QBF Proof Complexity

by

Leroy Nicholas Chew

Submitted in accordance with the requirements
for the degree of Doctor of Philosophy

The University of Leeds
School of Computing
May 2017

Declarations

The candidate confirms that the work submitted is his/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

List of Publications

1. Olaf Beyersdorff, Leroy Chew, and Karteek Sreenivasaiah. A Game Characterisation of Tree-like Q-resolution Size. To appear in *Journal of Computer and System Sciences*.
2. Olaf Beyersdorff, Leroy Chew, Meena Mahajan, and Anil Shukla. Feasible Interpolation for QBF Resolution Calculi. To appear in *Logical Methods in Computer Science*.

Refereed Contributions in Conference Proceedings

List of Publications

3. Olaf Beyersdorff, Leroy Chew, Meena Mahajan, and Anil Shukla. Understanding Cutting Planes for QBFs. In *Proc. IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'16)*, pages 40:1–40:15. LIPIcs, 2016.
4. Olaf Beyersdorff, Leroy Chew, Renate A. Schmidt, and Martin Suda. Lifting QBF Resolution Calculi to DQBF. In *Proc. Theory and Applications of Satisfiability Testing International Conference (SAT'16)*, pages 490–499. Springer, 2016.
5. Olaf Beyersdorff, Leroy Chew, Meena Mahajan, and Anil Shukla. Are Short Proofs Narrow? QBF Resolution is not Simple. In *Proc. Symposium on Theoretical Aspects of Computer Science (STACS'16)*. pages 15:1–15:14. LIPIcs, 2016.
6. Olaf Beyersdorff, Ilario Bonacina, and Leroy Chew. Lower Bounds: From Circuits to QBF Proof Systems. In *Proc. ACM Conference on Innovations in Theoretical Computer Science (ITCS'16)*, pages 249–260. ACM, 2016.
7. Olaf Beyersdorff, Leroy Chew, Meena Mahajan, and Anil Shukla. Feasible Interpolation for QBF Resolution Calculi. In *Proc. International Colloquium on Automata, Languages, and Programming (ICALP'15)*, pages 180–192. Springer, 2015.
8. Olaf Beyersdorff, Leroy Chew, and Karteek Sreenivasaiah. A Game Characterisation of Tree-like Q-resolution Size. In *Language and Automata Theory and Applications (LATA'15)*, pages 486–498. Springer, 2015.

9. Olaf Beyersdorff, Leroy Chew, and Mikoláš Janota. Proof Complexity of Resolution-Based QBF Calculi. In *Proc. Symposium on Theoretical Aspects of Computer Science (STACS'15)*, pages 76–89. LIPIcs, 2015.
10. Olaf Beyersdorff, Leroy Chew, and Mikoláš Janota. On Unification of QBF Resolution-Based Calculi. In *Proc. International Symposium on Mathematical Foundations of Computer Science (MFCS'14)*, pages 81–93, 2014.

Refereed Contributions in Workshop Proceedings

List of Publications

11. Olaf Beyersdorff, Leroy Chew, and Mikoláš Janota. Extension Variables in QBF Resolution. In *Proc. Beyond NP, Papers from the AAAI Workshop*, AAAI Press, 2016.

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

- Chapter 4 contains work from [10]. I was the main author. The proofs of this paper were a result of discussion including all authors.
- Chapter 5 and 6 contain work from [9]. I was the main author. The proofs of this paper were a result of discussion including all authors.
- Chapter 7 contains work from [1,8] ([8] is the conference version of [1]). I was the main author. The proofs of this paper were a result of discussion including all authors.
- Chapter 8 contains work from [2,7] ([7] is the conference version of [2]). Beyersdorff, Mahajan and Shukla, provided the feasible interpolation argument for LQU+-Res, I provided the feasible interpolation for IRM-calc. In Chapter 8, the theorem for LQU+-Res is referenced only.
- Chapter 9 contains work from [5]. Shukla was the main author. The proofs of this paper were a result of discussion including all authors. This chapter only references the work in this paper without proof.
- Chapter 10 contains work from [4]. I was the main author. The proofs of this paper were a result of discussion including all authors.
- Chapter 11 contains work from [11]. I was the main author. The proofs of this paper were a result of discussion including all authors.
- Chapter 12 contains work from [3]. Shukla was the main author. I contributed Theorem 19. This chapter only references the work in this paper without proof.
- Chapter 13 contains work from [6]. I was the main author. The proofs of this paper were a result of discussion including all authors.

Acknowledgements

I would like to take this opportunity to express my great indebtedness to my main supervisor Olaf Beyersdorff, who first persuaded me to undertake a PhD with him which is a decision I have never regretted. He has helped me throughout the entire process by introducing me to key contacts, collaborators and important papers to read and review in addition to locating and organising venues where I was able to talk about my work. Olaf has helped me familiarise myself with many new aspects of research; his supervision was beyond excellent.

I am particularly grateful to Judi Drew since much of this research has been dependent upon her help with all of my arrangements regarding travel and expenses.

I would also like to express special thanks to Alicja Piotrkowicz, who has painstakingly read through my thesis drafts and provided me with a wealth of valuable feedback. I would further like to wish her luck with her own thesis.

Regarding my co-authors, I have to express my gratitude. Not only did Mikoláš Janota help me understand QBF and solving, but our conversations laid the foundations of the problems tackled in this thesis. Among other things, Karteek Sreenivasaiah helped me with my writing style and proofs. Working in India with Meena Mahajan and Anil Shukla was invaluable since their enthusiasm for our project and their insights in complexity helped advance important developments in the research topic. Ilario Bonacina helped me produce some of my best results. Martin Suda and Renate Schmidt provided some needed insights into our work and its relations to first order logic. Moreover, I would like to thank Uwe Egly and Pavel Pudlák who I have also had detailed discussions with about proof complexity.

Thanks to all the staff at the School of Computing, to my second supervisor Kristina Vuskovíc and to John Stell who reviewed my work at the end of the year. I would also like to acknowledge my proof complexity colleagues: Jan Pich, Luke Hinde, Josh Blinkhorn, Sarah Sigley and Jude Clymo. I was fortunate to have shared my workspace with others who were working on similar problems to me as this meant I had people around to discuss complexity with. Thanks is also due to the many other research students who I have spoken with over the years about research and my thesis: Sam, Matt, Elaine, Christian, Aryana, Bernhard, Harriet, Jacob, Carla and Anja. Every conversation has helped me get this far.

Furthermore, I would like to recognise the support that the many friends I have made over the years at the various groups and societies I have joined in Leeds have given me. Thank you for giving structure to my time outside of work and expanding my education outside of my academic field.

Finally, I am grateful to my family, particularly my parents who have throughout my entire life supported my academic progress through personal and financial sacrifices.

Dedicated to the memory of my grandparents
Elsie and Geoffrey Chew

Abstract

Quantified Boolean Formulas (QBF) and their proof complexity are not as well understood as propositional formulas, yet remain an area of interest due to their relation to QBF solving. Proof systems for QBF provide a theoretical underpinning for the performance of these solvers. We define a novel calculus IR-calc, which enables unification of the principal existing resolution-based QBF calculi and applies to the more powerful Dependency QBF (DQBF).

We completely reveal the relative power of important QBF resolution systems, settling in particular the relationship between the two different types of resolution-based QBF calculi. The most challenging part of this comparison is to exhibit hard formulas that underlie the exponential separations of the proof systems. In contrast to classical proof complexity we are currently short of lower bound techniques for QBF proof systems. To this end we exhibit a new proof technique for showing lower bounds in QBF proof systems based on *strategy extraction*. We also find that the classical lower bound techniques of the *prover-delayer* game and *feasible interpolation* can be lifted to a QBF setting and provide new lower bounds.

We investigate more powerful proof systems such as *extended resolution* and *Frege systems*. We define and investigate new QBF proof systems that mix propositional rules with a reduction rule, we find the strategy extraction technique also works and directly lifts lower bounds from circuit complexity. Such a direct transfer from circuit to proof complexity lower bounds has often been postulated, but had not been formally established for propositional proof systems prior to this work.

This leads to strong lower bounds for restricted versions of QBF Frege, in particular an exponential lower bound for QBF Frege systems operating with $\mathsf{AC}^0[p]$ circuits. In contrast, any non-trivial lower bound for propositional $\mathsf{AC}^0[p]$-Frege constitutes a major open problem.

Table of Contents

List of Figures

List of Notations

AC^0	class of problems decidable by constant-depth polynomial size circuits
$\mathsf{AC}^0[p]$	class of problems decidable by AC^0 circuits with mod p gates
$\mathcal{B}_{\mathrm{v}}$	bounded variables
coNP	class of problems whose complement is in NP
dom	set of domain variables
$\forall$Exp+Res	for-all expansion plus resolution, a refutation system for QBF
EXPTIME	class of problems decidable in exponential time
FO-res	first order resolution
Frege	Frege proof systems
$\mathcal{F}_{\mathrm{v}}$	free variables
G	QBF analogue of LK
G_i	G with cuts restricted to prenex $\Sigma_i^q \cup \Pi_i^q$-formulas
$G(t)$	$\mathrm{KBKF}(t)$ or $\mathrm{KBKF}_{\mathrm{lq}}(t)$
h	index of the positive in derivation clause of KBKF
iff	if and only if
ind	index level
inst	instantiation function
IR-calc	QBF calculus with instantiation and resolution
IRM-calc	QBF calculus with instantiation, resolution and merging
LK	Gentzen's sequent calculus
LQPARITY	a modified QPARITY formula for hardness in LD-Q-Res
LD-Q-Res	long-distance Q-resolution
LQU^+-Res	QU-resolution with long distance universal and existential resolution
lv	quantifier level
max	the function that returns the maximum value of a set
min	the function that returns the minimum value of a set
mod	modulo
NC^1	class of logarithmic depth and bounded fan-in circuits
NEXPTIME	class of problems decidable in non-deterministic exponential time
NP	class of problems decidable in non-deterministic polynomial time
PARITY	function that returns true if and only if an odd number of variables are 1
P	class of problems decidable in polynomial time
P/poly	class of problems decidable by non-uniform polynomial size circuits
PSPACE	class of problems decidable in polynomial space
PSUM_i	partial sum of x_j variables up to $j = i$
QPARITY	a false formula with Skolem functions that are parity

Q-Res	Q-resolution, a refutation system for quantified boolean formulas
QUPARITY	a modified QPARITY formula for hardness in LQU$^+$-Res
QU-Res	Q-resolution with universal resolution
RES	resolution
$\mathsf{restrict}_l(\sigma)$	restriction of annotation σ to universal literals left of l
s_f	the minimum proof size function for proof system f
SAT	the propositional satisfiability problem
TAUT	the propositional tautology problem
TC^0	class of problems decidable by AC^0 circuits with threshold gates
var	the function taking formulas to their set of variables
w	width
z	last 0 variable in the prover delayer game for KBKF
$*$	merged complementary values
$\bot$	contradiction
$\circ$	completion operator
$\forall$	universal quantifier
$\forall$Exp+Res	universal expansion plus resolution calculus
$\forall$-Red	universal reduction
$\forall$-Red*	universal reduction for merged literals
$\exists$	existential quantifier
$\equiv_p$	is p equivalent to
$\triangleq$	defined as equal to
Γ	arbitrary alphabet
Γ^*	set of all strings of Γ
$\leq_p$	is p-simulated by
$\mathbb{N}$	set of natural numbers
$\mathbb{Z}$	set of integers
$\mathbb{Z}^+$	set of positive integers
Π_i^P	The universal class on the ith level of the polynomial hierarchy
Π_i^q	QBF with i quantifier levels with leading $\forall$ quantifier
Σ_i^P	The existential class on the ith level of the polynomial hierarchy
Σ_i^q	QBF with i quantifier levels with leading $\exists$ quantifier
$\varnothing$	the empty set
$\vdash_P$	syntactic entailment under proof system P
$\models$	semantic entailment

Acronyms

CDCL	Conflict Driven Clause Learning
CEGAR	Counterexample Guided Abstraction Refinement
CNF	Conjunctive Normal Form
CP	Cutting Planes
DAG	Directed Acyclic Graph
D-IR	DQBF Instantiation Resolution
DNF	Disjunctive Normal Form
DPLL	The algorithm of Davis, Putney, Logemann and Loveland
DQBF	Dependency Quantified Boolean Formulas
FO	First-Order logic
IR	Instantiation Resolution
IRM	Instantiation Resolution Merge
KBKF	the formulas of Kleine Büning, Karpinski and Flögel
L$\forall$R	Long-distance Universal Resolution
L$\exists$R	Long-distance Existential Resolution
QBF	Quantified Boolean Formulas
QCDCL	Quantified boolean formula version of the Conflict Driven Clause Learning algorithm
QDPLL	QBF version of the algorithm of Davis, Putnam, Logemann and Loveland
QRAT	Quantified boolean formula Resolution Asymmetric Tautology
S$\forall$R	Short-distance Universal Resolution
S$\exists$R	Short-distance Existential Resolution
w.l.o.g.	without loss of generality

Part I

Background

Chapter 1

Introduction

In this chapter we will introduce the general background and give an overview of our contributions, detailing the relevant chapters.

1.1 Background

Proof Complexity. Computational complexity is an important field in both theoretical and practical computer science. The most studied topic is understanding the *running time* of *algorithms*. *Proof complexity* is structured in a similar fashion. However, instead of algorithms, the focus is on *proofs* and, instead of time, the main measure is *proof size*. Proof complexity is relevant to a wide range of logics, but typically the focus is on *propositional logic*.

An area in logic that is linked to proof complexity is *satisfiability solving (SAT)*. The SAT problem is as follows: a statement with variables is written in purely propositional formal logic, and it needs to be decided whether there is any consistent 0/1 assignment to the variables. SAT is inextricably linked to non-deterministic polynomial-time decision problems (NP). The SAT problem is the canonical NP-complete problem [42] and thus SAT is linked to long-standing complexity open problems such as P vs NP (deterministic vs non-deterministic polynomial-time). The programs that decide whether an instance of the SAT problem is satisfiable or not are called *SAT solvers*. The idea is that since SAT is NP-hard, other NP decision problems can be feasibly transformed into SAT problems. Therefore it makes sense from a practical point of view to focus specifically on making fast and memory efficient SAT solvers. Modern SAT solvers such as MiniSat [51], MapleCOMSPS [87] and Riss [89] compete on huge instances of SAT, with up to millions of variables.

Resolution is a remarkably simple inference tool based on case analysis. It forms the main rule of the resolution proof system, which is the best understood proof system. The motivation behind so many investigations into resolution is that proof size in resolution corresponds to running time for the most common type of SAT solvers.

From a proof complexity perspective, resolution is considered as a weak system, witnessed by the wealth of proof size lower bounds (cf. [108] for a survey). Lower bounds in stronger proof systems have been a part of ongoing investigations for many decades and bring about some of the most important open problems in proof complexity.

Another important proof complexity connection is to complexity class problems. The famous class NP, has an alternative natural definition in proof complexity [45]. Open complexity problems such as NP vs coNP and NP vs PSPACE can be seen as natural proof complexity problems.

Finally, proof complexity is linked to ‘bounded arithmetic’. In complexity theory, first order theories of arithmetic correspond to the levels of the *polynomial hierarchy* of complexity classes. A

neat connection can be made with proof complexity. Propositional proofs can be translated to first order formulas, where provability in bounded arithmetic translates to short size proofs. (see [15] for a survey).

Quantified Boolean Formulas. There are strong links between formal logic and complexity. The fact that SAT is NP-complete, means that NP as a whole can be studied just by investigating SAT. *Quantified Boolean Formulas (QBF)*, which augment propositional logic with *Boolean quantifiers*, are important because they also have a strong relationship with another complexity class. Just as propositional SAT is considered as the canonical NP-complete problem, QBF truth or falsity is considered as the canonical PSPACE-complete decision problem, where PSPACE is the class of all languages decidable in a polynomial amount of required memory. Furthermore, restrictions on the quantifiers in QBF give rise to languages that are complete for the appropriate classes in the polynomial hierarchy.

Consequently, the above properties make QBF very important in complexity theory. This also gives it practical importance. Due to its PSPACE completeness, QBF can express formulas more succinctly than SAT and thus applies to further fields such as formal verification or planning [14,104]. These advantages broaden the amount the computational tasks that we may wish to rewrite as QBFs compared to that of propositional logic. We wish to advance understanding of algorithms that find solutions to QBF problems by better understanding QBF proofs. This is similar to the approach in propositional proof complexity to propositional satisfiability (SAT).

SAT solving has been found to be tremendously successful [91], despite the NP-hardness of the underlying problem. QBF is seen as one of many possible "next steps" after SAT. Hence, due to the successes of SAT solving, QBF has seen an increase in interest over the past 15 years. Currently, many QBF solvers such as DepQBF [88], RAReQS [72], GhostQ [78], and CAQE [101], to name but a few, compete on thousands of QBF instances. However, QBF solvers are still significantly behind in terms of the performance compared to SAT solvers. QBFs introduce the additional difficulty of having to deal with quantification, as well as solver-dependent differences in *how*. Two separate paradigms exists in QBF solving. Firstly, QBF solvers can take the state-of-the-art *Conflict Driven Clause Learning (CDCL)* technique from SAT solving, and use a 'reduction' rule to deal with quantification. On the other hand, QBF solvers can take a different approach where quantifier expansion can remove the quantification in order to use SAT-based reasoning on the formulas.

Prior to this thesis, a handful of proof systems existed for QBF. Much like how runs of existing SAT solvers correspond to lengths of resolution proofs, modern QBF solvers correspond to QBF resolution systems. CDCL style solvers, such as DepQBF, correspond to the QBF resolution system Q-resolution (Q-Res) [77] and its variants. However for expansion based solvers, such as RAReQS, a correspondence to Q-Res was not found. A basic expansion based proof system $\forall$Exp+Res was developed for this reason [74]. In comparison to proposition resolution, these QBF resolution systems were not well understood. Firstly, the relationship between CDCL style and expansion proof systems were not properly understood. Secondly, QBF should present an additional source

of difficulty, however there was a lack of QBF examples giving size lower bounds in proofs. Propositional resolution has an important advantage here, it has a wealth of *lower bound techniques.* In QBF there were only a few results [11, 36, 74, 116] based on ad-hoc methods, rather than general techniques.

1.2 Contributions

We will highlight our main theorems and contributions in this section.

QBF Resolution. The first contribution is a better understanding of the relationship between CDCL and expansion-based QBF solving. We do this through the framework of proof complexity. We design a new system– IR-calc, that incorporates both CDCL and expansion-based proofs. IR-calc is actually an improvement on an earlier system – ∀Exp+Res. The key improvement is that *partial expansion* is used instead of full expansion. Since IR-calc uses a very simple improvement to ∀Exp+Res, it remains machine amenable. The hope is, that, by having a system like IR-calc that unifies CDCL and expansion, solvers that operate in a similar way to IR-calc can be motivated. Ideally, these would have the advantages of both types of solver.

Furthermore, towards this task we show that CDCL in QBF (QCDCL) and expansion have mutual advantages. Indeed, there are formulas that have short proofs in expansion based proof systems but not in QCDCL proofs and vice versa.

While CDCL and expansion is an insightful dichotomy to focus on, it is not the only source of variation in QBF calculi. Q-Res has a number of improvements such as QU-Res [116], LD-Q-Res [10, 119] and LQU$^+$-Res [11]. These mainly deal with partially lifting some of the very tight restrictions in Q-Res. We show that an improved IR-calc known as IRM-calc can capture the additional power of one of these– namely LD-Q-Res.

We can summarize the calculi and their strengths in the Hasse diagram in Figure 1. The calculi are arranged by their power to *simulate* other systems, marked in Figure 1 by a line. This means that proofs can be transferred to the higher proof system (vertically higher in Figure 1) without a super-polynomial blow-up in size. We actually find that these are the entire simulations due to the existence of *separations* where we find a family of formulas with polynomial size proofs in one system but only super-polynomial size proof in another.

These calculi also form the basis of an investigation into the more expressive logic of *Dependency Quantified Boolean Formulas (DQBF)* where the ordering of the quantifiers is made explicit. QBFs form a special case of these, namely when the ordering is linear. The results of these investigations find that most of these calculi are either incomplete or unsound in the DQBF setting. However, our new calculus IR-calc is both sound and complete in DQBF due to its motivation from first order logic.

