal{P}^i_\pi$. Therefore both clauses contain all the literals assigned 0 by α. The derivation

$$\frac{C_1 \cup \{u\} \qquad C_2 \cup \{\bar{u}\}}{C_1 \cup C_2}$$

has been replaced by

$$\frac{C_1 \cup \{l \mid \bar{l} \in \alpha\}}{C_1 \cup C_2 \cup \{l \mid \bar{l} \in \alpha\}}$$

which is a pure weakening step.

For $C \in \pi(P, i_\alpha)$, C' only differs from C in universal literals at level i that are falsified by α. Any parent of C' cannot contain any of the literals satisfied by α because either it belongs to $\pi(P, i_\alpha)$ or it was derived by universal reduction at level $j \leq i$ so cannot contain any literals from level i. Therefore the extra literals in C' do not block the proof step used to derive it. The pivot variable is the same as the pivot used in deriving C.

The ordering of clauses in $e'_i(\pi)$ respects the ordering of derivations. The result follows from this together with Lemma 4.2.6. □

Lemma 4.2.8. *$e'_i(\pi)$ contains no universal resolution steps with pivot in X_i.*

Proof. Every clause in $e'_i(\pi)$ is either derived by universal reduction at level $j \leq i$ or is newly generated for a clause in $\pi(P, i_\alpha)$ for some $P \in \mathcal{P}^i_\pi$ (by a simple modification of Lemma 4.1.10). For any such P, $\pi(P, i_\alpha)$ contains no universal resolution steps with pivot in X_i. □

As before, observe that no new universal reduction steps at level $j \leq i$ are introduced in constructing $e'_i(\pi)$. We expand π from the inner-most to outer-most universal levels in Ψ.

Definition 4.2.9. *The modified complete expansion of π is denoted $E'(\pi)$ and is defined as*

$$E'(\pi) = \begin{cases} e'_1(e'_3 \dots (e'_{k-1}(\pi))) & \text{if } \mathcal{Q}_1 = \forall, \\ e'_2(e'_4 \dots (e'_{k-1}(\pi))) & \text{if } \mathcal{Q}_1 = \exists. \end{cases}$$

Intermediate stages are labelled π'_j (where $\mathcal{Q}_j = \forall$), so that

$$\pi'_j = e'_j(\pi'_{j+2}) = e'_j(e'_{j+2} \dots (e'_{k-1}(\pi))).$$

Lemma 4.2.10. *$E'(\pi)$ is a refutation of Ψ in Q-Res with weakening.*

Proof. $E'(\pi)$ is a QU-Res refutation by Lemma 4.2.7 but contains no universal resolution at any level by repeated application of Lemma 4.2.8. All derivation steps in $E'(\pi)$ are based on derivations in π or are weakening steps deriving a clause that was previously derived by universal resolution. Therefore it is a proof in Q-Res with weakening. □

Lemma 4.2.11. $|E'(\pi)| \leq |\pi|^{1+k/2}$.

Proof. The proof is identical to Lemma 4.1.16. □

The weakening can be removed from $E'(\pi)$ without increase in proof size and we can easily annotate the clauses to reach an $\forall$Exp + Res proof.

The annotation of clauses proceeds as before, in fact it is simpler because we added universal literals into the clauses as part of the expansion stage. The only clauses that do not contain a literal for every variable in X_i are those which are derived by universal reduction at some level $j \leq i$, so the universal literals in each clause are sufficient to define the annotation which will be applied to every existentially quantified literal in that clause. The equivalent of Lemma 4.1.18 is immediate from the construction of $e_i'(\pi)$ and Lemma 4.2.6.

Theorem 4.2.12. *Let Ψ be a QBF with k blocks in the quantifier prefix and π a QU-Res refutation of Ψ. Then there is a Q-Res refutation and an $\forall$Exp + Res refutation of Ψ, each of size at most $|\pi|^{1+k/2}$.*

Proof. From π, generate the Q-Res refutation $E'(\pi)$ of Ψ as described above. This has size at most $|\pi|^{1+k/2}$.

To generate the $\forall$Exp + Res refutation from $E'(\pi)$ observe that for every clause $C \in E'(\pi)$, if $x \in C$ is existentially quantified, then C contains exactly one literal of every universally quantified u with $\mathrm{lv}(u) \leq \mathrm{lv}(x)$. Each $x \in C$ is annotated with the assignment that does not satisfy the universal literals of C, and the universal literals are then removed from C. It is straightforward to observe that the annotations must obey the rules of $\forall$Exp + Res, and clearly this proof also has size at most $|\pi|^{1+k/2}$. □

Theorem 4.2.13. *For every k, Q-Res and QU-Res are p-equivalent on Σ_k^b formulas. For every k, $\forall$Exp + Res p-simulates QU-Res on Σ_k^b formulas but the reverse simulation does not hold for $k \geq 3$, and there are Σ_3^b formulas providing an exponential separation.*

Proof. The simulation follows from the construction given above, and the separation between $\forall$Exp + Res and QU-Res is again given by the QParity$_n$ formulas. By definition QU-Res can p-simulate Q-Res. □

Theorem 4.2.14. *Tree-like* $\forall$Exp+Res *and tree-like* Q-Res *p-simulate tree-like* QU-Res.

Proof. In the case of tree-like QU-Res no clause needs to be copied twice during the construction of $e'_i(\pi)$ and so there is no increase in proof size at each quantifier level. Therefore the restriction to formulas with bounded quantifier complexity is not required. □

4.3 Long Distance Q-Resolution for QBFs with Bounded Quantifier Complexity

4.3.1 QBFs Based on Parity

The QBFs QParity$_n$ were introduced in Beyersdorff *et al.* (2015). The formulas are parameterised by a natural number n and express the claim that, given a set of Boolean inputs $\{x_i\}_{i=1}^n$, the parity of the number of 1s in $\{x_i\}$ is different from z for all choices of $z \in \{0,1\}$.

Figure 4.3 shows a parity circuit. Each t_i calculates the exclusive or of t_{i-1} and x_i, (denoted $t_{i-1} \oplus x_i$), which is true when exactly one of t_{i-1} and x_i is true. Equivalently, each t_i calculates the parity of $x_1 \dots x_i$, so that $t_n \equiv \textsc{Parity}(x_1, \dots, x_n)$.

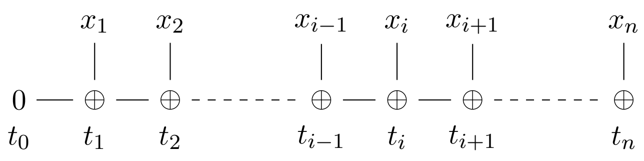

Figure 4.3: An example of a parity circuit for input of size n.

The formulas are false due to the addition of a universally quantified variable z, and the requirement that $t_n = \neg z$. The matrix is falsified whenever $z = \textsc{Parity}(x_1, \dots, x_n)$. The QBFs are given as

$$\textsc{QParity}_n = \exists x_1 \dots \exists x_n \forall z \exists t_0 \dots \exists t_n$$
$$(z \vee t_n) \wedge (\neg z \vee \neg t_n) \wedge (\neg t_0) \wedge \bigwedge_{i=1}^{n} (t_i = t_{i-1} \oplus x_i)$$

where each $(t_i \equiv t_{i-1} \oplus x_i)$ is expressed by the four clauses

$$(\neg t_{i-1} \vee x_i \vee t_i),\ (t_{i-1} \vee \neg x_i \vee t_i),\ (t_{i-1} \vee x_i \vee \neg t_i),\ (\neg t_{i-1} \vee \neg x_i \vee \neg t_i).$$

The QParity$_n$ formulas are known to require exponential-size refutations in Q-Res, but have polynomial-size LD-Q-Res refutations. This provides a separation between Q-Res and LD-Q-Res on formulas with only three quantifier blocks. Since LD-Q-Res is an extension of Q-Res it is immediate to see that LD-Q-Res p-simulates Q-Res, so LD-Q-Res is strictly stronger than Q-Res, including on QBFs with bounded quantifier complexity.

The QParity$_n$ formulas also have polynomial-size $\forall$Exp + Res refutations, and can be modified so that they remain easy for $\forall$Exp + Res but are hard for LD-Q-Res (Beyersdorff *et al.*, 2015). The short LD-Q-Res refutations of QParity$_n$ iteratively derive clauses $(t_i \vee z^*)$ and $(\neg t_i \vee z^*)$ for decreasing i, which finally allows resolution with $(\neg t_0)$ and universal reduction to derive the empty clause. The refutation relies on the fact that z does not appear in the four clauses expressing $(t_i \equiv t_{i-1} \oplus x_i)$. In the modified formulas, $(\neg t_{i-1} \vee x_i \vee t_i)$ is replaced with $(\neg t_{i-1} \vee x_i \vee t_i \vee z) \wedge (\neg t_{i-1} \vee x_i \vee t_i \vee \neg z)$, and similarly for the other clauses in $(t_i \equiv t_{i-1} \oplus x_i)$, for each i. The short refutations are no longer possible. In particular, it is not possible to resolve $(\neg t_i \vee z^*)$ with $(\neg t_{i-1} \vee x_i \vee t_i \vee z)$ since z appears before t_i in the quantifier prefix.

We will now modify the QParity$_n$ formulas in a different way so that they require exponential-size refutations in $\forall$Exp + Res but still have polynomial-size LD-Q-Res refutations. This shows that when restricted to QBFs with bounded quantifier complexity LD-Q-Res and $\forall$Exp + Res remain incomparable, as in the general case.

4.3.2 Modified Parity Formulas

In the original QParity$_n$ formulas there is only one universal variable z. The universal variable z must be set to the parity of the x_i variables in order for the formula to evaluate to false. We modify the formulas so that there is a universal variable z_i for every node of the circuit in Figure 4.3 (or equivalently, for each existentially quantified t_i). We add clauses to express that if $t_i = \neg z_i$ then the next node in the circuit, t_{i+1}, is no longer constrained to be the parity of t_i and x_{i+1}. As a result, if any of the z_i are not set to equal Parity$(x_1, \ldots, x_i)$ then the remaining t variables are unconstrained and the formula is easily satisfied. However, the formula as a whole remains false, with a unique strategy for the universal player: each universal variable z_i must calculate Parity$(x_1, \ldots, x_i)$ in order for the formula to evaluate to false.

To refute QParity$_n$ using $\forall$Exp + Res, the formula is first expanded by substituting both 0 and 1 for z, which causes it to double in size, and this expanded formula is refuted with a polynomial-sized Resolution refutation. For the modified formulas, an $\forall$Exp + Res refutation will need to consider every assignment to the z_i variables, which gives exponentially many different annotations in the parameter n. This is because of the requirement to set z_i depending on the value of Parity$(x_1, \ldots, x_i)$. Due to these exponentially many annotations, it follows that the refutation itself must be of exponential size.

The QBFs QParity$_n$ have polynomial-size refutations in LD-Q-Res, we will show that the same technique also gives polynomial-size refutations of the modified formulas

QParity$_n$-LD. The QBFs are defined as

$$\begin{aligned}\text{QParity}_n\text{-LD} = {} & \exists x_1 \dots x_n \forall z_1 \dots z_n \exists t_0 \dots t_n \\ & (z_n \vee t_n) \wedge (\neg z_n \vee \neg t_n) \wedge (\neg t_0) \wedge (t_1 = t_0 \oplus x_1) \\ & \wedge \bigwedge_{i=2}^{n} (t_{i-1} = \neg z_{i-1}) \vee (t_i = t_{i-1} \oplus x_i)\end{aligned}$$

where $(t_{i-1} = \neg z_{i-1}) \vee (t_i = t_{i-1} \oplus x_i)$ is expressed by the four clauses

$(\neg t_{i-1} \vee x_i \vee t_i \vee \neg z_{i-1})$, $(t_{i-1} \vee \neg x_i \vee t_i \vee z_{i-1})$, $(t_{i-1} \vee x_i \vee \neg t_i \vee z_{i-1})$, $(\neg t_{i-1} \vee \neg x_i \vee \neg t_i \vee \neg z_{i-1})$.

Lemma 4.3.1. *The formulas* QParity$_n$-LD *have polynomial-sized refutations in LD-Q-Res.*

Proof. We will derive $(t_i \vee z_n^* \vee \dots \vee z_{i+1}^* \vee z_i)$ and $(\neg t_i \vee z_n^* \vee \dots \vee z_{i+1}^* \vee \neg z_i)$ for $i = n-1, \dots, 1$. For $i = n$ we begin with the clauses $(z_n \vee t_n)$ and $(\neg z_n \vee \neg t_n)$ from the input formula.

$$1.\ \frac{(\neg t_{i-1} \vee \neg x_i \vee \neg t_i \vee \neg z_{i-1}) \qquad (t_i \vee z_n^* \vee \dots \vee z_{i+1}^* \vee z_i)}{(\neg t_{i-1} \vee \neg x_i \vee z_n^* \vee \dots \vee z_{i+1}^* \vee z_i \vee \neg z_{i-1})}$$

$$2.\ \frac{(\neg t_{i-1} \vee x_i \vee t_i \vee \neg z_{i-1}) \qquad (\neg t_i \vee z_n^* \vee \dots \vee z_{i+1}^* \vee \neg z_i)}{(\neg t_{i-1} \vee x_i \vee z_n^* \vee \dots \vee z_{i+1}^* \vee \neg z_i \vee \neg z_{i-1})}$$

$$3.\ \frac{(t_{i-1} \vee x_i \vee \neg t_i \vee z_{i-1}) \qquad (t_i \vee z_n^* \vee \dots \vee z_{i+1}^* \vee z_i)}{(t_{i-1} \vee x_i \vee z_n^* \vee \dots \vee z_{i+1}^* \vee z_i \vee z_{i-1})}$$

$$4.\ \frac{(t_{i-1} \vee \neg x_i \vee t_i \vee z_{i-1}) \qquad (\neg t_i \vee z_n^* \vee \dots \vee z_{i+1}^* \vee \neg z_i)}{(t_{i-1} \vee \neg x_i \vee z_n^* \vee \dots \vee z_{i+1}^* \vee \neg z_i \vee z_{i-1})}$$

These resolution steps are all permitted in LD-Q-Res because the two parent clauses do not contain any of the same z_i variables.

Next, we resolve on x_i the results of lines 1 and 2, and the results of lines 3 and 4. Because x_i appears before any of the z variables in the quantifier prefix this is permitted in LD-Q-Res even though the clauses contain the opposing literals of z_i and both contain $z_n^*, \dots, z_{i+1}^*$. These resolution steps introduce z_i^*.

$$\frac{(\neg t_{i-1} \vee \neg x_i \vee z_n^* \vee \dots \vee z_{i+1}^* \vee z_i \vee \neg z_{i-1}) \qquad (\neg t_{i-1} \vee x_i \vee z_n^* \vee \dots \vee z_{i+1}^* \vee \neg z_i \vee \neg z_{i-1})}{(\neg t_{i-1} \vee z_n^* \vee \dots \vee z_i^* \vee \neg z_{i-1})}$$

$$\frac{(t_{i-1} \vee x_i \vee z_n^* \vee \dots \vee z_{i+1}^* \vee z_i \vee z_{i-1}) \qquad (t_{i-1} \vee \neg x_i \vee z_n^* \vee \dots \vee z_{i+1}^* \vee \neg z_i \vee z_{i-1})}{(t_{i-1} \vee z_n^* \vee \dots \vee z_i^* \vee z_{i-1})}$$

Finally, we have $(\neg t_1 \vee z_n^* \vee \cdots \vee z_2^* \vee \neg z_1)$ and $(t_1 \vee z_n^* \vee \cdots \vee z_2^* \vee z_1)$, and derive the empty clause $(\bot)$ as follows:

$$\frac{(t_0 \vee x_1 \vee \neg t_1) \qquad (t_1 \vee z_n^* \vee \cdots \vee z_2^* \vee z_1)}{(t_0 \vee x_1 \vee z_n^* \vee \cdots \vee z_2^* \vee z_1)}$$

$$\frac{(t_0 \vee \neg x_1 \vee t_1) \qquad (\neg t_1 \vee z_n^* \vee \cdots \vee z_2^* \vee \neg z_1)}{(t_0 \vee \neg x_1 \vee z_n^* \vee \cdots \vee z_2^* \vee \neg z_1)}$$

$$\frac{\dfrac{\dfrac{(t_0 \vee x_1 \vee z_n^* \vee \cdots \vee z_2^* \vee z_1) \qquad (t_0 \vee \neg x_1 \vee z_n^* \vee \cdots \vee z_2^* \vee \neg z_1)}{(t_0 \vee z_n^* \vee \cdots \vee z_1^*)} \qquad (\neg t_0)}{(z_n^* \vee \cdots \vee z_1^*)}}{\bot}$$

□

Now we will show that every possible assignment to the z_i variables is required in the $\forall\mathsf{Exp}+\mathsf{Res}$ refutation. Since each axiom can only contain a single annotation, and the size of the formula is linear in the number of z_i variables, this shows that the number of axiom clauses required in the refutation is exponential in the size of the formula.

Lemma 4.3.2. $\mathrm{QP\small{ARITY}}_n$-LD *require exponential-size refutations in* $\forall\mathsf{Exp}+\mathsf{Res}$.

Proof. Let α be an arbitrary assignment to the z_i variables. We will show that the propositional formula that results from expanding $\mathrm{QP\small{ARITY}}_n$-LD for every assignment except α is satisfiable. Therefore the $\forall\mathsf{Exp}+\mathsf{Res}$ refutation must use clauses annotated with α, but since α is chosen arbitrarily it follows that all 2^n possible annotations are required in the $\forall\mathsf{Exp}+\mathsf{Res}$ refutation.

For any choice of α, a satisfying assignment must have $t_0 = 0$. Set $x_1 = \alpha(z_1)$ and $x_i = \alpha(z_{i-1}) \oplus \alpha(z_i)$ for every $i \geq 2$. It follows that $\alpha(z_i) = \mathrm{P\small{ARITY}}(x_1, \ldots, x_i)$.

Let τ be any assignment to the z_i variables that is different from α. Let j be the minimum index such that $\tau(z_j) \neq \alpha(z_j)$.

For each choice of τ we need to satisfy $(z_n \vee t_n)$, $(\neg z_n \vee \neg t_n)$, and $(t_1 = t_0 \oplus x_1)$, and for every $i \in \{1, \ldots n-1\}$ the clauses representing $(t_i = \neg z_i) \vee (t_{i+1} = t_i \oplus x_{i+1})$. Some of these are satisfied during the expansion by the choice of τ, the rest must be satisfied by the choice of values for the t_i^τ variables.

If $\tau(z_i) = 1$ then the two clauses containing z_i are satisfied, so we have the annotated clauses $(\neg t_i^\tau \vee x_{i+1} \vee t_{i+1}^\tau)$ and $(\neg t_i^\tau \vee \neg x_{i+1} \vee \neg t_{i+1}^\tau)$. If $\tau(z_i) = 0$ then the two clauses containing $\neg z_i$ are satisfied, leaving $(t_i^\tau \vee \neg x_{i+1} \vee t_{i+1}^\tau)$ and $(t_i^\tau \vee x_{i+1} \vee \neg t_{i+1}^\tau)$.

Set $t_0^\tau = 0$ and for $i \in \{1, \ldots, j\}$ set $t_i^\tau = t_{i-1}^\tau \oplus x_i = \mathrm{P\small{ARITY}}(x_1, \ldots, x_i)$. Now the clause $(t_1 = t_0 \oplus x_1)$ and whichever two clauses remain for $i-1$ under assignment τ, are all satisfied by this choice of t_i.

$$\begin{tikzcd}
\forall\mathsf{Exp}+\mathsf{Res} \arrow[rr, dotted, no head] \arrow[dr, no head] & & \mathsf{LD\text{-}Q\text{-}Res} \arrow[dl, no head] \\
& \mathsf{Q\text{-}Res} \arrow[r, equal] & \mathsf{QU\text{-}Res}
\end{tikzcd}$$

Figure 4.4: Relationships between QBF proof systems for formulas with bounded quantifier complexity.

Observe that $t_j^\tau = \textsc{Parity}(x_1, \ldots, x_j) = \alpha(z_j) \neq \tau(z_j)$.

If $j = n$ and $\tau(z_n) = 1$ then $(z_n \vee t_n)$ was satisfied during the expansion and we are left with $(\neg t_n^\tau)$ which is satisfied. Similarly if $\tau(z_n) = 0$ both of $(z_n \vee t_n)$ and $(\neg z_n \vee \neg t_n)$ are satisfied, so all clauses in the part of the expansion relating to τ are satisfied and we are done.

If $j < n$ then for $i > j$ we will assign $t_i^\tau = \neg\tau(z_i)$ (and already we have $t_j^\tau = \neg\tau(z_j)$. If $\tau(z_i) = 0$ then the clauses which remain for i under τ both contain t_i^τ. If $\tau(z_i) = 1$ they both contain $\neg t_i^\tau$. These are now all satisfied for $i \geq j$, and whichever of (t_n^τ) and $(\neg t_n^\tau)$ remains must likewise be satisfied.

For every choice of $\tau \neq \alpha$ we have made assignments to the variables t_i^τ which ensure that all clauses belonging to the part of the expanded formula relating to τ are satisfied. Therefore the expanded formula which doesn't include any clauses relating to α is true, and to refute $\text{QP}\textsc{arity}_n\text{-}\textsc{ld}$ in $\forall$Exp + Res we must require some clause that is annotated with α. Since α was chosen arbitrarily it follows that it is only possible to refute $\text{QP}\textsc{arity}_n\text{-}\textsc{ld}$ if all annotations appear in the refutation. There are 2^n such assignments, so the refutation must contain at least 2^n clauses. The formula itself contains only $O(n)$ clauses, thus the size of the refutation is exponential in the size of the formula. $\square$

Theorem 4.3.3. *LD-Q-Res and $\forall$Exp + Res are incomparable on QBFs with bounded quantifier complexity.*

Proof. Lemmas 4.3.1 and 4.3.2 show that $\forall$Exp + Res cannot p-simulate LD-Q-Res on a family of QBFs with 3 quantifier blocks. The converse is shown in Beyersdorff *et al.* (2015). $\square$

The results of this chapter allow us to simplify part of the diagram of p-simulations and separations from Figure 3.6 in Chapter 3 in the case of QBFs with bounded quantifier complexity. Our new results are summarised in Figure 4.4. Solid lines indicate that the upper system p-simulates the lower system. Dotted lines indicate that the systems are incomparable. Double lines indicate that the systems are p-equivalent.

Conclusion

The results presented here demonstrate proof-theoretic advantages of $\forall$Exp + Res over Q-Res when quantifier complexity is bounded. We have shown that wherever a Q-Res refutation can be generated for a problem with fixed number of quantifier alternations there is also an $\forall$Exp + Res refutation with only a polynomial increase in size. The converse does not hold, as demonstrated by the QParity formulas. The cost of transforming a DAG-like Q-Res proof into an $\forall$Exp + Res proof by this construction depends on the number of quantifier alternations. However for tree-like proofs we can do the transformation while keeping the size the same, no matter what the number of quantifier alternations.

The construction can be extended to also show that QU-Res is simulated by both Q-Res and $\forall$Exp + Res for QBFs with a fixed number of quantifier alternations or if the proofs are required to be tree-like.

In Janota & Marques-Silva (2015), a family of formulas demonstrated that $\forall$Exp + Res cannot simulate Q-Res. These formulas had short proofs in Q-Res and required exponential size proofs in $\forall$Exp + Res. Our results explain some necessary conditions for such an example. Firstly, the number of quantifier levels must not be bounded above by a constant. Secondly, the short Q-Res proofs must be DAG-like. The clause re-use and the increasing number of quantifier levels also have to come together in a specific way.

The simulation becomes less efficient as the number of alternations increase, and we have noted the correlation between this result and empirical observations regarding the effect of quantifier alternations on solver performance. While this connection between the theoretical and empirical results is interesting we also recognise that the proof systems $\forall$Exp + Res and Q-Res greatly simplify the QBF solving algorithms. Because of unit-propagation in QCDCL solving, sometimes the proofs are better represented in the LD-Q-Res proof system, which is not simulated by $\forall$Exp + Res even for QBFs with bounded quantifier complexity. In addition, QBF solvers sometimes use dependency schemes (Lonsing & Biere, 2010), which are not taken into account here.

The specific advantage of $\forall$Exp + Res over Q-Res and QU-Res does not imply that proof systems using universal reduction are generally weak. For example the systems Frege $+\forall$red and Extended Frege $+\forall$red (Beyersdorff *et al.*, 2016a) are very strong systems, and finding lower bounds for them is equivalent to solving major open problems in circuit complexity or propositional proof complexity (Beyersdorff & Pich, 2016).

Chapter 5

Tree-Like Expansion Proofs

In the previous chapter we showed that $\forall$Exp + Res is strictly stronger than Q-Res when acting on QBFs with bounded quantifier complexity. This chapter considers a restriction on the proof system, instead of restricting the input QBFs. We show that Q-Res cannot p-simulate tree-like $\forall$Exp + Res, firstly showing formulas that separate tree-like $\forall$Exp + Res from tree-like Q-Res, then formulas to separate tree-like $\forall$Exp + Res from general (DAG-like) Q-Res, LD-Q-Res and even LQU$^+$-Res.

We use Prover-Delayer games to model the proof systems, with strategies in the games implying upper or lower bounds on proof size.

5.1 Proof Systems as Games

As well as characterising the QBF satisfiability problem as a game between two players, we can also describe some proof systems as games. This can be useful in allowing concise arguments for upper and lower bounds on proof size in these systems. We will use games that characterise the size of proofs in tree-like Resolution and tree-like Q-Res, the rules of which are given here with a brief explanation of how to determine proof size from strategies for the two players.

Characterising Tree-Like Resolution A game for finding lower bounds on the size of tree-like Resolution refutations was introduced in Pudlák (2000); Pudlák & Impagliazzo (2000), and later refined to also yield upper bounds in Beyersdorff *et al.* (2013) and to also characterise hardness for satisfiable formulas in Beyersdorff & Kullmann (2014). We use here the game of Beyersdorff *et al.* (2013), which is played on an unsatisfiable CNF Φ. There are two players: the Prover aims to make Φ evaluate to false, and since Φ is unsatisfiable they must be able to succeed eventually; the Delayer scores points while the game is ongoing, and aims to maximise their score.

The game proceeds in rounds through which an assignment to the variables of Φ is built up. During each round:

1. The Prover queries some variable x which is not currently assigned a value,
2. The Delayer responds with weights p_0 and p_1, both at least 0, and with $p_0+p_1=1$,
3. The Prover chooses a value $b \in \{0,1\}$ to assign to x and the Delayer scores $\log(\frac{1}{p_b})$ points.

The game ends when the current assignment falsifies any clause in Φ. If the Delayer chooses $p_b = 1$ they force x to be assigned the value b and score 0 points in that round.

In a tree-like Resolution proof image a walk along a path which starts at the root and proceeds towards the leaves. Each node in the tree is associated with a clause, which in turn corresponds to a partial assignment that falsifies that clause. A resolution step with pivot x corresponds to extending the assignment by either x or $\neg x$, depending on which parent is chosen as the next node in the walk.

This gives an intuitive idea of how the game and proof tree are related. Points scored by the Delayer under optimal strategies for the two players are related to the size of the proof tree not explored due to the assignment which was made in that round. When these ideas are formalised, we have the following theorems (Beyersdorff *et al.*, 2013):

Theorem 5.1.1. *If Φ has a tree-like Resolution proof of size at most S then there is a Prover strategy such that the Delayer scores at most $\log(\lceil \frac{S}{2} \rceil)$ points.*

Theorem 5.1.2. *If Φ has shortest tree-like Resolution proof of size S then there is a Delayer strategy such that the Delayer scores at least $\log(\lceil \frac{S}{2} \rceil)$ points.*

Since $\forall\mathsf{Exp}+\mathsf{Res}$ is a Resolution proof on annotated variables we can use the Prover-Delayer game to prove bounds on the size of tree-like proofs in this system.

Characterising Tree-Like Q-Resolution The above game was modified in Beyersdorff *et al.* (2017b) to describe tree-like Q-Res refutations. The Prover must be able to forget assignments – this can be allowed in the Resolution game but is not necessary – and the game must be extended to describe universal reduction steps.

Through the game the players are building an assignment to the variables of a PCNF Ψ. Each round consists of the following stages:

1. The Prover sets universal variables. Any universally quantified variable u can be assigned a value providing that all existentially quantified variables x with $\mathrm{lv}(x) > \mathrm{lv}(u)$ are currently unassigned,

2. The Delayer can declare an assignments for any existentially quantified variables that are currently unassigned,
3. The Prover queries an existentially quantified variable x which is not currently assigned,
4. The Delayer responds with weights p_0 and p_1, both at least 0 with $p_0 + p_1 = 1$,
5. The Prover selects a value $b \in \{0, 1\}$ to assign to x and the Delayer scores $\log(\frac{1}{p_b})$ points,
6. The Prover can forget any assignments.

Again, the game ends when some clause in Ψ is falsified by the current assignment.

Resolution steps in the proof correspond to the assignments made to existentially quantified variables, as before, and the points scored represent the size of the tree not being explored because of this choice. Universal reduction is represented by the Prover's choice of assignments to universally quantified variables. There is no branching on these values, and no points are scored by the Delayer. The following theorems are from (Beyersdorff *et al.*, 2017b).

Theorem 5.1.3. *If Ψ has a tree-like Q-Res refutation of size at most S then there is a Prover strategy such that the Delayer scores at most $\log(\lceil \frac{S}{2} \rceil$ points.*

Theorem 5.1.4. *If Ψ has shortest tree-like Q-Res refutation of size S then there is a Delayer strategy such that the Delayer scores at least $\log(\lceil \frac{S}{2} \rceil$ points.*

We now use these games to give upper and lower bounds in tree-like Q-Res and tree-like $\forall$Exp + Res to demonstrate separations between the two systems.

5.2 Separating Tree-Like Expansion From Tree-Like Q-Resolution

From Janota & Marques-Silva (2015) we know that tree-like $\forall$Exp + Res p-simulates tree-like Q-Res. We show that this simulation is strict by constructing a sequence of QBFs with polynomial-size tree-like proofs in $\forall$Exp + Res but requiring exponential-size proofs in tree-like Q-Res.

Let $[n] = \{1, \dots, n\}$ and let R be a ternary relation. Let $\Psi_n := \forall x \exists y \forall z R(x, y, z)$ where x, y, and z take values in $[n]$. Then we can define QBFs Θ_n expressing the contradiction that there is some interpretation of the relation R so that Ψ_n and $\neg\Psi_n$ are both true.

A general method for expressing first-order formulas as a sequence of QBFs is discussed in Chapter 9, where we also generalise the proof of Lemma 5.2.2. The formula

Θ_n has n Boolean variables for each variable in Φ_n corresponding to the n possible values for this variable.

$$\begin{aligned}
\Theta_n = \exists_{i,j,k\in[n]} R_{i,j,k} \forall x_1 \dots x_n \exists y_1 \dots y_n \exists u_1 \dots u_n \forall v_1 \dots v_n \exists w_1 \dots w_n \\
(y_1 \vee \dots \vee y_n) \wedge (u_1 \vee \dots \vee u_n) \wedge (w_1 \vee \dots \vee w_n) \qquad (5.1)\\
\wedge \bigwedge_{i,j,k\in[n]} (R_{i,j,k} \vee \neg x_i \vee \neg y_j \vee \bigvee_{i'\in[n],i'\neq i} x_{i'}) \qquad (5.2)\\
\wedge \bigwedge_{i,j,k\in[n]} (\neg R_{i,j,k} \vee \neg u_i \vee \neg v_j \vee \neg w_k \vee \bigvee_{j'\in[n],j'\neq j} v_{j'}) \qquad (5.3)
\end{aligned}$$

Lemma 5.2.1. *The formulas Θ_n require exponential-size proofs in tree-like Q-Res.*

Proof. We state a strategy for the Delayer and show that it guarantees a score of $\Omega(n)$ points before the formula is made false, for any Prover strategy. Let $x = a$ denote that $x_a = 1$ and $x_i = 0$ for all $i \neq a$ are in the current assignment, and let $x_a \leftarrow 1$ denote that $x_a = 1$ is added to the current assignment.

The Delayer will always declare assignments (scoring no points) according to the following rules, if it is possible to do so:

1. **If** $x = a$ and $y_b = 1$ **then** $R_{a,b,c} \leftarrow 1$ for all c,
2. **If** $x = a$ and $R_{a,b,c} = 0$ for some b, c **then** $y_b \leftarrow 0$,
3. **If** $x = a$ **then** $u_a \leftarrow 0$,
4. **If** $u_a = 1$, $v = b$ and $w_c = 1$ **then** $R_{a,b,c} \leftarrow 0$,
5. **If** $u_a = 1$, $v = b$ and $R_{a,b,c} = 1$ **then** $w_c \leftarrow 0$,
6. **If** $R_{a,b,c} = 1$ for any b, c **then** $u_a \leftarrow 0$.

Assignments are made according to these rules until no further rule can be applied. For any other existentially quantified variable that is queried the Delayer will respond with equal weights for both assignments, and will score one point whichever choice the Prover makes.

Together, these rules make it impossible to falsify the clauses in lines 5.2 and 5.3 of Θ_n. Recall that the Prover may not assign values to universally quantified variables without first forgetting any assignments to existentially quantified variables that appear later in the prefix.

Rule 3 ensures that there is no conflict between rules 1 and 4 and no other rules could be in conflict. Rules 1 and 2 ensure that clauses in 5.2 are never falsified. If rule 6 can be applied then it will ensure clauses in 5.3 are not falsified, and if it is not possible to apply this rule (because $u_a = 1$ already) then rules 4 and 5 will together ensure these clauses will not be falsified. Therefore Θ_n will become false by violating one of the clauses in line 5.1. Now we show that whichever of these clauses is made false the Delayer must have scored $\Omega(n)$ points in the game.

Suppose that the game ends with $w_i = 0$ for all $i \in [n]$. For each w_i, either the variable was set by a Prover choice, giving one point to the Delayer, or it was forced to take the value 0 by rule 5. In the latter case, to force the assignment of w_c we must already have $u_a = 1$, $v = b$ and $R_{a,b,c} = 1$ for some a and b. A different R variable is required for each value of c. It is not possible that $R_{a,b,c}$ was forced by rule 1, since rule 3 ensures that we cannot have $x = a$ and $u_a = 1$ simultaneously. Therefore, whenever rule 1 is invoked, we cannot have $u_a = 1$ and so rule 6 ensures that $u_a = 0$ is forced, violating our assumption that $u_a = 1$ and rule 5 can be applied. Consequently, $R_{a,b,c}$ must have been assigned by a Prover choice and this earlier choice has already given one point to the Delayer. The Delayer has scored at least one point for each w_i that is assigned 0, giving a total of at least n points if the game ends by violating the clause $(w_1 \vee \cdots \vee w_n)$.

Suppose instead that the game ends with $y_i = 0$ for all $i \in [n]$. Each y_i may have been set through a Prover choice, giving the Delayer one point. Alternatively, if some y_b was forced by rule 2 then $x = a$ and $R_{a,b,c} = 0$ for some c. We require a different R variable for each value of b, and it may have been assigned by a Prover choice (scoring one point) or by rule 4. In the latter case, since $w_c = 1$ is required and no rule may force this the Delayer must have scored one point for this assignment. The value of v cannot be changed without forgetting all w_i and therefore each application of rule 4 requires a new query on some w_i. The Delayer therefore scores at least one point for each y_i that is assigned 0, giving a total of at least n points if the game ends by violating the clause $(y_1 \vee \cdots \vee y_n)$.

If the game ends with $u_i = 0$ for all $i \in [n]$ then each variable may have been assigned by a Prover choice (scoring one point) or by either rule 3 or rule 6. Rule 3 can only be used to force one variable since a reassignment of x requires forgetting the current assignments to all u_i. If rule 6 was used then $R_{a,b,c}$ may have been assigned by a Prover choice (scoring one point) or by rule 1, in which case y_b must have been assigned by a Prover choice since no rule could force $y_b = 1$. There must be a new query made for each value of a because all y_i are forgotten when any x_i is changed. The Delayer scores one point for all but one u_i that is assigned 0, giving a total of at least $n - 1$ points if the game ends by violating the clause $(u_1 \vee \cdots \vee u_n)$.

In any case, the Delayer scores $\Omega(n)$ points by the end of the game, which implies that tree-like Q-Res refutations of Θ_n require size $2^{\Omega(n)}$. □

Lemma 5.2.2. *The formulas Θ_n have polynomial-size proofs in tree-like $\forall$Exp+Res.*

Proof. A proof in $\forall$Exp + Res is simply a Resolution proof, so we use the Prover-Delayer game for Resolution and show that there is a Prover strategy which limits the points scored by the Delayer by the end of the game.

We begin by showing that, for any annotation α, some u_i^α can be assigned 1 (or the game ends) for $O(log(n))$ points. The Prover queries each u_i^α for $i \in [n]$ in turn until one of them has been assigned 1. For $i < n$, given weights p_0 and p_1 for u_i^α the Prover will set $u_i^\alpha = 1$ if $p_1 \geq \frac{1}{n-i+1}$ and set $u_i^\alpha = 0$ otherwise.

Suppose that following this strategy results in the assignment $u_a^\alpha = 1$, with $a < n$. Now no u_i^α for $i > a$ will be queried, so no points are scored for these variables. All u_i^α for $i < a$ were assigned 0 so in each case $p_1 < \frac{1}{n-i+1}$. Since $p_0 + p_1 = 1$ therefore $p_0 > \frac{n-i}{n-i+1}$ and the Delayer scores at most $\log(n-i+1) - \log(n-i)$ for this assignment. For $i = a$ we have that $p_1 \geq \frac{1}{n-a+1}$ and the Delayer scores at most $\log(n-a+1)$. The total score is at most

$$\log(n-a+1) + \sum_{i=1}^{a-1} \log(n-i+1) - \log(n-i),$$

which simplifies to $log(n)$.

In the case that all u_i^α for $i < n$ are assigned 0 the Delayer has scored at most $\sum_{i=1}^{n-1} \log(n-i+1) - \log(n-i)$, giving a total of $\log(n)$. For the final query on u_{n-1}^α the Prover simply makes the assignment that gives Delayer the least score, which ensures this assignment gives at most 1 point. Either the game ends because all u_i^α have been assigned 0, or we have some a such that $u_a^\alpha = 1$, and in either case the Delayer has scored at most $\log(n) + 1$ points. The same argument obviously applies to the sets of variables y_i^α and w_i^α.

Now the Prover strategy is very simple and is shown in Algorithm 4. The notation $x = a$ in an annotation means that the annotation includes $0/x_i$ for all $i \neq a$ and $1/x_a$.

The Prover seeks values a, b, and c with $u_a^{x=1} = 1$, $y_b^{x=a} = 1$ and $w_c^{x=1,v=b} = 1$. The Delayer scores at most $3(\log(n) + 1)$ through these queries. The game may end with a failure to assign one of a, b or c a value, violating a clause from 5.1. Otherwise the Prover can query $R_{a,b,c}$, which gives a maximum of one more point to the Delayer. If $R_{a,b,c} = 0$ then the clause $(R_{a,b,c} \vee \neg y_b^{x=a})$ is made false, if $R_{a,b,c} = 1$ then the clause $(\neg R_{a,b,c} \vee \neg u_a^{x=1} \vee \neg w_c^{x=1,v=b})$ is made false. The tree-like Resolution proof on the expanded formula has size $O(n^3)$. $\square$

Consequently, tree-like $\forall$Exp + Res is strictly stronger than tree-like Q-Res.

Algorithm 4 Prover Strategy

```
function QUERYEXISTENTIAL($x$, $\alpha$)
    for $i \leftarrow 1$ to $n-1$ do
        Query Delayer on $x_i^\alpha$ for weights $p_0$, $p_1$
        if $p_1 \geq \frac{1}{n-i+1}$ then $x_i^\alpha \leftarrow 1$; return $i$
        else $x_i^\alpha \leftarrow 0$
    Query Delayer on $x_n^\alpha$ for weights $p_0$, $p_1$
    if $p_1 \geq \frac{1}{2}$ then $x_n^\alpha \leftarrow 1$; return $n$
    else $x_n^\alpha \leftarrow 0$

$a \leftarrow$ QUERYEXISTENTIAL($u$, $\{x = 1\}$)
$b \leftarrow$ QUERYEXISTENTIAL($y$, $\{x = a\}$)
$c \leftarrow$ QUERYEXISTENTIAL($w$, $\{x = 1, v = b\}$)
Query Delayer on $R_{a,b,c}$ and choose the value that gives the Delayer the least score.
```

Theorem 5.2.3. *Tree-like* $\forall$Exp + Res *p-simulates tree-like* Q-Res*, but tree-like* Q-Res *does not simulate* $\forall$Exp + Res*, and there are QBFs providing an exponential separation.*

Proof. It is known from (Janota & Marques-Silva, 2015) that tree-like $\forall$Exp + Res p-simulates tree-like Q-Res. The QBFs Θ_n and Lemmas 5.2.1 and 5.2.2 show that the converse cannot hold. □

5.3 Separating Tree-Like Expansion From Stronger Calculi

Even when Q-Res is not restricted to tree-like proofs it still cannot simulate tree-like $\forall$Exp + Res. To show this we recall the formulas QPARITY$_n$ from Section 4.3.1. The formulas are known to have short proofs in $\forall$Exp + Res, the refutations inductively derive clauses $(t_i^{1/z} \vee \neg t_i^{0/z})$ and $(\neg t_i^{1/z} \vee t_i^{0/z})$ for each i, which for $i = n$ generates a contradiction with the axioms $(t_n^{0/z})$ and $(\neg t_n^{1/z})$. This refutation requires the re-use of derived clauses and would have exponential size if it was expanded to have the structure of a tree. We now show that there are short proofs in tree-like $\forall$Exp + Res as well.

Theorem 5.3.1. QPARITY$_n$ *have polynomial-size tree-like* $\forall$Exp + Res *proofs.*

Proof. We expand out all the clauses of QPARITY$_n$ based on the two settings to the single universal variable z and use the Prover-Delayer game for Resolution. The Prover strategy is given in Algorithm 5.

For the four unit clauses, if the Delayer does not declare the values then the Prover

Algorithm 5 Prover Strategy

```
$i = 0$, $j = n$.
The Prover queries the variables of the unit clauses $\neg t_0^{0/z}, \neg t_0^{1/z}, t_n^{0/z}, \neg t_n^{1/z}$. The Delayer is forced to satisfy these clauses or get a constant score.
while $j - i > 1$ do
    $k \leftarrow \lfloor \frac{i+j}{2} \rfloor$.
    The Prover queries the variable $t_k^{1/z}$.
    The Prover chooses the value that gives the Delayer the least score.
    The Prover queries the variable $t_k^{0/z}$.
    The Prover chooses the value that gives the Delayer the least score.
    if $t_k^{0/z} = t_k^{1/z}$ then $i \leftarrow k$ else $j \leftarrow k$
The Prover now queries $x_j$, and chooses the value that gives the Delayer the least score.
  ▷ The game ends here because $t_i^{0/z} = t_i^{1/z}$, $t_j^{0/z} \neq t_j^{1/z}$, and for $c \in \{0, 1\}$, there are clauses expressing $t_j^{c/z} \equiv x_j \oplus t_i^{c/z}$.
```

can choose whichever value gives the least score. The Delayer scores a maximum of four points and if any of the unit clauses are falsified then the game ends.

The main idea is to notice that both of $t_0^{0/z}$ and $t_0^{1/z}$ must have the same assignment, but $t_n^{0/z}$ and $t_n^{1/z}$ must have different assignments. The Prover uses a binary search to find the value of i such that $t_i^{0/z}$ and $t_i^{1/z}$ are assigned the same value but $t_{i+1}^{0/z}$ and $t_{i+1}^{1/z}$ have opposite values. Since x_{i+1} is not annotated, the same value must be used in both $t_{i+1}^{0/z} = t_i^{0/z} \oplus x_{i+1}$ and $t_{i+1}^{1/z} = t_i^{1/z} \oplus x_{i+1}$. Both statements are required to hold, but the right hand sides are equal and the left hand sides are not. The binary search to find i takes $\lceil \log(n) \rceil$ rounds and the Delayer can score at most 2 points per round, hence a total score of at most $2\lceil \log(n) \rceil$. It follows that there are tree-like $\forall\mathsf{Exp}+\mathsf{Res}$ refutations with size $O(n^2)$. □

QPARITY$_n$ are known to be hard for Q-Res and QU-Res based on a strategy extraction argument. Balabanov & Jiang (2012) showed that both systems admit strategy extraction and the strategies which are extracted from the proofs can be efficiently expressed in a restricted form which they call a Right-First-And-Or formula. These formulas belong to AC^0, the class of Boolean functions expressible by polynomial-size constant depth circuits. It is not possible to compute the parity function using such a circuit, PARITY$\notin \mathsf{AC}^0$. The QBFs QPARITY$_n$ have a single universally quantified variable z whose unique winning strategy is the parity of the n existentially quantified variables which appear prior to z in the quantifier prefix. The QBFs have linear size in n. If Q-Res (or QU-Res) was able to refute this formula with a polynomial-size proof (in the size of the formula) then we would also be able to extract a strategy for z that had polynomial size in the size of the proof, and therefore the strategy is also of polynomial size in n. The strategy would give a circuit in AC^0 for computing Parity, but this is known to be impossible. Therefore, Q-Res and QU-Res cannot have polynomial-size refutations of the $\mathrm{QParity}_n$ formulas.

There are short proofs of $\mathrm{QParity}_n$ in extensions of Q-Res, but the formulas can be modified slightly to give QBFs that are also hard for these systems (Beyersdorff *et al.*, 2015). Specifically, the following QBFs are hard for LD-Q-Res and LQU^+-Res respectively. For LD-Q-Res the modification is to duplicate clauses and insert z in one copy and $\neg z$ in the other.

$$\begin{aligned}
&\mathrm{QParity}_n^{LDQ} := \exists x_1 \ldots \exists x_n \forall z \exists t_0 \ldots \exists t_n \\
&(t_n \vee z) \wedge (\neg t_n \vee \neg z) \wedge (\neg t_0 \vee z) \wedge (\neg t_0 \vee \neg z) \\
&\wedge \bigwedge_{i=1}^{n} (t_i = t_{i-1} \oplus x_i \vee z) \wedge \bigwedge_{i=1}^{n} (t_i = t_{i-1} \oplus x_i \vee \neg z)
\end{aligned}$$

where $(t_i = t_{i-1} \oplus x_i \vee l)$ is expressed by the four clauses

$$(\neg t_{i-1} \vee x_i \vee t_i \vee l),\ (t_{i-1} \vee \neg x_i \vee t_i \vee l),\ (t_{i-1} \vee x_i \vee \neg t_i \vee l),\ (\neg t_{i-1} \vee \neg x_i \vee \neg t_i \vee l).$$

The full universal expansion of this formula is identical to the full universal expansion of $\mathrm{QParity}_n$, so the Prover strategy in Algorithm 5 can be used for $\mathrm{QParity}_n^{LDQ}$ without modification and again proves the existence of short tree-like $\forall\mathsf{Exp}+\mathsf{Res}$ proofs for these formulas. Although there is a polynomial-sized refutation of $\mathrm{QParity}_n$ in LD-Q-Res, the modified formulas $\mathrm{QParity}_n^{LQD}$ require exponential-sized LD-Q-Res proofs. This shows that LD-Q-Res cannot simulate tree-like $\forall\mathsf{Exp}+\mathsf{Res}$.

For LQU^+-Res we further modify the formulas with a duplicate universal variable.

$$\begin{aligned}
&\mathrm{QParity}_n^{LQU} := \exists x_1 \ldots \exists x_n \forall z_1, z_2 \exists t_0 \ldots \exists t_n \\
&(t_n \vee z_1 \vee z_2) \wedge (\neg t_n \vee \neg z \vee \neg z_2) \wedge (\neg t_0 \vee z_1 \vee z_2) \wedge (\neg t_0 \vee \neg z_1 \vee \neg z_2) \\
&\wedge \bigwedge_{i=1}^{n} (t_i = t_{i-1} \oplus x_i \vee z_1 \vee z_2) \wedge \bigwedge_{i=1}^{n} (t_i = t_{i-1} \oplus x_i \vee \neg z_1 \vee \neg z_2)
\end{aligned}$$

where $(t_i = t_{i-1} \oplus x_i \vee l_1 \vee l_2)$ is expressed by the four clauses

$$(\neg t_{i-1} \vee x_i \vee t_i \vee l_1 \vee l_2),\ (t_{i-1} \vee \neg x_i \vee t_i \vee l_1 \vee l_2),\ (t_{i-1} \vee x_i \vee \neg t_i \vee l_1 \vee l_2),\ (\neg t_{i-1} \vee \neg x_i \vee \neg t_i \vee l_1 \vee l_2).$$

The full expansion of $\mathrm{QParity}_n^{LQU}$ is identical to that of $\mathrm{QParity}_n^{LDQ}$ except that any variable annotated by $0/z$ in $\mathrm{QParity}_n^{LDQ}$ is now annotatied with $0/z_1, 0/z_2$, and similarly any variable that was previously annotated with $1/z$ is now annotated with $1/z_1, 1/z_2$. Since this just amounts to a relabelling of variables it immediately follows that an equivalent $\forall\mathsf{Exp}+\mathsf{Res}$ refutation can be used. Consequently, we see that there are formulas which require exponential-sizes proofs in unrestricted LQU^+-Res but have polynomial-sized proofs in tree-like $\forall\mathsf{Exp}+\mathsf{Res}$.

Corollary 5.3.2. *LQU^+-Res cannot p-simulate tree-like $\forall\mathsf{Exp}+\mathsf{Res}$, and there are QBFs providing an exponential separation.*

5.4 Short Tree-Like Expansion Proofs for QBFs Based on Thin Circuits

The proof method shown above for $\mathrm{QP{\scriptsize ARITY}}_n$ formulas can be extended to other QBFs based on Boolean circuits with a constant bound on the number of incoming edges to each gate. We assume, without loss of generality, that each gate has at most two incoming edges.

Definition 5.4.1. *Let C be a Boolean circuit over variables $x_1, \dots, x_n$, with gates $g_1, \dots, g_m$, each of which computes a Boolean function. The output gate is g_m.*

The set of variables $X = \{x_1, \dots, x_n\}$ contains the input variables of C. To construct Q-C we define a second set of variables $T = \{t_1, \dots, t_m\}$. The variable t_j corresponds to gate g_j of C.

$$\text{Q-}C = \exists X \forall z \exists T \ (z \vee t_m) \wedge (\neg z \vee \neg t_m) \wedge \bigwedge_{j=1}^{S} [t_j \textit{ is consistent with gate } g_j]$$

The statement $[t_j$ is consistent with gate $g_j]$ depends on t_j and at most two other variables in $X \cup T$, which correspond to the inputs of gate g_j. It can therefore be written as a short CNF formula. For example, if $g_k = g_i \wedge x_j$ then we add clauses $(t_k \vee \neg t_i \vee \neg x_j), (\neg t_k \vee t_i), (\neg t_k \vee x_j)$. If $g_j = \neg g_i$ then we add clauses $(t_j \vee t_i)$ and $(\neg t_j \vee \neg t_i)$ (recall that the combination of $\wedge$ and $\neg$ is sufficient to construct circuits computing any Boolean function). The size of Q-C is $O(m)$, where m is the number of gates in the circuit C. In particular, if C is polynomial-sized in n, then so is Q-C.

The clauses $(z \vee t_m)$ and $(\neg z \vee \neg t_m)$ require that the output of the final gate g_m has the opposite value from z. Since the t_i variables appear after z in the quantifier prefix this means that the assignment to z forces a particular assignment for t_m, which is equivalent to fixing the output of the circuit, i.e. the value of $C(x_1, \dots, x_n)$. The QBF Q-C therefore expresses the contradiction $\exists x_1 \dots x_n \forall z \ [z \neq C(x_1, \dots, x_n)]$.

It is known that QBFs of this form have short refutations in $\forall\mathsf{Exp}+\mathsf{Res}$ (see Proposition 5.4.2). We will further show that there are short refutations for this family of formulas in tree-like $\forall\mathsf{Exp}+\mathsf{Res}$.

Proposition 5.4.2 (Proposition 28 in Beyersdorff *et al.* (2015))**.** *For every family of polynomial-size circuits $\{C_n\}$, the QBF family $\{Q\text{-}C_n\}$ has polynomial-size proofs in n in DAG-like $\forall\mathsf{Exp}+\mathsf{Res}$.*

Proof (sketch). Fix a circuit C_n and let m be the number of gates it contains. We prove by induction on $i \in [m]$ that $t_i^{0/z} = t_i^{1/z}$.

Since the input X variables are outermost in the prefix they are never annotated in the $\forall\mathsf{Exp}+\mathsf{Res}$ proof. Resolution on variables in X is sufficient to show the base case, that $t_1^{0/z} = t_1^{1/z}$.

For the inductive step, assume we have clauses stating that $t_j^{0/z} = t_j^{1/z}$ for $j < i$. The inputs to gate g_i are in $\{g_j \mid j < i\} \cup X$, and we have axiom clauses expressing the consistency requirements for $t_i^{0/z}$ and $t_i^{1/z}$, so it is possible to derive $t_i^{0/z} = t_i^{1/z}$ in a short proof fragment.

From $t_m^{0/z} = t_m^{1/z}$, resolution with the unit clauses $t_m^{0/z}$ and $\neg t_m^{1/z}$ derives the empty clause. $\square$

This proof reuses the derivations of $t_j^{0/z} = t_j^{1/z}$ for all steps relating to any gate which has g_j as an input. Such reuse of derived clauses is not permitted in tree-like $\forall\mathsf{Exp}+\mathsf{Res}$. Instead, we generalise the binary search Prover strategy idea from Theorem 5.3.1. The technique works because the circuits underlying the QBFs have small "width". As long as C has polynomial length $p(n)$ and constant "width" bounded by b, the Prover can use binary search to devise a strategy where the Delayer scores at most $c\log(p(n))$ for some constant c and hence the proof size is at most $p(n)^c$ which is polynomial in n.

Further, if we relax our desire for polynomial-size proofs to just quasi-polynomial size we can allow circuits with non-constant width. The restriction of poly-logarithmic width results in quasi-polynomial size proofs.

The following definition allows us to make this intuition more formal.

Definition 5.4.3 (Layered Circuits, and Circuit Width)**.** *A circuit is* layered *if its gates can be partitioned into disjoint sets S_i for $1 \leq i \leq \ell$, such that for each i, and for each gate in layer S_i, all its inputs are either input variables or the outputs of gates in S_{i-1}. The output gate is in the final layer S_ℓ.*

The width *of a layered circuit is the maximum number of gates in any layer; width$(C) = \max\{|S_i| \mid i \in \ell]\}$.*

Theorem 5.4.4. *Let C be a layered circuit of size m and width w, and let $Q\text{-}C$ be the corresponding QBF. Then $Q\text{-}C$ has a proof, in tree-like $\forall\mathsf{Exp}+\mathsf{Res}$, of size $m^{O(w)}$.*

Proof. Consider the expanded CNF obtained from Q-C. Let U be the set of $t^{0/z}$ variables and V be the set of $t^{1/z}$ variables, while the set of variables X remains without any annotation. Let the clauses expressing that each variable t_i is consistent

Algorithm 6 Prover Strategy in Theorem 5.4.4

```
$i = 0$, $j = \ell$.
The Prover queries the variables $t_m^{0/z}$ and $t_m^{1/z}$ of $W_l$. The Delayer is forced to satisfy
the unit clauses or get a constant score.
The Prover queries the variables of $W_0$, and chooses the values that give the Delayer
the least score.
while $j - i > 1$ do
    $k \leftarrow \lfloor (i+j)/2 \rfloor$.
    while some $t$ variables in $W_k$ are unassigned do
        The Prover queries an unassigned $t_a^{0/z}$ in $W_k$.
        The Prover chooses the value that gives the Delayer the least score.
        The Prover queries the variable $t_a^{1/z}$.
        The Prover chooses the value that gives the Delayer the least score.
        Prover queries all remaining $X$ variables in $W_k$.
        Prover chooses the values that give the Delayer the least score.
    if $t_b^{0/z} \neq t_b^{1/z}$ for some pair of variables in $W_k$ then $j \leftarrow k$ else $i \leftarrow k$.
        ▷ Once $j - i = 1$, the game ends in a contradiction witnessed by the variables
corresponding to some gate at layer $j$.
```

with the computation of gate g_i be denoted $F(X, T)$. Following expansion we have two copies of these clauses – one in which all variables in T are annotated with $0/z$ and one in which the variables of T are annotated with $1/z$. This CNF is denoted $G(X, U, V)$, with

$$G(X, U, V) = F(X, U) \wedge F(X, V) \wedge t_m^{0/z} \wedge \neg t_m^{1/z}$$

A proof in (tree-like) $\forall\mathsf{Exp+Res}$ that Q-C is false is a proof in (tree-like) Resolution that the CNF $G(X, U, V)$ is unsatisfiable.

Let the number of layers in C be ℓ where $\ell \leq m$.

Let W_ℓ be the set of (annotated) variables corresponding to the output gate, i.e. $W_\ell = \{t_m^{0/z}, t_m^{1/z}\}$. For $i \in [\ell]$, let $W_{i-1} \subseteq X \cup U \cup V$ be the variables feeding into gates at layer i of C. Note that $W_0 \subseteq X$.

The Prover uses binary search to identify a layer j such that there is a pair of variables $t_b^{0/z}$ and $t_b^{1/z}$ in W_j which are assigned different values, but with all pairs of variables $t_a^{0/z}$ and $t_a^{1/z}$ in W_{j-1} having the same values assigned. The strategy is given in Algorithm 6.