DQBF is connected back again to regular QBF solving and proof systems. This manifests as dependency schemes, which detect the dependencies in the linear order that are spurious, i.e. whether the dependency can be removed and preserve soundness. This relates to a few of the calculi

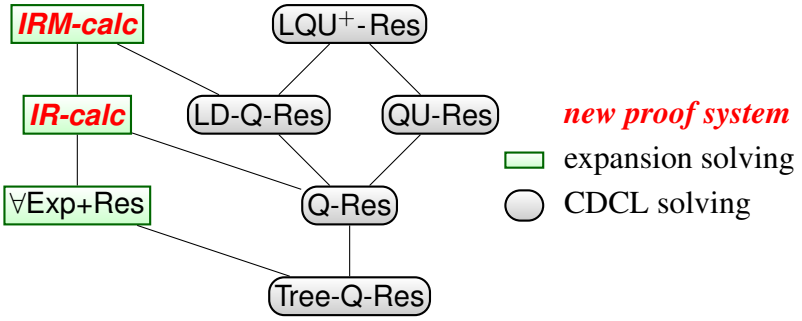


Fig. 1. Relative strength of QBF resolution calculi

that we have not mentioned here such as $Q(D)$-Resolution [111] which takes into a consideration dependency scheme D. The QRAT proof system from [70] also appears to use a dependency scheme in its reduction rule. These are omitted from the comparison because we believe dependency schemes are an orthogonal approach worthy of its own separate full investigation.

Lower Bound Techniques. Lower bounds are important in proof complexity. They can be used to highlight the relative strengths of the proof systems. There is also a vital practical importance to solvers, especially where calculi are directly related. We typically care about super-polynomial lower bounds, where a theorem cannot be proven in a proof that is bounded by some polynomial.

Since propositional logic is a sub-case of QBF, propositional lower bounds have an effect on QBF lower bounds, indeed lower bounds in resolution count as lower bounds in QBF resolution systems. This do not really illuminate on difficulty in QBF and do not provide separations between the QBF resolution systems.

As discussed earlier, propositional resolution has many lower bound techniques. We find that some of these classical techniques work in a QBF setting. The first of these is a two player game between a Prover and a Delayer. This was adapted from a working technique for propositional resolution [27–29]. A false QBF is chosen and the Delayer tries to score points while the Prover tries to refute the QBF and end the game. The logarithm of the score corresponds to the proof size in tree-like Q-Res and QU-Res.

Theorem 20. *If QBF ϕ has a tree-like QU-Resolution proof of size at most s, then there exists a Prover strategy such that any Delayer scores at most $\lg\lceil\frac{s}{2}\rceil$ points.*

We also show that *feasible interpolation* [80], which is one of the few examples of proof lower bound techniques that uses circuit complexity, also works for certain classes of QBFs. Feasible interpolation looks at implication formulas and allows interpolating circuits to be extracted from the proofs in polynomial time. Hence any implication with only exponential-size interpolating circuits must have exponential-size proofs. Feasible interpolation in a QBF setting differs from the propositional setting because quantification allows more complicated formulas to be used as lower bounds.

Theorem 30. *All QBF resolution calculi in Figure 1 have the feasible interpolation property.*

However, not every technique can be lifted from propositional logic. An example would be the most important technique from propositional resolution which is the size-width relationship [13]. This takes a second measure, the width, which is the maximum size of a proof line and shows that the size grows exponentially in that measure. While size-width technique does hold in some QBF tree-like calculi, this relation is refuted from holding in the main QBF systems, even using a improved notion of existential width [22].

Theorem 42 (Beyersdorff, Chew, Mahajan, Shukla [22]). *There is a family of false 3-PCNFs with linear size Q-resolution proofs and linear width.*

In addition to these lifted techniques, we introduce a new technique only available in QBF– *strategy extraction*. This technique is rather remarkable and involves using lower bounds from circuit complexity. In particular we use the class AC^0 which is the class of problems expressible in polynomial size circuits with *bounded depth*. We show that families of formulas can be generated from hard functions for AC^0.

Theorem 8. *Let f_n be a family of functions not in AC^0, then there are formulas $\mathcal{Q}$-f_n that do not have polynomial size proofs in QU-Res.*

A link between circuit and proof complexity for propositional logic is conjectured, yet there are very few examples of it materialising, feasible interpolation being another rare exception. In fact, we see a connection between strategy extraction and feasible interpolation– with feasible interpolation being akin to special case of strategy extraction.

QBF Lower Bounds. With the lower bound techniques mentioned above we can generate important lower bounds, particularly *separating formulas* that have short proofs in one system but not in another. We work towards the goal on completely separating the calculi in Figure 1. The first step is to separate QCDCL and expansion-based solving.

Resolution is known to be directly related to the CDCL algorithm, which means that any (super-polynomial) lower bound for resolution is also a lower bound for the CDCL algorithm. This is similar in the QBF setting. Lower bounds for Q-Res roughly correspond to lower bounds in QCDCL. On the expansion side of things, lower bounds for ∀Exp+Res correspond to lower bounds in expansion solving.

We show that without the hybridisation introduced in IR-calc, the difference between QCDCL and expansion solvers really exists via lower bounds. We find a family of QBF problems– QPARITY that were hard for all QCDCL proof systems but could be proved easy in ∀Exp+Res (the converse was already known [73]). The family of formulas essentially code that the parity function on a subset of the variables is both true and not true– an obvious contradiction. However, the strategy extraction technique shows that all Q-Res proofs must have size similar to the parity function in bounded depth circuits, and hence using the parity lower bound result from Håstad [67] (cf. Chapter 5, Theorem 7), all proofs must be exponential in size.

Theorem 8. *Any QU-Res refutation of* QPARITY *is of exponential size.*

We continue to separate the known resolution systems and show every possible separation. To do this we use the example from Kleine Büning et. al [77] and show a new lower bound for IR-calc. This does not involve any general lower bound technique, but does use a counting on assignments to universal variables.

Theorem 16. *All proofs of* KBKF *in IR-calc have exponential size.*

We fine-tune our hard examples to show separations for the advanced versions of the calculi. Along with the strategy extraction lower bounds and previously known lower bounds we manage to separate all the QBF resolution systems in Figure 1.

We can use our other techniques for more bounds. Our game technique from Theorem 20 can be used to show a new lower bound for tree-like QU-Res. Feasible interpolation can show a new lower bound for all the QBF resolution systems. This result uses the unconditional hardness of the clique problem in monotone circuits.

Theorem 33. *The* CLIQUE-*Co*CLIQUE *QBFs require exponential-size proofs in all the QBF resolution proof systems of Figure 1.*

Beyond QBF Resolution. While the initial focus of this work was on resolution and its relationship to solving, proof complexity is much broader domain. Our extended goals are to see how our research extrapolates outside of the typical settings, which are usually related to solving, and how it works in wider proof complexity.

Resolution is actually considered as a weak proof system because it has known lower bounds. An additional rule, known as extension, allows new extension variables to abbreviate more complicated boolean functions. This gives phenomenal improvement, as the system becomes equivalent to one of the most powerful propositional systems– Extended Frege (eFrege). In Tseitin transformations, extension variables are used to encode arbitrary propositional formulas in *Conjunctive Normal Form*. More generally, allowing the extension rule in proofs is known to shorten proof size drastically for many examples. This makes extension variables also very interesting in the context of solving, and indeed modern proof checking formats such as the ones from [69, 70] incorporate the use of extension variables.

Extension can also work in QBF resolution [75], although how extension variables have to be placed in the quantification order is not immediate. We use lower bounds to help shed some light on this issue. We find that restrictively placing the extension variables in the innermost quantification level, as in the weaker system, leads to lower bounds. This confirms an experimental observation by Jussila et. al. [75] with a rigorous theoretical argument.

Theorem 52. *Weak extended Q-resolution does not simulate extended Q-resolution.*

Another way to go beyond resolution is to look at other styles of proof system. An example would be the cutting planes proof system, which incorporates ideas from the NP-hard problem

of integer programming. Instead of working with propositional formulas, it works with sets of linear inequalities. The cutting planes system can be adapted for QBF refutation, by using the same reduction rule as in the QBF resolution system– Q-Res.

Frege Systems. We find the same principle of adding the reduction rule to a proof system to be generally applicable. The central proof systems under consideration here are Frege systems. Frege systems are textbook style systems that operate on propositional lines with sound deduction rules. Each Frege system defines its own axioms and rules, but all Frege systems are equivalent under simulation. Frege can be augmented with extension variables to give the calculus eFrege. Frege and eFrege are very powerful systems and finding lower bounds in these systems have been ongoing problems for decades.

We introduce a new family of proof systems for QBF that utilise the rules of Frege and eFrege systems– Frege + $\forall$red and eFrege + $\forall$red. These systems add the $\forall$-Red rule and we show that they are sound and refutationally complete for QBF. A remarkable aspect of these systems is that when adding the reduction rule, the strategy extraction theorem becomes applicable. The theorem introduces a direct connection between *circuit complexity* and QBF Frege systems. Hence, circuit size lower bounds can be turned into lower bounds for QBF Frege systems. In propositional proof complexity such a connection has been conjectured but is not yet known.

Theorem 63 (Strategy Extraction). *Given a false QBF $\mathcal{Q}\phi$ and a refutation π of $\mathcal{Q}\phi$ in* eFrege + $\forall$red*, it is possible to extract in linear time (w.r.t. $|\pi|$) a collection of polynomial size circuits D computing a winning strategy on the universal variables of ϕ.*

Furthermore, if we restrict the circuit complexity of the lines of Frege + $\forall$red we extract circuits of exactly of that complexity. This means that we can utilise the full range of conditional and unconditional lower bounds in circuit complexity and find lower bounds and separations in QBF proof systems. This also allows us to slightly extend the picture of proof systems with known lower bounds further than when projected down to the propositional case. Bounded depth Frege (AC^0-Frege) systems do have lower bounds, however bounded depth Frege systems with $\mod p$ gates ($\mathsf{AC}^0[p]$-Frege) do not. Since the strategy extraction theorem again holds for Frege systems, this allows us find some very strong lower bounds for these systems which are still open in the propositional case.

Corollary 13. *There are exponential lower bounds and separations for the QBF proof system* $\mathsf{AC}^0[p]$-Frege + $\forall$red *for all primes p;*

AC^0-Frege can be tuned to individual depth bounds for d which is written as AC^0_d-Frege. In propositional logic there are no known separating formulas with constant depth. However we can show this in QBF.

Corollary 21. *There is an exponential separation of the hierarchy of constant-depth systems* AC^0_d-Frege + $\forall$red *by formulas of depth independent of d.*

The circuit class TC^0 adds threshold gates to AC^0 circuits. We show a separation for the corresponding Frege systems in QBF.

Corollary 19. *There is an exponential separation of* $\mathsf{AC}^0[p]$-Frege+$\forall$red *from* TC^0-Frege+$\forall$red.

Although it would appear that the reduction rule creates an additional source of hardness, it is sufficient to create a very strong calculus. For eFrege+$\forall$red it was later shown [32] that lower bounds could only come from eFrege lower bounds or strategy extraction lower bounds. In other words lower bounds could not be found unless a breakthrough is made in propositional proof complexity or a breakthrough is made in circuit complexity. The other direction also is true and we provide that in this thesis.

Theorem 69. eFrege+$\forall$red *is not polynomially bounded if and only if there are* PSPACE*-functions that do not have non-uniform polynomially bounded circuits or* eFrege *is not polynomially bounded.*

A similar result works for Frege+$\forall$red. The technique in [32] was to use a formalisation of our strategy extraction theorem within the Frege calculus itself. This has larger implications for calculi beyond Frege+$\forall$red. Any QBF proof system simulating Frege+$\forall$red (such as the proof system G from [81]) would have the same difficulty in finding lower bounds.

1.3 Organisation

Part I contains background work that supplements the main content presented in later chapters. Chapter 2 covers the preliminaries in propositional logic, SAT solving and proof complexity. In Chapter 3 we look at the logic of Quantified Boolean Formulas as well as its own solving and proof complexity. These sections also highlight important literature and the results therein. Further discussions on related literature can be found in relevant chapters where appropriate.

The remaining chapters provide the main content of this thesis. Part II is titled "QBF Resolution" and concerns itself with the modern QBF resolution systems and their improvements. Part III is "QBF Proof Systems Beyond Resolution" and looks at stronger systems and their QBF counterparts.

In Chapter 4 we present a new calculus, IR-calc, which unifies concepts from both CDCL and expansion-based solving in QBF. The calculus is shown to naturally simulate the base proof systems from each solving technique. A stronger calculus IRM-calc is also shown to simulate more. The simulations show the completeness of these systems. We show the soundness by strategy extraction– that the witnesses of the universal variables that refute the formula can be calculated from the proof in polynomial time.

In Chapters 5 and 6 we generate new lower bounds for QBF calculi. In fact, using these bounds we show all possible separations for the resolution systems of QBF. Both new and old techniques are used to generate and confirm the lower bounds. The main new technique is the strategy extraction lower bound technique in Chapter 5. In this we show a new lower bound for QCDCL proof systems using lower bounds from circuit complexity, and indeed we show that expansion based calculi are

exponentially separated from all QCDCL systems. However, this does not give us all the lower bounds, so in Chapter 6 we use the most well-known QBF lower bounds from [77].

In Chapters 7 through 9 we look at the standard techniques from propositional logic in the wider QBF setting. Specifically in Chapter 7 we adapt the Prover-Delayer game from [27–29] for tree-like QBF resolution systems. We use this to show some alternative proofs for lower bounds and show a new lower bound. In Chapter 8 we adapt the feasible interpolation technique from [80, 99], that uses circuit lower bounds to show proof size lower bounds. In the QBF setting, we remark that there is a similarity between this technique and the strategy extraction technique, namely they both use lower bounds from circuit complexity. In the end of this section we show a connection between the two techniques: for every feasible interpolation problem we can create a strategy extraction example with the same proof size lower bounds. Not every technique can be adapted to QBF. In Chapter 9 we present from [22] that the size-width relation from Ben-Sasson and Widgerson [13] fails, unless we use tree-like calculi.

In Chapter 10 we go beyond the QBF setting and study the more expressive Dependency QBF (DQBF) language. The focus is still on QBF proof systems as we investigate all the QBF resolution systems applied in a natural way to the DQBF setting. We find that only one of these proof systems suffices in the DQBF setting; the IR-calc introduced in Section 4.

In Chapter 11 the lower bound from strategy extraction is used again to separate two improvements on Q-resolution, each of which allows extension variables under different quantifier level conditions. One system places all extension variables in the innermost quantification block, whereas the stronger system allows extension variables to be placed in any block as long as they are right of all variables they are defined on. Extension variables are especially important in proof complexity because while propositional resolution is a fairly weak proof system with many known lower bounds, *extended* resolution has no known lower bound and simulates even the most powerful systems.

In Chapter 12, we look at a slightly stronger system than resolution, namely the cutting planes proof system. We find that the cutting planes proof system is both sound and complete for QBF with one additional rule. The strategy extraction theorem and feasible interpolation result also holds for this system.

The most important systems above resolution are arguably the Frege systems. In propositional logic they have no known lower bound in general. We adapt a version of Frege called Frege+$\forall$red to work for QBFs and find a number of interesting properties that hold in it. The most important result here is that we show the strategy extraction theorem again holds for Frege systems, and this used to show lower bounds for important fragments of the system.

1.4 List of Publications

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

1. Olaf Beyersdorff, Leroy Chew, and Karteek Sreenivasaiah. *A Game Characterisation of Tree-like Q-resolution Size.* To appear in *Journal of Computer and System Sciences*.

2. Olaf Beyersdorff, Leroy Chew, Meena Mahajan and Anil Shukla. *Feasible Interpolation for QBF Resolution Calculi.* To appear in *Logical Methods in Computer Science*.
3. Olaf Beyersdorff, Leroy Chew, Meena Mahajan, and Anil Shukla. *Understanding Cutting Planes for QBFs.* In *Proc. IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'16)*, pages 40:1–40:15. LIPIcs, 2016.
4. Olaf Beyersdorff, Leroy Chew, Renate A. Schmidt, and Martin Suda. *Lifting QBF Resolution Calculi to DQBF.* In *Proc. Theory and Applications of Satisfiability Testing. International Conference (SAT'16)*, pages 490–499. Springer, 2016.
5. Olaf Beyersdorff, Leroy Chew, Meena Mahajan, and Anil Shukla. *Are Short Proofs Narrow? QBF Resolution is not Simple.* In *Proc. Symposium on Theoretical Aspects of Computer Science (STACS'16)*. pages 15:1–15:14. LIPIcs, 2016.
6. Olaf Beyersdorff, Ilario Bonacina, and Leroy Chew. *Lower Bounds: From Circuits to QBF Proof Systems.* In *Proc. ACM Conference on Innovations in Theoretical Computer Science (ITCS'16)*, pages 249–260. ACM, 2016.
7. Olaf Beyersdorff, Leroy Chew, and Mikoláš Janota. *Proof Complexity of Resolution-Based QBF Calculi.* In *Proc. Symposium on Theoretical Aspects of Computer Science (STACS'15)*, pages 76–89. LIPIcs, 2015.
8. Olaf Beyersdorff, Leroy Chew, and Mikoláš Janota. *On Unification of QBF Resolution-Based Calculi.* In *Proc. International Symposium on Mathematical Foundations of Computer Science (MFCS'14)*, pages 81–93, 2014.
9. Olaf Beyersdorff, Leroy Chew, and Mikoláš Janota. *Extension Variables in QBF Resolution.* In *Proc. Beyond NP, Papers from the AAAI Workshop*, AAAI Press, 2016.

The following publications were published during my time as a research student but are not featured in this thesis :

10. Olaf Beyersdorff and Leroy Chew. *The complexity of theorem proving in circumscription and minimal entailment.* In *Proc. 7th International Joint Conference on Automated Reasoning (IJCAR'14)*, pages 403–417, 2014.
11. Olaf Beyersdorff and Leroy Chew. *Tableau vs. sequent calculi for minimal entailment.* In *Proc. 15th International Workshop on Non-Monotonic Reasoning (NMR'14)*, volume abs/1405.1565, 2014.

Chapter 2

Preliminaries

We begin our investigation into QBF proof complexity by first looking at propositional proof complexity. QBFs are relatively simple modifications of propositional formulas and there are important concepts in propositional logic that apply also to QBF. In particular, many of the proof systems in QBF take inspiration from propositional systems.

In complexity theory, complexity classes group problems according to their inherent difficulty. There is a three-way connection between solving, proof complexity and complexity classes, which we will discuss.

In this chapter we cover the main preliminaries for proof complexity. In Section 2.1 we introduce the propositional logic notation. SAT solving and proof complexity are covered by Sections 2.2 and 2.3, respectively. In Section 2.5 we cover some important relations between logic and complexity.

2.1 Propositional Logic

Boolean Operations. The foundation of Boolean algebra is the set of *Boolean constants* $\{0, 1\}$, namely false and true respectively. From here, we can then start considering the common connectives on these constants. The first connective we introduce is the binary connective $\rightarrow$. This is the logical implication symbol and $p \rightarrow q$ means that if p is true q is also true. Formally this is given below.

$$p \rightarrow q = \begin{cases} 0 & p = 1, q = 0 \\ 1 & \text{otherwise} \end{cases}$$

We define the connectives $\neg, \vee, \wedge$ to mean negation, disjunction and conjunction respectively. Negation considers when a proposition is not true, disjunction is a logical 'or' and conjunction is a logical 'and'. Formally they can be defined as below by using $\rightarrow$, this allows us to inductively describe propositional formulas using only $\rightarrow$.

$$\neg p = p \rightarrow 0$$

$$p \vee q = (\neg p) \rightarrow q$$

$$p \wedge q = \neg((\neg p) \vee (\neg q))$$

$$p \oplus q = ((\neg p) \vee (\neg q)) \wedge ((p) \vee (q))$$

While it is useful in some instances to consider $0, 1, \rightarrow$ as the only operations and all others as derived operations, in some instances due to the commutative nature of $\vee$ and $\wedge$ we sometimes only

consider $0, 1, \wedge, \vee, \neg$ as the only connectives and exclude $\rightarrow$. Note that $(\neg p) \vee q = p \rightarrow q$. The last element to establish the entire set of propositional formulas– is a set of countably infinitely many variables $\{x_1, x_2, \dots\}$.

Definition 1. *A formula is defined inductively.* 0 *is a formula.* 1 *is a formula. For every* $i \in \mathbb{N}$*,* x_i *is a formula when* x_i *is a propositional variable. If* p *and* q *are formulas then,* $(p \rightarrow q)$ *is a formula.*

The connectives follow an order of operations $\neg, \wedge, \vee, \rightarrow, \oplus$. So a formula $\neg x \vee y \wedge \neg z$ is equivalent to $(\neg x) \vee (y \wedge (\neg z))$.

We introduce the notation $A[x/y]$ indicating that in the formula A every occurrence of formula y is replaced by formula x. The notation $\mathrm{var}(A)$ denotes the set of propositional variables that are present in formula A, this can be defined inductively with the rules $\mathrm{var}(x_i) = \{x_i\}$, $\mathrm{var}(0) = \mathrm{var}(1) = \varnothing$, $\mathrm{var}(p \rightarrow q) = \mathrm{var}(p) \cup \mathrm{var}(q)$.

An *assignment* to a propositional formula ϕ is a function $\alpha : \mathrm{var}(\phi) \rightarrow \{0, 1\}$. A *satisfying assignment* is any assignment α to ϕ such that when every variable x in ϕ is replaced by $\alpha(x)$, ϕ evaluates to 1. Equivalently, a *model* M is a subset of $\mathrm{var}(\phi)$ such that the following assignment is a satisfying assignment.

$$\alpha(x) = \begin{cases} 1 & \text{if } x \in M \\ 0 & \text{if } x \notin M \end{cases}$$

Conjunctive Normal Form. Since associativity holds for $\wedge$ and $\vee$ we can consider unbounded versions of these with arbitrary arity; $\bigwedge$ and $\bigvee$. Note that in the special case where we have the empty disjunction we consider the formula to be equivalent to 0 and symmetrically where we have the empty conjunction we consider the formula to be equivalent to 1.

Finally this gives us a way (although by no means a unique way) to express every propositional formula by the commonly used conjunctive normal form (CNF) or disjunctive normal form (DNF). We will see this below.

Literals are special kinds of formulas. We can represent all literals as l_i where $l_{2i} = x_i$ and $l_{2i+1} = (\neg x_i)$ when x_i is a propositional variable. In other words a literal is a propositional variable or a negation of a propositional variable. The literals x_i, $\neg x_i$ are complimentary to each other, so $\neg\neg x_i$ is simplified to x_i accordingly. In practice we use other letters to denote literals and variables.

A *clause* is a disjunction of literals. A *term* is a conjunction of literals. Because disjunctions are commutative and idempotent, we can treat clauses as sets of literals, likewise with terms. A *CNF* is a conjunction of clauses. A *DNF* is a disjunction of terms. Every formula is equivalent to a DNF and to a CNF.

Circuits. We differentiate between a formula and *circuit*. A formula can be considered as a tree with variable (or constant) leaves and connective nodes. We generalise this concept to a circuit where we allow it to be a *directed acyclic graph*. Each node is a gate connected to an unbounded number of input nodes.

2.2 The SAT Problem

The *SAT problem* is the language of all propositional formulas Φ that have at least one satisfying assignment; dually it can be seen as the set of all formulas that are not self-contradictory.

Given an assignment α on formula Φ, it can be checked in polynomial time (in the size of Φ) whether α is a satisfying assignment. A description of any α can be expressed in polynomial size of Φ, hence it provides a polynomial-size witness that can be checked in polynomial-time. Therefore this is exactly what is needed to be in the class NP. The problem is that there may be exponentially many assignments to $\mathrm{var}(\Phi)$ and checking them all exhaustively would not be efficient.

The Cook-Levin Theorem [42] states that SAT is in fact NP-hard, which means that all other NP problems are reducible to SAT. Since SAT is in NP, this means that SAT is NP-complete. This is hugely important to the complexity classes and gives a canonical NP-complete problem. This also has practical considerations, suppose we have a decision problem L in NP. By Cook's Theorem we know we can reduces instances of the L problem to instances of the SAT problem and then run the best SAT solver to decide this. Satisfiability or more specifically unsatisfiability, has a clear connection to semantic implication: $p_1, \ldots p_n \vDash q$ if and only if $p_1, \ldots p_n, \neg q$ is unsatisfiable.

Due to its NP completeness, SAT solvers are hugely competitive and consequently SAT solvers can solve industrial instances with millions of variables.

The DPLL Algorithm. The base algorithm for SAT-solving is the DPLL algorithm [48] named after Davis, Putnam, Logeman and Loveland. It is remarkably simple but forms the basis of even modern SAT-solvers. The DPLL algorithm is an enhancement of the original DLL algorithm [49] (cf. Algorithm 1).

As follows: the input is a CNF, which is presented as a set of clauses (with clauses themselves being sets of literals), then branching occurs on the variable values until either the branch yields a satisfying assignment of the formula, or the assignment contradicts the formula.

The DPLL algorithm (cf. Algorithm 2) improves on DLL by adding *unit propagation* and *pure-literal elimination*. Unit propagation restricts assignments to those that satisfy unit clauses. Pure literal elimination checks for variables that only occur as one polarity and removes all clauses that contain them. These two concepts are applied iteratively each time after each branching procedure until fix-point.

CDCL Solving. The DLL and DPLL algorithms use *chronological backtracking*, that is it backtracks to the previous variable only once they have searched both values of the current variable.

One way to avoid this is as follows: wherever DPLL reaches a contradiction, observe which of the assigned variables resulted in the contradiction. The set of literals that correspond to these assigned variables is known as the *conflict set* [50]. A non-chronological backtracking algorithm may instead backtrack to the last decision variable in the conflict set.

Another way the same idea can be implemented is through *Conflict Driven Clause Learning (CDCL)*. For example, whenever we generate a conflict set P we observe that $\{\neg l \mid l \in P\}$ is an

Algorithm 1 The DLL algorithm

function DLL(Φ)
  if $\Phi = \varnothing$ **then**
    return 1
  else if $\varnothing \in \Phi$ **then**
    return 0
  else
    Pick a as any variable that appears in Φ.
    $\Phi_{a=0} \leftarrow \{D | D = C \setminus \{a\}, C \in \Phi, \neg a \notin C\}$
    $\Phi_{a=1} \leftarrow \{D | D = C \setminus \{\neg a\}, C \in \Phi, a \notin C\}$
    if DLL($\Phi_{a=0}$) $= 1$ **then**
      return 1
    else if DLL($\Phi_{a=1}$) $= 1$ **then**
      return 1
    else
      return 0

Algorithm 2 The DPLL algorithm

function DPLL(Φ)
  while Φ contains a unit clause $\{l\}$ **do**
    $\Phi \leftarrow (\Phi \setminus \{l\})|_{l=1}$
  for every literal l that is pure in Φ **do**
    $\Phi \leftarrow \Phi_{l=1}$
  if $\Phi = \varnothing$ **then**
    return 1
  else if $\varnothing \in \Phi$ **then**
    return 0
  else
    Pick a as any variable that appears in Φ.
    $\Phi_{a=0} \leftarrow \{D | D = C \setminus \{a\}, C \in \Phi, \neg a \notin C\}$
    $\Phi_{a=1} \leftarrow \{D | D = C \setminus \{\neg a\}, C \in \Phi, a \notin C\}$
    if DPLL($\Phi_{a=0}$) $= 1$ **then**
      return 1
    else if DPLL($\Phi_{a=1}$) $= 1$ **then**
      return 1
    else
      return 0

implicant of the original clauses. Once we backtrack we then also add this clause as if it were one of the premises. This is called clause learning and has the advantage that it involves both search and inference.

Practical Techniques. Modern propositional SAT solvers typically use CDCL solving. However numerous other improvements are commonly put inside the solver. For example, data structures make a huge practical difference, so it is usually inefficient to list every literal and its assignment in each clause. Efficiency can be obtained by *lazy* data structures, which preserve the soundness of the algorithm but whose data structures do not contain accurate information. Examples include the Head and Tail data structure [118] and the Watched Literal data structure [92].

Branching in CDCL can also be guided by heuristics and there are many different examples of such [90], including those that work with lazy data structures [92]. Variable selection can also include restarts, and random restarts have been effective in countering the possibility of an exponential time implementation [63]. Finally, while clause learning can help to improve speed, unrestricted clause learning will eventually use up increasing amounts of memory, which in practice is limited. In addition, this can also cause speed issues. The solution involves deletion of unnecessary clauses.

2.3 Proofs

The concept of proof is a central idea in mathematics and logic. A proof is required to be correct, in which case it is formally called *sound*. Proofs should also be easy to check, we do not want exhaustive details that should be in the proof to be left to the reader.

In order to analyse proofs formally, and eventually from a complexity point of view, we need to fix the frameworks that proofs can exist in. To do this *proof systems* are defined. Proof systems dictate which strings constitute a proof of an element of a particular language (i.e. propositional tautologies). Proof systems can be a line-based derivation or follow a more static structure.

As well as being sound and allowing proofs to be easily checked, a proof system needs to be complete for a particular language by ensuring that every theorem has a proof. Typically there are two kinds of completeness- firstly, *implicational completeness* requires that any semantic implication of a set of statements Γ can be derived when we take Γ as a premise. Secondly, *refutational completeness* only requires that the contradiction symbol (0 or sometimes the empty clause) can be reached from an unsatisfiable set of formulas. A proof system that is implicationallly complete is also refutationally complete. The converse is not always true, but refutational systems are usually sufficient in practice for determining if an implication holds. For example, in classical logic if Γ implies a then $\Gamma \cup \{\neg a\}$ is contradictory.

Formally, a *proof system* [45] for a language L over alphabet $\mathsf{\Gamma}$ is a polynomial-time computable partial function $f : \mathsf{\Gamma}^\star \rightharpoonup \mathsf{\Gamma}^\star$ with $rng(f) = L$.

The partial function f actually gives a proof checking function. Soundness is given by $rng(f) \subseteq L$ and completeness is given by $rng(f) \supseteq L$. The polynomial-time computability is an indication of

feasibility, relying on the complexity notion that if something is in polynomial time it is considered feasible. Individual authors may wish to modify the definition of a proof system, particularly changing the condition of it being polynomial time, but these variations are ignored in this thesis and we will always require proofs to be checked in polynomial time.

Proof Complexity. From the definition of a proof system, we can start defining proof size. For a proof system f for language L and string $x \in L$ we define $s_f(x) = \min(|w| : f(w) = x)$. Thus the partial function s_f tells us the minimum proof size of a theorem. We can overload the notation by setting $s_f(n) = \max(s_f(x) : |x| \leq n)$ where $n \in \mathbb{N}$. We do not focus on the exact numerical proof size, but how proof size behaves asymptotically. For a function $t : \mathbb{N} \to \mathbb{N}$, a proof system f is called *t-bounded* if $\forall n \in \mathbb{N}, s_f(n) \leq t(n)$.

A proof system f is *polynomially bounded* if there is a polynomial $p(x)$ such that $s_f(n) \leq p(n)$. Cook and Reckhow [45] proved that $\mathsf{NP} = \mathsf{coNP}$ if and only if there is a polynomially bounded proof system of propositional tautologies, where coNP is the class of all languages whose complements are in NP. A *super-polynomial lower bound* is an infinite family of formulas Φ_n where there is no polynomial p such that the shortest proof of each formula is $\leq p(|\Phi_n|)$. An *exponential lower bound* is an infinite family of formulas Φ_n where there is an exponential function $f = 2^{n^{\Omega(1)}}$ such that the shortest proof of each formula is $\geq f(|\Phi_n|)$.

Proof systems are compared by simulations. We say that a proof system f *simulates* g ($g \leq f$) if there exists a polynomial p such that for every g-proof π_g there is an f-proof π_f with $f(\pi_f) = g(\pi_g)$ and $|\pi_f| \leq p(|\pi_g|)$. If π_f can even be constructed from π_g in polynomial time, then we say that f *p-simulates* g ($g \leq_p f$). Two proof systems f and g are *(p-)equivalent* ($g \equiv_{(p)} f$) if they mutually (p-)simulate each other. Figure 2 details the p-simulation relations between the main proof systems in propositional logic. Note that every system below the dotted line has a known super-polynomial lower bound.

A *separation* is where we demonstrate that a simulation cannot occur. Typically, to do this we have to find a super-polynomial lower bound Φ_n in one proof system, but show that Φ_n has polynomial-size proofs in the other.

Proof Systems for Propositional Logic. A *line-based proof system* operates with axioms and rules to reach a conclusion. A proof π can be seen as a sequence of lines $L_1, \ldots L_n$ where if ϕ was the premise of the proof then $\phi \wedge L_1 \wedge L_2 \wedge \cdots \wedge L_i$ is semantically equivalent to ϕ for any $i : 1 \leq i \leq n$.

Axiom lines are formulas that can be derived in any situation, these can take the form of polynomial-time recognisable tautologies (not all tautologies are polynomial-time recognisable unless P=coNP). Alternatively in some systems, particularly refutational ones, the whole of, or a conjunct of the premise may be derived using the axiom rule, depending on the proof system.

Rules take one or more previous lines and use some (usually limited) recognition of semantic implication to derive a new line. For example, in resolution, the resolution rule notices that if both $C \vee x$ and $D \vee \neg x$ are lines before L_i then $\phi \wedge L_1 \wedge \cdots \wedge L_i \vDash C \vee D$.

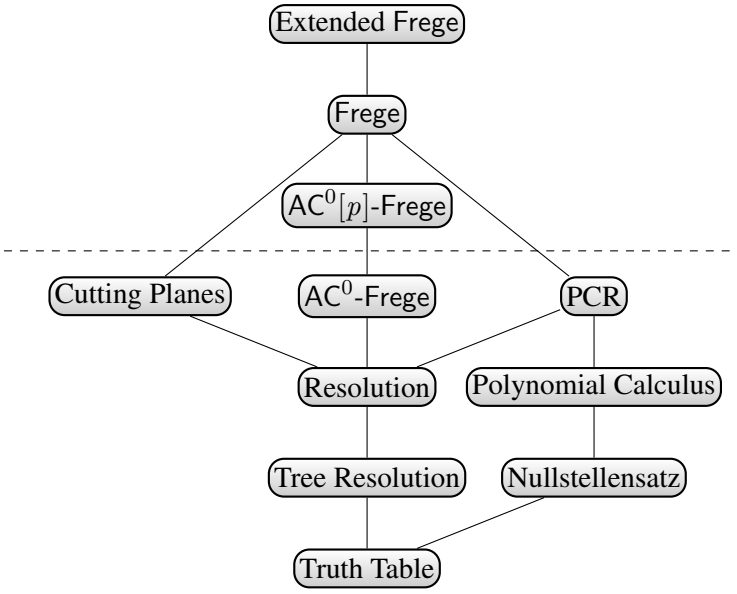


Fig. 2. A simulation Hasse diagram for propositional proof systems

Resolution, introduced by Blake [34] and Robinson [107], is a refutational proof system manipulating unsatisfiable CNFs as sets of clauses, and clauses as sets of literals. The only inference rule is $\frac{C \vee x \qquad D \vee \neg x}{C \cup D}$ where C, D denote clauses and x is a variable. In other forms particularly when Resolution operates on ordered multisets, factoring and exchange rules are used, but these are unnecessary when the clauses are sets. A resolution refutation derives the empty clause $\perp$.

The following observation underpins the key connection between SAT solving and proof complexity.

Observation 1 *If ϕ is input into the* DLL *algorithm and it returns* 0 *then ϕ has a tree-like resolution refutation with the same or fewer number of lines as the total number of calls to* DLL *used in total.*

Proof. We will construct a tree of partial assignments as the algorithm progresses and then compare the assignments to the lines of a tree-like resolution proof. We start with the empty assignment and add more values as we make recursive calls. When we call $\mathsf{DLL}(\Phi_{a=0})$ we add $a = 0$ to the assignment and similarly we add $a = 1$ in the branch where we call $\mathsf{DLL}(\Phi_{a=1})$. We continue to add these assignments in the respective branches.

Notice that each assignment must eventually contradict one of the clauses in ϕ. In order to construct the refutation we mark each leaf with the clause it refutes. We then resolve where the branches meet on the variable a that is split on, unless the variable does not appear in one of the clauses. In that case we simply copy that clause without the variable (and pick arbitrarily if both do not have the variable). The invariant we preserve by this process is that the assignment always refutes the clause associated with it and since we end up with the empty assignment we must have the empty clause at the end. □

One of the most general and important line-based systems are the *Frege systems* [45]. These are based on axiom schemes of tautologies with additional sound rules to make it complete. In the context here we only look at Frege systems as classical propositional proof systems, although Frege systems can exist for other languages.

For now we will fix a complete connective set $\{\rightarrow, 0, 1\}$ for simplicity, but the set $\{\neg, \vee, \wedge, 0, 1\}$ is commonly used and this will be more appropriate later when talking about CNFs. Each Frege system is defined with a finite number of propositional rules–

$$\frac{C_1, C_2, \dots C_k}{D}$$

where $C_1, C_2, \dots C_k, D$ are propositional formulas in the variables $x_1, \dots x_m$ and $C_1 \wedge C_2 \wedge \dots \wedge C_k \vDash D$ (where $\vDash$ denotes classical truth functional semantic entailment). Nullary rules gives us axioms $\frac{}{D}$ which require D to be a tautology. We consider substitutions $\sigma = f_1, \dots f_m / x_1, \dots x_m$ of propositional variables $x_1, \dots x_m$ to propositional formulas $f_1, \dots f_m$. We can denote a substitution on formula D as σD. A proof consists of a sequence $L_1, L_2, \dots L_n$ where L_i has the requirement that $\frac{C_1, C_2, \dots C_k}{D}$ is a rule in the system and that there is some σ such that $\sigma C_1, \sigma C_2, \dots \sigma C_k$ are all previous lines in the proof and $\sigma D = L_n$. A Frege system has an additional requirement that it must be implicationally complete for the language (note that the language technically changes if we change the connective set). An example of a Frege system is given in Figure 3.

$$\frac{}{1} \qquad \frac{}{x_1 \rightarrow (x_2 \rightarrow x_1)} \qquad \frac{}{((x_1 \rightarrow 0) \rightarrow 0) \rightarrow x_1}$$

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

Fig. 3. A Frege system for connectives $\rightarrow, 0, 1$

It was first shown in [45] that for a fixed connective set κ any two Frege systems p-simulate each other, furthermore [103] extended this mutual p-simulation to systems with a different connective set. This means that an exponential lower bound for one Frege system would show a lower bound for all Frege systems. However, to this day, neither such an exponential nor super-polynomial lower bound has been found.

This does not mean that Frege simulates every known proof system. There is a proof system that is conjectured to be stronger than Frege– an improvement known as *Extended Frege* (eFrege). Extended Frege takes a Frege system and allows the introduction of new variables that do not appear in any previous line of the proof. These variables abbreviate formulas. The rule works by introducing the axiom of $v \leftrightarrow f_1 \bar{\wedge} f_2$ for new variable v (not in appearing in f_1 or f_2).

Because there is no known super-polynomial lower bound for Frege there is no known separation between Extended Frege and Frege. Instead one might think of the analogous relation of circuits versus formulas. Extended Frege is a very powerful system, it was shown [15, 79] that any propositional proof system f can be simulated by $e\mathsf{Frege} + ||\phi||$ where ϕ is a polynomially recognisable axiom scheme.

Sequent Calculi. An alternative to Frege systems are Gentzen systems. While Frege systems use formulas as their lines, Gentzen systems use sequents.

Gentzen's system LK is one of the first and best studied proof systems [61]. It operates with sequents. Formally, a *sequent* is a pair (Γ,Δ) with Γ and Δ defined as finite ordered multisets of formulas. A sequent is usually written in the form $\Gamma \longrightarrow \Delta$. In propositional logic $\Gamma \longrightarrow \Delta$ is true if every model for $\bigwedge \Gamma$ is also a model of $\bigvee \Delta$. The system can be used both for propositional and first-order logic; the propositional rules are displayed in Fig. 4.

$$\frac{}{A \longrightarrow A}\ (\longrightarrow) \qquad \frac{}{\bot \longrightarrow}\ (\bot\text{-introduction}) \qquad \frac{}{\longrightarrow \top}\ (\top\text{-introduction})$$

$$\frac{\Gamma, A, B, \Delta \longrightarrow \Sigma}{\Gamma, B, A, \Delta \longrightarrow \Sigma}\ (\text{exchange-l}) \qquad \frac{\Gamma \longrightarrow \Sigma, A, B, \Delta}{\Gamma \longrightarrow \Sigma, B, A, \Delta}\ (\text{exchange-r})$$

$$\frac{A, A, \Gamma \longrightarrow \Sigma}{A, \Gamma \longrightarrow \Sigma}\ (\text{contraction-l}) \qquad \frac{\Gamma \longrightarrow \Sigma, A, A}{\Gamma \longrightarrow \Sigma, A}\ (\text{contraction-r})$$

$$\frac{\Gamma \longrightarrow \Sigma}{\Delta, \Gamma \longrightarrow \Sigma}\ (\text{weakening-l}) \qquad \frac{\Gamma \longrightarrow \Sigma}{\Gamma \longrightarrow \Sigma, \Delta}\ (\text{weakening-r}) \qquad \frac{\Gamma \longrightarrow \Sigma, A}{\neg A, \Gamma \longrightarrow \Sigma}\ (\neg\text{-l})$$

$$\frac{A, \Gamma \longrightarrow \Sigma}{\Gamma \longrightarrow \Sigma, \neg A}\ (\neg\text{-r}) \qquad \frac{A, \Gamma \longrightarrow \Sigma}{B \wedge A, \Gamma \longrightarrow \Sigma}\ (\bullet\wedge\text{-l}) \qquad \frac{A, \Gamma \longrightarrow \Sigma}{A \wedge B, \Gamma \longrightarrow \Sigma}\ (\wedge\bullet\text{-l})$$

$$\frac{\Gamma \longrightarrow \Sigma, A \qquad \Gamma \longrightarrow \Sigma, B}{\Gamma \longrightarrow \Sigma, A \wedge B}\ (\wedge\text{-r}) \qquad \frac{A, \Gamma \longrightarrow \Sigma \qquad B, \Gamma \longrightarrow \Sigma}{A \vee B, \Gamma \longrightarrow \Sigma}\ (\vee\text{-l})$$

$$\frac{\Gamma \longrightarrow \Sigma, A}{\Gamma \longrightarrow \Sigma, B \vee A}\ (\bullet\vee\text{-r}) \qquad \frac{\Gamma \longrightarrow \Sigma, A}{\Gamma \longrightarrow \Sigma, A \vee B}\ (\vee\bullet\text{-r})$$

$$\frac{\Gamma \longrightarrow \Sigma, A \qquad A, \Gamma \longrightarrow \Sigma}{\Gamma \longrightarrow \Sigma}\ (\text{cut})$$

Fig. 4. Rules of the sequent calculus LK [61]

The full version of LK is known to be p-equivalent to Frege systems. However a weaker system without the cut rule (see Figure 4) known as *Cut-free Gentzen* is known to be a complete calculus, yet there is an exponential separation between the cut-free and full version.