There are variables $t_b^{0/z} \neq t_b^{1/z}$ in W_j, but for all t_a variables in W_{j-1}, $t_a^{0/z} = t_a^{1/z}$. Furthermore all X variables in W_{j-1} are assigned some value. So $t_b^{0/z} \neq t_b^{1/z}$ must cause a contradiction with the copies of the clauses expressing consistency of gate g_b with its inputs.

For every layer i where the Prover queries variables from W_i, there are at most $4w$ variables to query, and this is the maximum score for the Delayer on W_i. The Prover looks at no more than $\lceil \log \ell \rceil$ sets W_i. Since $\ell \leq m$ it follows that there are tree-like $\forall\mathsf{Exp}+\mathsf{Res}$ proofs of size $m^{O(w)}$. $\square$

Corollary 5.4.5. *Suppose $\{C_n\}$ is a family of layered circuits with width bounded by a constant. Then the family of QBFs Q-C_n has polynomial-size proofs in tree-like $\forall\mathsf{Exp}+\mathsf{Res}$.*

Conclusion

For tree-like systems it was already known that $\forall\mathsf{Exp}+\mathsf{Res}$ p-simulates Q-Res. In this chapter we have shown that the converse is not true and even DAG-like Q-Res cannot p-simulate tree-like $\forall\mathsf{Exp}+\mathsf{Res}$. The separation given by the QParity$_n$ formulas is in some sense optimal since two quantifier alternations is the least possible number of alternations to have separating formulas between these systems.

We have shown that the Q-C formulas, which can give lower bounds in QCDCL style systems (Beyersdorff *et al.*, 2016a), are easy for tree-like $\forall\mathsf{Exp}+\mathsf{Res}$ when the circuits have a specific structure. The false Q-C formulas are also Σ_3^b so we have found a class of separating formulas which all have the minimum quantifier complexity needed to separate Q-Res and $\forall\mathsf{Exp}+\mathsf{Res}$.

Chapter 6

Strategy Extraction in QRAT

As well as deciding whether a given QBF is true or false, QBF algorithms usually also provide a proof. The proof is checked against the rules of a sound proof system and acts as a verification that the result given by the solver is correct. The QRAT proof system (Heule *et al.*, 2014b) is sufficiently strong to simulate the reasoning steps of all current QBF solvers and pre-processors and so offers a standard format for solvers to output proofs, both for true and false QBFs.

In many settings it is also desirable to find functions that witness whether a QBF is true or false. If a QBF is true then there must exist functions for the existentially quantified variables that certify this. The functions represent a winning strategy for the existential player on this QBF and are called Skolem functions. When Skolem functions witnessing the truth of a QBF are substituted into the matrix of the formula this yields a tautologous propositional formula.

Similarly, if a QBF is false then there must exist certifying functions for the universally quantified variables, representing a winning strategy for the universal player, which are called Herbrand functions. When Herbrand functions witnessing the falsity of a QBF are substituted into the matrix of the formula the result is an unsatisfiable propositional formula.

QBFs are useful to model a wide variety of problems and the Herbrand or Skolem functions may correspond to, for example, a piece of code in a program synthesis problem, a feasible plan in robotic planning, or an example showing that a system or plan is unsafe. The ability to efficiently extract Skolem or Herbrand functions from a proof is called strategy extraction.

Many QBF proof systems which use the universal reduction rule are known to have polynomial time strategy extraction (Balabanov & Jiang, 2011; Egly *et al.*, 2013). For QBF systems with universal expansion some strategy extraction results are known using a different technique (Beyersdorff *et al.*, 2015; Goultiaeva *et al.*, 2011).

QRAT is able to simulate both the universal reduction and expansion rules and uses an exponentially stronger form of propositional reasoning than resolution. With this power it has been shown to simulate a number of different QBF proof systems, including LD-Q-Res and $\forall$Exp + Res (Kiesl & Seidl, 2019; Kiesl *et al.*, 2017).

Strategy extraction on a universal checking format like QRAT would allow a work-flow in which a proof is extracted from a solver in the standard format, verified and then the Skolem/Herbrand functions that give the winning strategy are derived from that proof. This avoids having to extract strategies directly from solvers while they are running, which may affect performance.

Conversely, the property of strategy extraction can actually provide a source of weakness in QBF proof systems. As we have seen, for example, Q-Res can always extract strategies as bounded depth circuits so QBFs with winning strategies that cannot be expressed in small bounded depth circuits necessarily have large Q-Res proofs. This is a similar argument to the proof size lower bound technique based on feasible interpolation (Krajíček, 1997; Pudlák, 1997) in which if a propositional proof system can extract Craig interpolants (Craig, 1957a) in polynomial time then super-polynomial interpolant size lower bounds become super-polynomial proof size lower bounds.

Heule *et al.* (2014a) showed that Skolem functions to certify that a QBF is true can be extracted in polynomial time from a QRAT proof. In this chapter we show that Herbrand functions may be extracted from proofs in a restricted version of refutational QRAT, but that it is not possible in general to efficiently extract Herbrand functions certifying falsity from proofs in unrestricted QRAT (unless P = PSPACE). This is shown by demonstrating that there are short QRAT proofs of formulas that have PSPACE-hard strategies. Thus we show an asymmetry between the refutation of false QBF and proof of true QBF in the QRAT system. We demonstrate that this is due to the presence of universal expansion steps which manifest from the powerful reduction rules in the full QRAT proof system. The proof system $\forall$Exp + Res also uses universal expansion but allows polynomial time strategy extraction (Beyersdorff *et al.*, 2015). We end the chapter by strengthening a connection previously explored in Beyersdorff *et al.* (2017a) between strategy extraction and feasible interpolation.

We begin by introducing redundancy properties which are then used to define the QRAT proof system.

6.1 Redundancy Properties

Definition 6.1.1 (Redundant Clause). *Let Φ be a propositional formula in CNF and let C be a clause. Suppose Φ is satisfiable if and only if $\Phi \cup \{C\}$ is satisfiable. Then we say C is redundant in Φ. Similarly, for a closed QBF $\Psi = \Pi\Phi$ in PCNF, clause C is redundant in $\Pi\Phi$ when $\Pi\Phi$ and $\Pi'\Phi \cup \{C\}$ have the same truth value, where Π and Π'*

are identical except that Π' *may contain additional variables from* C *that do not appear in* Φ.

A redundancy property defines a set of pairs (Φ, C) where C is redundant in Φ. The simplest redundancy property is Tautology (T). If a clause C is a tautology then for some variable x, C contains both x and $\neg x$. C is true under any assignment to the variables so its inclusion or exclusion cannot affect the overall truth value of a formula in CNF.

In general, deciding whether a clause in a propositional formula is redundant is coNP-hard, but there are several useful redundancy properties that can be checked in polynomial time. These form the basis of many QBF solving and pre-processing techniques. We define three further redundancy properties for propositional formulas: Asymmetric Tauology (AT), Resolution Tautology (RT) and Resolution Asymmetric Tautology (RAT). RAT will then be extended in the next section for QBFs and form the basis of the QRAT proof system.

Definition 6.1.2. *If* C *is a clause, then* $\bar{C}$ *is the conjunction of unit clauses containing the negations of the literals in* C. *That is, if* $C = (l_1 \vee \dots \vee l_n)$ *then* $\bar{C} = (\bar{l}_1) \wedge \dots \wedge (\bar{l}_n)$.

Recall that unit propagation is defined by the procedure shown in Algorithm 7. We denote that the clause D can be derived by unit propagation applied to Φ by $\Phi \vdash_1 D$. Unit propagation is a polynomial-time procedure.

Algorithm 7 Unit Propagation

```
function UnitPropagate($\Phi$)
    while $\Phi$ contains unit clause $\{l\}$ do
        $\Phi \leftarrow \{C \setminus \{\bar{l}\} \mid C \in \Phi, l \notin C\}$
    return $\Phi$
```

Definition 6.1.3 (Asymmetric Tautology)**.** *Clause* C *is an asymmetric tautology with respect to* CNF Φ *if* $\Phi \wedge \bar{C} \vdash_1 \bot$.

Observation 6.1.4. *If clause* C *is an asymmetric tautology with respect to* CNF Φ *then* $\Phi \cup \{C\}$ *and* Φ *have the same models.*

Proof. Let α be an assignment such that $\alpha \models \Phi$. Let n be the number of literals in C. Since C is an asymmetric tautology with respect to Φ there is a sequence of assignments $\tau_0, \dots, \tau_m$ generated through unit propagation. $\tau_0 = \{\}$, and $\tau_i = \tau_{i-1} \cup \{l_i\}$, where for $i = 1, \dots, n$ we have $(l_i) \in \bar{C}$, and for $i = n+1, \dots m$ there is some clause $(D \vee l_i) \in \Phi$ with D falsified by τ_{i-1}.

Since unit propagation must derive the empty clause we know that Φ is not satisfied by τ_m, so τ_m disagrees with α on the assignment of some variable i.e. there exists some literal $l \in \tau_m$ such that $\bar{l} \in \alpha$. Let j be the minimum index such that τ_j disagrees with α.

For every literal $l \in \tau_j$ either $(l) \in \bar{C}$ or $(D \vee l) \in \Phi$ with D falsified by τ_{j-1}. In the latter case, since we assume that τ_{j-1} agrees with α, and α cannot falsify a clause in Φ, it follows that $l \in \alpha$. Therefore there exists some $(l) \in \bar{C}$ (i.e. $\bar{l} \in C$) with $\bar{l} \in \alpha$, and so $\alpha \models C$. □

The redundancy properties T and AT can be generalised by considering the resolvents of a clause: if all of the resolvents of clause C on literal l are (asymmetric) tautologies with respect to a CNF Φ then C is also redundant with respect to Φ.

Definition 6.1.5 (Resolution Tautology)**.** *Clause C is a resolution tautology with respect to* CNF Φ *if either C is a tautology or C contains a literal l such that for every clause $D \in \Phi$ with $\bar{l} \in D$ the resolvent of C and D on l (i.e. $C \setminus \{l\} \cup D \setminus \{\bar{l}\}$) is a tautology.*

The properties AT and RT recognise different redundancies.

Example 6.1.6.

$$\begin{aligned} C_0 &= (x_1 \vee x_2 \vee x_5) & C_4 &= (x_1 \vee \neg x_3) \\ C_1 &= (\neg x_1 \vee x_2) & C_5 &= (x_1 \vee x_4) \\ C_2 &= (\neg x_2 \vee \neg x_3) & C_6 &= (\neg x_4 \vee \neg x_5) \\ C_3 &= (x_2 \vee x_3) \end{aligned}$$

C_4 is an asymmetric tautology: $\tau = \{\neg x_1, x_3\}$, clause C_5 adds x_4, then C_6 adds $\neg x_5$, and C_0 adds x_2. $\tau = \{\neg x_1, x_3, x_4, \neg x_5, x_2\}$ falsifies clause C_2.

C_4 is not a resolution tautology because the only resolvent on x_1 is $(x_2 \vee \neg x_3)$ and the only resolvent on x_3 is $(x_1 \vee x_2)$, neither of which is a tautology.

Definition 6.1.7 (Resolution Asymmetric Tautology (RAT))**.** *Clause C is a resolution asymmetric tautology with respect to* CNF Φ *if either C is an asymmetric tautology or C contains a literal l such that for every clause $D \in \Phi$ with $\bar{l} \in D$ the resolvent of C and D on l is an asymmetric tautology. We say C has RAT on l with respect to Φ.*

RAT generalises the both the resolution tautology and asymmetric tautology properties. Any clause which is a resolution tautology or asymmetric tautology with respect to a formula Φ also has RAT with respect to Φ, but the converse does not hold.

Example 6.1.8.

$$\begin{aligned} C_0 &= (x_1 \vee x_2) & C_2 &= (x_2 \vee \neg x_3) \\ C_1 &= (\neg x_1 \vee x_3) & C_3 &= (x_1 \vee x_3) \end{aligned}$$

C_3 has RAT on x_3, but is not as resolution tautology or asymmetric tautology. Resolving with C_2 gives $(x_1 \vee x_2)$. This is not a tautology, but it is an asymmetric tautology (since it is already in the formula).

The RAT property does not necessarily preserve models but does preserve satisfiability.

Observation 6.1.9. *If clause C has RAT with respect to* CNF *Φ then $\Phi \cup \{C\}$ and Φ are satisfiability equivalent.*

We can prove a slightly more general statement, showing that any model preserving redundancy property can be lifted by considering also the resolution environment of the redundant clause.

Observation 6.1.10. *Let $\mathcal{P}$ be a redundancy property that preserves models, Φ a* CNF *and C a clause. If C contains a literal l such that $C \setminus \{l\} \cup D \setminus \{\bar{l}\}$ has property $\mathcal{P}$ with respect to Φ for every $D \in \Phi$ with $\bar{l} \in D$, then $\Phi \cup \{C\}$ is satisfiable if any only if Φ is satisfiable.*

Proof. If Φ has no models then also $\Phi \cup \{C\}$ has no models. Let $\alpha \models \Phi$. If α is a model of C then we are done. Otherwise, since all $C \setminus \{l\} \cup D \setminus \{\bar{l}\}$ have property $\mathcal{P}$, also $\alpha \models C \setminus \{l\} \cup D \setminus \{\bar{l}\}$ for all $D \in \Phi$ with $\bar{l} \in D$. Since α does not satisfy C, α must satisfy all $D \setminus \{\bar{l}\}$. Let α' be identical to α except that l evaluates to true. Now Φ is still satisfied because all the clauses that contain $\bar{l}$ are satisfied by other variables, and C is satisfied by l. □

We can think of a clausal refutation of a CNF as a sequence of steps that progressively add redundant clauses to the formula until the empty clause can be added. For a QBF the notion of redundancy also considers the quantifier prefix, but we can similarly characterise a clausal refutation as adding redundant clauses until the empty clause is added, and a clausal proof of truth as removing redundant clauses until we have the empty formula. The RAT property is used as the basis of the RAT proof system with practical applications in proof checking for SAT solvers. Extending the RAT property to QBF provides the basis for the QRAT proof system.

It was shown in Heule *et al.* (2014a) that Skolem functions witnessing the truth of true QBFs can be extracted in polynomial time from a QRAT proof. Here we investigate the extraction of Herbrand functions from QRAT refutations.

6.2 The QRAT Proof System

The QRAT system uses some inference rules that are lifted directly from propositional reasoning, and others that depend on the QRAT property specific to QBFs. For these we must take the order of variables in the quantifier prefix into account.

Definition 6.2.1 (Outer Clause, Outer Resolvent)**.** *Let $\Pi\Phi$ be a closed PCNF and let C be a clause not in Φ. Let Π' be a prefix including the variables of C and Φ such that Π is a sub-prefix of Π' that contains the variables of Φ only. Suppose C contains a literal l. Consider all clauses D in Φ with $\bar{l} \in D$. The outer clause O_D of D is*

$$O_D = \{k \in D \mid \mathrm{lv}(k) \leq_\Pi \mathrm{lv}(l),\ k \neq \bar{l}\}.$$

The outer resolvent $\mathcal{R}(C, D, \Pi, l)$ is defined as $C \cup O_D$ when l is existentially quantified, and $C \setminus \{l\} \cup O_D$ when l is universally quantified.

As stated, the definition of O_D assumes that l appears in Π. It is possible for C to contain variables that are not included in $\Pi\Phi$, however in this case there are no $D \in \Phi$ with $\bar{l} \in D$ and so no outer clauses O_D need to be defined.

Definition 6.2.2 (Quantified Implied Outer Resolvent (QIOR))**.** *Clause C has property QIOR on literal l with respect to* PCNF $\Pi\Phi$ *if $l \in C$ and $\Phi \models \mathcal{R}(C, D, \Pi, l)$ for all $D \in \Phi$ with $\bar{l} \in D$.*

The following two theorems are proved in Heule *et al.* (2017).

Theorem 6.2.3. *Let $\Pi\Phi$ be a closed PCNF and let C be a clause not in Φ. Let Π' be a prefix including the variables of C and Φ such that Π is a sub-prefix of Π' that contains the variables of Φ only. If C has QIOR on an existentially quantified literal $l \in C$ with respect to $\Pi\Phi$ then $\Pi\Phi$ and $\Pi'\Phi \wedge C$ have the same truth value.*

Theorem 6.2.4. *Let $\Pi\Phi$ be a closed PCNF and let C be a clause not in Φ. If $(C \vee l)$ has QIOR on universally quantified literal l with respect to $\Pi\Phi$ then $\Pi\Phi \wedge (C \vee l)$ and $\Pi\Phi \wedge C$ have the same truth value.*

Because it is hard to check whether $\Phi \models \mathcal{R}(C, D, \Pi, l)$ we instead use unit propagation to define the QRAT property.

Definition 6.2.5 (Quantified Resolution Asymmetric Tautology(QRAT))**.** *Clause C has QRAT on literal l with respect to* PCNF $\Pi\Phi$ *if C contains the literal l and $\Phi \vdash_1 \mathcal{R}(C, D, \Pi, l)$ for every clause $D \in \Phi$ with $\bar{l} \in D$.*

This is equivalent to requiring that the outer resolvent of C and D on l is an asymmetric tautology for all $D \in \Phi$ with $\bar{l} \in D$.

A QRAT proof begins with a PCNF $\Pi\Phi$ that is then edited throughout the proof by satisfiability preserving rules. New conjuncts can be added and clauses can be altered or deleted. Some rules are based on the whole current status of the QBF and not only on a feature of the clause being modified. A refutation ends when the empty clause has been added to the formula, for a true QBF a proof ends when the matrix is empty. We now state the rules from Heule *et al.* (2014b) that define the QRAT proof system.

Definition 6.2.6 (Asymmetric Tautology Addition (ATA))**.** *Let $\Pi\Phi$ be a closed PCNF and let C be a clause not in Φ. Let Π' be a prefix including the variables of C and Φ such that Π a sub-prefix of Π' that contains the variables of Φ only. We can make the following inference when C is an asymmetric tautology with respect to Φ.*

$$\frac{\Pi\Phi}{\Pi'\Phi \wedge C}\ \textit{(ATA)}$$

The QRAT proof system is defined so that $\Pi'\Phi \wedge C$ replaces $\Pi\Phi$.

Definition 6.2.7 (Asymmetric Tautology Elimination (ATE))**.** *Let $\Pi'\Phi$ be a closed PCNF and let C be a clause not in Φ. Let Π be a prefix including the variables of C and Φ such that Π' is a sub-prefix of Π that contains the variables of Φ only. We can make the following inference when C is an asymmetric tautology with respect to Φ.*

$$\frac{\Pi\Phi \wedge C}{\Pi'\Phi}\ \textit{(ATE)}$$

ATA and ATE are sound by Observation 6.1.4.

Definition 6.2.8 (Quantified Resolution Asymmetric Tautology Addition (QRATA))**.** *Let $\Pi\Phi$ be a closed PCNF and let C be a clause not in Φ. Let Π' be a prefix including the variables of C and Φ such that Π a sub-prefix of Π' that contains the variables of Φ only. We can make the following inference when C has QRAT with respect to $\Pi\Phi$ on an existentially quantified literal l.*

$$\frac{\Pi\Phi}{\Pi'\Phi \wedge C}\ \textit{(QRATA on } l\textit{)}$$

Correctness follows from Theorem 6.2.3. QRATA allows new variables to be introduced that were not in the initial prefix, and so allows us to add extension variables (as defined in Section 3.1). Although ATA also allows new variables to be introduced it is not sufficiently powerful to introduce extension variables in general.

Example 6.2.9. *Suppose we want to add extension variable t that is equivalent to $(\neg x \vee \neg y)$. First add clause $(\neg x \vee \neg y \vee \neg t)$ quantifying t with higher level than x and y. This can be added with QRATA on $\neg t$. Because t does not occur in any other clause the condition is vacuously satisfied.*

The two other clauses $(x \vee t)$ and $(y \vee t)$ can also be added via QRATA on t. The only clause D that has to be checked is $(\neg x \vee \neg y \vee \neg t)$. The outer clause $O_D = (\neg x \vee \neg y)$ so we need to show unit propagation on $\Phi \wedge \bar{C} \wedge x \wedge y$ derives the empty clause for $C = (x \vee t)$ and $C = (y \vee t)$. Since $\bar{C}$ therefore includes $(\neg x)$ or $(\neg y)$ (respectively) this is immediate.

Definition 6.2.10 (Quantified Resolution Asymmetric Tautology Elimination (QRATE))**.** *Let $\Pi'\Phi$ be a closed PCNF and let C be a clause not in Φ. Let Π be a prefix including the variables of C and Φ such that Π' a sub-prefix of Π that contains the variables of Φ only. We can make the following inference when C has QRAT with respect to $\Pi\Phi$ on an existentially quantified literal l.*

$$\frac{\Pi\Phi \wedge C}{\Pi'\Phi}\ (\textit{QRATE on } l)$$

Again, correctness follows from Theorem 6.2.3.

Definition 6.2.11 (Quantified Resolution Asymmetric Tautology Universal (QRATU))**.** *Let $\Pi\Phi\wedge(C\vee l)$ be a closed PCNF. Let Π' be a sub-prefix of Π that contains the variables of Φ and C only. If l is universally quantified in Π and C has QRAT with respect to $\Pi\Phi$ on l then we can make the following inference.*

$$\frac{\Pi\Phi \wedge (C \vee l)}{\Pi'\Phi \wedge C}\ (\textit{QRATU on } l)$$

Correctness of QRATU follows from Theorem 6.2.4.

Definition 6.2.12 (Extended Universal Reduction (EUR))**.** *Let $\Pi\Phi \wedge (C \vee l)$ be a closed PCNF. Let Π' be a sub-prefix of Π that contains the variables of Φ and C only. Let l be a universally quantified literal. Consider extending C by*

$$C \leftarrow C \cup \{k \in D \mid \mathrm{lv}(k) >_\Pi \mathrm{lv}(l) \textit{ or } k = \bar{l}\}$$

where $D \in \Phi$ is any clause with some $p \in C$ and $\bar{p} \in D$ such that $\mathrm{lv}(p) >_\Pi \mathrm{lv}(l)$. Continue extending C in this way until we reach a fixed point, denoted ε.

We can make the following inference when $\bar{l} \notin \varepsilon$.

$$\frac{\Pi\Phi \wedge (C \vee l)}{\Pi'\Phi \wedge C}\ (\textit{EUR})$$

EUR encompasses the universal reduction rule ($\forall$-Red) used in Q-Res but EUR is strictly stronger than $\forall$-Red because it uses a dependency scheme from Slivovsky & Szeider (2014). The procedure $C \leftarrow C \cup \{k \in D \mid \mathrm{lv}(k) >_\Pi \mathrm{lv}(l) \text{ or } k = \bar{l}\}$ finds every clause that can be reached from C via resolution steps with pivot variable later in the quantified prefix than l. Suppose that $\Pi'\Phi \wedge C$ is false. Then there is a winning universal strategy witnessing the fact, and we can play according to this strategy until $\mathrm{var}(l)$. Now the remaining formula must be false, so there is a Q-Res refutation of it. Suppose that the clause C, restricted according to the assignments made so far, is used in this Q-Res refutation. Then all the other clauses used in the refutation can be reached via resolution steps with pivot variable later in the quantified prefix than l. If none of these clauses contain $\bar{l}$ then clearly the universal player can assign l to be false and the restricted QBF remains false. As a result, l could be added into the clause C without affecting the truth of the restricted formula. If, on the other hand, C is not used in the Q-Res refutation of the restricted QBF, then C can be modified without affecting the refutation at all, so the QBF remains false. Since the same argument holds for any restricted QBF reached by playing according to any winning universal strategy, we can conclude that when $\Pi'\Phi \wedge C$ is false and $\bar{l} \notin \varepsilon$ then also $\Pi\Phi \wedge (C \vee l)$ is false.

While refutational QRAT works perfectly well with the standard universal reduction rule, Extended Universal Reduction is used because it allows QRAT to simulate expansion steps used in QBF pre-processing. Note that this is the *only* pre-processing rule that required the Extended Universal Reduction rule. In principle, QRAT could be augmented with any sound dependency scheme used as the basis of its reduction rule.

Definition 6.2.13 (Clause Deletion)**.** *Let $\Pi'\Phi$ be a closed PCNF and let C be a clause not in Φ. Let Π be a prefix containing all the variables in both Φ and C such that Π' is a sub-prefix of Π which contains the variables of Φ only. In refutational QRAT clauses may be arbitrarily deleted without checking if they conform to a rule. This is not permitted in satisfaction QRAT.*

$$\frac{\Pi\Phi \wedge C}{\Pi'\Phi}$$

Because some of the rules consider every clause contained in Φ, clause deletion may not be superficial. A clause may need to be deleted in order for EUR to be performed, for example. Similarly for QRATA and QRATU on l, the property considers all other clauses that contain the complimentary literal $\bar{l}$.

The rules of QRAT together are refutationally complete and simulate Extended Q-Res (Q-Res with the addition of extension variables). Because refutational QRAT permits arbitrary clause deletion the other rules for clause deletion, QRATE and ATE, are not necessary. Without Clause Deletion, QRAT is a sound and complete system for proving true QBF. When checking a QRAT proof of satisfiability it is only necessary to check the lines which remove clauses by QRATE and ATE.

A Weaker Version of QRAT

Let QRAT(X) be QRAT with the EUR rule replaced by reduction rule X. This means that the standard QRAT is given by QRAT(EUR). An alternative would be to use the universal reduction rule from Q-Res, which allows

$$\frac{\Pi\Phi \wedge (C \vee l)}{\Pi\Phi \wedge C}\ (\forall\text{-Red})$$

for universally quantified l whenever $\text{lv}(l) > \text{lv}(x)$ for all existentially quantified x in C. We call this simplest version QRAT(UR).

EUR is not essential to the underlying techniques of QRAT and QRAT(UR) can still simulate the main pre-processing techniques except for universal expansion. In addition EUR is not required to simulate Extended Q-Res as this can be done with the simpler reduction rule. Extension clauses are added using QRATA on the extension variable, resolution uses ATA, and the universal reduction rule is the same in QRAT(UR) and Extended Q-Res.

We now come to the first main result of this chapter: refutational QRAT(UR) has strategy extraction.

6.3 Strategy Extraction in QRAT(UR)

In order to show that QRAT(UR) has polynomial-time strategy extraction on false QBFs we will inductively compute a winning universal strategy for formulas at each step of the QRAT(UR) proof, starting from the final proof step.

For the inductive step we need to construct a winning strategy σ for the formula prior to some proof step from σ', a known winning strategy for the formula after that proof step. We prove that this is possible for each derivation rule in refutational QRAT(UR) (i.e. ATA, QRATA, QRATU, $\forall$-Red, Clause Deletion). The strategy σ is composed of Herbrand functions σ_u for each universal variable u. A Herbrand function σ_u for QBF $\Pi\Phi$ takes as input an assignment to existentially quantified variables x with $\mathrm{lv}(x) \leq_\Pi \mathrm{lv}(u)$ and outputs 0 or 1, an assignment for u.

Definition 6.3.1. *Let $\Psi = \Pi\Phi$ be a closed PCNF and let σ be a universal strategy for Ψ. Let $\tau_\exists$ be an assignment to the existentially quantified variables of Ψ and τ an extension of $\tau_\exists$ to all of the variables of Ψ. Then τ is consistent with σ if for all universally quantified u in Ψ we have that $\tau(u) = \sigma_u(\tau_\exists)$.*

In $\sigma_u(\tau_\exists)$ only the assignments to variables earlier than u in the prefix are actually relevant, the later part of the assignment does not affect the output of σ_u.

Lemma 6.3.2. *If $\Pi'\Phi \wedge C$ is derived from $\Pi\Phi$ by ATA, and σ' is a winning universal strategy for $\Pi'\Phi \wedge C$, then we can construct a winning universal strategy σ for $\Pi\Phi$.*

Proof. Clause C is added by ATA so $\Phi \wedge \bar{C} \vdash_1 \bot$. Therefore, any assignment that falsifies C also falsifies Φ.

σ' is a strategy that ensures $\Pi'\Phi \wedge C$ is falsified given any existential assignment. Let $\tau_\exists$ be an assignment to the existential variables and τ its extension consistent with the Herbrand functions of σ'. We know that τ falsifies Φ or C since σ' is a winning strategy for $\Pi'\Phi \wedge C$. But if τ falsifies C then, since C is an asymmetric tautology with respect to Φ, also τ falsifies Φ.

Let $\sigma = \sigma'$ except that if $\Pi \neq \Pi'$ then any existential input variable that does not appear in Φ can be restricted in σ to 0 or 1 arbitrarily (there is a winning strategy for either assignment, so we just pick one). If a universal variable is in Π' but not Π it will have no effect on the outcome of the game for $\Pi\Phi$ and there is no strategy for this variable in σ.