Cutting Planes. Not every line-based proof system uses propositional formulas as its internal lines. A good example is the cutting planes proof system which uses linear inequalities.

A translation from CNF to a system of linear equalities that preserves satisfiability is necessary. We will use θ to denote the translation.

- for a propositional variable x, $\theta(x) = x$.
- for a propositional variable x, $\theta(\neg x) = 1 - x$.
- for a clause C, where n is the number of literals in C, The inequality $\theta(C) = \sum_{l \in C} \theta(l) \geq 1$.
- for a CNF F, $\theta(F) = \{\theta(C) \mid C \in F\} \cup \{x \geq 0, -x \geq -1 \mid x \text{ is a variable}\}$.

The cutting planes proof system is refutational, its aim is to derive the contradiction $0 \geq 1$ from a inconsistent set of linear inequalities. The rules of the cutting planes proof systems are presented in Figure 5.

$$\frac{\sum_k a_k x_k \geq a \qquad \sum_k b_k x_k \geq b}{\sum_k (a_k + b_k) x_k \geq a + b} \text{ (Addition)} \qquad \frac{\sum_k c_k x_k \geq c}{\sum_k dc_k x_k \geq dc} \text{ (Multiplication)}$$

$$\frac{\sum_k q p_k x_k \geq p}{\sum_k p_k x_k \geq \lceil \frac{p}{q} \rceil} \text{ (Division)}$$

Fig. 5. Rules for the cutting planes proof system, $d, q \in \mathbb{Z}^+$

2.4 First-order Logic.

First-order logic is a more expressive language than propositional logic. In propositional logic, atomic formulas can only be true or false, but in first-order logic we allow atomic formulas to be contingent on *first-order variables*. This is done via *predicates* where the truth is solely dependent on the values of a finite number of ordered first-order variable arguments. The number of arguments of an individual predicate is known as the arity. When the arity is zero the formulas are equivalent to propositions. In order to make further use of variables, variables can be quantified via universal or existential *quantifiers*.

First-order Formulas. A *first order term* is either a variable or a function where each argument is an first order term itself. *Atomic formulas* are predicates in which every argument is a term. *Well formed formulas* are grammatically correct formulas. If ϕ is a well formed formula it must satisfy one of the following conditions.

- ϕ is an atomic formula.
- ψ is a well formed formula and ϕ is $\neg\psi$.

- χ and ψ are well formed formulas and ϕ is either $\chi \wedge \psi$, $\chi \vee \psi$ or $\chi \rightarrow \psi$.
- ψ is a well formed formula x is a free variable in χ (it does not appear in any quantifier). and ψ is either $\forall x\psi$ or $\exists x\psi$.

A *sentence* is a well formed formula where every variable in the sentence is quantified.

First-order Semantics. The semantics of first-order logic were most famously formulated inductively by Tarski [114], it follows very similarly to propositional logic but with the following considerations for quantifiers.

- $\exists x\psi$ is true if and only if there is some value c for which $\psi[c/x]$ is true.
- $\forall x\psi$ is true if and only if $\neg\exists x\neg\psi$ is true.

A Decidable Fragment in First-order Logic. Determining whether a first-order logic formula is true under all interpretations is not decidable, the best that can be done is under fragments of first-order logic. One decidable fragment is the *Bernays-Schönfinkel class* also known as EPR (a shortening of Effectively PRopositional). This class have formulas in prefix form $\Pi\phi$, $\Pi = \exists x_1, \ldots x_n \forall y_1, \ldots y_m$ and ϕ does not contain quantifiers or function symbols (except for zero-arity functions, which are constant symbols).

First-order Resolution. Proof systems for first-order logic and EPR exist. In Figure 6 we define the first-order resolution of Robinson (FO-res) [106] for EPR. The system is a refutation system for prefix formulas $\Pi\phi$ where ϕ is in conjunctive normal form (for first-order logic literals are atomic formulas or their negations). By unifier of $L(X)$ and $L(Y)$, we mean the first-order term substitution θ to the universal variables of X and Y such that $L(X)\theta = L(Y)\theta$. σ is more general than θ if there is another substitution ν such that ν applied to σ is equal to θ. The most general unifier is the unique unifier that is more general than all other unifiers.

$$\frac{C \cup \{L(X)\} \qquad D \cup \{\neg L(Y)\}}{(C \cup D)\sigma} \text{ (Res)}$$

Where σ is the most general unifier of $L(X)$ and $L(Y)$ and C and D do not contain common variables.

$$\frac{C \cup \{L(X_1), \ldots L(X_j)\}}{(C \cup \{L(X_1)\})\sigma} \text{ (Fact)} \qquad\qquad \frac{C}{C\theta} \text{ (Inst)}$$

Where σ is the most general unifier of $L(X_1), \ldots L(X_j)$. θ is a term substitution for the universal variables of C.

Fig. 6. Rules for Robinson's FO-res for EPR

While first-order logic uses a different machinery to that of QBFs, both use quantification albeit in different ways. We will introduce Boolean quantification in Chapter 3 alongside QBFs themselves. We will revisit the fragment EPR in Chapter 10.

2.5 Relationship between Logic and Complexity

Circuit Classes. Complexity theory aims to understand the inherent difficulty of various objects in theoretical computer science. Computational complexity primarily looks at problems, languages or tasks and the important objects are algorithms. For these complexity is usually measured by running time, leading to the famous class P which includes all problems that have a polynomial time algorithm. Other measures such as space (the amount of memory required) can also be considered, leading to complexity classes such as PSPACE where polynomial space algorithms exist.

In Section 2.1 we defined and differentiated formulas and circuits and there are many ways to measure complexity using these. For example, i) binary size of a circuit description ii) number of gates needed iii) depth (longest directed path from an input node to an output node).

Like computational complexity, circuit complexity concerns itself with languages or problems. In circuit complexity, these languages consist of boolean functions with multi-arity boolean domains. The complexity classes usually concern themselves with the following question:

Can all the functions g in language L be found using a family of circuits $c(g)$ (for each g) such that the size/depth/other measure of $c(g)$ is bounded by $O(f(g))$ where f is the growth function and $c(g)$ is subject to additional constraints?

This then brings up the question of uniformity versus non-uniformity. A uniform family of circuits is a family that can be generated quickly by a computer program. Normally for family $\{C_n \mid n \in \mathbb{N}\}$ we require a Turing machine M to output the circuit C_n (as a binary string) on input 1^n in polynomial time. A non-uniform family does not have these restriction on circuits. While the terminology is confusing, every uniform family is also non-uniform.

The class $\mathsf{P/poly}$ is the class of languages such that if $L \in \mathsf{P/poly}$ there is a non-uniform family of boolean circuits $\{C_i \mid i \in \mathbb{N}\}$, such that for every binary string $|x| = i$, $x \in L$ if and only if $C_i(x) = 1$. There is the additional important distinction that there is some polynomial p such that $|C_i| = O(p(i))$.

The class AC^0 is the smallest class from the AC-hierarchy, it is the class of languages that have non-uniform polynomial-size circuits that are constant depth. In AC^0 the circuits can only use NOT gates and AND, OR gates of unbounded fan-in. With this restriction the majority function, which asks whether the number of 1 bits outnumbers the number of 0 bits, is not in AC^0 [67]. Allowing threshold gates, which ask whether the number of 1 values are above a specified threshold, allows new kinds of circuits. In the setting of non-uniform bounded depth polynomial-size circuits we use the class TC^0 when we have threshold gates of unbounded fan-in.

Another family of problems that are not in AC^0 are the $\mathrm{mod}\ p$ functions, which test whether an input's sum value equals $1 \bmod p$. The functions themselves can be added individually as gates to get the $\mathsf{AC}^0[p]$ class when bounded depth and polynomial.

AC^0 circuits can also be given individual depth restrictions, AC^0_d is the class of functions that can be represented by a family of polynomial size non-uniform circuits of depth at most constant d.

Finally we look at the class NC^1, this class contains all functions that have logarithmic depth non-uniform circuits where all gates have bounded fan-in.

Completeness. As discussed in Section 2.2, the SAT problem is NP-complete. This means that all NP problems can be transformed into SAT in polynomial time, and that SAT itself is an NP problem. This is not the only case of a logical problem being complete for a natural complexity class. In Figure 7, we show some of these important connections that are related to this work.

Complexity Class	Language Complete for Class
NP	SAT
coNP	TAUT
PSPACE	QBF
NEXPTIME	EPR

Fig. 7. Some important logics complete for complexity classes

First-order logic in general is not decidable, so one way to understand it from a complexity point of view is to study decidable fragments such as EPR. It is known that the EPR problem in general is not solvable in polynomial time and instead complete for non-deterministic exponential time (NEXPTIME) [86].

PSPACE is the class of all problems that can be decided with a polynomial (in the size of the input) amount of memory. The true Quantified Boolean Formulas (QBF) form a PSPACE-complete language. The QBF logic will be our main focus for this thesis. In Chapter 3 we will discuss the important preliminaries for QBF.

Chapter 3

Quantified Boolean Formulas

Quantified Boolean Formulas (QBFs) express formulas from propositional logic more succinctly and thus apply to further fields such and verification and planning. In Chapter 2 we introduced important practical and theoretical aspects of propositional logic, in this chapter we do the same for QBF.

In this chapter we study the preliminaries specific to QBF. We define the semantics of QBF in Section 3.1, using both the standard and game semantics. In Section 3.2 we discuss the solving techniques for QBF, including both CDCL and expansion techniques. We then introduce a number of proof systems for QBF in Section 3.3.

3.1 Quantified Boolean Formulas

Quantified Boolean Formulas extend propositional logic with quantifiers $\forall, \exists$ [76]. The standard semantics are that $\forall x.\, \Psi$ is satisfied by the same truth assignments as $\Psi[0/x] \wedge \Psi[1/x]$ and $\exists x.\, \Psi$ is satisfied by the same truth assignments as $\Psi[0/x] \vee \Psi[1/x]$.

Bound and Free Variables. We now express what it means to be a bound variable in a QBF. This can be described formally using set $\mathcal{B}_{\mathrm{v}}(\phi)$ for QBF ϕ, where every element is a bound variable. This can be defined inductively as below. We let x denote any propositional variable.

$$\mathcal{B}_{\mathrm{v}}(0) = \mathcal{B}_{\mathrm{v}}(1) = \mathcal{B}_{\mathrm{v}}(x) = \varnothing, \quad \mathcal{B}_{\mathrm{v}}(p \to q) = \mathcal{B}_{\mathrm{v}}(p) \cup \mathcal{B}_{\mathrm{v}}(q)$$

$$\mathcal{B}_{\mathrm{v}}(\exists x.\, \Psi) = \mathcal{B}_{\mathrm{v}}(\forall x.\, \Psi) = \mathcal{B}_{\mathrm{v}}(\Psi) \cup \{x\}$$

Similarly the set of free variables is given by $\mathcal{F}_{\mathrm{v}}(\Psi)$. We let x denote any propositional variable.

$$\mathcal{F}_{\mathrm{v}}(0) = \mathcal{F}_{\mathrm{v}}(1) = \varnothing, \quad \mathcal{F}_{\mathrm{v}}(x) = \{x\}$$

$$\mathcal{F}_{\mathrm{v}}(p \to q) = \mathcal{F}_{\mathrm{v}}(p) \cup \mathcal{F}_{\mathrm{v}}(q), \quad \mathcal{F}_{\mathrm{v}}(\exists x.\, \Psi) = \mathcal{F}_{\mathrm{v}}(\forall x.\, \Psi) = \mathcal{F}_{\mathrm{v}}(\Psi) \setminus \{x\}$$

In order to make manipulation of QBFs easier, we only allow QBFs Ψ where $\mathcal{F}_{\mathrm{v}}(\Psi) \cap \mathcal{B}_{\mathrm{v}}(\Psi)$ is empty, so, in fact, $\mathcal{F}_{\mathrm{v}}(\Psi) = \mathrm{var}(\Psi) \setminus \mathcal{B}_{\mathrm{v}}(\Psi)$. We also avoid quantifying the same variable twice. In both these cases we create new "fresh" variables to replace any duplicated ones. For example, using the semantic expansion definition of QBFs, $\forall y \exists x \phi = \exists x \phi[0/y] \wedge \exists x' \phi[1/y]$, we distinguish x from x' which is bound elsewhere.

Closed QBF. A closed QBF is a QBF where all variables are bound (cf. Example 1). A closed QBF must be either true or false, since if we semantically expand all the quantifiers we have a boolean connective structure on $0, 1$. Intuitively this gives a naïve exponential time algorithm for finding whether a QBF is true or false.

Example 1. $\exists y(\neg y \vee \neg z) \vee \forall x(z \wedge x)$ would *not* be a closed QBF because of the free variable z. Instead in order to make this closed z needs to be quantified i.e. $\forall z(\exists y(\neg y \vee \neg z) \vee \forall x(z \wedge x))$.

Prenex QBF. A prenex QBF is a QBF where all quantification is done outside of the propositional connectives. A prenex QBF Φ therefore consists of a propositional part ϕ called the matrix and a prefix of quantifiers Π and can be written as $\Phi = \Pi \cdot \phi$. When the propositional matrix of a prenex QBF is a CNF, then we have a PCNF (cf. Example 2). We can feasibly transform any QBF into prenex form by moving the quantifiers to the outside.

If we have that each variable is bound only once then this causes no issues. We then can transform the matrix into a CNF using Tseitin variables, these Tseitin variables need to be quantified and the quantifier of the new variable must occur to the right of all variables they depend on.

A prenex QBF without any variables in the prefix is just a propositional formula.

Example 2. $\forall x(\exists y \exists z(y \vee \neg z \wedge x)) \wedge (\forall a \exists b(x \vee a \wedge \neg b))$ is *not* in prenex form. However this can be easily remedied by taking $\exists y \exists z$ and $\forall a \exists b$ to the left as $\forall x \exists y \exists z \forall a \exists b(y \vee \neg z \wedge x) \wedge (x \vee a \wedge \neg b)$ ($\forall x \forall a \exists b \exists y \exists z(y \vee \neg z \wedge x) \wedge (x \vee a \wedge \neg b)$ also works). This still is not a PCNF but $(y \vee \neg z \wedge x)$ can be expanded to $(y \vee \neg z) \wedge (y \vee x)$ and $(x \vee a \wedge \neg b)$ can be expanded to $(x \vee a) \wedge (x \vee \neg b)$ using distributivity rules. This gives the following PCNF:

$$\forall x \exists y \exists z \forall a \exists b(y \vee \neg z) \wedge (y \vee x) \wedge (x \vee a) \wedge (x \vee \neg b).$$

Quantifier Order. Let $\mathcal{Q}_1 X_1 \ldots \mathcal{Q}_k X_k.\, \phi$ be our prenex QBF, where for $1 \leq i \leq k$ $Q_i \in \{\exists, \forall\}$ and X_i are pairwise disjoint sequences of variables. If $x \in X_i$, we say that x is at *level* i and write $\mathrm{lv}(x) = i$. We write $\mathrm{lv}(l)$ for $\mathrm{lv}(\mathrm{var}(l))$. In contrast to the level, the *index* $\mathrm{ind}(x)$ provides the more detailed information on the actual position of x in the prefix, i.e. all variables are indexed by $1, \ldots, n$ from left to right. We remark that if variables are permuted such that are still in the same level, truth is still preserved.

QBF Game Semantics. Often it is useful to think of a closed prenex QBF $\mathcal{Q}_1 X_1 \ldots \mathcal{Q}_k X_k.\, \phi$ as a *game* between the *universal* and the *existential player*. In the i-th step of the game, the player $\mathcal{Q}_i$ assigns values to all the variables X_i. The existential player wins the game if and only if the matrix ϕ evaluates to 1 under the assignment constructed in the game. The universal player wins if and only if the matrix ϕ evaluates to 0. Given a universal variable u with index i, a *strategy for* u is a function, which maps assignments of 0/1 values to the variables of lower index than u to $\{0, 1\}$ (the intended response for u). Therefore a QBF is false if and only if there exists a *winning strategy* for

the universal player, i.e. if the universal player has a strategy for all universal variables that wins any possible game [65] [5, Sec. 4.2.2] [95, Chap. 19].

The natural games semantics mean that QBFs are good at coding two-player games. Indeed, "Connect4" can be neatly coded in QBF [60], in addition to subproblems in checkers [4].

3.2 QBF Solving

In comparison to SAT, in the case of the *QBF problem*, instead of asking whether a satisfying assignment exists, we ask whether a closed QBF is true or false. We can also ensure every input QBF is in prenex form. Instead of being NP-complete it has a connection to PSPACE. This means that other PSPACE problems are reducible to QBF [95]. Note the PSPACE algorithm below in Algorithm 3. Note that the function Simplify may vary on the exact solver used but always performs both unit propagation and pure literal elimination. Likewise, the choice of l is intentionally ambiguous as the heuristic used can be decided in each individual solver.

In order to verify that this algorithm uses only polynomial space we observe that the algorithm calls other instances of itself but there can only be as many running instances as there are variables in the prefix at any one time.

Algorithm 3 The QDPLL algorithm

function QDPLL($\mathcal{Q}_1 X_1, \ldots, \mathcal{Q}_k X_k, \Phi$)
  Simplify Φ
  if $\Phi = \varnothing$ **then**
    return 1
  else if $\varnothing \in \Phi$ **then**
    return 0
  else
    Pick l as any literal whose variable x appears in X_1.
    if $\mathcal{Q}_1 = \forall$ and QDPLL($\mathcal{Q}_1 X_1 \setminus \{x\}, \mathcal{Q}_2 X_2 \ldots, \mathcal{Q}_k X_k, \Phi \cup \{\{l\}\}) = 0$ **then**
      return 0
    else if $\mathcal{Q}_1 = \exists$ and QDPLL($\mathcal{Q}_1 X_1 \setminus \{x\}, \mathcal{Q}_2 X_2 \ldots, \mathcal{Q}_k X_k, \Phi \cup \{\{l\}\}) = 1$ **then**
      return 1
    else
      return QDPLL($\mathcal{Q}_1 X_1 \setminus \{x\}, \mathcal{Q}_2 X_2 \ldots, \mathcal{Q}_k X_k, \Phi \cup \{\{\neg l\}\}$)

Much like the relation between DPLL and resolution there is a relation between QDPLL and QBF resolution. This is preserved when clauses-learning is added (QCDCL).

CEGAR Solving. *Counterexample guided abstraction refinement (CEGAR)* solving [41] is based on the principle of quantifier expansion. However, full expansion would lead to an exponential blow up every time. Instead, it queries winning strategies for the variables at the $n+1$th level in order to find counteractive winning strategies at the nth level. An example of a CEGAR algorithm is given in Algorithm 4.

Algorithm 4 A CEGAR algorithm that outputs winning strategies

```
function CEGAR SOLVE(QX. Φ)
    if Φ has no quant then
        return (Q = ∃) ? SAT(φ) : SAT(¬φ)
    ω ← ∅
    while true do
        α ← (Q = ∃) ? ⋀_{μ∈ω} Φ[μ] : ⋁_{μ∈ω} Φ[μ]
        τ′ ← CEGAR solve(prenex(QX. α))
        if τ′ = NULL then
            return NULL
        τ ← {l | l ∈ τ′ ∧ var(l) ∈ X}
        μ ← CEGAR solve(Φ[τ])
        if μ = NULL then
            return τ
        ω ← ω ∪ {μ}
```

CEGAR solving uses a different paradigm than QDPLL. While traces of QDPLL algorithms can be transformed in Q-Res, CEGAR has a relation to a different calculus known as ∀Exp+Res. This calculus will be introduced and explained in Chapter 4.

3.3 Proof Systems for Quantified Boolean Formulas

Resolution Systems. *Q-resolution* by Kleine Büning, Karpinski, and Flögel [77] is a resolution-like refutation system that operates on QBFs in prenex form where the matrix is a CNF. It uses the propositional resolution rule on existential variables with a side condition that prevents tautological clauses (for Figure 8 recall that $\neg\neg z = z$ for literals). We also forbid tautologies from being introduced in the proof system via axiom. In addition Q-resolution has a universal reduction rule to remove universal variables.

$$\frac{}{C}\ (\text{Ax}) \qquad\qquad \frac{C \vee x \qquad D \vee \neg x}{C \vee D}\ (\text{Res})$$

Ax: C is a non-tautological clause in the propositional matrix.
Res: variable x is existential and if literal $z \in C$, then $\neg z \notin D$.

$$\frac{C \vee u}{C} \qquad \frac{C \vee \neg u}{C}\ (\forall\text{-Red})$$

variable u is universal and all other existential variables $x \in C$ are left of u in the quantifier prefix.

Fig. 8. The rules of Q-Res [77]

Example 3. We wish to refute the following PCNF in Q-Res:

$$\exists x_1 \exists x_2 \forall y_1 \forall y_2 \exists x_3 (x_1 \vee y_1 \vee x_3) \wedge (\neg x_1 \vee \neg y_1 \vee x_3) \wedge (\neg x_2 \vee y_2 \vee \neg x_3) \wedge (x_2 \vee \neg y_2 \vee \neg x_3)$$

This QBF can be shown as false by a semantic argument, using the two-player game. As long as the universal player plays $y_1 \leftarrow x_1$ and $y_2 \leftarrow \neg x_2$, then whatever the existential player sets for x_3 one clause gets refuted.

A refutation is given in Figure 9. Note that we cannot perform universal reduction on any of the axioms. We first can resolve the x_3 variables in every possible combination. The resulting clauses unblock universal literals which can then be reduced. Finally we can use resolution to reach the empty clause.

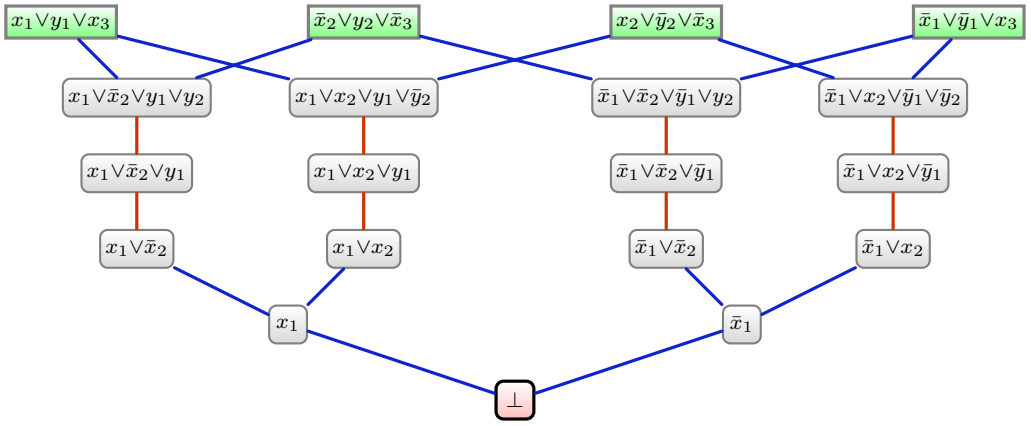


Fig. 9. An example of a Q-Res refutation

Q-Res has a number of augmentations. *Long-distance Q-resolution (LD-Q-Res)* appears originally in the work of Zhang and Malik [119] and was formalized into a calculus by Balabanov and Jiang [10]. It merges complementary literals of a universal variable u into the special literal u^*. These special literals behave like universal literals in that they can be reduced. The issue comes with the condition of merging. In particular, different literals of a universal variable u may be merged only if $\mathrm{ind}(x) < \mathrm{ind}(u)$, where x is the resolution variable. There are some restrictions on merging given in Figure 11. LD-Q-Res uses the rules L$\exists$R, $\forall$-Red and $\forall$-Red * of Figure 10.

Example 4. We again use the same false QBF from Example 3:

$$\exists x_1 \exists x_2 \forall y_1 \forall y_2 \exists x_3 (x_1 \vee y_1 \vee x_3) \wedge (\neg x_1 \vee \neg y_1 \vee x_3) \wedge (\neg x_2 \vee y_2 \vee \neg x_3) \wedge (x_2 \vee \neg y_2 \vee \neg x_3)$$

Here we will use L$\exists$R from Figure 10. Recall that our resolution pivot must be to the left of our merged universal variable, so we can use x_1 or x_2. We find examples of both that merge y_1 and y_2

$$\frac{}{C}\ \text{(Axiom)} \qquad \frac{D \cup \{u\}}{D}\ (\forall\text{-Red}) \qquad \frac{D \cup \{u^*\}}{D}\ (\forall\text{-Red}^*)$$

C is a clause in the matrix. Literal u is universal and $\mathrm{lv}(u) \geq \mathrm{lv}(l)$ for all $l \in D$.

$$\frac{C_1 \cup U_1 \cup \{x\} \qquad C_2 \cup U_2 \cup \{\neg x\}}{C_1 \cup C_2 \cup U}\ \text{(Res)}$$

We consider four instantiations of the Res-rule:
S∃R: x is an existential variable. If literal $l \in C_1$, then $\neg l \notin C_2$. $U_1 = U_2 = U = \emptyset$.
S∀R: x is an universal variable. Otherwise same conditions as S∃R.
L∃R: x is an existential variable.
If $l_1 \in C_1, l_2 \in C_2$, if $\mathrm{var}(l_1) = \mathrm{var}(l_2) = z$ then $l_1 = l_2 \neq z^*$. U_1, U_2 contain only universal literals with $\mathrm{var}(U_1) = \mathrm{var}(U_2)$. $\mathrm{ind}(x) < \mathrm{ind}(u)$ for each $u \in \mathrm{var}(U_1)$.
If $w_1 \in U_1, w_2 \in U_2$, $\mathrm{var}(w_1) = \mathrm{var}(w_2) = u$ then $w_1 = \neg w_2$, $w_1 = u^*$ or $w_2 = u^*$. $U = \{u^* \mid u \in \mathrm{var}(U_1)\}$.
L∀R: x is an universal variable. Otherwise same conditions as L∃R.

Fig. 10. The rules of Q-Res-based proof systems

respectively. Despite the restrictions on merged variables, y_1^* and y_2^* are indeed allowed in the same clause when we resolve on x_3. ∀-Red * steps finalise the proof deriving the empty clause. Here is a good example of the benefits of the merged variables– notice how the proof (given in Figure 12) is considerably more compact compared to the proof in Figure 9.

QU-resolution (QU-Res) [116] removes the restriction from Q-Res that the resolved variable must be an existential variable and allows resolution of universal variables. The rules of QU-Res are S∃R, S∀R and ∀-Red. *LQU$^+$-Res* [11] extends LD-Q-Res by allowing short and long distance resolution pivots to be universal. However, the pivot is never a merged literal z^*. LQU$^+$-Res uses the rules L∃R, L∀R, ∀-Red and ∀-Red *.

Stronger Calculi. In addition to calculi that work with resolution, there are calculi that work with Gentzen sequent systems. The central calculus here is the system G [81] which is the propositional system LK with four additional rules that deal with boolean quantifiers:

∀-introduction

$$\frac{\phi(\psi/x), \Gamma \longrightarrow \Delta}{\forall x\, \phi, \Gamma \longrightarrow \Delta}\ (\forall\text{-l}) \qquad \frac{\Gamma \longrightarrow \Delta, \phi(p/x)}{\Gamma \longrightarrow \Delta, \forall x\, \phi}\ (\forall\text{-r})$$

and ∃-introduction

$$\frac{\phi(p/x), \Gamma \longrightarrow \Delta}{\exists x\, \phi, \Gamma \longrightarrow \Delta}\ (\exists\text{-l}) \qquad \frac{\Gamma \longrightarrow \Delta, \phi(\psi/x)}{\Gamma \longrightarrow \Delta, \exists x\, \phi}\ (\exists\text{-r}).$$

Where ϕ and ψ are arbitrary QBF and p is a boolean variable.

For $i \geq 0$, G_i is a subsystem of G with cuts restricted to prenex $\Sigma_i^q \cup \Pi_i^q$-formulas (QBF restricted to i quantifier alternations) [43, 81].

C, D are clauses without x, u or v literals, u, v are universal.

$$\text{ind}(x) < \text{ind}(u)\ \frac{C \vee x \vee u \qquad D \vee \bar{x} \vee \bar{u}}{C \vee D \vee u^*} \qquad \text{ind}(u) < \text{ind}(x)\ \frac{C \vee x \vee u \qquad D \vee \bar{x} \vee \bar{u}}{\text{no resolution on } x}$$

$$\text{ind}(x) < \text{ind}(u)\ \frac{C \vee x \vee u^* \qquad D \vee \bar{x} \vee \bar{u}}{C \vee D \vee u^*} \qquad \text{ind}(u) < \text{ind}(x)\ \frac{C \vee x \vee u^* \qquad D \vee \bar{x} \vee \bar{u}}{\text{no resolution on } x}$$

$$\text{ind}(x) < \text{ind}(u)\ \frac{C \vee x \vee u \qquad D \vee \bar{x} \vee u^*}{C \vee D \vee u^*} \qquad \text{ind}(u) < \text{ind}(x)\ \frac{C \vee x \vee u \qquad D \vee \bar{x} \vee u^*}{\text{no resolution on } x}$$

$$\text{ind}(x) < \text{ind}(u)\ \frac{C \vee x \vee u^* \qquad D \vee \bar{x} \vee u^*}{C \vee D \vee u^*} \qquad \text{ind}(u) < \text{ind}(x)\ \frac{C \vee x \vee u^* \qquad D \vee \bar{x} \vee u^*}{\text{no resolution on } x}$$

$$\frac{C \vee x \vee u^* \qquad D \vee \bar{x}}{C \vee D \vee u^*} \qquad \frac{C \vee x \vee u^* \qquad D \vee \bar{x} \vee v^*}{C \vee D \vee u^* \vee v^*}$$

Fig. 11. Merging rules and restrictions

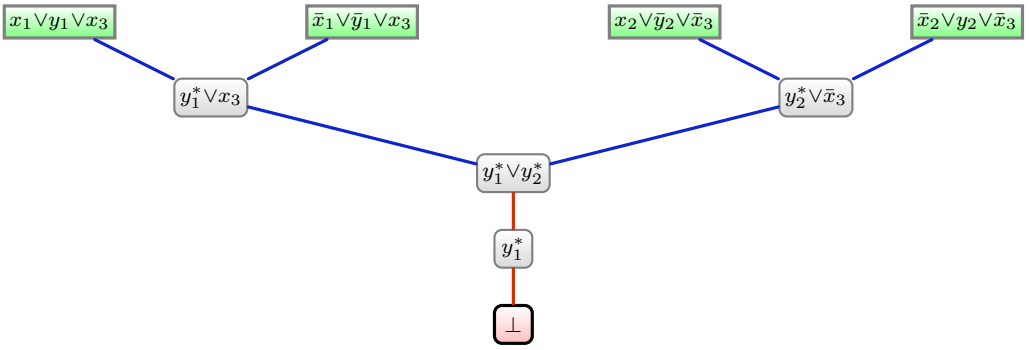


Fig. 12. An example of a LD-Q-Res refutation

Correspondence to Solving. With the exception of the powerful system G, the QBF proof systems discussed in this chapter correspond roughly to versions of the QDPLL and QCDCL algorithm. This means that Q-Res proofs can be extracted from runs of QCDCL solvers such as DepQBF. However, CEGAR solving and other expansion based techniques do not have this relation. In Chapter 4 we introduce expansion based proof systems that can deal with expansion based techniques.

Part II

QBF Resolution

Chapter 4

Expansion-based Resolution Calculi

In Chapter 3, we introduced the QBF proof system Q-resolution and its improvements LD-Q-Res, QU-Res and LQU$^+$-Res. These form the QBF Conflict-Driven Clause-Learning (QCDCL) style proof systems. However, many QBF solvers do not fit under the QCDCL style of solving– in particular, expansion based solvers such as RAReQS [72], Ghost-Q [78] and CAQE [101].

Recently, a proof system ∀Exp+Res was introduced [74] with the motivation to trace expansion-based QBF solvers [71]. ∀Exp+Res also uses resolution, but is conceptually very different from Q-Res. Conceptually, the variants of Q-resolution correspond to solving QBF by *search and clause-learning*. In contrast, expansion-based systems correspond to solving QBF by *rewriting quantifiers into Boolean connectives*, such rewrites may be iterative and may require introduction of fresh variables. Hence, the division between expansion-based systems and Q-resolution based systems is naturally reflected in solving techniques.

In this chapter we create a new calculus IR-calc that naturally unifies the concepts in Q-Res and ∀Exp+Res. The inspiration for this calculus comes from a known QBF connection to first order logic [109], in particular the fragment EPR, also known as the Bernays-Schönfinkel class. The rules of IR-calc simplify the rules of first order resolution [107] in way that is congruent with the expansion based ∀Exp+Res. In this way IR-calc is a natural evolution of ∀Exp+Res that uses the partial expansion of universal variables as an advantage.

We offer an additional improvement IRM-calc, which simulates IR-calc and LD-Q-Res. IRM-calc roughly corresponds to IR-calc but introduces merged annotations to simulate the merged universal variables in LD-Q-Res.

As with any calculi, we need to establish soundness. For this we use the two-player game semantics for QBF given in Section 3.1. We show that any refutation in IRM-calc has a winning strategy for the universal player. In fact, we show that such a strategy can be extracted in polynomial time from the refutation. A similar technique is used for Q-Res [65] and LD-Q-Res [53].

Completeness of the calculi comes from their simulations of other complete calculi. IR-calc simulates both Q-Res and ∀Exp+Res and IRM-calc simulates LD-Q-Res.

We organise this chapter in the following way. Firstly, we define the new expansion calculi IR-calc and IRM-calc (Section 4.1), with examples (Section 4.2). We then show their soundness by strategy extraction — a very desirable property in practise (Section 4.3), and show that they simulate previously defined QBF resolution systems (Section 4.4), hence showing completeness. Since these calculi simulate both expansion and reduction calculi, they help unify the two concepts by looking at a single powerful concept: instantiation.

The work in this chapter appeared at the 39th International Symposium on Mathematical Foundations of Computer Science Science [18].

4.1 Introducing the Expansion Calculi IR-calc and IRM-calc

In contrast to the QBF resolution systems from the previous section the expansion calculi we will consider here use only existential variables. Before giving the technical details, let us try to explain the idea of expansion of universal variables.

Consider the QBF $\exists x \forall u \exists y.\, \phi(x, u, y)$. Expanding the universal variable u, this is semantically equivalent to $\exists x \exists y^0 \exists y^1.\, \phi(x, 0, y^0) \wedge \phi(x, 1, y^1)$, where we need to create two fresh copies of the existential variables right of u (in this case we create two copies y^0 and y^1, but keep x). In fact, instead of just renaming the variable y to y^0 and y^1 it will be more convenient to record in the annotation, which expansion caused the renaming. In our example, we would name y^0 as $y^{0/u}$ and y^1 as $y^{1/u}$ to remember that we created the copies of y by expanding u.

In this way we expand all the universal variables in the QBF, which results in a purely existentially quantified formula. Of course, in general this will produce an exponential increase in the formula size. However, in some cases it may suffice to expand a universal variable in just one polarity and still preserve the falsity of the QBF. Consider the example $\forall u \exists x.\, (u \vee x) \wedge (u \vee \neg x)$, where we can just expand u by 0 to obtain the false QBF $\exists x^{0/u}.\, x^{0/u} \vee \neg x^{0/u}$. We refer to [74] for more background information on expansion.

This approach is in line with the workings of the highly competitive QBF solver RAReQS [72], which gradually expands all universal variables to obtain a purely existential formula that is then passed to a SAT solver. Proof theoretically, RAReQS corresponds to the calculus $\forall$Exp+Res [74], introduced below. On a technical note, the solver also simultaneously expands the formula's negation, which enables proving true formulas.

We now start to set up the formal framework allowing us to define our new calculi. The framework hinges on the concept of annotated clauses.

Definition 2.

1. *An* extended assignment *is a partial mapping from the universal variables to* $\{0, 1, *\}$. *We denote an extended assignment by a set or list of individual replacements i.e.* $0/u, */v$ *is an extended assignment.*
2. *An* annotated clause *is a clause where each literal is annotated by an extended assignment to universal variables.*
3. *For an extended assignment* σ *to universal variables we write* $l^{\mathsf{restrict}_l(\sigma)}$ *to denote an annotated literal where* $\mathsf{restrict}_l(\sigma) = \{c/u \in \sigma \mid \mathrm{lv}(u) < \mathrm{lv}(l)\}$.
4. *Two (extended) assignments* τ *and* μ *are called* contradictory *if there exists a variable* $x \in \mathsf{dom}(\tau) \cap \mathsf{dom}(\mu)$ *with* $\tau(x) \neq \mu(x)$.

For extended assignments, we remark that this $*$ notation has nothing to do with the proof complexity convention to map unassigned variables to $*$. Rather, a variable u will be mapped to $*$ if u^* appears in some clause. This will made precise later. In contrast to extended assignments, an assignment has range $\{0, 1\}$, as usual. To highlight the difference, we will sometimes refer to assignments as 0/1 assignments.

Further we define operations that let us modify annotations of a clause by *instantiation*.

Definition 3. *For (extended) assignments τ and μ, we write $\tau \circ \mu$ for the assignment σ defined as follows: $\sigma(x) = \tau(x)$ if $x \in \mathsf{dom}(\tau)$, otherwise $\sigma(x) = \mu(x)$ if $x \in \mathsf{dom}(\mu)$. The operation $\tau \circ \mu$ is referred to as* completion *because μ provides values for variables that are not defined in τ.*

The completion operation is associative and therefore we can omit parentheses. However, it is *not* commutative. The following properties hold: (i) for non-contradictory μ and τ, we have $\mu \circ \tau = \tau \circ \mu = \mu \cup \tau$. (ii) $\tau \circ \tau = \tau$.

We also need an auxiliary operation of instantiation, which completes the annotations of literals in a clause by a partial (extended) assignment.

Definition 4. *For an extended assignment τ and an annotated clause C, the function $\mathsf{inst}(\tau, C)$ returns the annotated clause $\left\{ l^{\mathsf{restrict}_l(\sigma \circ \tau)} \mid l^\sigma \in C \right\}$.*

We are now ready to describe the expansion QBF systems.

The first of these is the *calculus ∀Exp+Res from [74]*. In Figure 13 we present an adapted version of this calculus so that it is congruent with the new concepts presented here (semantically it is the same as in [74]). The axiom rule for ∀Exp+Res simply downloads a clause by picking a *total* 0/1 assignment to the universal variables, applies it to the clause, and records the used assignment in the annotations.

Note that τ is always a complete assignment to all universal variables. However, when appearing as an annotation to an existential variable x, it is truncated to the universal variables left of x. This is reflected in the notation $\mathsf{restrict}_x(\tau)$ (cf. Definition 2). For example, if we have a formula $\forall u \exists x \forall v \exists y.\, \phi$ and τ assigns u and v to 0, then a clause $u \vee x \vee v \vee y$ from ϕ would be instantiated to $x^{0/u} \vee y^{0/u,0/v}$.

The resolution rule of ∀Exp+Res is just the propositional resolution rule. Note, however, that the pivot annotations need to match exactly in both parent clauses. This makes sense, because in our framework, different annotations formally lead to distinct variables.

In ∀Exp+Res, proofs can be easily split into two phases. The first consists only of axiom downloads, removing all universal variables and annotating the existential variables. This is followed by the second phase, consisting only of classical resolution steps on the new annotated variables, cf. Section 4.2 for an example.

We now describe *our first new system IR-calc*. Like ∀Exp+Res it completely eliminates universal variables and operates with annotated clauses. The main difference is that, unlike in ∀Exp+Res, where existentials are always annotated by all universals preceding them in the prefix, the system IR-calc can deal with partial assignments. The calculus introduces clauses from the matrix and allows to **i**nstantiate and **r**esolve clauses; hence the name IR-calc. It comprises the rules in Figure 14.