Every assignment consistent with σ falsifies Φ so σ is a winning universal strategy for $\Pi\Phi$. $\square$

Lemma 6.3.3. *If $\Pi'\Phi \wedge C$ is derived from $\Pi\Phi$ by QRATA, and σ' is a winning universal strategy for $\Pi'\Phi \wedge C$, then we can construct a winning universal strategy σ for $\Pi\Phi$.*

Proof. C contains some existential literal l such that for every $D \in \Phi$ with $\bar{l} \in D$ and outer clause O_D, $\Pi'\Phi \wedge \bar{C} \wedge \bar{O}_D \vdash_1 \bot$. For notational convenience we split C into three parts: the literals before l in the prefix, those after l in the prefix, and l itself. We define

$$A = \{k \in C \mid k \neq l, \mathrm{lv}(k) \leq_\Pi \mathrm{lv}(l)\},$$

$$B = \{k \in C \mid \mathrm{lv}(k) >_\Pi \mathrm{lv}(l)\}.$$

Therefore $C = (A \vee l \vee B)$ and

$$\Pi'\Phi \wedge \bar{A} \wedge \bar{l} \wedge \bar{B} \wedge \bar{O}_D \vdash_1 \bot,$$

and the derivation to be considered is rewritten.

$$\frac{\Pi\Phi}{\Pi'\Phi \wedge (A \vee l \vee B)} \text{(QRATA on } l\text{)}$$

Initially we assume $\Pi = \Pi'$. Let u be a universally quantified variable in Π. If $\mathrm{lv}(u) <_\Pi \mathrm{lv}(l)$ then $\sigma_u = \sigma'_u$. If $\mathrm{lv}(u) >_\Pi \mathrm{lv}(l)$ then it is necessary to consider two cases (we don't need to consider $\mathrm{lv}(u) = \mathrm{lv}(l)$ since l is existentially quantified).

Let $\tau_\exists$ be an assignment to the existentially quantified variables and let τ denote the extension of $\tau_\exists$ consistent with σ'.

$$\sigma_u(\tau_\exists) = \begin{cases} \sigma'_u(\tau'_\exists) & \text{if } \tau(A \vee l) = 0 \text{, but for every clause } D \text{ with } \bar{l} \in D \\ & \text{if } O_D \text{ is the outer clause of } D \text{ then } \tau(O_D) = 1 \\ & \text{where } \tau'_\exists \text{ differs from } \tau_\exists \text{ only on variable } l \\ & \text{such that } l \text{ is satisfied by } \tau'_\exists, \\ \sigma'_u(\tau_\exists) & \text{otherwise.} \end{cases}$$

Both $(A \vee l)$ and O_D only contain literals whose level is no greater than $\mathrm{lv}(l)$, and $\mathrm{lv}(u) > \mathrm{lv}(l)$ so σ_u only depends on variables from earlier levels in the quantifier prefix as required.

We need to show that σ actually falsifies $\Pi\Phi$. Now consider an assignment τ which is consistent with σ. We will show that $\tau(\Phi) = 0$. σ and σ' are identical for variables earlier than l in Π', so for these variables τ is also consistent with σ'.

Suppose that τ satisfies A, then $\tau(A \vee l) = 1$ so for all u with $\mathrm{lv}(u) > \mathrm{lv}(l)$ we have $\sigma_u(\tau_\exists) = \sigma'_u(\tau_\exists)$. Therefore σ and σ' are identical. Because τ is consistent with σ' we know it falsifies $\Pi'\Phi \wedge (A \vee l \vee B)$. Since we have assumed τ satisfies A it cannot falsify $(A \vee l \vee B)$ and instead τ falsifies $\Pi\Phi$.

Now suppose that τ falsifies A but satisfies the outer clauses of every D with $\bar{l} \in D$. Consider an assignment to the existentially quantified variables only which, for these variables, is identical to τ except that l evaluates to true. Call this assignment $\tau'_\exists$. Let τ' be the extension of $\tau'_\exists$ which is consistent with σ'. The assignments τ' and τ must be identical on all variables earlier than l in Π'. Since τ' satisfies $(A \vee l \vee B)$ (because l is true in τ') it must falsify some clause in Φ. Now modifying τ' so that l evaluates to false cannot satisfy any additional clauses in Φ since, by assumption, the outer clauses of all D with $\bar{l} \in D$ are already satisfied. Under σ all universal variables occurring after l in the prefix are assigned according to σ' as if l were made true. This will falsify some clause in Φ regardless of which value is actually assigned to l.

If τ falsifies A but also falsifies the outer clause O_D of some clause D with $\bar{l} \in D$ then we know that the responses from σ here are defined to be identical to the original σ', and either τ falsifies Φ or τ falsifies $(A \vee l \vee B)$. Since $\Pi'\Phi \wedge \bar{A} \wedge \bar{l} \wedge \bar{B} \wedge \bar{O}_D \vdash_1 \bot$ and τ falsifies O_D, it follows that if τ falsifies $(A \vee l \vee B)$ then Φ is also falsified by τ.

If in fact $\Pi \neq \Pi'$ then Π' contains more variables than Π. To obtain strategies for the variables in Π we first construct the strategies as above, assuming prefix Π', then fix these as in Lemma 6.3.2 to not include the variables missing from Π. Existentially quantified variables not in Π are restricted in σ to 0 or 1 arbitrarily. Universally quantified variables not in Π simply have their strategies removed from σ. □

Example 6.3.4. *Consider a QBF with prefix $\exists abcl \forall u$ and matrix $\Phi = (a \vee b \vee \neg u) \wedge (\neg b \vee u) \wedge (b \vee l \vee u) \wedge (b \vee \neg l \vee u)$, and clause $(a \vee c \vee l \vee \neg u)$.*

The only clause that contains $\neg l$ in Φ is $(b \vee \neg l \vee u)$, so the only outer clause we need to consider is (b). $\exists abcl \forall u\ \Phi \wedge \neg b \wedge \neg a \wedge \neg c \wedge \neg l \wedge u \vdash_1 \bot$ so QRATA is possible on l.

A winning strategy for the universal player after the new clause is added is to play u to 1 if and only if a, c and l are all 0. In our strategy extraction we derive the strategy prior to when QRATA is used.

- *If any of a, c, l are 1, we can continue to set u to 0 and falsify some clause in Φ not containing $\neg u$.*
- *If a, b, c, l are all 0, we falsify the only outer clause (b) thus we know via unit*

propagation some other clause (here $(a \vee b \vee \neg u)$) will be falsified if we continue to falsify $(a \vee c \vee l \vee \neg u)$ by setting u to 1. We keep playing the old strategy for this reason.

- *If a, c, l are all 0 and b is 1, setting u to 1 no longer works as it only falsifies the added clause, we instead see what happens when l is flipped to 1. The strategy now says to set u to 0, and this leads to falsifying clause $(\neg b \vee u)$.*

Lemma 6.3.5. *If $\Pi'\Phi \wedge C$ is derived from $\Pi\Phi \wedge (C \vee l)$ by QRATU, and σ' is a winning universal strategy for $\Pi'\Phi \wedge C$, then we can construct a winning universal strategy σ for $\Pi\Phi \wedge (C \vee l)$.*

Proof. The rule QRATU removes universal literal l from $(C \vee l)$. As before, it is useful to have notation for sub-clauses of C containing the variables prior to and later than l in the prefix. Let $A = \{k \in C \mid k \neq l, \text{lv}(k) \leq_\Pi \text{lv}(l)\}$, and $B = \{k \in C \mid \text{lv}(k) >_\Pi \text{lv}(l)\}$ so that $(C \vee l) = (A \vee l \vee B)$.

$$\frac{\Pi\Phi \wedge (A \vee l \vee B)}{\Pi'\Phi \wedge (A \vee B)} \text{ (QRATU on } l)$$

For every clause $D \in \Phi$ with $\bar{l} \in D$ and outer clause O_D, it holds that

$$\Pi' \; \Phi \wedge \bar{A} \wedge \bar{B} \wedge \bar{O}_D \vdash_1 \bot.$$

We have a winning universal strategy σ' for $\Pi'\Phi \wedge (A \vee B)$ and we will use this to construct a winning strategy σ for $\Pi\Phi \wedge (A \vee l \vee B)$. Let u be a universally quantified variable in Π. If $\text{var}(l) \neq u$ then $\sigma_u = \sigma'_u$. If $\text{var}(l) = u$ and u does not appear in Π' then $\sigma_u = a$, where $a \in \{0, 1\}$ falsifies l.

Otherwise, if $\text{var}(l) = u$, for each assignment to the existentially quantified variables $\tau_\exists$ prior to u in Π we need to define the output of σ_u. Let τ denote the extension of $\tau_\exists$ consistent with σ'.

$$\sigma_u(\tau_\exists) = \begin{cases} a & \text{if } \tau(A) = 0 \text{, but for every clause } D \text{ with } \bar{l} \in D \\ & \text{if } O_D \text{ is the outer clause of } D \text{ then } \tau(O_D) = 1 \\ & \text{where } a \in \{0, 1\} \text{ is such that setting } u \text{ to } a \text{ falsifies } l, \\ \sigma'_u(\tau_\exists) & \text{otherwise.} \end{cases}$$

The definition ensures that the decision for how to assign u depends only on variables whose level is no greater than u.

We need to show that σ actually falsifies $\Pi\Phi \wedge (A \vee l \vee B)$. Let τ be an assignment that is consistent with σ. Other than the assignment to $\text{var}(l)$ we know that τ is also consistent with σ'.

If τ satisfies A then τ is also completely consistent with σ' so it must also falsify $\Phi \wedge (A \vee B)$. Since τ satisfies A, there is some clause in Φ that τ falsifies.

If τ falsifies A but satisfies the outer clauses of all D with $\bar{l}$ in D then it may be that τ is not consistent with σ' on $\mathrm{var}(l)$. We observe that there is an assignment τ' consistent with σ' which is identical to τ except possibly on $\mathrm{var}(l)$, which is assigned by τ so that l is falsified. Therefore τ' satisfies all the outer clauses and modifying τ' so that l evaluates to false (i.e. to τ) cannot satisfy any additional clauses in $\Pi'\Phi \wedge (A \vee B)$. Since σ' is a winning strategy, τ' falsifies $\Pi'\Phi \wedge (A \vee B)$. If τ' falsifies a clause in Φ then τ falsifies the same clause. If τ' falsifies $(A \vee B)$ then τ falsifies $(A \vee l \vee B)$.

If τ falsifies A, and for some D with $\bar{l} \in D$ the outer clause O_D of D is also falsified, then τ is fully consistent with σ', so it falsifies either Φ or $(A \vee B)$. If τ falsifies some clause in Φ then we are done. If τ falsifies $(A \vee B)$ then τ falsifies $O_D \vee (A \vee B)$ so it must also falsify Φ since $\Pi'\Phi \wedge \bar{A} \wedge \bar{B} \wedge \bar{O}_D \vdash_1 \bot$ by the condition for QRATU. □

Lemma 6.3.6. *(Balabanov & Jiang, 2012) If $\Pi'\Phi \wedge C$ is derived from $\Pi\Phi \wedge (C \vee l)$ by universal reduction, and σ' is a winning universal strategy for $\Pi'\Phi \wedge C$, then we can construct a winning universal strategy σ for $\Pi\Phi \wedge (C \vee l)$.*

Proof. The universal reduction rule removes l from clause $(C \vee l)$ where $\mathrm{lv}(x) \leq_\Pi \mathrm{lv}(l)$ for every literal $x \in C$.

$$\frac{\Pi\Phi \wedge (C \vee l)}{\Pi'\Phi \wedge C}\ (\forall\text{-Red})$$

Let $\sigma_u = \sigma'_u$ when $\mathrm{var}(l) \neq u$. If $\mathrm{var}(l) = u$ and u does not appear in Π' then $\sigma_u = a$ where $a \in \{0, 1\}$ falsifies l. Otherwise, if $\mathrm{var}(l) = u$ then for each assignment $\tau_\exists$ to the existential variables let τ be the extension of $\tau_\exists$ consistent with σ' and let σ_u be defined as follows.

$$\sigma_u(\tau_\exists) = \begin{cases} a & \text{if } \tau(C) = 0 \\ & \text{where } a \in \{0,1\} \text{ is such that setting } u \text{ to } a \text{ falsifies } l, \\ \sigma'_u(\tau_\exists) & \text{otherwise} \end{cases}$$

This deviates from σ' exactly when C is falsified, and always falsifies $(C \vee l)$ in that case. So $\Phi \wedge (C \vee l)$ is always falsified under σ. □

Lemma 6.3.7. *If $\Pi'\Phi$ is derived from $\Pi\Phi \wedge C$ by clause deletion, and σ' is a winning universal strategy for $\Pi'\Phi$, then we can can construct a winning universal strategy σ for $\Pi\Phi \wedge C$.*

Proof. Clause deletion allows to derive $\Pi'\Phi$ from $\Pi\Phi \wedge C$. We simply let $\sigma_u = \sigma'_u$. Since σ' always falsifies a clause from Φ then also σ will falsify a clause from Φ. □

Theorem 6.3.8. *QRAT(UR) has polynomial-time strategy extraction on false QBFs.*

Proof. We inductively show that we can compute a winning strategy σ on the current formula $\Pi\Phi$ during our steps of QRAT(UR). This strategy can be constructed as a circuit with polynomial size in the length of the QRAT(UR) proof of $\Pi\Phi$ and consists of Herbrand functions σ_u for each of the universal variables u in the formula, taking as input existentially quantified variables from earlier blocks in the quantifier prefix.

We work backwards in the proof. Initially (i.e. at the end of the proof) we have the empty clause in the formula so any universal strategy is winning. Therefore in the base case we take a strategy that sets all universal variables to 0.

For the inductive steps we construct a new strategy σ for $\Pi\Phi$ based on σ' for $\Pi'\Phi'$, which is possible by Lemmas 6.3.2, 6.3.3, 6.3.5, 6.3.6, 6.3.7. The circuits σ_u constructed for ATA and clause deletion steps are no larger than σ'_u. For universal reduction we have one copy of σ'_u and a circuit to check whether C is satisfied. For QRATU we need a circuit to determine if A and each of the outer clauses are satisfied. The result of this together with the output of σ'_u determines the final value for u. For QRATA a new circuit is added to decide whether $(A \vee l)$ and the outer clauses are satisfied. The output of this is used to possibly change the input to σ'_u, which can be achieved with a small sub-circuit. Crucially, only one copy of σ'_u is needed. On reaching the first line of the proof we have a strategy for the initial formula which can be constructed as a polynomial-size circuit, giving us strategy extraction. □

6.4 Unrestricted Refutational QRAT Does Not Have Strategy Extraction

So far we have considered the restricted version of QRAT which allows universal reduction ($\forall$-Red) but not extended universal reduction (EUR). In this section we will turn out attention to the unrestricted version of QRAT and show that polynomial-time strategy extraction is not possible in this system. Together these results clearly point to EUR as the reason that QRAT lacks strategy extraction. We further demonstrate that the ability to perform universal expansion is key.

We begin by constructing formulas that have short refutations in QRAT but may require a large strategy for one of the universal variables. The idea is to use the conjunction of a QBF and its negation (the negated formula is defined over new variables). This gives rise to pairs of variables in the conjunction, and in each pair one variable is existentially quantified and the other is universally quantified.

Recall that we can conceptualise a QBF as a two player game with one player responsible for making assignments to the existentially quantified variables and the other responsible for making assignments to the universally quantified variables. Assignments are made in the order of the quantifier prefix. Then the negation of a QBF can be thought of as representing the same game as the original, but with the role of the players switched.

We interleave the prefixes of the original QBF and its negation so that each existential assignment is made before the associated universal assignment. Only one of the conjuncts needs to be made false so a simple winning strategy for the universal player is to always copy the associated existential assignments. The situation is analogous to forcing the existential player to play against themselves, and requiring them to win from both positions.

The easy winning strategy is essential for the short proofs, but despite the guaranteed win, it is PSPACE-hard in the worst case to find out which game the universal player wins prior to playing because this depends on the exact choices made by the existential player throughout the game. Modifying the formulas with an extra universal variable that requires the calculation of who wins generates a family of QBFs that have a large universal strategy but still retain short QRAT proofs. As we will see, a single universal expansion or EUR step allows us to return to the previous easy problem.

6.4.1 Duality Formulas

In this section we will define formulas that represent the conjunction of a QBF and its negation, and show how a short QRAT proof can be uniformly obtained for these formulas.

Let X be a set of variables $X = \{x_1, \ldots, x_{2n}\}$. $\Pi\Phi(X)$ is a QBF where Π is a prefix binding all variables in X. More specifically, $\Pi = \forall x_1 \exists x_2 \forall x_3 \ldots \exists x_{2n}$ and $\Phi(X)$ is a CNF in the variables X.

We also define a second set of $2n$ variables $X' = \{x'_1, \ldots, x'_{2n}\}$ and an alternative prefix $\Pi' = \exists x'_1 \forall x'_2 \exists x'_3 \ldots \forall x'_{2n}$. The QBF $\Pi\Phi(X) \wedge \Pi' \neg\Phi(X')$ is necessarily false for any choice of CNF $\Phi(X)$. However this QBF is not in PCNF, which many proof systems, including QRAT, require.

The formula $\neg\Phi(X')$ can be transformed into a CNF $\bar{\Phi}(X', T)$ by the use of Tseitin variables $T = \{t_K \mid K \in \Phi(X)\}$. We overload the $'$ notation:

- For a literal l if $l = x_i$ then $l' = x'_i$ and if $l = \neg x_i$ then $l' = \neg x'_i$,
- For each clause $K \in \Phi(X)$ we denote the corresponding clause in $\Phi(X')$ as K' so that $K' = \bigvee_{l \in K} l'$.

The CNF $\bar{\Phi}(X', T)$ is required to be true precisely when $\Phi(X')$ is false. First, introduce clauses stating that Tseitin variable t_K is true if and only if clause K' is satisfied. Then $\Phi(X')$ is false if and only if at least one t_K is false, so we will also add a clause specifying that this must hold.

Therefore $\bar{\Phi}(X', T)$ contains the following clauses:

- $(\neg t_K \vee K')$ for each clause K in $\Phi(X)$,
- $(\bar{l}' \vee t_K)$ for each literal l in K,
- $\left(\bigvee_{K \in \Phi(X)} \neg t_K\right)$.

Next the QBF is put into prenex form so that all of the variables (including the Tseitin variables) are quantified together before both $\Phi(X)$ and $\bar{\Phi}(X', T)$. We place every universal variable to the right of its existential version. The auxiliary T variables must be placed at the end of the prefix and are existentially quantified. Thus, from any PCNF $\Psi = \Pi\Phi$ we generate a formula $\mathsf{Duality}(\Pi\Phi)$ which encodes in PCNF the claim that both Ψ and its negation are true.

$$\mathsf{Duality}(\Pi\Phi) = \exists x_1' \forall x_1 \exists x_2 \forall x_2' \dots \exists x_{2n-1}' \forall x_{2n-1} \exists x_{2n} \forall x_{2n}' \exists T \; \Phi(X) \wedge \bar{\Phi}(X', T)$$

6.4.2 Short proofs of Duality Formulas

The technique for showing that the $\mathsf{Duality}$ formulas have short proofs is inspired by a result in Beyersdorff *et al.* (2018). The authors demonstrate short $\mathsf{Frege} + \forall\mathsf{red}$ proofs of a family of QBFs that take a representation of a graph and state that there is a k-clique (Clique) and also that there is no k-clique (Co-Clique). The short proofs exploit the fact that the Co-Clique part of the formula was structured in a similar way to the Clique part.

We generalise this approach to give short proofs of the $\mathsf{Duality}$ formulas. We will first give a sketch proof of how this is done using $\mathsf{Frege} + \forall\mathsf{red}$ rules to provide intuition for the argument before describing the steps of the QRAT refutation in detail.

$\mathsf{Frege} + \forall\mathsf{red}$ is simply a propositional Frege system augmented with the $\forall$-Red rule for removing universally quantified variables by allowing the substitution of a Boolean value for universally quantified u in any previously derived line where $\mathrm{lv}(u) > \mathrm{lv}(x)$ for all existentially quantified x in the proof line.

The clauses in $\mathsf{Duality}(\Pi\Phi)$ state $\bigwedge_K (t_K \leftrightarrow K')$, $\bigvee_K \neg t_K$ and $\bigwedge K$ where clause K' is identical to clause K with all instances of x_i replaced with x_i' (for all i). From an assumption $\bigwedge_{i=1}^{2n} (x_i \leftrightarrow x_i')$ we can find a contradiction in polynomially many Frege steps. We give an outline of the derivation.

$$\dfrac{\bigvee_K \neg t_K \qquad \dfrac{\bigwedge K \qquad \dfrac{\bigwedge_K (t_K \leftrightarrow K') \qquad \bigwedge_{i=1}^{2n} (x_i \leftrightarrow x_i')}{\bigwedge_K (t_K \leftrightarrow K)}}{\bigwedge_K t_K}}{\bot}$$

Therefore we conclude that $\bigvee_{i=1}^{2n} \neg(x_i \leftrightarrow x'_i)$.

Now, starting from the variables quantified innermost in the prefix, we perform universal reduction on all universally quantified x'_{2j} and x_{2j-1}. The first universal reduction step sets x'_{2n} to 0, and we substitute this into $\bigvee_{i=1}^{2n} \neg(x_i \leftrightarrow x'_i)$.

$$\neg(x_{2n} \leftrightarrow 0) \vee \bigvee_{i=1}^{2n-1} \neg(x_i \leftrightarrow x'_i)$$

This simplifies to

$$x_{2n} \vee \bigvee_{i=1}^{2n-1} \neg(x_i \leftrightarrow x'_i).$$

Reduction can also be done by substituting 1 for x'_{2n}, which simplifies to

$$\neg x_{2n} \vee \bigvee_{i=1}^{2n-1} \neg(x_i \leftrightarrow x'_i).$$

We can resolve these two disjunctions together and conclude $\bigvee_{i=1}^{2n-1} \neg(x_i \leftrightarrow x'_i)$.

Now x_{2n-1} is the innermost universally quantified variable. The same sequence of steps is applied for each universal variable in reverse level order and leads to a contradiction which completes the proof.

This idea also works in QRAT and the short refutations we construct have a uniform structure. These refutations do not use EUR. The $\forall$-Red rule is sufficient.

Theorem 6.4.1. *Given a formula* Duality$(\Pi\Phi)$ *we can construct a polynomial-size QRAT(UR) refutation of* Duality$(\Pi\Phi)$*.*

Proof. Let $|K|$ be the number of literals in clause $K \in \Phi$. We have that $|\,\text{Duality}(\Pi\Phi)| \geq |\Phi| \geq \Sigma_{K\in\Phi}|K|$. Recall $\Pi\Phi$ has $2n$ variables.

Extension Variables The refutation begins by using QRATA (Definition 6.2.8) to introduce an extension variable eq_{x_i} for each $x_i \in X$. Each eq_{x_i} is existentially quantified and is introduced to the prefix so that

- $\text{lv}(\text{eq}_{x_i}) > \text{lv}(x_i)$ and $\text{lv}(\text{eq}_{x_i}) > \text{lv}(x'_i)$,
- $\text{lv}(\text{eq}_{x_i}) < \text{lv}(x_j)$ and $\text{lv}(\text{eq}_{x_i}) < \text{lv}(x'_j)$ for every $j > i$.

This is possible since both x_i and x'_i appear before x_j and x'_j in the quantifier prefix whenever $j > i$.

For each $x_i \in X$ use QRATA to add four clauses:

- $(\neg x_i \vee x'_i \vee \neg\text{eq}_{x_i})$,
- $(x_i \vee \neg x'_i \vee \neg\text{eq}_{x_i})$,

- $(\neg x_i \vee \neg x'_i \vee \mathrm{eq}_{x_i})$,
- $(x_i \vee x'_i \vee \mathrm{eq}_{x_i})$.

Recall that adding a clause by QRATA requires that we have an existential literal l in the new clause C such that $\Phi \wedge \bar{C} \wedge \bar{O}_D \vdash_1 \bot$ for all D with $\bar{l} \in D$. For the first two clauses this is vacuously satisfied with $l = \neg\mathrm{eq}_{x_i}$ since eq_{x_i} does not appear positively anywhere in the formula.

To add the third and fourth clauses let $l = \mathrm{eq}_{x_i}$ and consider the two outer clauses $(\neg x_i \vee x'_i)$ and $(x_i \vee \neg x'_i)$. The QRATA condition is satisfied for $(\neg x_i \vee \neg x'_i \vee \mathrm{eq}_{x_i})$ because $x_i \wedge x'_i \wedge x_i \wedge \neg x'_i \vdash_1 \bot$ and $x_i \wedge x'_i \wedge \neg x_i \wedge x'_i \vdash_1 \bot$, and similarly for the final clause.

For each of the original $2n$ variables in $\Pi\Phi$ we have added four clauses of constant size. Following $O(n)$ steps the formula has increased in length by $O(n)$ characters.

Non Equivalence of X and X' The next three steps are equivalent to those in the derivation of $\bigvee_{i=1}^{2n} \neg(x_i \leftrightarrow x'_i)$ in the sketch proof above. By ATA (definition 6.2.6) we derive:

- $(\bigvee_{i=1}^{2n} \neg\mathrm{eq}_{x_i} \vee t_K \vee \bar{l})$ for every $K \in \Phi(X)$ and every $l \in K$.

  For every such l we already have clauses $(\bar{l} \vee l' \vee \neg\mathrm{eq}_{var(l)})$ and $(\bar{l}' \vee t_K)$ in Φ. The condition for ATA is satisfied since

$$\bigwedge_{i=1}^{2n} \mathrm{eq}_{x_i} \wedge \neg t_K \wedge l \wedge (\bar{l} \vee l' \vee \neg\mathrm{eq}_{var(l)}) \wedge (\bar{l}' \vee t_K) \vdash_1 \bot.$$

- $(\bigvee_{i=1}^{2n} \neg\mathrm{eq}_{x_i} \vee t_K)$ for every $K \in \Phi(X)$.

  The condition for ATA is satisfied since

$$\bigwedge_{i=1}^{2n} \mathrm{eq}_{x_i} \wedge \neg t_K \wedge \bigwedge_{l \in K} (\bigvee_{i=1}^{2n} \neg\mathrm{eq}_{x_i} \vee t_K \vee \bar{l}) \wedge K \vdash_1 \bot.$$

- $(\bigvee_{i=1}^{2n} \neg\mathrm{eq}_{x_i})$.

  The condition for ATA is satisfied since

$$\bigwedge_{i=1}^{2n} \mathrm{eq}_{x_i} \wedge \bigwedge_{K \in \Phi(x)} (\bigvee_{i=1}^{2n} \neg\mathrm{eq}_{x_i} \vee t_K) \wedge \bigvee_{K \in \Phi(X)} \neg t_K \vdash_1 \bot.$$

Each clause has $O(n)$ literals and there are at most $|\Phi|$ clauses of each type. In $O(|\Phi|)$ proof steps the formula has increased in length by $O(n|\Phi|)$.

Removing the Universal Variables Finally, we want to derive $(\bigvee_{i=1}^{j-1} \neg \mathrm{eq}_{x_i})$ for $j = 2n \dots 1$ (thus $j = 1$ means that we have derived the empty clause). Assuming that we already have $(\bigvee_{i=1}^{j} \neg \mathrm{eq}_{x_i})$ we can use ATA to add:

- $(\bigvee_{i=1}^{j-1} \neg \mathrm{eq}_{x_i} \vee x_j \vee x'_j)$.

  For every x_i we have clause $(x_i \vee x'_i \vee \mathrm{eq}_{x_i})$. The ATA condition is satisfied since

$$\bigwedge_{i=1}^{j-1} \mathrm{eq}_{x_i} \wedge \neg x_j \wedge \neg x'_j \wedge (x_j \vee x'_j \vee \mathrm{eq}_{x_j}) \wedge \bigvee_{i=1}^{j} \neg \mathrm{eq}_{x_i} \vdash_1 \bot.$$

- $(\bigvee_{i=1}^{j-1} \neg \mathrm{eq}_{x_i} \vee \neg x_j \vee \neg x'_j)$.

  For every x_i we have $(\neg x_i \vee \neg x'_i \vee \mathrm{eq}_{x_i})$ and the ATA condition is satisfied since

$$\bigwedge_{i=1}^{j-1} \mathrm{eq}_{x_i} \wedge x_j \wedge x'_j \wedge (\neg x_j \vee \neg x'_j \vee \mathrm{eq}_{x_j}) \wedge \bigvee_{i=1}^{j} \neg \mathrm{eq}_{x_i} \vdash_1 \bot.$$

In clauses $(\bigvee_{i=1}^{j-1} \neg \mathrm{eq}_{x_i} \vee x_j \vee x'_j)$ and $(\bigvee_{i=1}^{j-1} \neg \mathrm{eq}_{x_i} \vee \neg x_j \vee \neg x'_j)$, whichever of x_j and x'_j is universally quantified is innermost in Π by the construction of $\mathsf{Duality}(\Pi\Phi)$ and the decision of where to introduce the variables eq_{x_i} in the prefix. Without loss of generality, assume x'_j is universally quantified so we can use universal reduction to derive clauses $(\bigvee_{i=1}^{j-1} \neg \mathrm{eq}_{x_i} \vee x_j)$ and $(\bigvee_{i=1}^{j-1} \neg \mathrm{eq}_{x_i} \vee \neg x_j)$, then ATA allows to add the resolvent $(\bigvee_{i=1}^{j-1} \neg \mathrm{eq}_{x_i})$.