The axiom download rule is similar to ∀Exp+Res, but only annotates the existential literals with those preceding universals which actually occur in the clause. For example, if $\forall u \exists x \forall v \exists y.\, \phi$

$$\frac{}{\left\{l^{\mathsf{restrict}_l(\tau)} \mid l \in C, l \text{ is existential}\right\} \cup \{\tau(l) \mid l \in C, l \text{ is universal}\}} \text{(Axiom)}$$

C is a clause from the matrix and τ is an assignment to all universal variables.

$$\frac{C_1 \cup \{x^\tau\} \qquad C_2 \cup \{\neg x^\tau\}}{C_1 \cup C_2} \text{(Res)}$$

Fig. 13. The rules of ∀Exp+Res (adapted from [74])

contains the clause $x \vee v \vee y$ we download it in IR-calc as $x \vee y^{0/v}$, whereas in ∀Exp+Res we also need to choose a value for u and could either download it as $x^{0/u} \vee y^{0/u,0/v}$ or $x^{1/u} \vee y^{1/u,0/v}$.

The resolution rule is identical to ∀Exp+Res. Again, annotations in the pivot need to match precisely, and for this reason we also adopt an instantiation rule, which can increase annotations in the clause. This can enable further resolution steps. Again, in instantiation steps we need to truncate the annotations to universal variables left of the annotated existential variable, as indicated by the notation $l^{\mathsf{restrict}_l(\sigma \circ \tau)}$ in Definition 4.

Unlike in the ∀Exp+Res system, IR-calc proofs are no longer separated into an annotation and a resolution phase, but can mix instantiation and resolution steps in the proof by "delayed instantiations" as and when they are needed. An example is contained in Section 4.2.

$$\frac{}{\left\{l^{\mathsf{restrict}_l(\tau)} \mid l \in C, l \text{ is existential}\right\}} \text{(Axiom)}$$

C is a non-tautological clause from the matrix. $\tau = \{0/u \mid u \text{ is universal in } C\}$, where the notation $0/u$ for literals u is shorthand for $0/x$ if $u = x$ and $1/x$ if $u = \neg x$.

$$\frac{x^\tau \vee C_1 \qquad \neg x^\tau \vee C_2}{C_1 \cup C_2} \text{(Resolution)} \qquad \frac{C}{\mathsf{inst}(\tau, C)} \text{(Instantiation)}$$

τ is an assignment to universal variables with $\mathsf{rng}(\tau) \subseteq \{0, 1\}$.

Fig. 14. The rules of IR-calc

Note that in ∀Exp+Res, propositional variables are introduced so that their annotations assign *all* relevant variables. In this way, each literal corresponds to a value of a Skolem function in a specific point. In contrast, in IR-calc, variables are annotated "lazily", i.e., it enables us to reason about multiple points of Skolem functions at the same time. This is analogous to *specialization* of free variables by constants in first-order logic (FO). Similarly, resolution in IR-calc is analogous to resolution in Robinson's FO resolution [107]. Chapter 10 and the recent paper [52] further explore

Axiom and instantiation rules as in IR-calc in Figure 14.

$$\frac{x^{\tau\cup\xi} \vee C_1 \qquad \neg x^{\tau\cup\sigma} \vee C_2}{\mathsf{inst}(\sigma, C_1) \cup \mathsf{inst}(\xi, C_2)} \text{ (Resolution)}$$

$\mathsf{dom}(\tau)$, $\mathsf{dom}(\xi)$ and $\mathsf{dom}(\sigma)$ are mutually disjoint. $\mathsf{rng}(\tau) = \{0, 1\}$

$$\frac{C \vee b^{\mu} \vee b^{\sigma}}{C \vee b^{\xi}} \text{ (Merging)}$$

$\mathsf{dom}(\mu) = \mathsf{dom}(\sigma)$. $\xi = \{c/u \mid c/u \in \mu, c/u \in \sigma\} \cup \{*/u \mid c/u \in \mu, d/u \in \sigma, c \neq d\}$

Fig. 15. The rules of IRM-calc

the connection between IR-calc and FO resolution. From the FO perspective, IR-calc appears to be a very natural proof system.

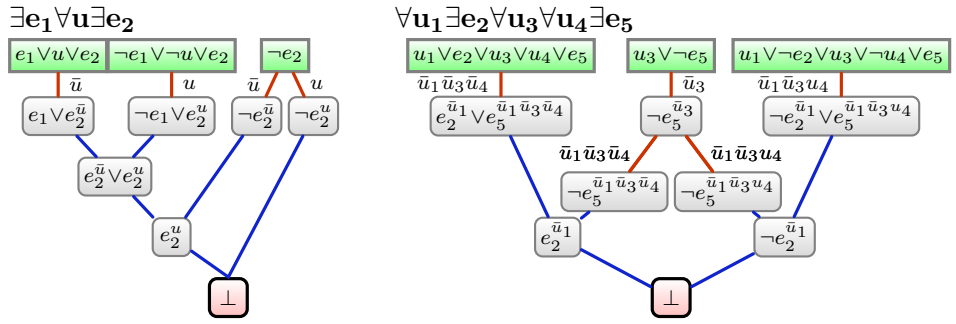
Our second new system IRM-calc is an extension of IR-calc where we allow as annotations extended assignments with range $\{0, 1, *\}$. The purpose of $*$ is similar to the role of $*$ in LD-Q-Res (cf. the remarks in Chapter 3).

Axioms and instantiation rules of IRM-calc are handled as in IR-calc, which do not create $*$. The annotation $*$ may be introduced by a new rule called *merging*. It merges two literals on the same existential variable to one copy where all conflicting annotations produce $*$. For example, the merge of $x^{0/u,0/v,0/w}$ and $x^{0/u,1/v,*/w}$ produces $x^{0/u,*/v,*/w}$.

The resolution rule is defined slightly differently, annotations of the pivot need not necessarily match before application of the rule, but are matched "on the fly" while resolving (the resolution rule is designed such that this matching is possible). More precisely, the resolution rule in Figure 15 uses pivot x, annotated as $x^{\tau\cup\xi}$ in the first parent clause and as $\neg x^{\tau\cup\sigma}$ in the second parent, where ξ and σ are partial extended assignments with disjoint domain. Thus, by instantiating the first parent with σ and the second parent with ξ we create a pivot $x^{\tau\cup\xi\cup\sigma}$ with matching annotations and obtain a sound resolution step. The resolution rule can now deal with $*$, but when $\xi = \sigma = \emptyset$ we have exactly the resolution rule from Figure 14. Thus IRM-calc encompasses IR-calc.

IRM-calc is defined in Figure 15, an example is contained in Section 4.2.

Intuitively, a literal $x^{0/u}$ asserts that any interpretation of the Skolem function for x must yield 1 whenever $u = 0$. Thus the clause $x^{1/u} \vee x^{0/u}$ asserts that for any value of the remaining relevant universal variables, the Skolem function must yield 1 on $u = 1$ or on $u = 0$. Note that this does *not* mean that the function has to be constantly 1. Merging enables compressing such clauses. For instance, in IR-calc, to refute the clauses $\{\neg x^{1/w},\ x^{0/u} \vee x^{1/u}\}$ one would apply the instantiation rule to obtain $\{\neg x^{0/u,1/w}, \neg x^{1/u,1/w}, x^{0/u,1/w} \vee x^{1/u,1/w}\}$ and then proceed as in classical propositional resolution. In contrast, IRM-calc enables merging the binary clause into the unit clause $x^{*/u}$, which in turn gives a contradiction with the original clause $\neg x^{1/w}$.



(a) An example proof in ∀Exp+Res

(b) An example proof in IR-calc

(c) An example proof in IRM-calc

Fig. 16. Proof examples

4.2 Proof Examples

We illustrate the expansion calculi from Section 4.1 with a few examples. For existential literal l and universal variable u we write l^u as a shorthand for $l^{1/u}$ and $l^{\bar{u}}$ for $l^{0/u}$.

Example 5. Figure 16(a) exemplifies a proof in ∀Exp+Res. In this case there is only one universal variable u. The variable needs to be instantiated whenever a new clause is introduced into the proof. In the case of the clause $\neg e_2$ there are two options for instantiation. In the case of the two other clauses only one of the instantiations is useful. Note that only e_2 is annotated by u because $\mathrm{lv}(e_1) < \mathrm{lv}(u)$.

Example 6. Figure 16(b) exemplifies a proof in IR-calc; most notably, $\neg e_5$ needs to be annotated only by $\bar{u}_3$ when it enters the proof.

Example 7. Figure 16(c) shows an IRM-calc refutation, containing $*$ in an annotation of e_5.

Example 8. Consider the (true) QBF $\exists x \forall u w \exists b.\, (x \vee u \vee b) \wedge (\neg x \vee \neg u \vee b) \wedge (u \vee w \vee \neg b)$. In both calculi axioms yield $(x \vee b^{0/u})$, $(\neg x \vee b^{1/u})$, and $(\neg b^{0/w,0/u})$. IR-calc enables deriving $(b^{0/u} \vee b^{1/u})$ from the first two clauses. IRM-calc further enables deriving $b^{*/u}$ by merging. Intuitively, $(b^{0/u} \vee b^{1/u})$ means that the existential player must play so that for any assignment

to w it holds that $b = 1$ if $u = 0$ or if $u = 1$. For instance, the player might choose to play $b = 1$ if $w = 0, u = 1$ and if $w = 1, u = 0$ (and otherwise $b = 0$). The clause $b^{*/u}$ can be seen as a shorthand for the clause $(b^{0/u} \vee b^{1/u})$. Note that it would be *unsound* to derive the clause (b) (with no annotation). This would mean that b must be 1 regardless of the moves of the universal player. However, b needs to be 0 when $u = w = 0$ due to the axiom $(\neg b^{0/w,0/u})$.

Example 9. Consider again the QBF $\exists x \forall u w \exists b.\, (x \vee u \vee b) \wedge (\neg x \vee \neg u \vee b) \wedge (u \vee w \vee \neg b)$ from Example 8. If the third clause of the formula is changed to $\neg b$, the formula becomes false, which is shown by instantiating $\neg b$ to $\neg b^{0/u}$ and to $\neg b^{1/u}$, using those to obtain x and $\neg x$ by resolution and deriving the empty clause.

Example 10. Consider the QBF $\exists x \forall u \exists bc.\, (x \vee u \vee b) \wedge (\neg x \vee \neg u \vee c) \wedge (\neg b \vee c) \wedge (\neg c)$. The following derivation is possible in IR-calc. Resolving $x \vee b^{0/u}$ and $\neg x \vee c^{1/u}$ yields $b^{0/u} \vee c^{1/u}$. Instantiating $\neg b \vee c$ by $0/u$ gives $\neg b^{0/u} \vee c^{0/u}$, resolving this with the previous resolvent yields $c^{0/u} \vee c^{1/u}$. Refutation can be obtained by instantiation of $\neg c$ once by $0/u$ and once by $1/u$ and subsequent two resolution steps. In IRM-calc it is possible to obtain $c^{*/u}$ by merging and resolve that directly with $\neg c$, which yields $\bot$.

4.3 Soundness and Extraction of Winning Strategies

The purpose of this section is twofold: to show how to obtain a *winning strategy* for the universal player from an IR-calc proof, and, to show that IR-calc is *sound*. First we show how to obtain a winning strategy for the universal player from a proof. From this, the soundness of the calculus follows because a QBF is false if and only if such a strategy exists. We will then explain how to extend this technique to IRM-calc.

The approach we follow is similar to the one used for Q-Res [65] or LD-Q-Res [53]. For this approach to work for a QBF proof system P we need two properties:

1. The proof system P shall be *closed under restrictions*, i.e., if π is a P-proof of $\mathcal{Q}X.\, \Phi$ (where Φ may contain further quantifiers) and α is an assignment to the variables X, then we can efficiently construct from π and α a P-proof π_α of $\Phi|_\alpha$.
2. Restricting a proof π by an assignment to the leftmost block of existential variables to a proof π_α as in item 1, we can *read off from* π_α *a response* for the universal player for the next round.

We therefore consider QBFs in the form $\Gamma = \exists E \forall U.\, \Phi$, where E and U are sets of variables and Φ is a QBF (potentially with further quantification beginning with $\exists$). Suppose π is an IR-calc refutation of Γ, and let ϵ be a total assignment to the variable block E. The assignment ϵ represents a move of the existential player. We reduce π to a refutation π_ϵ of the restricted formula $\forall U.\, \Phi|_\epsilon$. To obtain a response of the universal player, we construct from π_ϵ an assignment μ to the variables U such that reducing π_ϵ by μ gives a refutation of $\Phi|_{\epsilon \cup \mu}$.

Now let $\pi_{\epsilon,\mu}$ be the proof resulting from restricting π_ϵ by μ. The game continues with $\Phi|_{\epsilon\cup\mu}$ and $\pi_{\epsilon,\mu}$. In each of these steps, two quantifier levels are removed from the given QBF and a

refutation for each of the intermediate formulas is produced. This guarantees a winning strategy for the universal player because in the end the existential player will be faced with an unsatisfiable formula without universal variables.

We now implement this approach for IR-calc. We start with item 1, showing closure of IR-calc under restrictions.

Lemma 1. *Let π be an IR-calc refutation of Φ and let ϵ be a partial assignment of existential variables from Φ. We can then construct from π a refutation π_ϵ of $\Phi|_\epsilon$.*

Proof. Let ϵ be a partial assignment to the existential variables of Φ and let $\pi = (C_1, \ldots, C_m = \bot)$. We describe the construction of the restricted proof π_ϵ and simultaneously argue that it is a valid refutation. For this we define inductively clauses C'_i such that

1. if $C'_i \neq \top$ then $C'_i \subseteq C_i$ and C'_i has a valid IR-calc derivation in π_ϵ.
2. if $C'_i = \top$ then ϵ satisfies C_i, i.e., $\epsilon(x) = 1$ for some $x^\tau \in C_i$.

We note that from these two conditions it follows that π_ϵ contains a refutation, because $C_m = \bot$ and hence $C'_m = \bot$, as ϵ does not satisfy C_m.

We now inductively construct C'_i and show the two items above.

Let C_i be derived by the axiom rule. If ϵ satisfies a literal l with $l^\tau \in C_i$ for some annotation τ we set $C'_i = \top$. Otherwise we define $C'_i = C_i \setminus \{l^\tau \mid l \text{ is a literal falsified by } \epsilon \text{ and } \tau \text{ is an annotation}\}$.

If C_i is derived by instantiating C_j by τ and $C'_j \neq \top$, then we set $C'_i = \mathsf{inst}(\tau, C'_j)$. Obviously, $C'_i \subseteq C_i$. If $C'_i = \top$ then also $C'_j = \top$ and ϵ satisfies C_j by inductive hypothesis. Hence ϵ also satisfies C_i in the sense of condition 2.

Assume now that C_i was derived by resolution from C_j and C_k with pivot $x^\tau \in C_j$ and $\neg x^\tau \in C_k$. We distinguish four cases.

In the first case we have $x^\tau \in C'_j$ and $\neg x^\tau \in C'_k$. We perform the resolution step and set $C'_i = C'_j \cup C'_k \setminus \{x^\tau, \neg x^\tau\}$. We then have $C'_i \subseteq C_j \cup C_k \setminus \{x^\tau, \neg x^\tau\} = C_i$, fulfilling condition 1.

In the second case we have $x^\tau \notin C'_j$ and $C'_j \neq \top$, and set $C'_i = C'_j$. Then $C'_i \subseteq C_j \setminus \{x^\tau\} \subseteq C_i$, fulfilling condition 1. (In the case of $\neg x^\tau \notin C'_k$ and $C'_k \neq \top$ we analogously set $C'_i = C'_k$.)

The third case is $x^\tau \in C'_j$ and $C'_k = \top$. Then we set $C'_i = \top$. We have to show condition 2. By inductive hypothesis, ϵ satisfies C_k, hence ϵ satisfies some literal $l \in C_k$. Necessarily, $l \neq \neg x^\tau$, because otherwise ϵ falsifies x^τ and thus $x^\tau \notin C'_j$, contradicting our assumption. Thus $l \in C_k \setminus \{\neg x^\tau\} \subseteq C_i$, which satisfies condition 2.

In the last case, $C'_j = C'_k = \top$. We set $C'_i = \top$. By inductive hypothesis, ϵ satisfies both C_j and C_k, hence by soundness of the resolution rule also C_i.

Lemma 2. *Similarly as in Lemma 1, we can show closure of IRM-calc under restrictions.*

Proof. The proof is similar in structure, but we have to be careful of the following:

– We also need to consider the merge rule.

– IRM-calc has an instantiation part to the resolution rule, so dummy instantiation steps may have to occur in the reduced proof in place of resolution steps.
– We cannot instantiate $*$ annotations outside of the resolution rule, so we have to instantiate instead by a constant (e.g. 0). This does not cause any invalid inferences as $*$ is more restrictive than constants.
– Annotations may differ between the original and reduced proof on $*$, so the clauses in the reduced proof may not be subsets of the clauses in the original proof. In order to make sure the clauses in the reduced proof are smaller than the original, we inductively define an injection from the reduced clause to the original.
– To define such an injection, it will be convenient to treat contraction of two identical literals to one literal as a separate rule and not perform contraction automatically in the proof system. This does not affect the complexity of the calculus. Contraction can also be thought of as a special case of the merging rule.

Details of the reduction are shown in Figure 17. By induction on the derivation depth we show that this construction yields a valid IRM-calc refutation π_ϵ of $\forall U.\, \Phi|_\epsilon$.

The induction hypothesis states that any derived clause C' in the reduced proof corresponding to clause C in the original proof has a valid derivation $\pi_{C'}$. Further, if C' is non-tautologous there is an injection $f_C : C' \to C$, where $f_C(l^{\sigma'}) = l^\sigma$ and σ satisfies the following: $\mathsf{dom}(\sigma') = \mathsf{dom}(\sigma)$ and for every $c/u \in \sigma'$ there is exactly one $d/u \in \sigma$ where $d = c$ or $d = *$. If C' is tautologous then C is satisfied by ϵ.

The purpose of this injection is to ensure that in the reduced proof the clauses have at most the number of literals in the original proof. This along with the condition on tautological clauses ensures that the reduced proof is a refutation. The condition on f_C regarding annotations allows resolution to occur whenever the pivot literals are present.

Base Case. In the axiom step (A), the clause in the reduced proof either has some literals removed, or is set to $\top$ when ϵ satisfies the clause.

Instantiation. An instantiation step (I) keeps the number of literals exactly the same, giving rise to a bijection between literals in C and $\mathsf{inst}(\sigma, C)$.

Merging. Consider the first merge case (M1) in Figure 17. In the reduced clause the literals l^τ and l^σ become $l^{\tau'}$ and $l^{\sigma'}$, respectively. Some $*$ in the annotations τ, σ might be 0 or 1 in τ' or σ'. However, the domains of τ, τ', σ, and σ' are all equal. The reduced proof will contain a merge of $l^{\tau'}$ and $l^{\sigma'}$.

In order to satisfy the annotation condition, we note that wherever a $*/u$ appears in ξ', a $*/u$ will appear in ξ, since it either comes from a $*$ itself or a $0/1$ conflict. Either the $0/1$ conflict is present in the original merge or there is a $*$ there by the condition from the induction hypothesis. This means our condition is satisfied.

Now consider the second merge case (M2) in Figure 17. If the reduced clause contains only one of the literals, say $l^{\tau'}$, the merge step is not performed in π_ϵ. The injection is constructed

Rule	Original proof step leading to clause D	Proof step reduced by ϵ & case condition	Construction of f_D
A	$\overline{\bigvee_{l\in C, l\in\exists} l^{[\tau]}}$ C is a clause in ϕ. τ falsifies all $\forall$ lit in C.	$\overline{\bigvee_{l\in C, l\in\exists, l\notin \mathrm{var}(\epsilon)} l^{[\tau]}}$ ϵ does not satisfy C.	$f_D(l^\tau) = l^\tau$
I	$\dfrac{C}{\bigvee_{l^\tau\in C} l^{\tau\circ\sigma}}$	$\dfrac{C'}{\bigvee_{l^\tau\in C'} l^{\tau\circ\sigma}}$	$f_D(\mathsf{inst}(\sigma, l^\tau))$ $= \mathsf{inst}(\sigma, f_C(l^\tau))$
M1	$\dfrac{C = K \vee l^\tau \vee l^\sigma}{K \vee l^\xi}$	$\dfrac{K'\vee l^{\tau'}\vee l^{\sigma'}}{K'\vee l^{\xi'}}$ $f_C(l^{\tau'}) = l^\tau$ $f_C(l^{\sigma'}) = l^\sigma$	$f_D(x) = \begin{cases} f_C(x), & x \in K' \\ l^\xi, & x = l^{\xi'} \end{cases}$
M2		$\dfrac{C'=K'\vee l^{\tau'}}{K'\vee l^{\tau'}}$ $f_C(l^{\tau'}) = l^\tau$ $\forall l^{\sigma'} \in C', f_C(l^{\sigma'}) \neq l^\sigma$	$f_D(x) = \begin{cases} f_C(x), & x \in K' \\ l^\xi, & x = l^{\tau'} \end{cases}$
M3		$\dfrac{C'}{C'}$ $\forall l^{\sigma'} \in C', l^\tau \neq f_C(l^{\sigma'}) \neq l^\sigma$	$f_D(x) = f_C(x)$
R1	$\dfrac{C_1 \qquad C_2}{\mathsf{inst}(\xi, K_1) \vee \mathsf{inst}(\sigma, K_2)}$ $C_1 = K_1 \vee x^{\tau\circ\sigma}$ $C_2 = K_2 \vee \neg x^{\tau\circ\xi}$ τ, σ, ξ pairwise disjoint $\mathsf{rng}(\tau) = \{0, 1\}$	$\dfrac{K_1'\vee x^{\tau\circ\sigma'} \quad K_2'\vee\neg x^{\tau\circ\xi'}}{\mathsf{inst}(\xi', K_1')\vee\mathsf{inst}(\sigma', K_2')}$ $f_{C_1}(x^{\tau\circ\sigma'}) = x^{\tau\circ\sigma}$ $f_{C_2}(\neg x^{\tau\circ\xi'}) = \neg x^{\tau\circ\xi}$	$f_D(\mathsf{inst}(\beta, l^\alpha)) =$ $\begin{cases} \mathsf{inst}(\beta, f_{C_1}(l^\alpha)) & l^\alpha \in C_1' \\ \mathsf{inst}(\beta, f_{C_2}(l^\alpha)) & l^\alpha \in C_2' \end{cases}$
R2		$\dfrac{C_1' \quad C_2'}{\mathsf{inst}(\xi', C_1')}$ $\forall l^\alpha \in C_1', f_{C_1}(l^\alpha) \neq x^{\tau\circ\sigma}$ $\xi'(u) = \begin{cases} 0 & \xi(u) = * \\ \xi(u) & \text{else} \end{cases}$	$f_D(\mathsf{inst}(\xi', l^\alpha))$ $= \mathsf{inst}(\xi, f_{C_1}(l^\alpha))$
R3		$\dfrac{K_1' \vee x^{\tau\circ\sigma'} \qquad \top}{\top}$ $f_{C_1}(x^{\tau\circ\sigma'}) = x^{\tau\circ\sigma}$	

Fig. 17. Transformation of IRM-calc proof steps under the restriction ϵ with conditions and construction of the injection f_D

straightforwardly: the literal $l^{\tau'}$ in the new proof is mapped to the merged literal in the original proof and all other literals remain the same. The third case (M3) is similar to the second, but simpler.

Merging does not affect the presence of a literal that is satisfied by ϵ.

Contraction. Contraction is a special case of merging, so the same argument applies.

Resolution. Consider the first resolution case (R1) in Figure 17. The function f_D is an injection due to the induction hypothesis, as well as the fact that we do not perform contraction in this step. We need to show the required properties hold. Suppose, without loss of generality, $l^{\alpha'} \in K_1'$ and $f_{C_1}(l^{\alpha'}) = l^\alpha$. Since $\mathsf{dom}(\alpha) = \mathsf{dom}(\alpha')$ and $\mathsf{dom}(\xi) = \mathsf{dom}(\xi')$, then $\mathsf{dom}(\alpha \circ \xi) = \mathsf{dom}(\alpha) \cup \mathsf{dom}(\xi) = \mathsf{dom}(\alpha') \cup \mathsf{dom}(\xi') = \mathsf{dom}(\alpha' \circ \xi')$. This shows that the domains remain the same under f_D, which is one part of the condition. For the remaining part we consider two cases; either $u \in \mathsf{dom}(\alpha)$ or $u \notin \mathsf{dom}(\alpha)$, and we let $c \in \{0, 1\}$.

For $u \in \mathsf{dom}(\alpha)$ we have

$$c/u \in \alpha \Rightarrow c/u \in \alpha' \Rightarrow c/u \in \alpha' \circ \xi' \qquad \text{and} \qquad */u \in \alpha' \Rightarrow */u \in \alpha \Rightarrow */u \in \alpha \circ \xi.$$

Now suppose $u \in \mathsf{dom}(\alpha \circ \xi)$ and $u \notin \mathsf{dom}(\alpha)$. Then

$$c/u \in \xi \Rightarrow c/u \in \xi' \Rightarrow c/u \in \alpha' \circ \xi' \qquad \text{and} \qquad */u \in \xi' \Rightarrow */u \in \xi \Rightarrow */u \in \alpha \circ \xi.$$

Hence the inductive claim holds.

Consider now the second case (R2). Here C_1' is simply instantiated, crucially the pivot literal is missing from C_1', so f_D is an injection. In order to show the inductive condition, we suppose again that $l^{\alpha'} \in K_1'$ and $f_{C_1}(l^{\alpha'}) = l^{\alpha}$. Since $\mathsf{dom}(\alpha) = \mathsf{dom}(\alpha')$ and $\mathsf{dom}(\xi) = \mathsf{dom}(\xi')$, then $\mathsf{dom}(\alpha \circ \xi) = \mathsf{dom}(\alpha' \circ \xi')$. Now let $c \in \{0, 1\}$ and suppose $u \in \mathsf{dom}(\alpha)$. As in (R1) we have $c/u \in \alpha \Rightarrow c/u \in \alpha' \Rightarrow c/u \in \alpha' \circ \xi'$ and $*/u \in \alpha' \Rightarrow */u \in \alpha \Rightarrow */u \in \alpha \circ \xi$.

Now suppose $u \in \mathsf{dom}(\alpha \circ \xi)$ and $u \notin \mathsf{dom}(\alpha)$. $*/u$ cannot be in ξ' because $*$ annotations become 0. All $0/1$ annotations remain the same. This satisfies the inductive condition.

Finally, consider the third case (R3), which has a tautological resolvent. By induction hypothesis ϵ satisfies C_2, but ϵ cannot satisfy $\neg x$ since this would prevent any x literal from appearing anywhere in the reduced proof, such as the one we require for $f_{C_1}(x^{\tau \circ \sigma'}) = x^{\tau \circ \sigma}$. Hence ϵ satisfies K_2 and thus satisfies D.

We now proceed with item 2 of our general approach to strategy extraction, i.e., we show that the strategy for the universal player on U can be read off from π_ϵ. In fact, we show a slightly more general statement for arbitrary IR-calc proofs.

Lemma 3. *Let π be an IR-calc refutation of a QBF starting with a block of universally quantified variables U. Consider the set of annotations μ on variables U that appear anywhere in π. Then μ is non-contradictory.*

Proof. The proof proceeds by induction on the derivation depth. Let μ_C denote the set of annotations to variables in U appearing anywhere in the derivation of C (i.e., we only consider the connected component of the proof dag with sink C). The induction hypothesis states:

(i) The set μ_C is non-contradictory.
(ii) For every literal $l^\sigma \in C$, it holds that $\mu_C \subseteq \sigma$.

Base Case. Condition (i) is satisfied by the axioms, because we are assuming that there are no complementary literals in clauses in the matrix. Condition (ii) is satisfied because all existential literals are at a higher level than the variables of U.

Instantiation. Let $u \in U$ and $C = \mathsf{inst}(c/u, C')$ in the proof π. By induction hypothesis, u either appears in the annotations of all the literals l^ξ in C' or it does not appear in any of them. In the first case, the instantiation step is ineffective. In the second case, c/u is added to all literals in C.

By the induction hypothesis, u does not appear in any annotation of any clause in the sub-proof deriving C', and hence C is the first clause containing u.

Resolution. Let C be derived by resolving $x^\tau \vee C_1$ and $\neg x^\tau \vee C_2$. Let $u \in U$. Consider the following cases:

Case 1. $c/u \in \tau$. By induction hypothesis, c/u appears in all annotations of C_1, C_2 and hence in all annotations of the resolvent.

Case 2. $u \notin \mathsf{dom}(\tau)$. Then u does not appear as annotation anywhere in the derivation of either of the antecedents and neither it will appear in the resolvent.

Remark 1. We can show that Lemma 3 also holds for IRM-calc.

Proof. To do this we add a third condition to the inductive claim.

(iii) $*/u \notin \mu_C$, for any $u \in U$.

Since $*$ is only introduced by the merging rule, we argue that we do not obtain any annotations involving $*/u$ for $u \in U$. This holds because by condition (i) there are never contradictory annotations on u.

Therefore we obtain winning strategies. Given a QBF $\exists E \forall U. \Phi$ and an assignment ϵ to the leftmost existential block E, we construct an assignment to the universal variables U by collecting all the assignments μ to U appearing in annotations in π_ϵ (or instantiations when we have the empty clause instantiated immediately); any variable not appearing in π_ϵ is given an arbitrary value.

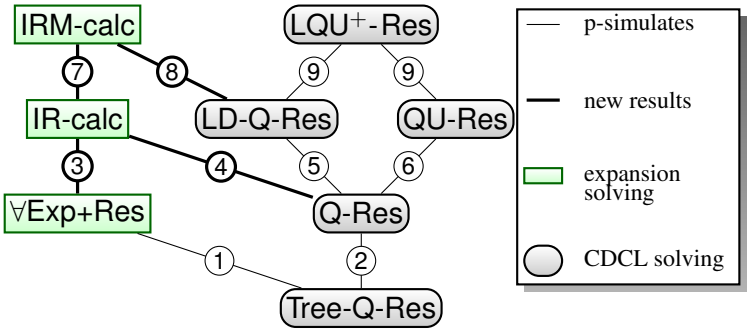
To obtain $\pi_{\epsilon,\mu}$, remove occurrences of U-variables from the annotations in the proof π_ϵ. This yields a valid refutation because by Lemma 3 for each variable in U only a single-value constant annotation can appear in the entire proof π_ϵ.

Theorem 2. *The construction above yields a winning strategy for the universal player.*

Proof. For any QBF $\Gamma = \exists E \forall U. \Phi$ and assignment ϵ to E, the above construction provides an IR-calc (IRM-calc) refutation $\pi_{\epsilon,\mu}$ of $\Phi|_{\epsilon \cup \mu}$. This process is iterated until no universal variables are left in the formula. Hence we get an IR-calc (IRM-calc) refutation of whatever was left from the matrix of Γ. Since an IR-calc (IRM-calc) refutation on a formula with no universal variables is in fact a classical propositional resolution refutation, we are left with an unsatisfiable formula, i.e. a formula with no winning move for the existential player. Hence, all the considered assignments correspond to a game won by the universal player. Since this process works for any assignment made by the existential player, it yields a winning strategy for the universal player.

The soundness of IR-calc (IRM-calc) follows directly from Theorem 2.

Corollary 1. *The calculi IR-calc and IRM-calc are sound.*



	Simulation
1	[74]
2	by def.
3	Thm. 4
4	Thm. 3
5	by def.
6	by def.
7	by def.
8	Thm. 5
9	by def.

Fig. 18. The simulation order of QBF resolution systems

4.4 Completeness and Simulations of Known QBF Systems

In this section we prove that our calculi simulate the main existing resolution-based QBF proof systems, doing so we add the new simulations shown in Figure 18, which in Chapters 5 and 6 we show are the only remaining simulations. As a by-product, this also shows completeness of our proof systems IR-calc and IRM-calc. We start by simulating Q-resolution, which is even possible with our simpler calculus IR-calc.

Theorem 3. *IR-calc p-simulates Q-Res.*

Proof. Let $C_1, \ldots, C_k$ be a Q-Res proof. We translate the clauses into $D_1, \ldots, D_k$, which will form the skeleton of a proof in IR-calc. Below we describe how this translation works and we repeat this in Figure 19.

- For an axiom C_i in Q-Res we introduce the same clause D_i by the axiom rule of IR-calc, i.e., we remove all universal variables and add annotations.
- If C_i is obtained via $\forall$-reduction from C_j, then $D_i = D_j$.
- Consider now the case that C_i is derived by resolving C_j and C_k with pivot variable x. Then $D_j = x^\tau \vee K_j$ and $D_k = \neg x^\sigma \vee K_k$. We instantiate to get $D'_j = \mathsf{inst}(\sigma, D_j)$ and $D'_k = \mathsf{inst}(\tau, D_k)$. Define D'_i as the resolvent of D'_j and D'_k. In order to obtain D_i we must ensure that there are no identical literals with different annotations. For this consider the set $\zeta = \{c/u \mid c/u \in t, l^t \in D'_i\}$ and define $D_i = \mathsf{inst}(\zeta, D'_i)$. This guarantees that we will always have fewer literals in D_i than in C_i, and we get a refutation.

Q-Res rule	IR-calc p-simulation
$\dfrac{}{C_i}$ (axiom)	$\dfrac{}{D_i = \{l^{\mathsf{restrict}_l(\tau)} \mid l \in C_i, l \in \exists\}}$ (axiom) $\tau = \{0/u \mid u \text{ is universal in } C\}$
$\dfrac{C_j}{C_i}$ (red)	$\dfrac{D_j}{D_i = D_j}$
$\dfrac{C_j = K_j \vee x \qquad C_k = K_k \vee \neg x}{C_i}$ (res)	$\dfrac{\dfrac{\dfrac{D_j = K'_j \vee x^\tau}{\mathsf{inst}(\sigma, D_j)}\text{ (inst)} \quad \dfrac{D_k = K'_k \vee \neg x^\sigma}{\mathsf{inst}(\tau, D_k)}\text{ (inst)}}{D'_i = \mathsf{inst}(\sigma, D_j) \cup \mathsf{inst}(\tau, D_k) \backslash \{x^{\tau \circ \sigma}, \neg x^{\tau \circ \sigma}\}}\text{ (res)}}{D_i = \mathsf{inst}(\zeta, D'_i)}\text{ (inst)}$ $\zeta = \{c/u \mid c/u \in t, l^t \in D'_i\}$

Fig. 19. A table outlining the IR-calc p-simulation of Q-Res

We will prove inductively on the lines of the proof starting from the axioms, that the resolution steps are valid, by showing that τ and σ are not contradictory and ζ does not contain contradictory annotations.

Induction Hypothesis. For all existential literals l we have $l \in C_i$ if and only if $l^t \in D_i$ for some annotation t. Additionally, if $0/u \in t$ for a literal u, then $u \in C_i$ (where for a variable x, we equivalently denote the annotation $1/x$ by $0/\neg x$).

Before proving the induction we argue that this yields the claim above. Assume for a contradiction that τ contradicts σ. This means that for some universal variable u, both u and $\neg u$ appear in C_i, which is not allowed; similarly if ζ contains contradictory annotations.

We now show the inductive claim by induction on the proof length.

Base case: Axiom. $l^t \in D_i$ if and only if $l \in C_i$ by definition. As annotations falsify all universal literals in the original clause, $0/u \in t$ for literal u implies $u \in C_i$.

Inductive step: $\forall$-Reduction. Suppose C_i is obtained via universal reduction from C_j. We have $l^t \in D_i$ iff $l^t \in D_j$, iff $l \in C_j$. Since l is existential it is not reduced and $l \in C_i$. If $0/u$ is in t, by inductive hypothesis we have $u \in C_j$. Further, u cannot be reduced in this step because it is blocked by l; hence $u \in C_i$. If $l \in C_i$ then $l \in C_j$, so by induction hypothesis there is some t such that $l^t \in D_j$, and hence $l^t \in D_i$. Assume now $l^t \in D_j$ and $0/u \in t$. By inductive hypothesis we have $u \in C_j$.

Resolution. Suppose that C_i is derived by resolving C_j and C_k over variable x, and $D_j = x^\tau \vee K'_j$ and $D_k = \neg x^\sigma \vee K'_k$. Then $l^t \in D_i$ iff $l^{t'} \in D'_i$ iff $l^{t'} \in \mathsf{inst}(\sigma, K'_j) \cup \mathsf{inst}(\tau, K'_k)$ iff $l^{t''} \in K'_j \cup K'_k$ iff $l \in C_j \cup C_k \backslash \{x, \neg x\}$ iff $l \in C_i$ (cf. Figure 19).

Without loss of generality, if $0/u \in t$ then there is some literal $p^{t' \circ \sigma} \in D'_i$ (with $p^{t'} \in D_j$) such that $0/u \in t' \circ \sigma$. If $0/u \in t'$ then $u \in C_j$ by inductive hypothesis, and if $0/u \in \sigma$ then $u \in C_k$, again by inductive hypothesis; hence $u \in C_i$. □

Despite its simplicity, IR-calc is powerful enough to also simulate the expansion proof system $\forall$Exp+Res from [74].

Theorem 4. *IR-calc p-simulates ∀Exp+Res.*

Proof. Let $C_1, \ldots, C_k$ be an ∀Exp+Res proof. We transform it into an IR-calc proof $D_1, \ldots, D_k$ as follows. If C_i is an axiom from clause C and assignment τ we construct D_i by taking the IR-calc axiom rule for C and then instantiating with $\mathsf{inst}(\tau, C)$. If C_i is derived by resolving C_j, C_k over variable x^τ, then D_i is derived by resolving D_j, D_k over variable x^τ. This yields a valid IR-calc proof because $l^t \in D_i$ iff $l^t \in C_i$, which is preserved under applications of both rules. □

∀Exp+Res rule	IR-calc p-simulation
$\dfrac{}{C_i = \{l^{\mathsf{restrict}_l(\sigma)} \mid l \in C, l \in \exists\}}$ (ax) σ is a complete assignment that does not satisfy C.	$\dfrac{\dfrac{}{D_i' = \{l^{\mathsf{restrict}_l(\tau)} \mid l \in C_i, l \in \exists\}}\text{(ax)}}{D_i = \mathsf{inst}(\sigma, D_i')}\text{(inst)}$ $\tau = \{0/u \mid u \text{ is universal in } C\}$
$\dfrac{C_j = K_j \vee x^\tau \qquad C_k = K_k \vee \neg x^\tau}{C_i}$ (res)	$\dfrac{D_j = K_j \vee x^\tau \qquad D_k = K_k \vee \neg x^\tau}{D_i}$ (res)

Fig. 20. A table outlining the IR-calc p-simulation of ∀Exp+Res

We now come to the simulation of a more powerful system than Q-resolution, namely LD-Q-Res from [10]. We show that this system is simulated by IRM-calc. Compared to Theorem 3, the proof uses a similar, but more involved technique.

Theorem 5. *IRM-calc p-simulates LD-Q-Res.*

Proof. Consider an LD-Q-Res refutation $C_1, \ldots, C_n$. We construct clauses $D_1, \ldots, D_n$, which will form the skeleton of the IRM-calc proof. The construction will preserve the following four invariants for $i = 1, \ldots, n$.

(1) For an existential literal l, it holds that $l \in C_i$ iff $l^t \in D_i$ for some t.
(2) The clause D_i has no literals l^{t_1} and l^{t_2} such that $t_1 \neq t_2$.
(3) If $l^t \in D_i$ with $0/u \in t$, then $u \in C_i$ or $u^* \in C_i$, likewise if $l^t \in D_i$ with $1/u \in t$, then $\neg u \in C_i$ or $u^* \in C_i$.
(4) If $l^t \in D_i$ with $*/u \in t$, then $u^* \in C_i$.

The actual construction proceeds as follows (also see Figure 21). If C_i is an axiom, D_i is constructed by the axiom rule from the same clause. If C_i is a ∀-reduction of C_j with $j < i$, then we set D_i equal to D_j. If C_i is obtained by a resolution step from C_j and C_k with $j < k < i$, the clause D_i is obtained by a resolution step from D_j and D_k, yielding clause K, and by performing some additional steps on K. Firstly, we let $\theta = \{c/u \mid c \in \{0,1\}, c/u \in t, l^t \in K\} \cup \{0/u \mid */u \in t, l^t \in K\}$. Here θ is chosen because we cannot instantiate by $*$. We perform instantiation on K by substitutions in θ, in any order, to derive K'. After this, all annotations in K' have the same domain. We merge all pairs of literals $l^\sigma, l^\tau \in K'$ with $\tau \neq \sigma$ (in any order) to derive D_i.

LD-Q-Res rule	IRM-calc p-simulation
$\frac{}{C_i}$ (axiom)	$\frac{}{D_i = \{l^{\mathsf{restrict}_l(\tau)} \mid l \in C_i, l \in \exists\}}$ (axiom) $\tau = \{0/u \mid u \text{ is universal in } C\}$
$\frac{C_j}{C_i}$ (red)	$\frac{D_j}{D_i = D_j}$
$\frac{C_j = K_j \vee U_j \vee x \qquad C_k = K_k \vee U_k \vee \neg x}{C_i = K_j \cup K_k \vee U}$ (res)	$\frac{\frac{\frac{D_j = K'_j \vee x^{\tau \cup \sigma} \qquad D_k = K'_k \vee \neg x^{\tau \cup \xi}}{K = \mathsf{inst}(\xi, K'_j) \cup \mathsf{inst}(\sigma, K'_k)}\text{(res)}}{K' = \mathsf{inst}(\theta, K)}\text{(inst)}}{D_i}\text{(merge)}$ $\theta = \{c/u \mid c \in \{0,1\}, c/u \in t, l^t \in K\} \cup \{0/u \mid */u \in t, l^t \in K\}$ Merge is on all pairs of literals $l^\sigma, l^\tau \in K'$ with $\tau \neq \sigma$.