For each of the $2n$ variables from Φ there are five proof steps in this final part of the refutation, each introducing a new clause of size $O(n)$, and in total the formula has increased in length by $O(n^2)$. The whole refutation therefore has size $O(|\mathsf{Duality}(\Pi\Phi)|^2)$. □

6.4.3 Making Strategies Hard

The formulas $\mathsf{Duality}(\Pi\Phi)$ have short winning strategies for the universal player, namely to always play so that $x_i = x'_i$. By construction one or other of $\Phi(X)$ or $\bar{\Phi}(X', T)$ will be falsified, but which sub-formula is falsified depends on the existential assignments. We know also that one of $\Pi\Phi(X)$ or $\Pi'\bar{\Phi}(X', T)$ is false and so has a winning strategy for the universal player. Deciding which sub-formula is false is PSPACE-hard and the winning strategy for the false formula could be much more complicated than the strategy for $\mathsf{Duality}(\Pi\Phi)$. We introduce formulas exploiting this hardness.

$$\mathsf{Select}(\Pi\Phi) = \forall u\ \mathbf{Q}\ \exists T\ \Phi_u(X) \wedge \bar{\Phi}_{\neg u}(X', T)$$

$$\text{where } \Phi_u(X) = \bigwedge_{K \in \Phi(X)} (K \vee u) \text{ and } \bar{\Phi}_{\neg u}(X', T) = \bigwedge_{K \in \bar{\Phi}(X', T)} (K \vee \neg u)$$

$$\text{and } \mathbf{Q} = \exists x'_1 \forall x_1 \exists x_2 \forall x'_2 \dots \exists x'_{2n-1} \forall x_{2n-1} \exists x_{2n} \forall x'_{2n}$$

When $u = 1$, all clauses in $\Phi_u(X)$ are satisfied, and the literal $\neg u$ is removed from every clause in $\bar{\Phi}_{\neg u}(X', T)$, so that we're left with $\bar{\Phi}(X', T)$. Similarly, when $u = 0$ we are left with $\Phi(X)$. Therefore, the winning strategy for the universal player requires knowing which of $\Phi(X)$ and $\bar{\Phi}(X', T)$ is false.

6.4.4 Short Proofs of Select Formulas in QRAT

We use the formulas $\mathsf{Select}(\Pi\Phi)$ to show that refutational QRAT does not have strategy extraction under a widely accepted complexity assumption.

Theorem 6.4.2. *QRAT has polynomial-size uniform proofs of* $\mathsf{Select}(\Pi\Phi)$ *for any QBF* $\Pi\Phi$.

Proof. The first step in the proof is to use Extended Universal Reduction (EUR) to remove u from all clauses in $\Phi_u(X)$ and $\neg u$ from all clauses in $\bar{\Phi}_{\neg u}(X', T)$. Using EUR to reduce l in C requires that $\bar{l}$ does not appear in ε, the fix-point of the procedure given in Definition 6.2.12. In other words, there is no inner resolution path between any clauses containing the removed literal and its negation. We can only add literals to the set used to build ε from clauses that share variables in common with the current set of literals. However, u and $\neg u$ appear in sections of the formula that have no other variables in common. As a result we can always reduce u (and $\neg u$) in $\mathsf{Select}(\Pi\Phi)$.

Having performed these (polynomially many) EUR steps the formula is identical to $\mathsf{Duality}(\Pi\Phi)$, which is uniformly refuted as in Theorem 6.4.1. □

Theorem 6.4.3. *Refutational QRAT does not admit strategy extraction unless* $\mathsf{P} = \mathsf{PSPACE}$.

Proof. Given a QBF $\Pi\Phi$, with Π a prefix and Φ a propositional formula in the variables of Π, create the formula $\mathsf{Select}(\Pi\Phi)$. Theorem 6.4.2 allows to find a refutation of $\mathsf{Select}(\Pi\Phi)$ in polynomial time. Suppose QRAT has polynomial-time strategy extraction. Then a winning strategy for the universal player can be efficiently extracted from the proof of $\mathsf{Select}(\Pi\Phi)$.

Since u is outermost in the prefix a winning strategy must give u a constant value, it may not depend on any existentially quantified variables. If the strategy sets $u = 0$ then all clauses in $\bar{\Phi}_{\neg u}(X', T)$ are immediately satisfied. Because it was a winning universal strategy we know that the remaining formula, after assigning u to 0 throughout, is still false and the remaining strategy witnesses this. Therefore, the rest of the extracted strategy is a winning universal strategy for $\Pi\Phi$, showing that $\Pi\Phi$ is false.

Similarly, if the strategy sets $u = 1$ then it must be the case that $\bar{\Phi}_{\neg u}(X', T)$ is false and so, by construction, $\Pi\Phi$ is true.

From Theorem 6.4.2 and the assumption that QRAT has polynomial-time strategy extraction we have constructed a decision procedure that is able to decide whether an arbitrary QBF is true or false in polynomial time. Deciding a QBF is a PSPACE-complete problem, so we have shown that if refutational QRAT has strategy extraction then P = PSPACE. □

In fact, the full power of EUR is not required. QRAT(UR) is capable of refuting the formulas Duality($\Pi\Phi$), and the initial EUR step can be replaced by universal expansion of u, producing a formula equivalent to Duality($\Pi\Phi$) with renamed variables. Even QBF solvers whose underlying proof system uses universal reduction to handle universally quantified variables often employ a pre-processing stage that includes universal expansion. The Select($\Pi\Phi$) formulas show that a single initial expansion step may be sufficient to prevent strategy extraction. We now explore the connection between universal expansion, strategy extraction, and feasible interpolation further.

6.5 Relation to Feasible Interpolation

The results of the previous section indicate that expansion steps may prevent strategy extraction. However there are proof systems and solvers that admit strategy extraction despite using universal expansion (for example, $\forall$Exp + Res). It is therefore clear that other rules in the proof system must play a role in determining whether or not strategy extraction is possible.

In Theorem 6.4.3, strategy extraction in QRAT for the formulas Select($\Pi\Phi$) implies the ability to efficiently decide the truth of $\Pi\Phi$. We can think of the strategy for u as a circuit deciding between $\Pi\Phi$ and $\neg(\Pi\Phi)$. In propositional logic the efficient extraction of these deciding circuits from a proof is a well studied technique known as feasible interpolation, and the circuit itself is called an interpolant.

Given a true propositional implication $\Phi_1(P,Q) \rightarrow \Phi_2(P,R)$ (or, equivalently, a false conjunction $\Phi_1(P,Q) \wedge \neg\Phi_2(P,R)$), Craig's interpolation theorem (Craig, 1957b) states that there is an interpolant $C(P)$ in only the joint variables P. Given an assignment to P, the circuit $C(P)$ indicates which of $\Phi_1(P,Q)$ and $\neg\Phi_2(P,R)$ is false. Feasible interpolation is a property of proof systems. A proof system has feasible interpolation (Krajíček, 1997; Pudlák, 1997) if and only if there is a polynomial time procedure that takes a proof of $\Phi_1(P,Q) \rightarrow \Phi_2(P,R)$ as an input and extracts an interpolating circuit $C(P)$. The definition is easily lifted to the QBF setting.

Definition 6.5.1. *Let Φ be a false QBF of the form $\exists P \Pi_1 \Pi_2\ \Phi_1(P,Q) \wedge \Phi_2(P,R)$, where P, Q and R are disjoint sets of variables, Π_1 is a quantifier prefix containing the variables in Q, and Π_2 is a quantifier prefix containing the variables in R. An interpolant for Φ is a Boolean circuit C such that for every Boolean assignment α to the variables of P, $C(\alpha) = 0$ implies that $\exists P \Pi_1 \Phi_1(P,Q)$ is false and $C(\alpha) = 1$ implies that $\exists P \Pi_2 \Phi_2(P,R)$ is false. A QBF proof system* f *has feasible interpolation if there* *is a polynomial time procedure that, given an* f *refutation* π *of* Φ *(in the required form given above), returns an interpolant for* Φ*.*

In Beyersdorff *et al.* (2017a) feasible interpolation was linked to strategy extraction by adding an extra universal variable with similarities to Section 6.4.3 and how the Select formulas are created from the Duality formulas. This connection allows us to express a necessary condition for efficient strategy extraction in QBF proof systems which allow universal expansion.

Theorem 6.5.2. *Given any propositional refutation system* f*, if the refutational QBF proof system* $\mathsf{f}+\forall\mathsf{Exp}$ *has polynomial-time strategy extraction then* f *must have feasible interpolation (provided refutations of* f *work independently of the variable names).*

Proof. Suppose $\mathsf{f}+\forall\mathsf{Exp}$ has strategy extraction and we have an f-refutation π of $\Phi_1(P,Q) \wedge \Phi_2(P,R)$ with P,Q,R disjoint sets of variables. We will show that we can find an interpolant in polynomial time.

We consider the following QBF

$$\exists P \forall u \exists Q \exists R (\Phi_1(P,Q) \vee u) \wedge (\Phi_2(P,R) \vee \bar{u}).$$

We can refute this formula in $\mathsf{f}+\forall\mathsf{Exp}$ using π. Expansion gives us

$$(\Phi_1(P,Q^{0/u}) \vee 0) \wedge (\Phi_2(P,R^{0/u}) \vee 1) \wedge (\Phi_1(P,Q^{1/u}) \vee 1) \wedge (\Phi_2(P,R^{1/u}) \vee 0)$$

but this immediately simplifies to

$$\Phi_1(P,Q^{0/u}) \wedge \Phi_2(P,R^{1/u}).$$

We can now refute this using π with the annotated variables from $Q^{0/u}$ in place of variables from Q, and the annotated variables from $R^{1/u}$ in place of variables from R. The provision that refutations work independently of variables names is important here as one could define f to be a pathological proof system that disallowed steps using variables named as in $R^{0/u}$, but allowed them named as in R.

We can then extract a strategy for u as a circuit in the variables P. However this circuit is also an interpolant for $\Phi_1(P,Q) \wedge \Phi_2(P,R)$. □

It would also be valuable to identify *sufficient* conditions for strategy extraction related to feasible interpolation. $\forall\mathsf{Exp}+\mathsf{Res}$ has strategy extraction via a technique of restricting the proof to find a universal response to any existential assignment. The outermost block of existentially quantified variables are assigned in the $\forall\mathsf{Exp}+\mathsf{Res}$ proof. In the restricted proof the assignments corresponding to the outermost block of universally quantified variables must be all the same and this indicates the correct universal assignment to use in this case. The proof is repeatedly restricted in this way until a complete set of universal responses is found and the formula is falsified.

The "reading off" which clauses actually contribute to the proof for a given assignment is a weak form of feasible interpolation and so we can say we have strategy extraction for $\mathsf{f}+\forall\mathsf{Exp}$ whenever refutational proof system f satisfies two conditions. Note that because of Theorem 6.5.2 feasible interpolation is implied by these two conditions (although this can be shown without Theorem 6.5.2). The extraction technique is inspired by the one used in Goultiaeva *et al.* (2011), but here we use it for expansion systems.

Theorem 6.5.3. $\mathsf{f}+\forall\mathsf{Exp}$ *has strategy extraction whenever the following two conditions hold.*

1. *From an* f *refutation* π *of* Φ *one can extract in polynomial time an* f *refutation* π_α *of* $\Phi|_\alpha$ *for any assignment* α *with* $|\pi_\alpha| \leq |\pi|$,
2. *From an* f *refutation* π_1 *of* $\Phi_1(Q) \wedge \Phi_2(R)$, *where* Q *and* R *share no common variables, one can extract in polynomial time an* f *refutation* π_2 *of either* $\Phi_1(Q)$ *or* $\Phi_2(R)$ *with* $|\pi_2| \leq |\pi_1|$.

Proof. Suppose we have a closed prenex QBF $\Psi = \exists X \forall Y \Pi \Phi$ where Π is a prefix in variables Z and Φ is a propositional matrix with variables in X,Y and Z. Suppose also that we have an $\mathsf{f}+\forall\mathsf{Exp}$ refutation π of Ψ. This is equivalent to an f proof π' of $\mathsf{subexp}_\pi(\Psi)$, the subset of the full expansion of Ψ used by π.

We will show that under conditions 1 and 2 we have a polynomial time procedure that takes any assignment α to X and outputs a response μ in Y and an $\mathsf{f}+\forall\mathsf{Exp}$ refutation of $\Pi\Phi|_{\alpha,\mu}$.

From π' we can extract in polynomial time an f refutation π'_α of $\mathsf{subexp}_\pi(\Psi)|_\alpha$ (by condition 1). Every clause in $\mathsf{subexp}_\pi(\Psi)|_\alpha$ is available by the axiom rule in an $\mathsf{f}+\forall\mathsf{Exp}$ refutation of $\forall Y \Pi \Phi|_\alpha$. Therefore, π'_α gives an $\mathsf{f}+\forall\mathsf{Exp}$ refutation π_α of $\forall Y \Pi \Phi|_\alpha$.

Now we find the response to α in universal variables $Y = \{y_1 \dots y_m\}$. We start with a response c to y_1 and find an $\mathsf{f}+\forall\mathsf{Exp}$ refutation of $\forall y_2 \dots y_m \Pi \Phi|_{\alpha, c/y_1}$, showing that the proof does not increase in size. Then we can repeat this for each variable of Y in turn.

Let μ_i be a Boolean assignment to variables $\{y_1, \dots, y_{i-1}\}$ for i with $1 \leq i \leq m$. Let π_i be an $\mathsf{f}+\forall\mathsf{Exp}$ refutation of the QBF $\forall y_i \dots y_m \Pi \Phi|_{\alpha,\mu_i}$. The variables of $\mathsf{subexp}_{\pi_i}(\forall y_i \dots y_m \Pi \Phi|_{\alpha,\mu_i})$ can be partitioned into $Z^{0/y_i} = \{z^\alpha \mid z \in Z, \alpha(y_i) = 0\}$ and $Z^{1/y_i} = \{z^\alpha \mid z \in Z, \alpha(y_i) = 1\}$. This completely partitions the variables because y_i is leftmost in the prefix.

$C \in \mathsf{subexp}_{\pi_i}(\Pi\Phi)|_{\alpha,\mu_i}$ cannot mix variables Z^{0/y_i} and Z^{1/y_i} since the axiom rule substitutes one or the other everywhere in C. Therefore we can re-write

$\mathsf{subexp}_{\pi_i}(\forall y_i \dots y_m \Pi\Phi|_{\alpha,\mu_i})$ as $\Phi_1(Z^{0/y_i}) \wedge \Phi_2(Z^{1/y_i})$ with f refutation π'_i (based on the $\mathsf{f}+\forall\mathsf{Exp}$ refutation π_i).

We define a new partial assignment μ_{i+1} with $\mu_{i+1}(y_j) = \mu_i(y_j)$ for $1 \leq j < i$. Condition 2 allows us to extract from π'_i an f refutation π'_{i+1} of either $\Phi_1(Z^{0/y_i})$ or $\Phi_2(Z^{1/y_i})$ in polynomial time. If $\Phi_1(Z^{0/y_i})$ is refuted then let $\mu_{i+1}(y_i) = 0$ and otherwise let $\mu_{i+1}(y_i) = 1$. Then π'_{i+1} becomes an $\mathsf{f}+\forall\mathsf{Exp}$ refutation π_{i+1} of $\forall y_{i+1} \dots y_m \Pi\Phi|_{\alpha,\mu_{i+1}}$ since $\mathsf{subexp}_{\pi_{i+1}}(\forall y_{i+1} \dots y_m \Pi\Phi|_{\alpha,\mu_{i+1}})$ is equal to either $\Phi_1(Z^{0/y_i})$ or $\Phi_2(Z^{1/y_i})$. Condition 2 guarantees $|\pi'_{i+1}| \leq |\pi'_i|$ so $|\pi_{i+1}| \leq |\pi_i|$ as well.

Once we get to μ_m we have a complete assignment to Y and a guarantee that the restricted QBF $\Pi\Phi|_{\alpha,\mu_m}$ is false by the $\mathsf{f}+\forall\mathsf{Exp}$ refutation π_m, with $|\pi_m| \leq |\pi|$.

We can repeat this procedure for every universal block and we end up with the false proposition $\perp$ and since our proofs are non-increasing in size in each step we guarantee this can be done in a polynomial time procedure. $\square$

Conclusion

In this chapter we have shown, perhaps surprisingly, that refutational QRAT does not have polynomial-time strategy extraction (provided P $\neq$ PSPACE), demonstrating an important asymmetry between the refutational and satisfaction versions of the proof system. Strategy extraction is often desirable in practice and our result shows that it may not be possible to create a general efficient tool for extracting strategies from QRAT traces output by solvers. We have shown that the difficulty of extracting strategies is related to a combination of the strength of the underlying propositional proof system and the type of universal reasoning employed. In particular, when QRAT's extended universal reduction rule is weakened to the universal reduction rule from Q-Res we have the proof system QRAT(UR), which we have shown does admit strategy extraction. Although most modern QBF solvers use universal expansion, at least during pre-processing, this will only prevent strategy extraction from the generated proofs if other strong inference steps are used.

Chapter 7

The Equivalence of Refutational QRAT and QRAT+

Lonsing & Egly (2018b) introduced an improvement to the QRAT proof system which generalises the properties that allow a clause to be recognised as redundant. Both the RAT and QRAT properties use unit propagation, which is a polynomial-time procedure and may equally be applied to both universally and existentially quantified variables. In the QBF setting, universal reduction is another common way of reasoning specifically about the universally quantified variables, and it can also be carried out in polynomial time. Therefore, Lonsing's and Egly's definition of QRAT+ replaces inferences $\vdash_1$ based on unit propagation with $\vdash_{1\forall}$, where $\vdash_{1\forall}$ is an inference using a combination of unit propagation *and* universal reduction.

To maintain soundness of the proof system when reasoning with universal reduction it is necessary to take the quantifier prefix into account. The usual requirement that universal reduction may only remove a universally quantified literal u from a clause if $\mathrm{lv}(u) > \mathrm{lv}(x)$ for all existentially quantified x in the clause applies. In addition to this, universal reduction is restricted to literals with a higher level than any literal in the clause being checked for redundancy.

In the first section of this chapter we define the requirement on $\vdash_{1\forall}$ and the QRAT+ proof system precisely. We then go on to show that, although QRAT+ can make inferences which are not possible in QRAT, the two systems are p-equivalent for refutations.

7.1 The QRAT+ Proof System

In the following definitions, let $\Pi\Phi$ be a closed PCNF and let C be a clause not in Φ. Let Π' be a prefix containing the variables of Φ and C such that Π is a sub-prefix of Π' containing the variables of Φ only.

Definition 7.1.1. *Let x be a variable in Π' and let $\mathcal{Q}_x$ be its quantifier. The abstraction of Π' with respect to C, denoted $\Pi'_{Abs(C)}$, is the quantifier prefix obtained from Π' by setting $\mathcal{Q}_x = \exists$ whenever $\mathrm{lv}(x) \leq \max_{l \in C}(\mathrm{lv}(l))$. If $\mathrm{lv}(x) > \max_{l \in C}(\mathrm{lv}(l))$ then $\mathcal{Q}_x$ is unchanged.*

Definition 7.1.2 (Quantified Asymmetric Tautology (QAT))**.** *C is a quantified asymmetric tautology with respect to $\Pi\Phi$ if and only if*

$$\Pi'_{Abs(C)}\ \Phi \wedge \bar{C} \vdash_{1\forall} \bot.$$

The reason for modifying the quantifier prefix is simply to prevent universal reduction steps on universal literals with level less than the maximum level of any literal in C. Without this restriction it would not be sound to assume that any clause C with $\Pi'\ \Phi \wedge \bar{C} \vdash_{1\forall} \bot$ is redundant, as the following example shows.

Example 7.1.3. *Let*

$$\Pi\Phi = \forall u_1 \exists x_1 \forall u_2 \exists x_2\ (u_1 \vee x_1) \wedge (\bar{u}_1 \vee \bar{x}_1) \wedge (u_2 \vee x_2) \wedge (\bar{u}_2 \vee \bar{x}_2).$$

Φ is true with the strategy that sets $x_1 \leftarrow \bar{u}_1$ and $x_2 \leftarrow \bar{u}_2$. Let $C = (x_1 \vee x_2)$. If universal reduction is allowed at any level then from $\Pi\ \Phi \wedge \bar{C}$ we can derive (u_1) by unit propagation and then $\bot$ by universal reduction. However, C cannot be redundant with respect to $\Pi\Phi$ since $\Pi\Phi \wedge C$ is false.

Now all of the rules from QRAT can be redefined to use quantified asymmetric tautologies in place of asymmetric tautologies.

Definition 7.1.4 (ATA+)**.** *If C is a* quantified *asymmetric tautology with respect to $\Pi\Phi$ then we can make the following inference.*

$$\frac{\Pi\Phi}{\Pi'\Phi \wedge C}\ (ATA+)$$

The definitions for QRATA and QRATU are similarly updated to use the QRAT+ property in place of the QRAT property.

Definition 7.1.5 (QRAT+)**.** *Clause C has QRAT+ on literal l in QBF $\Pi\Phi$ if and only if $(C \vee O_D)$ is a quantified asymmetric tautology with respect to $\Pi\Phi$ for all $D \in \Phi$ with $\bar{l} \in D$, where O_D is the outer clause of D with respect to l, $O_D = \{k \in D \mid \mathrm{lv}(k) \leq_{\Pi'} \mathrm{lv}(l), k \neq \bar{l}\}$.*

Definition 7.1.6 (QRATA+)**.** *If C has QRAT+ with respect to $\Pi\Phi$ on an existentially quantified literal l then we can make the following inference.*

$$\frac{\Pi\Phi}{\Pi'\Phi \wedge C}\ (QRATA+\ on\ l)$$

Definition 7.1.7 (QRATU+)**.** *If C has QRAT+ with respect to $\Pi\Phi$ on a universally quantified literal l then we can make the following inference.*

$$\frac{\Pi\Phi \wedge (C \vee l)}{\Pi'\Phi \wedge C} \text{ (QRATU+ on } l)$$

Lonsing and Egly do not differentiate between a refutational and satisfaction QRAT+. Here we focus on the refutational case where clauses can be deleted arbitrarily. Refutational QRAT+ therefore consists of ATA+, QRATA+, QRATU+, Extended Universal Reduction (EUR) and clause deletion. EUR and clause deletion are not changed in QRAT+.

7.2 Simulating QRAT+ by QRAT

The main idea of the proof is to explicitly derive in QRAT some of the intermediate steps used in the $\vdash_{1\forall}$ inferences of QRAT+. In particular, the clauses immediately preceding a universal reduction step in the $\vdash_{1\forall}$ procedure are derived as part of the main proof and the universal reduction can then also be carried out in the main proof. An example from Lonsing & Egly (2018b) serves as a simple illustration of how this method works.

Example 7.2.1. *Let $\Phi = \forall u_1, u_2 \exists x_1, x_2 \forall u_3 \exists x_3 \bigwedge_{i=0}^{7} C_i$ where*

$$C_0 = (\neg u_2 \vee \neg x_1 \vee \neg x_2) \qquad C_3 = (u_2 \vee x_1 \vee x_2) \qquad C_6 = (\neg x_1 \vee x_2 \vee \neg x_3)$$
$$C_1 = (\neg u_1 \vee \neg x_1 \vee x_2) \qquad C_4 = (\neg x_1 \vee \neg x_2 \vee x_3) \qquad C_7 = (\neg u_3 \vee x_3)$$
$$C_2 = (u_1 \vee x_1 \vee \neg x_2) \qquad C_5 = (u_3 \vee \neg x_3)$$

Both u_1 and u_2 can be removed by QRATU+ but not by QRATU.

In Φ, C_0 has QRAT+ on $\neg u_2$. The only clause containing u_2 is C_3 and the outer clause is empty. Unit propagation of x_1 and x_2 (from $\bar{C}_0$) allows us to derive x_3 (from C_4) then we use this to derive u_3 (from C_5) which is then removed by universal reduction. Therefore we can replace C_0 by $C_0 \backslash \{\neg u_2\}$.

This is not possible by QRATU, however it is possible to derive $C = (\neg x_1 \vee \neg x_2 \vee u_3)$ by ATA since $\Phi \wedge \bar{C} \vdash_1 \bot$. We have unit propagation on x_1 and x_2 as before, then use x_3 to derive u_3 which does not need to be removed by universal reduction because we can use unit propagation with $\neg u_3$ instead to derive the empty clause. Once C is derived we can remove u_3 by universal reduction to reach the target clause.

Now there are no clauses containing $\neg u_2$ we can remove u_2 from C_3 because the condition for QRATU is vacuously satisfied. A similar argument applies to u_1 in C_1 and C_2.

Lemma 7.2.2. *ATA+ is p-simulated by refutational QRAT(UR).*

Proof. As before, $\Pi\Phi$ is a PCNF and let C be a clause not in Φ. Let Π' be a prefix including the variables of C and Φ, and Π a sub-prefix of Π' containing the variables of Φ. ATA+ derives $\Pi'\Phi \wedge C$ from $\Pi\Phi$ when $\Pi'_{Abs(C)}\ \Phi \wedge \bar{C} \vdash_{1\forall} \bot$. However it may not be the case that $\Pi'\Phi \wedge \bar{C} \vdash_1 \bot$ because we may have used a universal reduction step (potentially multiple times).

The idea is that we break the single ATA+ step into several ATA and EUR steps. The parts of the $\vdash_{1\forall}$ inference that use unit propagation can be easily replicated in QRAT, so we focus on the parts of this inference that use universal reduction by explicitly deriving the clauses immediately prior to each universal reduction step during the QAT procedure.

Label these clauses L_i with the index i indicating the order in which they are derived (i.e. L_i is derived before L_j in the $\vdash_{1\forall}$ procedure if and only if $i < j$). In clause L_i the literal p_i is removed in the universal reduction step.

By considering the parts of the inference that use only unit propagation, we know that

1. $\Phi \wedge \bar{C} \vdash_1 L_1$,
2. $\Phi \wedge \bigwedge_{j<i} L_j \backslash \{p_j\} \wedge \bar{C} \vdash_1 L_i$, for each $i > 1$,
3. $\Phi \wedge \bigwedge_{i} L_i \backslash \{p_i\} \wedge \bar{C} \vdash_1 \bot$.

The conditions on reduction in QRAT+ ensure that $\text{lv}(p_i) \geq \text{lv}(x)$ for all $x \in L_i$, and also that $\text{lv}(p_i) \geq \text{lv}(x)$ for all $x \in C$ because we are using the modified prefix $\Pi'_{Abs(C)}$ when making inferences by $\vdash_{1\forall}$ so that every variable at a lower level than any variable in C is existentially quantified.

Induction Hypothesis: We can learn $(C \vee L_i \backslash \{p_i\})$ from $\Pi'\ \Phi \wedge \bigwedge_{j<i}(C \vee L_j \backslash \{p_j\})$ in a short proof using only ATA and universal reduction steps.

Base Case: We know $\Phi \wedge \bar{C} \vdash_1 L_1$ so $\Pi'\ \Phi \wedge \bar{C} \wedge \bar{L}_1 \vdash_1 \bot$ and we can add $(C \vee L_1)$ via ATA. Now p_1 can be removed by universal reduction since $\text{lv}(p_1) \geq_{\Pi'} \text{lv}(x)$ for all $x \in C \cup L_1$. We have learnt $(C \vee L_1 \setminus \{p_1\})$ in two proof steps.

Inductive Step: By the induction hypothesis we have $\Pi'\Phi \wedge \bigwedge_{j<i}(C \vee L_j \backslash \{p_j\})$. For each $j < i$, clearly $(C \vee L_j \backslash \{p_j\}) \wedge \bar{C} \vdash_1 L_j \backslash \{p_j\}$. We also know that $\Phi \wedge \bigwedge_{j<i} L_j \backslash \{p_j\} \wedge \bar{C} \vdash_1 L_i$. Joining these unit propagation inferences together gives that $\Pi'\Phi \wedge \bigwedge_{j<i}(C \vee L_j \backslash \{p_j\}) \wedge \bar{C} \wedge \bar{L}_i \vdash_1 \bot$. We learn $(C \vee L_i)$ and can remove p_i via universal reduction since $\text{lv}(p_i) \geq_{\Pi'} \text{lv}(x)$ for all $x \in C \cup L_i$.

Now we have learnt all the clauses $(C \vee L_i \backslash \{p_i\})$. These new clauses allow us to derive C via ATA. From $\Pi'\Phi \wedge \bigwedge_i (C \vee L_i \backslash \{p_i\}) \wedge \bar{C}$ we can derive each $L_i \backslash \{p_i\}$ using unit propagation, and we know that these together with Φ and $\bar{C}$ are sufficient to derive

$\perp$ by unit propagation. Hence $\Pi'\Phi \wedge \bigwedge_i (C \vee L_i \backslash \{p_i\}) \wedge \bar{C} \vdash_1 \perp$ and we can add C via ATA.

Finally, the clauses L_i and $L_i \setminus \{p_i\}$ are removed using clause deletion so we finish with $\Pi'\Phi \vee C$ as required. □

Similar arguments show that other rules in the QRAT+ refutation system can also be simulated by QRAT.

Lemma 7.2.3. *If C has QRAT+ in $\Pi\Phi$ on l then for every clause $D \in \Phi$ with $\bar{l} \in D$, the outer resolvent $C \cup \{k \in D \mid \text{lv}(k) \leq_\Pi \text{lv}(l), k \neq \bar{l}\}$ can be added to $\Pi\Phi$ via a sequence of polynomial size iterations of ATA and universal reduction.*

Proof. Let $\mathcal{R}_D$ denote the outer resolvent $C \cup \{k \in D \mid \text{lv}(k) \leq_\Pi \text{lv}(l), k \neq \bar{l}\}$. We have that $\Pi'_{Abs(C)}\Phi \wedge \bar{\mathcal{R}}_D \vdash_{1\forall} \perp$ for every $D \in \Phi$ with $\bar{l} \in D$. Let us fix some D and prove we can derive $\mathcal{R}_D$ via ATA and universal reduction steps. Since D is fixed we drop the subscript and let $\mathcal{R} = \mathcal{R}_D$.

Label the clauses immediately prior to universal reduction steps in the $\vdash_{1\forall}$ procedure as L_i (in order, so the first one derived is L_1) with the literal p_i to be removed.

We know that

1. $\Phi \wedge \bar{\mathcal{R}} \vdash_1 L_1$,

2. $\Phi \wedge \bigwedge_{j<i} L_j \setminus \{p_j\} \wedge \bar{\mathcal{R}} \vdash_1 L_i$, for each $i > 1$,

3. $\Phi \wedge \bigwedge_i L_i \setminus \{p_i\} \wedge \bar{\mathcal{R}} \vdash_1 \perp$.

Induction Hypothesis: We can learn $(\mathcal{R} \vee L_i \backslash \{p_i\})$ in a short proof using only ATA and universal reduction steps.

Base Case: We need to add $(\mathcal{R} \vee L_1)$ via ATA, we have that $\Pi'\Phi \wedge \bar{\mathcal{R}} \wedge \bar{L}_1 \vdash_1 \perp$ and therefore we can add $(\mathcal{R} \vee L_1)$ by ATA. We now need to reduce p_1. QRAT+ requires that $\text{lv}(p_i) \geq_{\Pi'} \text{lv}(x)$ for any literal $x \in L_1 \cup C$. Recall that $\mathcal{R}$ contains only literals with level less than or equal to l. Since l belongs to C we therefore have that $\text{lv}(p_i) \geq_{\Pi'} \text{lv}(x)$ for any literal $x \in \mathcal{R}$. Therefore it is valid to derive $(\mathcal{R} \vee L_1 \setminus \{p_1\})$ from $(\mathcal{R} \vee L_1)$ by universal reduction.

Inductive Step: By the induction hypothesis we have $\Pi'\Phi \wedge \bigwedge_{j<i} (\mathcal{R} \vee L_j \backslash \{p_j\})$ and want to add $\mathcal{R} \vee L_i$ via ATA.

For each $j < i$, $(\mathcal{R} \vee L_j \backslash \{p_j\}) \wedge \bar{\mathcal{R}} \vdash_1 L_j \backslash \{p_j\}$, and $\Phi \wedge \bigwedge_{j<i} L_j \backslash \{p_j\} \wedge \bar{\mathcal{R}} \vdash_1 L_i$. Together these give the derivation $\Pi'\Phi \wedge \bigwedge_{j<i} (\mathcal{R} \vee L_j \backslash \{p_j\}) \wedge \bar{\mathcal{R}} \wedge \bar{L}_i \vdash_1 \perp$. We can therefore learn $(\mathcal{R} \vee L_i)$. As before the conditions of QRAT+ and the fact that $\mathcal{R}$ is an outer resolvent mean that $\text{lv}(p_i) \geq_{\Pi'} \text{lv}(x)$ for any literal $x \in L_i \cup C \cup \mathcal{R}$, so $(\mathcal{R} \vee L_i \backslash \{p_i\})$ is derived from $(\mathcal{R} \vee L_i)$ by universal reduction.

For all i we have derived $(\mathcal{R} \vee L_i \backslash \{p_i\})$. $\Phi \wedge \bigwedge_i (\mathcal{R} \vee L_i \backslash \{p_i\}) \wedge \bar{\mathcal{R}} \vdash_1 L_i \backslash \{p_i\}$ for every i, and $\Phi \wedge \bigwedge_i L_i \backslash \{p_i\} \wedge \bar{\mathcal{R}} \vdash_1 \bot$ so we can add $\mathcal{R}$ by ATA. Finally, the intermediate clauses $\mathcal{R} \vee L_j \backslash \{p_j\}$ are removed by the clause deletion rule. □

Lemma 7.2.4. *The QRATA+ step is p-simulated by refutational* QRAT(UR)*.*

Proof. The QRATA+ step derives $\Pi'\Phi \wedge C$ from $\Pi\Phi$ when C has QRAT+ on existentially quantified l with respect to $\Pi\Phi$.

For each $D \in \Phi$ with $\bar{l} \in D$ with O_D the outer clause of D, then we can add $(C \vee O_D)$ via a short proof using ATA and universal reduction rules by Lemma 7.2.3. We can continue adding outer resolvents by this method to Φ even after one or more has already been added since the rules ATA and universal reduction used in Lemma 7.2.3 are not prohibited by the presence of additional clauses.

Now that we have all $(C \vee O_D)$ for every D we need to derive C. This is done by QRATA on l. Let Ω be the set of all outer clauses O_D. Then QRATA requires for each O_D that $\Pi'\Phi \wedge \bigwedge_{O \in \Omega} (C \vee O) \wedge \bar{C} \wedge \bar{O}_D \vdash_1 \bot$. This is clearly true since we can directly refute the clause $(C \vee O_D)$ using $\bar{C} \wedge \bar{O}_D$ in each case.

Once we have derived C we can freely delete all clauses from $\bigwedge_{O \in \Omega} (C \vee O)$. □

Lemma 7.2.5. *The QRATU+ step is p-simulated by refutational* QRAT(UR)*.*

Proof. The QRATU+ step derives $\Pi'\Phi \wedge C$ from $\Pi\Phi \wedge (C \vee l)$ where C has QRAT+ on universal literal l on $\Pi\Phi$.

Let $\Omega = \{O_D \mid O_D$ is the outer clause of some clause $D \in \Phi$ with $\bar{l} \in C\}$. Add $(O \vee C)$ for every $O \in \Omega$ (by Lemma 7.2.3). Now we require that $\Phi \wedge (\bigwedge_{O \in \Omega} (C \vee O)) \wedge \bar{C} \wedge \bar{O}_D \vdash \bot$ for each D with $\bar{l} \in D$. In each case a $(C \vee O)$ clause is refuted directly by $\bar{C} \wedge \bar{O}_D$. This allows to use QRATU to replace $\Pi'\Phi \wedge \bigwedge_{O \in \Omega} (C \vee O) \wedge (C \vee l)$ with $\Pi'\Phi \wedge \bigwedge_{O \in \Omega} (C \vee O) \wedge C$. We can then freely delete all clauses from $\bigwedge_{O \in \Omega} (C \vee O)$. □

Putting these lemmas together we see that any refutation in QRAT+ can be transformed in polynomial time into a refutation in QRAT.

Theorem 7.2.6. *Refutationally,* QRAT *is p-equivalent to* QRAT+*.*

Proof. A QRAT+ proof is a sequence of ATA+, QRATA+, QRATU+, EUR and clause deletion rules. This can be simulated by a QRAT proof, in other words a sequence of ATA, QRATA, QRATU, EUR and deletion steps. EUR and deletion rules remain the same in both systems. In Lemma 7.2.2 we showed that ATA+ steps are simulated by ATA and universal reduction steps, in Lemma 7.2.4 we showed that QRATA+ steps are simulated by ATA, QRATA and universal reduction steps and in Lemma 7.2.5 we showed that QRATU+ steps are simulated by ATA, QRATU and universal reduction steps. We can simulate universal reduction steps by EUR so we can do all this in refutational QRAT. By definition QRAT is p-simulated by QRAT+ also, therefore they are p-equivalent. □

Conclusion

Although QRAT+ enables some inferences to be made more efficiently than in QRAT it does not offer an exponential improvement compared to QRAT in the case of refutations. For satisfaction QRAT, a simulation is made more difficult since clauses added into the formula to simulate $\vdash_{1\forall}$ inferences cannot be removed using the arbitrary clause deletion rule. Since the definition of both the QRAT and QRAT+ properties considers all clauses in the formula it is possible that the presence of additional clauses could block future proof steps. Even in the refutational case, the use of QRAT+ instead of QRAT is still beneficial in providing a more succinct proof and arguably better modelling the practice of QBF solvers, in which universal reduction steps are carried out alongside unit propagation to simplify the formula between the main reasoning steps.

Chapter 8

Proof Size and Proof Width In Variants of Q-Resolution

Since one aim of proof complexity is to prove lower bounds on the size of proofs, we are especially interested in general methods for doing so. A successful approach in propositional proof complexity is based on a famous result of Ben-Sasson & Wigderson (2001).

Proof size is arguably the most important measure in proof complexity. Lower bounds on proof size in Resolution imply lower bounds on the running time of SAT solvers using the CDCL algorithm. The width of a Resolution proof is the maximum number of literals in any clause in the proof. If we know that a formula has a Resolution proof with small width then we can search for proofs with only narrow clauses, which reduces the search space and so bounds running time.

An upper bound on Resolution width w implies an upper bound of $O(n^w)$ on the proof length (where n is the number of variables in the formula, without repetition) due to the possible number of distinct clauses of width at most w. There are also k-CNFs (CNFs with all clauses containing k literals) with width w refutations in Resolution and requiring proofs of size $n^{\Omega(w)}$ (Atserias *et al.*, 2014).

The result of Ben-Sasson and Wigderson showed that whenever a short Resolution refutation exists, a narrow refutation can be constructed from it. Conversely, if every refutation of some family of formulas must contain a clause with large width, then no short refutation can exist. This relationship has been an important technique for proving lower bounds on Resolution proof length. Previously known lower bounds were re-derived using this method, simplifying the argument.

We would like to use the same technique in QBF proof complexity, but unfortunately Beyersdorff *et al.* (2016b) showed that it does not lift to the new context. This chapter adds to their investigation by showing that the original argument can be lifted to QBF refutations in level-ordered Q-Res and relates the proof size in that system to the width of Q-Res refutations. However this holds only for the tree-like systems. We also show negative results for the stronger systems of QU-Res and LD-Q-Res, thus answering a question of Beyersdorff *et al.* (2016b).

8.1 Relating Size and Width Between Two Variants of Q-Resolution

We begin with some definitions. Recall that the size of a Q-Res proof π is written $|\pi|$ and is the number of clauses in π (equivalently, the number of nodes in the associated tree or DAG).

Let Ψ be a formula in CNF or PCNF.

Definition 8.1.1. *The size of deriving a clause C from Ψ in proof system f, denoted $S_{\mathsf{f}}(\Psi \vdash C)$, is the minimum size of any f-proof of C from Ψ.*

We drop the subscripts indicating the proof system under consideration if it is already clear from the context.

Definition 8.1.2. *The width of a clause C is written $w(C)$ and is the number of existentially quantified literals it contains.*

Definition 8.1.3. *The width of Ψ is the maximum width over all clauses in Ψ and is denoted $w(\Psi)$.*

Definition 8.1.4. *The width of a derivation π is the maximum width of any clause contained in π.*

Definition 8.1.5. *The width of deriving a clause C from Ψ in proof system f is written $w_{\mathsf{f}}(\Psi \vdash C)$ and is the minimum width of any derivation of C from Ψ in f.*

Given a CNF Φ, Ben-Sasson & Wigderson (2001) showed that for tree-like propositional Resolution $w(\Phi \vdash \bot) \leq w(\Phi) + \log(S(\Phi \vdash \bot))$ and for DAG-like Resolution, $w(\Phi \vdash \bot) \leq w(\Phi) + O(\sqrt{n \log(S(\Phi \vdash \bot))})$ where n is the number of variables in Φ. The result does not claim that all short proofs are narrow, but that whenever a short proof exists there must also exist a narrow proof of the same formula. We focus here on the tree-like case and show that the same proof can be applied to a level-ordered Q-Res refutation, but that the constructed proof may not remain level-ordered. First we examine the reason that the initial proof must be level-ordered.

Suppose we have a Resolution refutation of some propositional formula Φ. The final step in the proof resolves x and $\neg x$. So we also have a derivation of x from Φ (and also a derivation of $\neg x$ from Φ), that is, Φ implies x (and $\neg x$). A crucial part of the argument rests on the observation that this derivation of x can be easily transformed into a refutation of $\Phi|_{\neg x}$, by simply restricting the proof by the assignment $\{\neg x\}$. This does not hold in general in Q-Res, even in the tree-like case. Indeed, it is possible to have a Q-Res derivation from Ψ to u, a universal variable not at the outermost level in the prefix, where $\Psi|_{\neg u}$ is not even false.

We begin with some observations about level-ordered Q-Res.

Definition 8.1.6. *For a Q-Res proof π an alternative measure of size is the number of lines in π which were not derived by universal reduction, denoted by $||\pi||$.*

Observe that if π is a derivation from Ψ, $|\pi| \leq ||\pi|| \times |\Psi|$ because each clause introduced as an axiom or by a resolution step can be followed by at most one universal reduction step for each universally quantified variable in Ψ. It follows that if $||\pi||$ is polynomially-bounded in $|\Psi|$ then so too is $|\pi|$.

Recall the definition of level-ordered Q-Res.

Definition 8.1.7. *Let π be a Q-Res-refutation of $\Psi = \Pi\Phi$. Then π is level-ordered if for every resolution step $\frac{(C\vee x) \quad (D\vee\neg x)}{(C\vee D)}$ in π we have that $\mathrm{lv}(y) \leq_\Pi \mathrm{lv}(x)$ for all $y \in C \cup D$.*

We will additionally assume that universal reduction steps are carried out as early as possible and in reverse level order. This cannot increase the number of non-reduction proof steps, though it may increase the number of universal reduction steps. The level of a proof step is the level of the pivot. It is straightforward to check that for level-ordered derivations the pivot of the final proof step can be assumed to have maximum level in the proof.

We now show that it is possible to restrict level-ordered proofs to create a new proof as required, which will allow the original construction to be lifted to the new situation.

Lemma 8.1.8. *Let u be a universally quantified literal in the outermost quantifier block of PCNF $\Psi = \Pi\Phi$, and let π be a Q-Res derivation of u from Ψ, then $\pi|_{\bar{u}}$ is a Q-Res refutation of $\Psi|_{\bar{u}}$. If π is level-ordered or tree-like these properties are maintained in $\pi|_{\bar{u}}$.*

Proof. For u outermost and universally quantified, π can only contain one of u or $\bar{u}$. To see why this is true, suppose π derives u and consider the final clause in π that contains $\bar{u}$, $(C \vee \bar{u})$. Clearly $u \notin C$. This clause must be used in a universal reduction step deriving C, which is only possible if there are no existentially quantified literals in the clause, since $\mathrm{var}(u)$ is assumed to be in the outermost block of Π. Universal reduction can never introduce new literals into a clause so C cannot be an ancestor of any clause containing u. Consequently, if π derives u then $\bar{u}$ does not appear in π. It follows that no clause $D \in \pi$ is satisfied by the assignment $\{\bar{u}\}$, so if $D \in \pi$ then

$D|_{\bar{u}} \in \pi|_{\bar{u}}$. Any proof step in π that did not have u as pivot still applies in the restricted proof. If $C \in \pi$ is derived by universal reduction on u then the parent and child of this step are identical in the restricted proof so the latter can be removed. □

Lemma 8.1.9. *Let x be an existentially quantified literal in PCNF Ψ, and π a* Q-Res *derivation of x from Ψ, then $\pi|_{\bar{x}}$ is a refutation of $\Psi|_{\bar{x}}$ in* Q-Res *with weakening. If π is level-ordered or tree-like these properties are maintained in $\pi|_{\bar{x}}$.*

Proof. Suppose $C \in \pi$ is derived by universal reduction from $(C \vee u)$. Then either both of these clauses contain $\bar{x}$ or neither of them do, so if $C|_{\bar{x}}$ is in $\pi|_{\bar{x}}$ then it is derived by universal reduction from $(C|_{\bar{x}} \vee u)$.

Suppose C is derived by resolution on some pivot other than x. If either parent contains $\bar{x}$ then so does C. So if $C|_{\bar{x}}$ is in $\pi|_{\bar{x}}$ then so are the restrictions of both its parents, and $C|_{\bar{x}}$ is derived by resolution on the same pivot.

If $(C \vee D)$ is derived by resolution on x from $(C \vee x)$ and $(D \vee \neg x)$ then the latter clause does not appear in $\pi|_{\bar{x}}$. However, $(C \vee x)|_{\bar{x}} = C$ so $(C \vee D)$ can be derived by weakening from C in $\pi|_{\bar{x}}$. □

The weakening steps can be removed without increase in proof size or width and without affecting the level-ordered and tree-like properties of the proof.

Lemma 8.1.10. *Let Ψ be a PCNF, C a clause, and x an existentially quantified literal in the outermost quantifier block of Ψ. In* Q-Res*, if $w(\Psi|_{\bar{x}} \vdash C) \leq W$ then also $w(\Psi \vdash C \vee x) \leq W + 1$.*

Proof. Let π be a Q-Res derivation of C from $\Psi|_{\bar{x}}$ with width W. Add x into every clause of π. Initial clauses may be obtained by weakening. If C is derived by resolution from A and B then $(C \vee x)$ can be derived from $(A \vee x)$ and $(B \vee x)$ (and neither clause contained $\bar{x}$). If C is derived by universal reduction from A then $(C \vee x)$ is derived by universal reduction from $(A \vee x)$, and since x is outermost it cannot block the reduction step. The width of the derivation is increased by 1. □

For universally quantified u the same argument applies but the width of the derivation is not increased since only existentially quantified literals contribute to the width of a clause. Therefore if $w(\Psi|_{\bar{u}} \vdash C) \leq W$ then also $w(\Psi \vdash C \vee u) \leq W$.

Lemma 8.1.11. *Let Ψ be a PCNF and x an existentially quantified literal in the outermost quantifier block of Ψ. In* Q-Res*, if $w(\Psi|_{\bar{x}} \vdash \bot) \leq W - 1$ and $w(\Psi|_{x} \vdash \bot) \leq W$ then $w(\Psi \vdash \bot) \leq \max\{W, w(\Psi)\}$. For u a universally quantified literal, if $w(\Psi|_{\bar{u}} \vdash \bot) \leq W$ then $w(\Psi \vdash \bot) \leq W$.*

Proof. If x is existentially quantified then since $w(\Psi|_{\bar{x}} \vdash \bot) \leq W - 1$ we have also that $w(\Psi \vdash x) \leq W$ by Lemma 8.1.10. Resolve x with every clause in Ψ containing $\bar{x}$, the resulting collection of clauses are exactly those in the matrix of $\Psi|_x$, and from these we can derive $\bot$ in width W. The total width of the derivation is $\max\{W, w(\Psi)\}$. If u is universally quantified then (u) can be derived in width W and universal reduction derives $\bot$. The total width of the derivation is W. $\square$

We can now state the relation between width in Q-Res and size in level-ordered Q-Res.

Theorem 8.1.12. *$w_Q(\Psi \vdash \bot) \leq w(\Psi) + \lceil \log(S_L(\Psi \vdash \bot)) \rceil$, where Q is* Q-Res *and L is level-ordered tree-like* Q-Res *and Ψ is a PCNF.*

Proof. We begin with a level-ordered refutation π of Ψ. Let $r = \lceil \log(S_L(\Psi \vdash \bot)) \rceil$, therefore $S_L(\Psi \vdash \bot) \leq 2^r$. If $r = 0$ then the empty clause is in Ψ, so $w_Q(\Psi \vdash \bot) = 0$ and we are done.

Otherwise the last step of the proof may be a universal reduction $\frac{x}{\bot}$ or a resolution step $\frac{x \quad \neg x}{\bot}$. The pivot x belongs to the outermost quantifier block that appears in the proof, since the proof is level ordered. Re-ordering the proof to ensure that this is the case cannot increase the number of axiom or resolution steps in the proof. Let $\bar{S}_L(\Psi \vdash \bot)$ denote the minimum number of non-reduction steps in a refutation of Ψ, which clearly must be less than $S_L(\Psi \vdash \bot)$. We will prove inductively that $w_Q(\Psi \vdash \bot) \leq w(\Psi) + \lceil \log(\bar{S}_L(\Psi \vdash \bot)) \rceil$.

In the case of universal reduction, consider π_x, the derivation of x. Then $\pi_x|_{\neg x}$ is a level-ordered refutation of $\Psi|_{\neg x}$ by Lemma 8.1.8. By induction on the number of variables in Ψ we have that $w_Q(\Psi|_{\neg x} \vdash \bot) \leq w(\Psi|_{\neg x}) + \lceil \log ||\pi_x|_{\neg x}|| \rceil$. By Lemma 8.1.11, $\Psi \vdash \bot$ has the same width as the restricted proof, and $w(\Psi|_{\neg x}) = w(\Psi)$, so the result follows.

In the case of resolution being the last step, consider π_x and $\pi_{\neg x}$, the level-ordered derivations of x and $\neg x$. By Lemma 8.1.9 we have that $\pi_x|_{\neg x}$ is a level-ordered refutation of $\Psi|_{\neg x}$ and $\pi_{\neg x}|_x$ is a level-ordered refutation of $\Psi|_x$, and $||\pi|| = ||\pi_x|| + ||\pi_{\neg x}|| + 1$. Without loss of generality, $||\pi_x|| \leq 2^{r-1}$ so by induction on r, there is a (possibly not level-ordered) proof with $w_Q(\Psi|_{\neg x} \vdash \bot) \leq w(\Psi|_{\neg x}) + r - 1 \leq w(\Psi) + r - 1$, and by induction on the number of variables in Ψ, $w_Q(\Psi|_x \vdash \bot) \leq w(\Psi|_x) + r \leq w(\Psi) + r$. Since x is outermost, by Lemma 8.1.11 we can use these two refutations to construct a refutation of Ψ with width at most $w(\Psi) + r$ and the result follows. $\square$

In the proof of Theorem 8.1.12, we begin with a small level-ordered proof and construct another proof from it which has small width. However, during the construction, the proof loses the level-ordered property. It is not possible in general to construct a level-ordered proof with small width.

Proposition 8.1.13. *There is a family of QBFs Φ_n with constant width and $O(n)$ variables such that $S(\Phi_n \vdash \bot) = O(n)$ and $w(\Phi_n \vdash \bot) = \Omega(n)$ in tree-like level-ordered* Q-Res.

Proof. Consider the following family of formulas:

$$\Phi_n = \exists x_1 \ldots x_n \forall z \exists a_1 \ldots a_n, y_0 \ldots y_n$$
$$(\neg y_0) \land (y_n) \land \bigwedge_{i \in [n]} (\neg x_i) \land (z \lor a_i) \land (y_{i-1} \lor \neg a_i \lor x_i \lor \neg y_i)\,.$$

All clauses are needed to refute Φ_n. Any level-ordered proof must carry out all resolution steps on y_i variables before resolving on x_i variables, and it is simple to verify that doing so must result in a clause that contains all x_i variables. There is a short tree-like level-ordered refutation which collapses $(y_{i-1} \lor \neg a_i \lor x_i \lor \neg y_i)$ together to $(\neg a_1 \lor \ldots \lor \neg a_n \lor x_1 \lor \ldots \lor x_n)$, then resolves this with all $(z \lor a_i)$, removes z and finally refutes $\bigwedge_{i \in [n]} (\neg x_i) \land (x_1 \lor \ldots \lor x_n)$, all of which takes linear size. □

Theorem 8.1.12 also cannot be lifted to general level-ordered Q-Res. A crucial part of the argument in the propositional case is to carefully select the next variable to use in restricting the refutation, but it is not possible in general to ensure that this variable belongs to a particular level of the prefix.

8.2 Size and Width for Stronger Proof Systems

It is left open in Beyersdorff *et al.* (2016b) whether the mainly negative picture regarding the size width relationship applies to stronger proof systems as well. We first recall their counter-example for Q-Res and then show that simple modifications to this example, inspired by Balabanov *et al.* (2014), can be used to lift the example to QU-Res and LD-Q-Res.

We use a family of QBFs introduced in Janota & Marques-Silva (2015). First we sketch the intuitive meaning of these formulas, showing why they are false, and then define the QBFs formally.

Given a grid of squares with n rows and columns, suppose that each square must be coloured either black or white so that either there is a row containing all black cells or there is a column containing all white cells. This is easily achieved. However, suppose instead that an adversary intervenes after all the cells have been coloured and specifies which of the two conditions should hold. It is not possible to colour the grid so that there is both a fully black row and a fully white column, so the adversary is always able to observe which condition cannot be satisfied and make an impossible request.

This situation can be encoded in the following QBFs.

$$\exists x_1^1 \dots x_n^1 \dots \dots x_1^n \dots x_n^n \; \forall z \; \exists a_1 \dots a_n, b_1 \dots b_n$$

$$\bigwedge_{i,j=1}^{n} \left(x_i^j \vee z \vee a_i\right) \wedge \bigwedge_{i,j=1}^{n} \left(\neg x_i^j \vee \neg z \vee b_j\right) \tag{8.1}$$

$$\wedge \;\; (\neg a_1 \vee \dots \vee \neg a_n) \wedge (\neg b_1 \vee \dots \vee \neg b_n) \tag{8.2}$$

The x_i^j variables indicate the colouring of the grid. There are n^2 such variables. If $x_i^j = 1$ then cell (i, j) is coloured black, otherwise it is coloured white. If $a_i = 0$ then row i is selected. Similarly, $b_j = 0$ selects column j. The universal variable z chooses whether the row or column condition will be tested. Since z can always be selected to make the choice of either a_i or b_j impossible, the QBF is false.

Modifications of these formulas show that the size-width relation does not hold for tree-like Q-Res, QU-Res and LD-Q-Res.

Proposition 8.2.1 (Beyersdorff *et al.* (2016b)). *There is a family of false QBFs Ψ_n over $O(n^2)$ variables, such that $w(\Psi_n) = 3$ and in tree-like Q-Res $S(\Psi_n \vdash \bot) = n^{O(1)}$, and $w(\Psi_n \vdash \bot) = \Omega(n)$.*

Proof. Consider the following family of formulas.

$$\Psi_n = \exists x_1^1 \dots x_n^1 \dots \dots x_1^n \dots x_n^n \forall z \exists a_1 \dots a_n, b_1 \dots b_n, p_0 \dots p_n, q_0 \dots q_n$$

$$\bigwedge_{i,j=1}^{n} \left(x_i^j \vee z \vee a_i\right) \wedge \bigwedge_{i,j=1}^{n} \left(\neg x_i^j \vee \neg z \vee b_j\right) \tag{8.3}$$

$$\wedge \neg p_0 \wedge \bigwedge_{i=1}^{n} (p_{i-1} \vee \neg a_i \vee \neg p_i) \wedge p_n \tag{8.4}$$

$$\wedge \neg q_0 \wedge \bigwedge_{j=1}^{n} (q_{j-1} \vee \neg b_j \vee \neg q_j) \wedge q_n \tag{8.5}$$

There are $O(n^2)$-size tree-like Q-Res refutations, given by the following procedure:

- Collapse the clauses in (8.4) and (8.5) to $(\neg a_1 \vee \dots \vee \neg a_n)$ and $(\neg b_1 \vee \dots \vee \neg b_n)$. This takes $O(n)$ steps.
- For each j use the clauses $(x_i^j \vee z \vee a_i)$ together with $(\neg a_1 \vee \dots \vee \neg a_n)$ to derive $(x_1^j \vee \ldots \vee x_n^j \vee z)$, then remove z by universal reduction (requiring $O(n)$ steps for each j). From $(x_1^j \vee \ldots \vee x_n^j)$ and the clauses $(\neg x_i^j \vee \neg z \vee b_j)$ derive $(b_j \vee \neg z)$ in $O(n)$ steps for each j.
- The n clauses of the form $(b_j \vee \neg z)$ together with $(\neg b_1 \vee \dots \vee \neg b_n)$ are used to derive $\neg z$, taking $O(n)$ steps. Finally we use universal reduction to reach the empty clause.

To show that any valid Q-Res refutation must be wide we show that the clause immediately following the first universal reduction step in any refutation of Ψ_n has width $\Omega(n)$.

Without loss of generality assume the first universal reduction step in the refutation removes z. Observe that any clause C in the proof that contains z must have some clause $(x_i^j \vee z \vee a_i)$ as an ancestor. Therefore, along the path between this clause and the first universal reduction of z the literal a_i must be removed by resolution. The only clauses in Ψ_n containing $\neg a_i$ are those in (8.4), so the clause $(p_{i-1} \vee \neg a_i \vee \neg p_i)$ must also be in the sub-derivation of C. But now also p_{i-1} and $\neg p_i$ need to be removed before we can perform universal reduction. In this way every clause of (8.4) is required in the sub-derivation of the clause immediately preceding the first universal reduction step and so also $\neg a_i$ appears in this sub-derivation for every value of i.

Since a_i appears positively only in clauses in (8.3), which also contain x_i^j, it is necessary to also include n of these clauses in the sub-derivation, all of which contribute a *different* x_i^j literal. Whenever x_i^j appears negatively it is alongside $\neg z$. By assumption, the sub-derivation cannot include any universal reduction step and therefore also cannot include any clause that contains $\neg z$. So every x_i^j in the sub-derivation must be positive, and there can be no resolution steps to remove them. This demonstrates that prior to the first universal reduction step we must have a clause that contains at least n different x_i^j literals in any valid Q-Res refutation of Ψ_n. ☐

Proposition 8.2.2. *There is a family of false QBFs Ψ'_n over $O(n^2)$ variables, such that $w(\Psi'_n) = 3$ and in tree-like QU-Res $S(\Psi'_n \vdash \bot) = n^{O(1)}$, and $w(\Psi'_n \vdash \bot) = \Omega(n)$.*

Proof. We modify Ψ_n to Ψ'_n by adding another universal variable z' at the same level as z and replacing (8.3) with

$$\bigwedge_{i,j=1}^{n} (x_i^j \vee z \vee z' \vee a_i) \wedge \bigwedge_{i,j=1}^{n} (\neg x_i^j \vee \neg z \vee \neg z' \vee b_j).$$

The size $O(n^2)$ refutation of Ψ_n is extended to a refutation of Ψ'_n by performing a universal reduction step on z' immediately after any universal reduction step on z. Except between each such pair of universal reduction steps this ensures that z' appears in exactly the same clauses as z throughout the proof, and similarly $\neg z$ and $\neg z'$ always appear together. Any resolution step that could be blocked by z' in Ψ'_n would already have been blocked by z in Ψ, so the proof remains valid.

The duplication of the universal variable also ensures that universal resolution cannot result in narrower proofs compared to Q-Res. This is simply because we have ensured that whenever a universal literal appears in a clause C of Ψ'_n, its complement appears only in clauses which also conflict with C on another universal variable, which serves to prevent the use of universal resolution early in the proof.

Any derived clause must contain all of the universal variables from its parents unless it is derived by universal reduction or universal resolution. Therefore universal resolution is forbidden before a universal reduction step has occurred, until this point we may only use existential resolution. The argument above readily applies to show that the clause immediately following the first universal reduction step in any refutation of Ψ'_n must have width $\Omega(n)$. □

The idea of duplicating universal variables can be applied to any formula to prevent universal resolution steps. This allows us to infer a lower bound for QU-Res from any lower bound for Q-Res.

Proposition 8.2.3. *Let $\Psi = \Pi\Phi$ be a closed PCNF and define $\Psi' = \Pi'\Phi'$ so that clause $C' \in \Phi'$ if and only if there is a clause $C \in \Phi$ with $C' = C \cup \{u' \mid u \in C\}$. Π is a sub-prefix of Π' and for every universally quantified variable u in Π there is a universally quantified variable u' in Π' with $\mathrm{lv}(u') = \mathrm{lv}(u)$. Given a QU-Res refutation π' of Ψ' there is a Q-Res refutation π of Ψ such that $|\pi| \leq |\pi'|$ and $w(\pi) \leq w(\pi')$.*

Proof. π' is a sequence of clauses $L_1, \dots, L_m$. Let u be a universally quantified literal in Ψ. We begin by finding the first clause L_i in π' which contains exactly one of u and u'. Without loss of generality we will assume L_i contains u but not u'. Before this point no universal resolution on u' was possible because u and $\neg u$ appear in all clauses with u' and $\neg u'$ and therefore prevent the resolution step. It follows that L_i was derived by universal reduction on u'.

Modify π' by inserting $L_i^2 = L_i \setminus \{u\}$ immediately after L_i. Any subsequent clauses that were derived from L_i are instead derived from L_i^2. No proof step can be blocked. If a clause was derived from L_i by universal resolution with u as the pivot then it can now be derived by weakening from L_i^2, otherwise it can be derived from L_i^2 using the same derivation step as before. No clause in $L_{i+1}, \dots, L_m$ has L_i as a parent, so the same argument applies to show that the next clause to contain only one of u and u' was derived by universal reduction. Again, add a second universal reduction.

Having applied this process to the whole proof the only clauses that contain u but not u' (or vice versa) are immediately preceding a universal reduction step and are not used in any other proof step. It follows that no universal resolution on u or u' is possible in the whole proof. Repeating this construction for all universally quantified variables in Ψ' we generate a proof of Ψ' which contains no universal resolution steps. To construct π simply remove every instance of u' for every universally quantified variable $u \in \Psi$. The additional universal reduction steps are also removed so $|\pi| \leq |\pi'|$. No new existentially quantified variables were introduced into any clause so $w(\pi) \leq w(\pi')$. □

In particular, the formulas defined in Theorem 4.0.1 in Chapter 4 have short (DAG-like) Q-Res refutations but require proofs with large width. As stated in Chapter 4 the formulas themselves contain a wide clause. They can easily be modified so that this clause is broken into several short clauses, in the same way as was done for Ψ_n above but any proof must still contain a wide clause. We can construct similar formulas with small size and large width refutations in QU-Res by doubling the universally quantified variables as demonstrated in Proposition 8.2.2. This shows that the size-width relation also fails for general QU-Res (the formulas Ψ'_n above cannot be used directly as they have a quadratic number of variables.)

Proposition 8.2.4. *There is a family of false QBFs Ψ''_n over $O(n^2)$ variables, such that $w(\Psi''_n) = 3$ and in tree-like LD-Q-Res $S(\Psi''_n \vdash \bot) = n^{O(1)}$, and $w(\Psi''_n \vdash \bot) = \Omega(n)$.*

Proof. In this case Ψ_n is modified by replacing the clauses in line (8.2) with

$$\neg y_0 \wedge \bigwedge_{i=1}^{n} (y_{i-1} \vee \neg a_i \vee \neg y_i \vee z) \wedge y_n \wedge \neg p_0 \wedge \bigwedge_{j=1}^{n} (p_{j-1} \vee \neg b_j \vee \neg p_j \vee \neg z) \wedge p_n.$$

This does not affect satisfiability since these clauses were already only relevant under one or other of the assignments to z. The same $O(n^2)$ refutation of the original formula applies to give the size upper bound, except that we begin by deriving $(\neg a_1 \vee \cdots \vee \neg a_n \vee z)$ and $(\neg b_1 \vee \ldots \neg b_n \vee \neg z)$ instead of $(\neg a_1 \vee \cdots \vee \neg a_n)$ and $(\neg b_1 \vee \cdots \vee \neg b_n)$.

For the width lower bound we show that long-distance resolution cannot be used prior to the first universal reduction step. The only long-distance resolution steps that could be performed would have some x_i^j variable as the pivot with z and $\neg z$ included in the two clauses being resolved. If we assume that this long-distance resolution is occurring before any universal reduction is possible then the derived clause must also contain some a, b, p or q literal.

Suppose the clause contains a_i. In order for the derived clause to form part of the refutation, a_i would need to be removed via resolution at some later point, but before $z*$ can be removed by reduction. This is now impossible. The only input clause that contains $\neg a_i$ also contains z. Inductively, any subsequent clause containing $\neg a_i$ must also contain z because universal reduction is blocked by $\neg a_i$. No such clauses can be resolved on pivot a_i with a clause that contains z^*. Similar arguments apply for all b, p or q literals.

As a result, no long-distance resolution step can apply before at least one universal reduction step. Until then we are restricted to standard resolution steps, and so the clause following the first universal reduction must have width $\Omega(n)$, using the argument of Proposition 8.2.1. □

Conclusion

We have demonstrated that the result of Ben-Sasson & Wigderson (2001) can be lifted to relate two variants of Q-Res, highlighting an interesting relationship between level-ordered and non level-ordered proofs in Q-Res. Removing either the restriction that the proof must be level-ordered, or the restriction that it must be tree-like, is sufficient to lose the desired behaviour. We have also answered the open question from Beyersdorff *et al.* (2016b) regarding extensions of Q-Res, by demonstrating how the counterexamples may be lifted to these stronger calculi.

Chapter 9

A Complexity Gap for QBF Resolution

The Complexity Gap Theorem (Riis, 2001) considers a translation of a first-order sentence ϕ to a sequence of false propositional formulas, and states that the complexity of refuting these propositional formulas in tree-like Resolution depends on whether ϕ has any models. The nth member of the sequence of propositional formulas is satisfiable if and only if ϕ has a model of size n. When ϕ has an infinite model but no finite models then all tree-like Resolution refutations of the related propositional formulas have size exponential in n. When ϕ also has no infinite model then there exist polynomial-size tree-like Resolution refutations of the propositional formulas.

This chapter considers whether the Complexity Gap Theorem holds for QBF extensions of Resolution. We first introduce a method to translate a first-order sentence ϕ to a sequence of QBFs. The translation will ensure that the nth member of the sequence has size polynomial in n, and is true precisely when ϕ has a model of size n.

We demonstrate that tree-like Q-Res will always require exponential-size refutations of these QBFs when ϕ has an infinite model but no finite model. However, unlike the propositional case, there exist formulas with no models but requiring exponential-size Q-Res refutations for the sequence of QBFs. In contrast, for tree-like $\forall$Exp + Res the gap theorem holds as in the propositional case. In this sense, $\forall$Exp + Res does not possess the same deficiency as Q-Res.

9.1 Rendering a First-Order Sentence As a Sequence of QBFs

We give a method to translate a first-order sentence ϕ to a family $\{\Phi_n\}_{n\in\mathbb{N}}$ of QBFs. The method is inspired by the encoding of ϕ into propositional formulas in conjunctive normal form (CNF) previously given in Riis (2001), and is similar to other translations used to encode QCSP instances as QBF (Gent *et al.*, 2004, 2008). We begin with a brief review of first-order logic.

A language of first-order logic is defined by a signature consisting of constants, function symbols, and relation symbols. In Boolean logic we have only two constants, 0 and 1. In first-order logic there are infinitely many constants available. Each function and relation symbol has some arity ≥ 0. The formulas of first-order logic are then defined recursively.

- **Terms**:
  - A variable is a term.
  - A constant is a term.
  - If f is a function of arity $k \geq 0$ and $t_1 \ldots t_k$ are terms then $f(t_1 \ldots t_k)$ is a term.
- **Formulas**:
  - If R is a relation of arity $k \geq 0$ and $t_1 \ldots t_k$ are terms then $R(t_1 \ldots t_k)$ is a formula.
  - If t_1 and t_2 are terms then $t_1 = t_2$ is a formula.
  - If ϕ is a formula then $\neg\phi$ is a formula.
  - If ϕ and ψ are formulas then $(\phi \vee \psi)$ and $(\phi \wedge \psi)$ are formulas.
  - If ϕ is a formula and x a variable then $\exists x\phi$ and $\forall x\phi$ are formulas.

Given a first-order language $\mathcal{L}$, an interpretation consists of a (possibly infinite) collection of elements A and a mapping I_A from constant symbols in $\mathcal{L}$ to members of A, for each relation R in $\mathcal{L}$ there is a relation on A and for each function f in $\mathcal{L}$ there is a function from A to A. Under the usual semantics of $\{\wedge, \vee, \neg, =, \exists, \forall\}$, if a formula ϕ of $\mathcal{L}$ evaluates to true under the interpretation given by $M = (A, I_A)$ then we say M is a model of ϕ. The size of M is the number of elements in A. If a formula has no models it is unsatisfiable.

A formula of first-order logic is in prenex normal form if it has a prefix consisting of quantifiers and the variables bound by them followed by a quantifier free formula. Every first-order formula is equivalent to a formula in prenex normal form. Without loss of generality we assume that there are no free variables since a formula containing a free variable is equivalent to one in which that variable is bound by a universal quantifier in the leftmost prefix block.

Constants are equivalent to functions of arity 0. It is also possible to remove all function symbols from a formula and write an equivalent formula using only relation symbols, since $f(t_1,\dots,t_{k-1}) = t_k$ is equivalent to $R(t_1,\dots,t_k)$ for some relation R. We therefore assume our first-order formulas have the following form:

$$\phi := \mathcal{Q}_1 x_1 \ldots\ldots \mathcal{Q}_k x_k \ \mathcal{D}_1(x_1 \ldots, x_k) \wedge \ldots \wedge \mathcal{D}_r(x_1,\dots,x_k)$$

with $\mathcal{Q}_j \in \{\forall, \exists\}$, and where each $\mathcal{D}_l$ is a disjunction of relations and their negations:

$$R_l^1(x_1,\dots,x_k) \vee \ldots \vee R_l^s(x_1,\dots,x_k).$$

We do not lose significant generality by assuming all extensional relations to be of arity k and all disjunctions to be of width s.

Given a first-order formula ϕ in this form, we want to generate a sequence of QBFs $\{\Phi_n\}_{n\in\mathbb{N}}$ where Φ_n is true exactly when ϕ has a model of size n. We are interested in such sequences where every Φ_n is false, which are generated by first-order formulas which have no models or whose only models are infinite.

Each first-order variable x in ϕ becomes n Boolean variables $x^1,\dots,x^n$. If x^j is made true this represents that x is assigned the value j from the n available constants. We introduce existentially quantified variables associated with a relational predicate $R_l^m(\lambda_1,\dots,\lambda_k)$ indicating that the tuple $(\lambda_1,\dots,\lambda_k)$ is in the relation R_l^m. In ϕ a variable x can only take on one value at a time, and must be given some value. We introduce clauses so that if any existentially quantified variable is not given exactly one value the QBF evaluates to false, and if any universally quantified variable is not given exactly one value then the QBF can be made to evaluate to true.

Let $[n] := \{1,\dots,n\}$. $\sum_{j\in[n]} x^j = 1$ asserts that precisely one of the variables x^j is true, i.e. it is an abbreviation for $\left(\bigvee_{i\in[n]} x^i\right) \wedge \bigwedge_{j\neq i\in[n]}(\neg x^i \vee \neg x^j)$. We can now define our sequence of QBFs.

$$\Psi_n := \exists_{\lambda_1,\dots,\lambda_k\in[n]} R_1^1(\lambda_1,\dots,\lambda_k)\ldots R_r^s(\lambda_1,\dots,\lambda_k) \tag{9.1}$$

$$\mathcal{Q}_1\ x_1^1 \ldots x_1^n \ldots \mathcal{Q}_k\ x_k^1 \ldots x_k^n \tag{9.2}$$

$$\bigwedge_{\{i \mid \mathcal{Q}_i=\forall\}} \sum_{j\in[n]} x_i^j = 1 \rightarrow \bigwedge_{l\in[r]\lambda_1,\dots,\lambda_k\in[n]} (x_1^{\lambda_1} \wedge \ldots \wedge x_k^{\lambda_k}) \rightarrow \mathcal{D}_l(\lambda_1,\dots,\lambda_k) \tag{9.3}$$

$$\wedge \bigwedge_{\{i \mid \mathcal{Q}_i=\exists\}} \sum_{j\in[n]} x_i^j = 1 \tag{9.4}$$

where the notation $\exists_{\lambda_1,\dots,\lambda_k\in[n]} R_1^1(\lambda_1,\dots,\lambda_k)\ldots R_r^s(\lambda_1,\dots,\lambda_k)$ indicates that we existentially quantify over all propositional variables of the form $R_l^m(\lambda_1,\dots,\lambda_k)$ for all tuples with $\lambda_1,\dots,\lambda_k \in [n]$.

By construction, Φ_n is true just in case ϕ has a model of size n. If the disjuncts $\mathcal{D}_l$ contain equality relationships between variables then these can be enforced by restriction of the $\lambda_1,\dots,\lambda_k \in [n]$ and if $\mathcal{D}_l$ only involve some subset of $x_1,\dots,x_k$ then only these need be mentioned in the relevant constraints of line 9.3. We call Boolean variables of the form $R_i^j(\lambda_1, \dots, \lambda_k)$, which are always existentially quantified in the outermost block, *relational variables.*

The quantifier-free part of Φ_n can be expanded to CNF with polynomial increase in size. In particular, line 9.3 is not in CNF but simplifies to a set of clauses of the form

$$\left(\mathcal{D}_l(\lambda_1, \dots, \lambda_k) \vee \bigvee_{i \in k} \neg x_i^{\lambda_i} \vee \bigvee_{\{i \mid \mathcal{Q}_i = \forall\}, j \neq \lambda_i} x_i^j \right)$$

9.2 The Exponential Lower Bound

We begin by showing that when ϕ has infinite models then any tree-like $\forall\mathsf{Exp}+\mathsf{Res}$ refutation of Φ_n has size $2^{\Omega(n)}$. Since $\forall\mathsf{Exp}+\mathsf{Res}$ p-simulates Q-Res in the tree-like case the result immediately transfers to Q-Res as well.

We use the game of Beyersdorff *et al.* (2013) for tree-like Resolution (introduced in Section 5.1) and state a strategy for the Delayer on any QBF generated through the above translation where the underlying first-order formula has an infinite model. The Delayer falsely claims that there is a model of ϕ of size n and responds to the Prover's queries based on information about the real (infinite) models of ϕ. We will assume, for notational simplicity, that all constants in the models being considered are called $c_1, c_2, \dots$, and that a model of size n has constants named $c_1, \dots, c_n$. The Prover and Delayer build up an assignment to the variables of the QBF, and these correspond to statements about the claimed first-order model. If $R_l^m(\lambda_1, \dots, \lambda_k)$ is assigned 1 this is equivalent to stating that the relation R_l^m holds for constants $c_{\lambda_1}, \dots, c_{\lambda_k}$ in the model that the Delayer claims exists for ϕ.

The remainder of the QBF represents the statement that for every assignment to the universally quantified variables of ϕ in $c_1, \dots, c_n$ there is an assignment to the existentially quantified variables from $c_1, \dots, c_n$ so that the constraints in the quantifier free part of ϕ hold, given some specific interpretation of the relations. Other than the relational variables the Prover can query, for an existentially quantified first-order variable x, any of $x^{j,\alpha}$ where α is an assignment to the universally quantified variables in the QBF prior to x^j. As previously noted, $x_i^j = 1$ in Φ_n corresponds to $x_i = c_j$ in ϕ. The annotation α records a (partial) assignment to the universal variables of ϕ, provided that it is "well behaved" in the sense that exactly one of the variables $u^1, \dots, u^n$ evaluates to 1 under α. In this case α indicates the value that u is assigned in ϕ. If $x^{j,\alpha}$ is assigned 1 this is equivalent to stating that in the claimed model of ϕ, when the universal assignments are made according to α, x is assigned the value c_j. Initially we will assume that α is indeed well behaved and will discuss what happens otherwise below. The assignment to variables of ϕ that corresponds to α will be denoted by $\tilde{\alpha}$. As usual, the annotation can only contain assignments for universal variables prior to x in the quantifier prefix.

Since there is no model of ϕ of size n, queries made by the Prover will eventually reveal a contradiction, with some clause of Φ_n being falsified. At this point the game ends, but we seek to show that the Delayer can ensure a score of $\Omega(n)$ points before this happens.

The Delayer's Strategy

Let $\mathcal{M}$ be the set of all models of ϕ. At any point in the game there is a record of QBF variables that have been assigned. For $x^{j,\alpha} = b$ in this record ($b \in \{0,1\}$) we define

$$\phi(\tilde{\alpha}, x^j, b) = \begin{cases} \phi|_{\tilde{\alpha}, x=c_j} \text{ if } b = 1, \\ (\phi|_{\tilde{\alpha}} \wedge x \neq c_j) \text{ if } b = 0. \end{cases}$$

Let $\mathbf{\Phi}$ denote the conjunction of all $\phi(\alpha, x^j, b)$ for the assignments $x^{j,\alpha} = b$ which have been made so far in the game.

We say that a model $M \in \mathcal{M}$ agrees with the current state of the game if the relations hold between the constants of M as specified by assignments to the relational variables of Φ_n, and M is a model of $\mathbf{\Phi}$. The subset of $\mathcal{M}$ that agrees with the current state of the game is denoted $\widetilde{\mathcal{M}}$.

We now return to the question of how to handle annotations that are not well behaved. Suppose we have such an assignment, α so there is some universally quantified variable u such that either all $u^1, \ldots, u^n$ are assigned 0 in α or at least two of them are assigned 1. Notice that if constraint D_l depends on universally quantified u in ϕ then every constraint in Φ_n that relates to D_l contains $(\neg u^j \vee \bigvee_{j' \neq j} u^{j'})$ for some $j \in [n]$. $\forall\mathsf{Exp}+\mathsf{Res}$ works on an expanded formula, and in the part of the expansion for α all the clauses related to D_l are satisfied. All that remains is the part of Φ_n relating to constraints in ϕ that do not depend on u, and the side conditions specifying that existentially quantified variables must be given one value. As such, if α is badly behaved with respect to u we set $\tilde{\alpha}$ to not assign any value to u and work with $\phi(\tilde{\alpha}, x^j, b) := \phi'(\tilde{\alpha}, x^j, b)$ where ϕ' is ϕ with all constraints D_l that depend on u removed. The Prover can't benefit from making such queries but would always get more information (so constrain $\widetilde{\mathcal{M}}$ more) by using a well behaved universal assignment in the annotations.

When the Prover makes a query on a relational variable $R_l^m(\lambda_1, \ldots, \lambda_k)$ the Delayer considers all $M \in \widetilde{\mathcal{M}}$. If all M agree that the relationship R_l^m holds for constants $c_{\lambda_1}, \ldots, c_{\lambda_k}$ then the Delayer responds with weights $p_1 = 1$, $p_0 = 0$. The Prover is forced to assign the variable to 1 and the Delayer scores no points. If all M agree that R_l^m does not hold for constants $c_{\lambda_1}, \ldots, c_{\lambda_k}$ then the Delayer responds with weights $p_1 = 0$, $p_0 = 1$. Again, the variable assignment is forced and the Delayer scores no points. If the models disagree then the Delayer responds with weights $p_1 = \frac{1}{2}$, $p_0 = \frac{1}{2}$. In this case the Delayer scores one point when the Prover decides the assignment.

When the Prover makes a query on $x^{j,\alpha}$, again the Delayer considers all $M \in \widetilde{\mathcal{M}}$, and if they all agree that when the universal variables of ϕ are set according to $\tilde{\alpha}$ then x is assigned c_j then the Delayer responds with weights $p_1 = 1, p_0 = 0$. If all the models agree that when the universal variables of ϕ are set according to $\tilde{\alpha}$ then x is not assigned c_j then the Delayer responds with weights $p_1 = 0, p_0 = 1$. If the models disagree then the Delayer responds with weights $p_1 = \frac{1}{2}, p_0 = \frac{1}{2}$ and the Delayer scores one point.

Lemma 9.2.1. *Using the strategy described above, the Delayer can only lose the game by violating a clause stating that for some set of existential variables $\{x^j\}_{j=1}^n$, exactly one must be set to true.*

Proof. Because we are following models that satisfy ϕ, each such model must satisfy every clause of the QBF except where the QBF makes a direct statement about the size of the model. At every stage of the game there remain infinite models that agree with everything stated so far, and for which x has some value under any universal assignment in ϕ, but that value may fall outside of the n elements permitted by the QBF. The statements that reference the size of the model are those stating that exactly one variable from each existentially quantified set $\{x^j\}_{j=1}^n$ must be true (i.e. that the assignment to variable x in the original sentence must correspond to one of the n elements in the universe). For the same reason, the clause will be violated because all variables are assigned 0, never because more than one is assigned 1 as no model can assign multiple values to x. $\square$

As a result of this, at least n variables in the QBF must be assigned a value in order for the Delayer to lose the game, and these variables must between them reference all n of the elements in the universe. The next idea is that a large proportion of these n assignments score a point for the Delayer – although perhaps not directly. Every query in the game on Φ_n makes reference to some of the n constants in the claimed model. Queries on relational variables $R_l^m(\lambda_1, \dots, \lambda_k)$ reference the constants $c_{\lambda_1}, \dots, c_{\lambda_k}$. Queries on $x^{j,\alpha}$ reference c_j and every constant used in $\tilde{\alpha}$.

Lemma 9.2.2. *At least $n - k$ of the n constants are referenced in a query for which the Delayer scores a point.*

Proof. For some existentially quantified variable x, the game ends because all of $\{x^{j,\alpha}\}_{j=1}^n$ are assigned to 0. Then the Prover must query $x^{j,\alpha}$ for all j at some point. Recall that k is the number of variables in ϕ and the maximum arity of all relations. The universal assignment α references $k' < k$ of the constants, without loss of generality suppose these are $c_1, \dots, c_{k'}$. Consider c_j for some $j > k'$. Suppose that nothing assigned by a Prover choice so far in the game has referenced c_j. By construction, there is at least one infinite model M that agrees with the choices made so far in the game, and this must assign some value to x given $\tilde{\alpha}$, although this value may be outside of $c_1, \dots, c_n$. However, there is no information from the queries made so far in the game which can distinguish c_j from any other elements in M which have not been referenced in a query decided by the Prover. In particular c_j cannot be distinguished from the constants outside of $c_1, \dots, c_n$. It follows that there must be a model in $\widetilde{\mathfrak{M}}$ in which x is assigned c_j in response to $\tilde{\alpha}$ (by relabelling the constants of M). Therefore, when $x^{j,\alpha}$ is queried, either c_j has already been referenced by a query which scored one point, or else this query can score one point. □

Lemma 9.2.3. *The Delayer scores $\Omega(n)$ points by the given strategy.*

Proof. At least $n - k$ constants must be referenced in a query that scores a point. Each query can only reference at most k constants. Therefore at least $\frac{n-k}{k}$ points are scored. k is fixed, it does not depend on n, so this gives $\Omega(n)$ points. □

Theorem 9.2.4. *Let ϕ be a first-order sentence which has an infinite model but no finite model. Then any tree-like $\forall$Exp+Res refutation of QBF Φ_n, representing the statement that there is a model for ϕ of size n, has size $2^{\Omega(n)}$.*

The result immediately transfers to tree-like Q-Res as well.

9.3 A Surprising Lower Bound

The Delayer's strategy above relies on having some model that satisfies ϕ, so clearly it cannot apply when ϕ has no models. If the gap theorem holds then we expect that when ϕ has no models there are polynomial-size refutations for the sequence $\{\Phi_n\}_{n\in\mathbb{N}}$. This does not hold for Q-Res.

Proposition 9.3.1. *Let $\theta := \forall x \exists y \forall z \exists u \forall v \exists w \; R(x, y, z) \wedge \neg R(u, v, w)$ and $\{\Theta_n\}_{n\in\mathbb{N}}$ be the sequence of QBFs expressing that θ has a model of size n. Although θ has no models, any tree-like Q-Res refutation of Θ_n must have size $2^{\Omega(n)}$.*

We have already shown an exponential lower bound on the size of tree-like Q-Res refutations of Θ_n in Chapter 5, Lemma 5.2.1. The formulas stated there are slightly different from those given by our translation in order to make the presentation cleaner, but the difference is only superficial.

Firstly, the QBFs in Chapter 5 lack any variables corresponding to z in θ, this is because these variables only appear in clauses where they are innermost and could be immediately removed by universal reduction in Q-Res. Including these variables does not affect the argument of Lemma 5.2.1 since the Delayer ensures that all relevant clauses are satisfied however the z variables are assigned.

Secondly, our translation includes clauses to ensure that no existentially quantified variable from θ can be assigned more than one value at a time, these do not appear in the formulas in Chapter 5. Intuitively, if x is existentially quantified in ϕ we can see that it would never be in the existential player's interest to set more than one x_i to 1 in Φ_n anyway since the x_i variables only appear positively in the clause $(x^1 \vee \cdots \vee x^n)$.

We can add a rule which says that as soon as one existential variable is assigned 1 then all other variables in that set are assigned 0, and if all but one are assigned 0 then the final variable in the set should be assigned 1. This rule will apply when it is not in conflict with the existing rules.

It is straightforward to check that the game still ends because some existential variable from θ has not been assigned a value. We need to show that the Prover cannot use the new rules to reduce the overall score for the Delayer.

Suppose that the game ends with $(w^1 \vee \cdots \vee w^n)$ falsified. In the previous round, without loss of generality, we had $w^1, \ldots, w^s$ assigned to 0 and $w^{s+1}, \ldots, w^n$ unassigned. If $s = n-1$ then w^n is forced to take the value 1. If $s = n-2$ then the Prover can make a query on w^{n-1} but this simply returns us to the previous situation. In order to achieve $w^i = 0$ for all i we would need at least two such w^i to be assigned 0 in the same round, before the new rule can be applied. However, it is not possible for a single Prover query to cause two values of w^i to be assigned 0 according to the original rules and v cannot be assigned while any w^i has a value assigned to it. Therefore, the clause $(w^1 \vee \cdots \vee w^n)$ cannot be made to evaluate to false without first forgetting the current assignments to all w^i and making new queries which will result in all w^i being forced by the original rules to take the value 0 in a single round.

The same reasoning applies if the game ends by falsifying the clauses $(y^1 \vee \cdots \vee y^n)$ or $(u^1 \vee \cdots \vee u^n)$.

We also see that although an existentially quantified variable can now be forced to take the value 1 this never happens for free. Since the most recent assignment of any of $v^1, \ldots, v^n$, when all w^i were necessarily unassigned, either at least one point has been scored in order to set some w^i to 1 or $\Omega(n)$ points have previously been scored in the game so that all $w^{i'}$ for $i' \neq i$ are forced to take the value 0 which then forces w^i to be assigned 1. Overall the Delayer still gains $\Omega(n)$ points during the game.

We have shown in Lemma 5.2.2 that Θ_n have polynomial-size refutations in $\forall\mathsf{Exp}+\mathsf{Res}$. The idea in this example can be generalised to give a polynomial-size upper bound for $\forall\mathsf{Exp}+\mathsf{Res}$ refutations whenever ϕ has no models.

9.4 The Polynomial Upper Bound

We show that when ϕ has no models the Prover can use a refutation of ϕ as a framework for a polynomial-size refutation in $\forall\mathsf{Exp}+\mathsf{Res}$. We will use a refutation of ϕ, specifically we assume an analytic tableau refutation. We begin by briefly introducing this proof method for first-order logic.

The principle is to break ϕ down into smaller components, creating a tree which represents it. Each node is labelled with some formula. The nodes of a single branch represent a conjunction. The different branches of the tree represent a disjunction. If two modes on a branch are and immediate contradiction (i.e. a formula A and its negation $\neg A$) then the formula represented by this branch has been proved false. The branch is said to be closed. To show that ϕ is false, every branch of the tableau must be closed.

The tree is built according to the following rules, each of which allows one or several nodes to be added as descendants of the current leaves of the tree. New nodes can only be added at the leaves, and nodes cannot be removed.

- **Axiom**: A new node can be added to any branch and labelled with a conjunct from ϕ.
- **And**: When a branch contains a node labelled $(A \wedge B)$ a new node labelled A and/or a new node labelled B can be added to this branch.
- **Or**: When a branch contains a node labelled $(A \vee B)$ we can create two children of the current leaf of that branch, thus splitting it in two. One of the new nodes is labelled A and the other is labelled B.
- **Universal**: When a branch contains a node labelled $\forall uA$ we can add a new node to this branch labelled $A[u'/u]$ where u' is a new free variable not in ϕ or anywhere in the tableau.
- **Existential**: When a branch contains a node labelled $\exists xA$ we can add a new node to this branch labelled $A[f(u'_1,\dots,u'_k)/x]$ where f is a new function symbol not appearing in ϕ or anywhere in the tableau and u'_i are the free variables in A. The new function f is a Skolem function for x.

Finally, a unification defines a substitution to the free variables of the tableau such that every branch contains an immediate contradiction. Each substitution could be a constant or a Skolem function.

This proof method is complete. If a first-order formula ϕ is false then there is a tableau proof of this (although it may not be straightforward to find it). We will assume that the Prover has access to a tableau refutation of ϕ.

Theorem 9.4.1. *Let ϕ be a first-order sentence without any models. Then the sequence of QBFs $\{\Phi_n\}$ have polynomial-size tree-like refutations in $\forall$Exp + Res.*

Proof. We use the unification witnessing that ϕ is false to structure a strategy for the Prover in the Prover-Delayer game of Beyersdorff *et al.* (2013). The unification is made up of substitutions to the free variables by constants or Skolem functions. Each Skolem function was introduced for some existential variable x from ϕ and is evaluated for an assignment to (a subset of) the universal variables of ϕ that appeared prior to x in the quantifier prefix.

Consider a line in the unification of the form $f(u'_1, \dots, u'_t)/y'$, in which each u'_i (and y') is a free variable introduced to the tableau in place of universally quantified variable u_i from ϕ, and f is a Skolem function introduced to the tableau in place of existentially quantified variable x from ϕ. If u'_i has already appeared earlier in the unification then it has a value $b_{u_i} \in [n]$ already assigned to it. All other universally quantified variables in ϕ that appear before x in the quantifier prefix can be arbitrarily assigned a value. Let $\tilde{\alpha}$ be the assignment to first-order universally quantified variables prior to x in the prefix of ϕ that sets $u_i = b_{u_i}$ if b_{u_i} is defined and otherwise sets each variable to c_1. Then α is the QBF assignment equivalent to $\tilde{\alpha}$, i.e. if $u = c_j \in \tilde{\alpha}$ then $u^j \in \alpha$ and $\neg u^{j'} \in \alpha$ for all $j' \neq j$. The Prover queries the value of x with the annotation α. Because x has been split into $x^1 \dots x^n$ by the translation to QBF the Prover will in fact query some or all of the $x^{i,\alpha}$ until one of them is made true. Then all other $x^{i,\alpha}$ would be forced to false so the Prover can move on to the next assignment in the unification.

Each line from the unification is handled in this way, in order, and the responses are remembered. Earlier responses can affect later queries by affecting the setting of α.

Once the assignments for the unification have been made the Prover can query relational variables to quickly reach a contradiction. Each branch of the tableau contains two entries that are directly contradictory under the unification. For each branch in turn the Prover queries the relation(s) that close the branch using the assignments determined in the first stage of the game. By construction, any sequence of Delayer answers results in an immediate contradiction and the QBF is made to evaluate to false. In particular, some clause in line 9.3 of the generated QBF is falsified.

We need to show that the Delayer scores $O(\log(n))$ points. The number of queries of the relational variables does not depend on n. For each query the Prover can select the value that gives Delayer the lowest score so each choice has a maximum value of 1 point (which occurs when $p_0 = p_1 = \frac{1}{2}$). This part gives the Delayer a constant score. There are also constantly many Skolem functions in the unification. Each of these requires (up to) n queries to assign a value to an (annotated) existential variable but we will show that the number of points remains limited to $O(\log(n))$ points.

To assign a value to x^{α} the Prover queries the $x^{i,\alpha}$ in order until one of them is assigned 1.

Induction Hypothesis: Immediately after querying $x^{j,\alpha}$ for each value of $j = 1, \dots n-1$ the Delayer has either scored $\log(n)$ points and some $x^{i,\alpha} = 1$ or has scored $\log(n) - \log(n-j)$ points and all $x^{i,\alpha} = 0$.

Base Case: The Prover first queries $x^{1,\alpha}$. The Delayer responds with weights p_0 and p_1. If $p_1 \geq \frac{1}{n}$ then set $x^{1,\alpha} = 1$ so Delayer scores at most $\log(n)$ points. Otherwise set $x^{1,\alpha} = 0$. In this case $p_0 > \frac{n-1}{n}$ so Delayer scores at most $\log(\frac{n}{n-1})$ points.

Inductive Step: Over $x^{i,\alpha}$ for $i = 1, \dots, j$ Delayer has either scored $\log(n)$ points and some $x^{i,\alpha} = 1$ or has scored $\log(n) - \log(n-j)$ points and all $x^{i,\alpha} = 0$. If some $x^{i,\alpha} = 1$ then we are done and Prover does not need to query $x^{j+1,\alpha}$. Otherwise the Prover sets $x^{j+1,\alpha} = 1$ if $p_1 \geq \frac{1}{n-j}$. The Delayer scores at most $\log(n-j)$ points for this, giving a total of $\log(n)$ points. If $p_1 < \frac{1}{n-j}$ then Prover sets $x^{j+1,\alpha} = 0$ and Delayer scores at most $\log(\frac{n-j}{n-j-1})$ points for this, giving a total of $\log(n) - \log(n-(j+1))$ points.

If all $x^{i,\alpha} = 0$ for $i = 1, \dots n-1$ then at the query for $x^{n,\alpha}$ simply choose the value that gives the Delayer the least score. If $x^{n,\alpha}$ is set to 0 then the game ends. In either case, the Delayer scores at most 1 point.

In total the Delayer has scored $O(\log(n))$ points, so the proof size is $n^{O(1)}$. $\square$

The reason that this approach fails in Q-Res is that the assignments to existentially quantified variables must be forgotten when resetting the universal assignment, so it is not possible to remember responses for lines early in the unification and apply them later. The contradiction from querying relational variables requires that all earlier responses can be recalled.

Conclusion

We have shown that PCNFs generated from first-order formulas by a natural translation exhibit a complexity gap in tree-like $\forall$Exp + Res, but that tree-like Q-Res is unable to achieve the polynomial upper bound in the case that the first-order formula has no models. This highlights a weakness in the universal reduction rule which prevents Q-Res from mimicking the refutation of the first-order proof as is possible in both $\forall$Exp + Res and when the first-order formula is directly translated into propositional logic.

Chapter 10

Conclusion

The main theme of this thesis has been comparing different approaches to reasoning about universally quantified variables in QBF proof systems that act on formulas in prenex conjunctive normal form, and we have demonstrated some consequences of the relative weakness of universal reduction.

Resolution with Universal Reduction and Expansion Through a careful study of several QBF systems based on Resolution, we have answered some open questions and shown some new connections between them. We have also given some theoretical context for empirically observed behaviour of different QBF solvers. We showed that for tree-like proofs, Q-Res and QU-Res are p-equivalent, and $\forall$Exp + Res is strictly stronger than both. The separating example can be generalised somewhat and also shows that stronger extensions of Q-Res cannot p-simulate even tree-like $\forall$Exp + Res.

Similarly when restricted to formulas with bounded quantifier complexity we have shown that (general, DAG-like) Q-Res and QU-Res are p-equivalent and are p-simulated by $\forall$Exp + Res. Prior to this it was known that QU-Res can give polynomial-size proofs for the formulas KBKF_n, which require exponential-size proofs in Q-Res, and it is clear from the definitions that QU-Res p-simulates Q-Res. The results in this thesis show that the situation in which universal resolution adds sufficient power for such a separation to exist is very limited. Not only must the formula have unbounded quantifier complexity, but also the proof must not be tree-like and the universal resolution steps must interact in a specific way. Even when such a separation is possible we can make a simple modification to any formula which has an exponential lower bound for proof size in Q-Res to create a QBF that is similarly hard for QU-Res. Given these results, we can see that from a theoretical point of view QU-Res offers minimal advantages over Q-Res.

For QBFs with bounded quantifier complexity we have shown that $\forall$Exp + Res p-simulates Q-Res but not LD-Q-Res. This highlights the importance of better understanding the relationship between QCDCL solvers and these two proof systems. It would also be interesting to investigate whether these results hold when the proof systems are augmented with dependency schemes.

It is somewhat surprising to find such a large class of formulas for which $\forall$Exp + Res is strictly stronger than Q-Res. Although they are both based on Resolution these two systems relate to different solving paradigms and were thought to exhibit orthogonal strengths. Again, the result shows that there are very specific requirements for a QBF that has short Q-Res proofs but requires large $\forall$Exp + Res proofs. Whether there are natural situations in which Q-Res p-simulates $\forall$Exp + Res is open. It is also open whether a separation exists between IR-calc and $\forall$Exp + Res for QBFs with bounded quantifier complexity. At present the only known formulas with polynomial-size proofs in IR-calc and requiring exponential-size proofs in $\forall$Exp + Res have unbounded quantifier complexity. However, the method used here does not immediately apply.

QRAT and QRAT+ We have shown that the proof system QRAT does not admit strategy extraction from refutations unless P = PSPACE. This is in contrast to QRAT proofs of true QBFs, which do admit extraction of Skolem functions. We have also shown that the cause of this difficulty is the extended universal reduction rule. When limited to using the standard universal reduction rule then refutational QRAT does allow extraction of Herbrand functions. Further, simply adding universal expansion without the full power of extended universal reduction is sufficient to prevent strategy extraction for QRAT. This shows that strategy extraction is closely linked to the type of universal reasoning used by a QBF proof system. We have formalised this connection by showing that a QBF proof system which includes universal expansion can only have strategy extraction when the underlying propositional proof system has feasible interpolation. We have also shown that QRAT+, which extends the QRAT rule by allowing inference by universal reduction as well as unit propagation, does not change this picture since QRAT+ and QRAT are p-equivalent.

An interesting open question is whether QRAT for true QBFs could be augmented with some new or stronger rule which would prevent extraction of Skolem functions. The cause of the asymmetry between the refutational and satisfaction variants of the proof system is not fully understood, but could be related to the inherent asymmetry of a CNF representation. Since one purpose of QRAT is to simulate current QBF solving and pre-processing techniques, which do admit strategy extraction, we could also investigate a restriction of QRAT which is still able to simulate all of these techniques, including universal expansion, but without losing strategy extraction, i.e. by restricting the power of the underlying propositional reasoning.

Applying Propositional Results to QBF Systems We revisited the relationship between the size and width of proofs in Resolution, lifting existing counter-examples to stronger systems. By carefully showing how the original proof of Ben-Sasson and Wigderson can be applied to Q-Res we have given some explanation beyond the counter-examples of why this important result fails to lift to the QBF setting.

Finally, we have show that the complexity gap theorem fails to lift to Q-Res and that this is directly related to a weakness in universal reduction which is not present in universal expansion. It is open whether dependency schemes may help to overcome this, and whether the gap result holds for extensions of Q-Res such as LD-Q-Res.

References

Atserias, A., Lauria, M. & Nordström, J. (2014). Narrow proofs may be maximally long. In *2014 IEEE 29th Conference on Computational Complexity (CCC)*, 286–297. 104

Audemard, G. & Simon, L. (2018). On the Glucose SAT solver. *International Journal on Artificial Intelligence Tools*, **27**, 1840001. 2

Balabanov, V. & Jiang, J.H.R. (2011). Resolution proofs and Skolem functions in QBF evaluation and applications. In *Computer Aided Verification (CAV)*, 149–164. 72

Balabanov, V. & Jiang, J.H.R. (2012). Unified QBF certification and its applications. *Formal Methods in System Design*, **41**, 45–65. 5, 31, 66, 85

Balabanov, V., Widl, M. & Jiang, J.H.R. (2014). QBF resolution systems and their proof complexities. In *Theory and Applications of Satisfiability Testing (SAT)*, 154–169. 109

Beame, P., Kautz, H.A. & Sabharwal, A. (2004). Towards understanding and harnessing the potential of clause learning. *Journal of Artificial Intelligence Research (JAIR)*, **22**, 319–351. 31

Beame, P., Beck, C. & Impagliazzo, R. (2012). Time-space tradeoffs in resolution: superpolynomial lower bounds for superlinear space. In *ACM Symposium on the Theory of Computing (STOC)*, 213–232. 4

Beek, P.V. (2006). Backtracking search algorithms. In *Handbook of Constraint Programming*, 85–134. 31

Ben-Sasson, E. & Nordström, J. (2011). Understanding space in proof complexity: Separations and trade-offs via substitutions. In *Innovations in Computer Science (ICS)*, 401–416. 4

Ben-Sasson, E. & Wigderson, A. (2001). Short proofs are narrow - resolution made simple. *Journal of the ACM*, **48**, 149–169. 4, 7, 104, 105, 113

Benedetti, M. & Mangassarian, H. (2008). QBF-based formal verification: Experience and perspectives. *JSAT*, **5**, 133–191. 4

Beyersdorff, O. & Clymo, J. (2018). Relating size and width in variants of Q-resolution. *Information Processing Letters*, **138**, 1–6. 7

Beyersdorff, O. & Kullmann, O. (2014). Unified characterisations of resolution hardness measures. In *Theory and Applications of Satisfiability Testing (SAT)*, 170–187. 4, 59

Beyersdorff, O. & Pich, J. (2016). Understanding Gentzen and Frege systems for QBF. In *Logic in Computer Science (LICS)*. 58

Beyersdorff, O., Galesi, N. & Lauria, M. (2013). A characterization of tree-like resolution size. *Information Processing Letters*, **113**, 666–671. 59, 60, 118, 124

Beyersdorff, O., Chew, L. & Janota, M. (2014). On unification of QBF resolution-based calculi. In *Mathematical Foundations of Computer Science (MFCS)*, 81–93. 28

Beyersdorff, O., Chew, L. & Janota, M. (2015). Proof complexity of resolution-based QBF calculi. In *Symposium on Theoretical Aspects of Computer Science (STACS)*, 76–89. 5, 31, 33, 45, 48, 53, 54, 57, 67, 69, 72, 73

Beyersdorff, O., Bonacina, I. & Chew, L. (2016a). Lower bounds: From circuits to QBF proof systems. In *ACM Conference on Innovations in Theoretical Computer Science (ITCS)*, 249–260. 29, 31, 58, 71

Beyersdorff, O., Chew, L., Mahajan, M. & Shukla, A. (2016b). Are short proofs narrow? QBF resolution is not simple. In *Symposium on Theoretical Aspects of Computer Science (STACS)*, 15:1–15:14. 104, 105, 109, 110, 114

Beyersdorff, O., Chew, L., Mahajan, M. & Shukla, A. (2017a). Feasible interpolation for QBF resolution calculi. *Logical Methods in Computer Science (LMCS)*, **13 (2)**. 5, 73, 94

Beyersdorff, O., Chew, L. & Sreenivasaiah, K. (2017b). A game characterisation of tree-like Q-resolution size. *Journal of Computer and System Sciences*, 82–101. 4, 60, 61

Beyersdorff, O., Chew, L., Mahajan, M. & Shukla, A. (2018). Understanding cutting planes for QBFs. *Information and Computation*, **262**, 141–161. 88

Beyersdorff, O., Chew, L., Clymo, J. & Mahajan, M. (2019). Short proofs in QBF expansion. In *Theory and Applications of Satisfiability Testing (SAT)*, 19–35. 7

Beyersdorff, O., Blinkhorn, J. & Mahajan, M. (2020). Hardness characterisations and size-width lower bounds for QBF resolution. In *Symposium on Logic in Computer Science (LICS)*, 209–223. 7

Biere, A., Lonsing, F. & Seidl, M. (2011). Blocked clause elimination for QBF. In *Conference on Automated Deduction CADE*, 101–115. 32

Biere, A., Fazekas, K., Fleury, M. & Heisinger, M. (2020). CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling entering the SAT Competition 2020. In *SAT Competition 2020*, 51–53. 2

Bonet, M.L. & Galesi, N. (2001). Optimality of size-width tradeoffs for resolution. *Computational Complexity*, **10**, 261–276. 4

Chew, L. & Clymo, J. (2019). The equivalences of refutational QRAT. In *Theory and Applications of Satisfiability Testing (SAT)*, 100–116. 7

Chew, L. & Clymo, J. (2020). How QBF expansion makes strategy extraction hard. In *Automated Reasoning*, 66–82. 7

Chvátal, V. & Szemerédi, E. (1988). Many hard examples for resolution. *Journal of the ACM*, **35**, 759–768. 3

Cook, S.A. (1971). The complexity of theorem-proving procedures. In *Symposium on Theory of Computing (STOC)*, 151–158. 1

Cook, S.A. & Reckhow, R.A. (1979). The relative efficiency of propositional proof systems. *The Journal of Symbolic Logic*, **44**, 36–50. 1, 21, 23

Craig, W. (1957a). Linear reasoning. a new form of the Herbrand-Gentzen theorem. *The Journal of Symbolic Logic*, **22**, 250–268. 73

Craig, W. (1957b). Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory. *The Journal of Symbolic Logic*, **22**, 269–285. 93

Davis, M. & Putnam, H. (1960). A computing procedure for quantification theory. *Journal of the ACM*, **7**, 210–215. 13, 22

Davis, M., Logemann, G. & Loveland, D.W. (1962). A machine program for theorem-proving. *Communications of the ACM*, **5**, 394–397. 3, 13

Egly, U., Lonsing, F. & Widl, M. (2013). Long-distance resolution: Proof generation and strategy extraction in search-based QBF solving. In *International Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR)*, 291–308. 72

Egly, U., Kronegger, M., Lonsing, F. & Pfandler, A. (2014). Conformant planning as a case study of incremental QBF solving. In *Artificial Intelligence and Symbolic Computation (AISC)*, 120–131. 4

Gent, I.P., Nightingale, P. & Rowley, A.G.D. (2004). Encoding quantified CSPs as quantified Boolean formulae. In *European Conference on Artificial Intelligence (ECAI)*, 176–180. 116

Gent, I.P., Nightingale, P., Rowley, A.G.D. & Stergiou, K. (2008). Solving quantified constraint satisfaction problems. *Artificial Intelligence*, **172**, 738–771. 116

Giunchiglia, E., Narizzano, M. & Tacchella, A. (2004). Qube++: An efficient QBF solver. In *Formal Methods in Computer-Aided Design*, 201–213. 4

Giunchiglia, E., Narizzano, M. & Tacchella, A. (2006). Clause/term resolution and learning in the evaluation of quantified Boolean formulas. *Journal of Artificial Intelligence Research (JAIR)*, **26**, 371–416. 32

Goultiaeva, A., Van Gelder, A. & Bacchus, F. (2011). A uniform approach for generating proofs and strategies for both true and false QBF formulas. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 546–553. 72, 95

Haken, A. (1985). The intractability of resolution. *Theoretical Computer Science*, **39**, 297–308. 3

Heule, M., Seidl, M. & Biere, A. (2014a). Efficient extraction of Skolem functions from QRAT proofs. In *Formal Methods in Computer-Aided Design*, 107–114. 73, 76

Heule, M., Seidl, M. & Biere, A. (2014b). A unified proof system for QBF preprocessing. In *International Joint Conference on Automated Reasoning (IJCAR)*, vol. 8562, 91–106, Springer. 72, 77

Heule, M.J., Seidl, M. & Biere, A. (2017). Solution validation and extraction for QBF preprocessing. *Journal of Automated Reasoning*, **58**, 97–125. 77

Heule, M.J.H., Kullmann, O. & Marek, V.W. (2016). Solving and verifying the Boolean Pythagorean triples problem via cube-and-conquer. In *Theory and Applications of Satisfiability Testing (SAT)*, 228–245. 2

Janota, M. (2016). On Q-resolution and CDCL QBF solving. In *Theory and Applications of Satisfiability Testing (SAT)*, 402–418. 32

Janota, M. & Marques-Silva, J. (2015). Expansion-based QBF solving versus Q-resolution. *Theoretical Computer Science*, **577**, 25–42. 5, 28, 32, 33, 34, 58, 61, 65, 109

Janota, M., Klieber, W., Marques-Silva, J. & Clarke, E.M. (2012). Solving QBF with counterexample guided refinement. In *Theory and Applications of Satisfiability Testing (SAT)*, 114–128. 4

Kautz, H.A. & Selman, B. (1992). Planning as satisfiability. In *European Conference on Artificial Intelligence (ECAI)*. 2

Kiesl, B. & Seidl, M. (2019). QRAT polynomially simulates ∀Exp+Res. In *Theory and Applications of Satisfiability Testing (SAT)*, 193–202. 73

Kiesl, B., Heule, M.J.H. & Seidl, M. (2017). A little blocked literal goes a long way. In *Theory and Applications of Satisfiability Testing (SAT)*, 281–297. 73

Kleine Büning, H., Karpinski, M. & Flögel, A. (1995). Resolution for quantified Boolean formulas. *Information and Computation*, **117**, 12–18. 24, 47, 48

Klieber, W., Sapra, S., Gao, S. & Clarke, E.M. (2010). A non-prenex, non-clausal QBF solver with game-state learning. In *Theory and Applications of Satisfiability Teasting (SAT)*, 128–142. 4

Krajíček, J. (1997). Interpolation theorems, lower bounds for proof systems and independence results for bounded arithmetic. *The Journal of Symbolic Logic*, **62**, 457–486. 73, 93

Larrabee, T. (1992). Test pattern generation using Boolean satisfiability. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **11**, 4–15. 2

Liang, J.H., Oh, C., Ganesh, V., Czarnecki, K. & Poupart, P. (2016). MapleCOMSPS, MapleCOMSPS LRB, MapleCOMSPS CHB. *SAT Competition*, 52. 2

Lonsing, F. & Biere, A. (2010). DepQBF: A dependency-aware QBF solver. *JSAT*, **7**, 71–76. 4, 18, 58

Lonsing, F. & Egly, U. (2018a). Evaluating QBF solvers: Quantifier alternations matter. In *Principles and Practice of Constraint Programming (CP)*, 276–294. 36, 46

Lonsing, F. & Egly, U. (2018b). QRAT+: generalizing QRAT by a more powerful QBF redundancy property. In *International Joint Conference Automated Reasoning (IJCAR)*, 161–177. 97, 99

Marques Silva, J.P. & Sakallah, K.A. (1996). GRASP - a new search algorithm for satisfiability. In *International Conference On Computer Aided Design (ICCAD)*, 220–227. 3, 14

Mironov, I. & Zhang, L. (2006). Applications of SAT solvers to cryptanalysis of hash functions. In *Theory and Applications of Satisfiability Testing (SAT)*, 102–115. 2

Peitl, T., Slivovsky, F. & Szeider, S. (2019). Dependency learning for QBF. *Journal of Artificial Intelligence Research*, **65**, 181–208. 18

Pipatsrisawat, K. & Darwiche, A. (2011). On the power of clause-learning SAT solvers as resolution engines. *Artificial Intelligence*, **175**, 512–525. 31

Pitassi, T., Beame, P. & Impagliazzo, R. (1993). Exponential lower bounds for the pigeonhole principle. *Computational Complexity*, **3**, 97–140. 3

Pudlák, P. (1997). Lower bounds for resolution and cutting planes proofs and monotone computations. *The Journal of Symbolic Logic*, **62**, 981–998. 3, 73, 93

Pudlák, P. (2000). Proofs as games. *American Math. Monthly*, 541–550. 59

Pudlák, P. & Impagliazzo, R. (2000). A lower bound for DLL algorithms for SAT. In *Proc. 11th Symposium on Discrete Algorithms*, 128–136. 3, 59

Pulina, L. & Seidl, M. (2017). Competitive evaluation of QBF solvers (2017). `http://www.qbflib.org/event_page.php?year=2017`. 46

Rabe, M.N. & Tentrup, L. (2015). CAQE: A certifying QBF solver. In *Formal Methods in Computer-Aided Design*, 136–143. 4

Reckhow, R.A. (1976). *On the lengths of proofs in the propositional calculus*. Ph.D. thesis, University of Toronto. 23

Riis, S. (2001). A complexity gap for tree-resolution. *Computational Complexity*, **10**, 179–209. 7, 115, 116

Robinson, J.A. (1965). A machine-oriented logic based on the resolution principle. *Journal of the ACM*, **12**, 23–41. 22

Samer, M. & Szeider, S. (2009). Backdoor sets of quantified Boolean formulas. *Journal of Automated Reasoning*, **42**, 77–97. 18

Slivovsky, F. & Szeider, S. (2014). Variable dependencies and Q-resolution. In *Theory and Applications of Satisfiability Testing (SAT)*, 269–284. 79

Slivovsky, F. & Szeider, S. (2015). Quantifier reordering for QBF. *Journal of Automated Reasoning*, 459–477. 18

Soos, M., Nohl, K. & Castelluccia, C. (2009). Extending SAT solvers to cryptographic problems. In *Theory and Applications of Satisfiability Testing (SAT)*, 244–257. 2

Tseitin, G.C. (1968). On the complexity of derivations in propositional calculus. In A.O. Slisenko, ed., *Studies in Mathematics and Mathematical Logic, Part II*, 115–125. 12

Urquhart, A. (1987). Hard examples for resolution. *Journal of the ACM*, **34**, 209–219. 3

Van Gelder, A. (2012). Contributions to the theory of practical quantified Boolean formula solving. In *Principles and Practice of Constraint Programming*, 647–663. 26, 47

Wimmer, R., Reimer, S., Marin, P. & Becker, B. (2017). HQSpre - an effective preprocessor for QBF and DQBF. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, 373–390. 32

Zhang, L. & Malik, S. (2002). Conflict driven learning in a quantified Boolean satisfiability solver. In *International Conference on Computer-Aided Design*, 442–449. 27

Zhang, W. (2014). QBF encoding of temporal properties and QBF-based verification. In *International Joint Conference on Automated Reasoning (IJCAR)*, 224–239. 4