Fig. 21. A table outlining the IRM-calc p-simulation of LD-Q-Res

To show that this construction yields a valid IRM-calc refutation, we first need to prove the invariants above. This proceeds by induction on i.

Base case (axiom). Because we do not remove or add any existential literals in the axiom case, condition (1) holds. Likewise we do not create duplicates, so (2) holds. Any 0/1 annotation corresponds exactly to the opposite literal appearing in the clause, by definition of the axiom rule, hence (3) holds. As we do not obtain any $*$ annotations from axioms, (4) holds.

Inductive step ($\forall$-reduction). Consider a $\forall$-reduction step from C_j to C_i on universal variable u. Because we do not alter the existential literals in a $\forall$-reduction and the corresponding clause D_i in the IRM-calc proof remains unchanged, conditions (1) and (2) are satisfied by induction hypothesis. For conditions (3) and (4) we note that D_j cannot contain any annotations involving u. This holds because u would only appear as annotation on existential literals with level higher than u. These cannot exist as they would be blocking the reduction by (1).

Resolution step. For this consider C_j, C_k being resolved in LD-Q-Res to obtain C_i as in Figure 21. As only the resolved variable is removed, which is removed completely due to condition (2), D_i fulfils (1). By induction hypothesis we know that there can be at most two copies of each variable when we derive K. Their annotations have the same domain in K', because instantiation by θ applies the entire domain of all annotations in the clause to all its literals. It then follows that all copies of identical literals are merged into one literal in D_i. Therefore (2) holds for D_i.

To prove (3) consider the case where $l^t \in D_i$ with $0/u \in t$. The case with $1/u \in t$ is analogous. We know that $0/u$ appearing in D_i means that $0/u$ must appear in K' as merging cannot produce a new annotation $0/u$. Existence of $0/u$ in K' means that either $*/u$ appears in K or $0/u$ appears in K. No new annotations are created in a resolution step, so either $*/u$ or $0/u$ must appear in one or more of D_j, D_k. By induction hypothesis this means that u or u^* appears in $C_j \cup C_k$, hence also in C_i.

To show condition (4), let $l^t \in D_i$ with $*/u \in t$. Then either $*/u$ is present in K', or $0/u$ and $1/u$ are present in K' and will be merged. In the first case it is clear that some $*/u$ annotation appears in K and thus in D_j or in D_k, in which case from (4) of the induction hypothesis u^* must appear in C_i. In the second case it is possible that $0/u$ in K' was obtained from $*/u$ in K. Thus as already argued, u^* must appear in C_i. If instead $1/u$, $0/u$ are both present in K then they must come from the original clauses D_j, D_k. If they both appear in the same clause D_j, then by condition (3) it must be the case that u^* appears in C_j and thus in C_i. If, however, they appear in different clauses, then by (3) either of the clauses C_j, C_k contains u^* or they contain literals over u of opposite polarity. Both situations merge the literals to $u^* \in C_i$.

We now show that these invariants imply that we indeed obtain a valid IRM-calc proof. We only need to consider the resolution steps. Suppose $x^{t_1} \in D_j$ and $\neg x^{t_2} \in D_k$ where C_j and C_k are resolved on x to get C_i in the LD-Q-Res proof. To perform the resolution step between D_j and D_k we need to ensure that we do not have $c/u \in t_1$, $d/u \in t_2$ where $c \neq d$ or $c = d = *$. Assume on the contrary that $*/u \in t_1$ and $c/u \in t_2$. By (4) we have $u^* \in C_j$, and by (3) some literal of u is in C_k. But as $\mathrm{lv}(u) < \mathrm{lv}(x)$ the LD-resolution of C_j and C_k on variable x is forbidden, giving a contradiction. Similarly, if there is $0/u \in t_1$ and $1/u \in t_2$, then either we get the same situation or we have two opposite literals of u in the different clauses C_j, C_k. In either case the resolution of C_j, C_k is forbidden. Hence the IRM-calc proof is correct.

It is not difficult to see that the IRM-calc proof is indeed a refutation and all steps of the construction can be performed in polynomial time, thus we obtain a p-simulation. □

We use IR-calc and IRM-calc for our proof complexity investigation in the remaining chapters. In particular, we show relevant lower bounds and separations in Chapter 6 and we show a connection to dependency QBF (DQBF) in Chapter 10. We will continue the comparison of QCDCL and expansion solving in Chapter 5.

Chapter 5

Strategy Extraction: Lower Bounds for CDCL Resolution Calculi

In Chapter 3 we introduced the CDCL-style QBF proof systems Q-resolution and its adaptations; LD-Q-Res, QU-Res and LQU$^+$-Res. Chapter 4 introduced ∀Exp+Res and the two new calculi IR-calc and IRM-calc, which are expansion-based QBF proof systems. These seven calculi as well as the tree-like version of Q-Res, give us all the QBF resolution calculi that are of part of our investigation into the simulation order of QBF resolution.

We already know many simulations for QBF resolution calculi; many of the systems naturally p-simulate each other by definition. There are other p-simulations such as the one in [74] and those in Theorems 3, 4 and 5. In fact, in this chapter and Chapter 6 we show these are the only simulations. The remaining relations are exponential separations where there are false formulas that have polynomial-size refutations in one system, but require exponential size in another system. We detail all simulations and separations in Figure 22.

This chapter further explores the dichotomy between CDCL-solving and expansion based solving. We primarily explore the relation between Q-Res and ∀Exp+Res.

We exhibit a new method to obtain lower bounds to the proof size in QBF proof systems, which directly allows to transfer circuit lower bounds to size of proof lower bounds. This method is based on the property of *strategy extraction*, which is known to hold for many resolution-based QBF proof systems.

The basic idea of our method is both conceptually simple and elegant: if we know that a family φ_n of false QBFs requires large winning strategies, then proofs of φ_n must be large in all proof systems with feasible strategy extraction. Now we need suitable formulas φ_n. Starting with a language L — for which we know (or conjecture) circuit lower bounds — we construct a family of false QBFs φ_n such that every winning strategy of the universal player for φ_n will have to compute L for inputs of length n. Consequently, a circuit lower bound for L directly translates into a lower bound for the winning strategy and therefore the proof size.

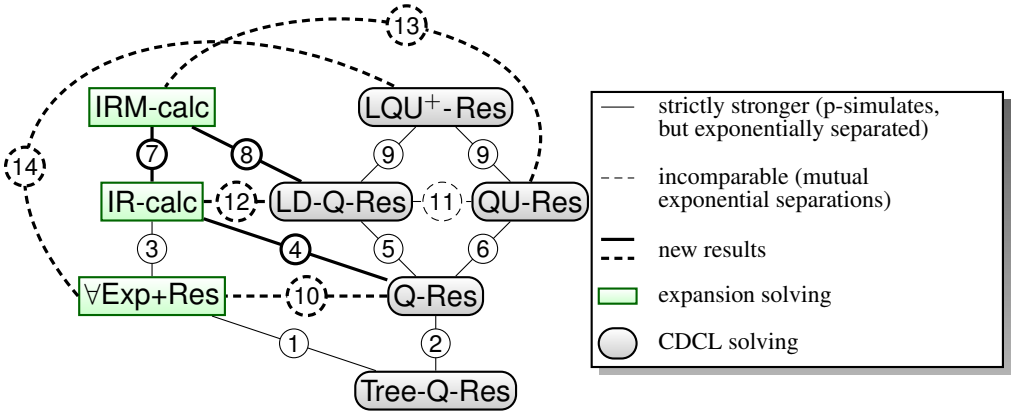
Carefully implemented, our method yields *unconditional lower bounds*. For Q-Res (and QU-Res) it is known that strategy extraction is computationally easy [10]; it is in fact possible in AC^0 as we verify here. Using the hardness of parity for AC^0 we can therefore construct formulas $\mathrm{QPARITY}_n$ that require exponential-size proofs in Q-Res (and QU-Res).

Conceptually, our lower bound method via strategy extraction is similar to the feasible interpolation technique [80], which is one of the most successful techniques in classical proof complexity. In feasible interpolation, circuit lower bounds are also translated into proof size lower bounds.

However, feasible interpolation only works for formulas of a special syntactic form, while our technique directly applies to arbitrary languages. It is a long-standing belief in the proof complexity community that there exists a direct connection between progress for showing lower bounds in circuit complexity and for proof systems (cf. [44]). For QBF proof systems our technique makes such a connection very explicit.

Section 5.1 shows the new lower bound for QU-Res (and thus for Q-Res). We use a new general technique that is widely applicable- *strategy extraction*. We illustrate this technique here with an exponential lower bound for parity formulas in QU-Res. This provides a separation between QU-Res and ∀Exp+Res. We then lift this lower bound to the long-distance systems LD-Q-Res and LQU$^+$-Res (Section 5.2).

The work in this chapter appeared at the 32nd Symposium on Theoretical Aspects of Computer Science [19].



	Simulation Separation			Incomparable
1	[74]	[74]	10	[74], Cor. 2
2	by def.	[36]	11	[11]
3	Thm. 4	[74], Thm. 3	12	Cor. 6, Cor. 4
4	Thm. 3	Cor. 2	13	Cor. 7, Thm. 5
5	by def.	[53]	14	Cor. 6, Cor. 4
6	by def.	[116]		
7	by def.	Thm. 5, Cor. 6		
8	Thm. 5	Cor. 4		
9	by def.	[11]		

Fig. 22. The simulation order of QBF resolution systems

5.1 Lower Bounds for Q-Res and QU-Res via Strategy Extraction

The lower bound argument hinges on strategy extraction, which is a widely used paradigm in QBF solving and proof systems. We recall that QU-Res admits strategy extraction via a computationally very restricted model, namely decision lists.

Definition 5 (decision list [105]). *A* decision list $D = (t_1, c_1), \ldots, (t_n, c_n)$ *is a finite sequence of pairs where* t_i *is a term and* $c_i \in \{0, 1\}$ *is a Boolean constant. Additionally, the last term is the empty term (equivalent to true). For an assignment* μ*, a decision list* D *evaluates to* c_i *if* i *is the least index such that* $\mu \models t_i$*, in such case we say that* (t_i, c_i) triggers *under* μ*.*

Intuitively decision lists code a succession of "if-then-else" queries in order that return true/false after a condition is satisfied. Winning strategies in form of decision lists can be efficiently extracted from QU-Res proofs:

Theorem 6 (Balabanov, Jiang, Widl [10,11]). *Given a Q-Res or QU-Res refutation* π *of QBF* ϕ*, there exists a winning strategy for the universal player for* ϕ*, such that each of its strategies for the universal variables are computable by a decision list of size polynomial in* $|\pi|$*.*

Balabanov et al. use a different form than decision lists, but it is semantically equivalent. We deem decision lists as more intuitive for our purposes. Note that that under our definition, a strategy for a universal variable may take as input the outputs of strategy functions with a smaller index (similarly as in the strategy construction by Goultiaeva et al. [65]).

The general idea behind our lower bound technique is as follows. First, we observe that we can define a family of QBFs ϕ_f such that every winning strategy of the universal player *must* compute a unique Boolean function f (Lemma 4). If we know that strategy extraction is possible by a weak computational model, say AC^0, we can carefully choose the Boolean formula ϕ_f such that the unique winning strategy f cannot be computed by AC^0 circuits. As the extracted strategy is polynomial in the proof, this implies a lower bound on the proof size. Thus we immediately turn circuit lower bounds to lower bounds for the proof size.

We will now implement this idea for the *parity function* $\textsc{Parity}(x_1, \ldots, x_n) = x_1 \oplus \cdots \oplus x_n$, which is the classical example of a function not computable in AC^0.[1]

Theorem 7 (Ajtai [1], Furst, Saxe, Sipser [59], Håstad [67]). $\textsc{Parity} \notin \mathsf{AC}^0$*. In fact, every non-uniform family of bounded-depth circuits computing* Parity *is of exponential size.*

We first observe how to construct a QBF that forces a unique winning strategy.

Lemma 4. *Consider the QBF* $\exists x_1, \ldots, x_n \forall z. (z \vee \phi_f) \wedge (\neg z \vee \neg \phi_f)$*, where* ϕ_f *is a propositional formula depending only on the variables* $x_1, \ldots, x_n$*. Let* $f : 2^n \to \{0, 1\}$ *be a Boolean function that returns 1 iff* ϕ_f *evaluates to true. Then there is a unique strategy for the universal player for* z*, which is* $z \leftarrow f$*.*

[1] The parity function determines if the number of true variables is odd ($\oplus$ denotes exclusive-or).

Proof. The strategy for z may only depend on the variables $x_1, \ldots, x_n$ and it must be so that the matrix evaluates to false under the given assignment μ to the x_i variables. By inspecting the matrix, z must be set to 0 whenever ϕ_f evaluates to 0 and the other way around. □

We will now use this idea specifically for the parity function. Consider the QBF

$$\Phi = \exists X \forall z \exists T.(F^+ \wedge F^-)$$

where F^+ is a CNF encoding of $z \vee \textsc{Parity}(X)$ and F^- encodes $\neg z \vee \neg\textsc{Parity}(X)$. Both F^+ and F^- use additional variables in T. More precisely, for $N > 1$ define QPARITY_N as follows. Let $\text{xor}(o_1, o_2, o)$ be the set of clauses $\{\neg o_1 \vee \neg o_2 \vee \neg o,\ o_1 \vee o_2 \vee \neg o,\ \neg o_1 \vee o_2 \vee o,\ o_1 \vee \neg o_2 \vee o\}$, which defines o to be $o_1 \oplus o_2$.

Definition 6. *Define* QPARITY_N *as*

$$\exists x_1, \ldots, x_N \forall z\, \exists t_2, \ldots, t_N.\ \text{xor}(x_1, x_2, t_2) \cup \bigcup_{i=3}^{N} \text{xor}(t_{i-1}, x_i, t_i) \cup \{z \vee t_N, \neg z \vee \neg t_N\}.$$

Note that since we want to encode parity in CNF, i.e. a bounded-depth formula, and $\textsc{Parity} \notin \mathsf{AC}^0$*, we need to use further existential variables (recall that existential* AC^0 *characterises all of* NP*). Choosing existential variables* t_i *to encode the prefix sums* $x_1 \oplus \cdots \oplus x_i$ *of the parity* $x_1 \oplus \cdots \oplus x_N$ *provides the canonical CNF formulation of parity.*

To use the lower bound of Theorem 7 we need to verify that QU-Res enables strategy extraction in AC^0. This holds as decision lists can be turned into bounded-depth circuits.

Lemma 5. *If* f_D *can be represented as a polynomial-size decision list* D*, then* $f_D \in \mathsf{AC}^0$.

Proof. Let $S = \{i \mid (t_i, 1) \in D\}$ be the indices of all pairs in D with 1 as the second component. Observe that f_D evaluates to 1 under μ iff one of the t_i with $i \in S$ triggers under μ. For each t_i with $i \in S$ construct a function $f_i = t_i \wedge \bigwedge_{l=1}^{i-1} \neg t_l$. Construct a circuit for the function $\bigvee_{i \in S} f_i$, which is equal to f_D and is computable in AC^0 as all t_i are just terms. □

We can now put everything together and turn the circuit lower bound of Theorem 7 into a lower bound for proof size in QU-Res.

Theorem 8. *Any QU-Res refutation of* QPARITY_N *is of exponential size in* N.

Proof. By Lemma 4 there is a unique strategy for the variable z in QPARITY_N, which is the PARITY function on N variables. From Theorem 6, there is a polynomial-time algorithm for constructing a decision list D_N from any QU-Res refutation of QPARITY_N. D_N must be a decision list computing the parity function. Such decision list can be converted in polynomial time into a circuit with bounded depth by Lemma 5. Due to Theorem 7 the bounded depth circuit and the decision list computing parity must be of exponential size in N. □

In contrast to this lower bound, the parity formulas are easy in ∀Exp+Res.

Lemma 6. *The formulas* QPARITY$_N$ *have polynomial-size* ∀*Exp+Res refutations.*

Proof. Instantiate all clauses in both polarities of z, which generates the clauses $\mathrm{xor}(x_1, x_2, t_2^{0/z}) \cup \bigcup_{i=3}^{N} \mathrm{xor}(t_{i-1}^{0/z}, x_i, t_i^{0/z}) \cup \{t_N^{0/z}\}$ and $\mathrm{xor}(x_1, x_2, t_2^{1/z}) \cup \bigcup_{i=3}^{N} \mathrm{xor}(t_{i-1}^{1/z}, x_i, t_i^{1/z}) \cup \{\neg t_N^{1/z}\}$.

Inductively, for $i = 2, \ldots, N$ derive clauses representing $t_i^{0/z} \Leftrightarrow t_i^{1/z}$. This lets us derive a contradiction using the clauses $t_N^{0/z}$ and $\neg t_N^{1/z}$. □

Theorem 8 together with Lemma 6 immediately give the following separations.

Corollary 2. *Q-Res and QU-Res do not simulate* ∀*Exp+Res, IR-calc, IRM-calc.*

This also has consequences for the complexity of strategy extraction in ∀Exp+Res.

Corollary 3. *Winning strategies for* ∀*Exp+Res cannot be computed in* AC^0*. This even holds when the system* ∀*Exp+Res is restricted to formulas with constant quantifier complexity.*

Proof. The formulas QPARITY$_N$ have polynomial-size ∀Exp+Res refutations by Lemma 6. Hence we cannot extract strategies in AC^0 as these would compute parity. □

Note, however, that strategy extraction for ∀Exp+Res and in fact even IRM-calc is in P due to Theorem 2.

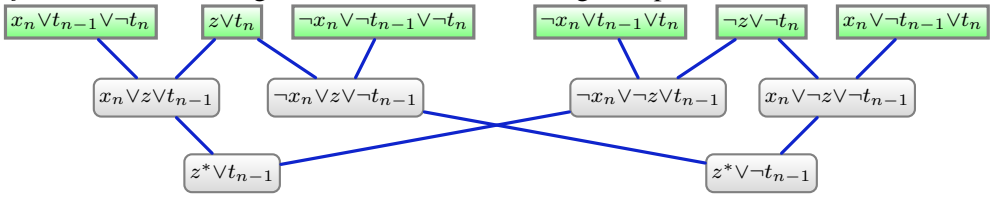
5.2 Extending the Lower Bound to LD-Q-Res and LQU$^+$-Res

We now aim to extend the lower bound from the previous section to stronger QBF proof systems using long-distance resolution. For this we cannot directly use the strategy extraction method from the last section. In fact the formulas do not work as a lower bound as we show in Theorem 9. However, we will slightly modify the parity formulas and then reduce the hardness of those in the stronger systems to the hardness of QPARITY in Q-Res. As the modified formulas remain easy for ∀Exp+Res, these lower bounds imply many new separations between the proof systems involved.

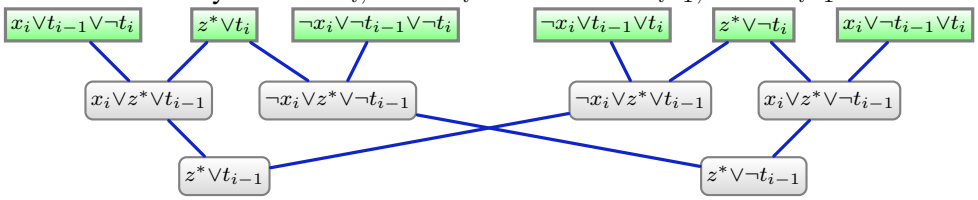
We first show that short proofs exists in LD-Q-Res for QPARITY.

Theorem 9. *There are* $O(N)$ *length LD-Q-Res refutations of* QPARITY$_N$.

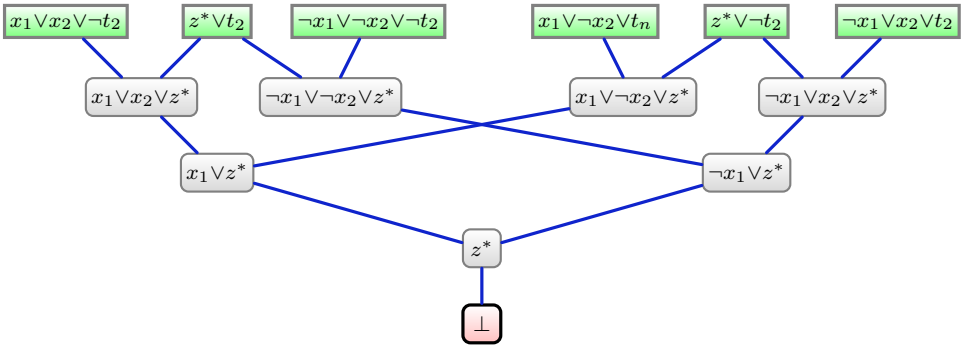
Proof. We can utilise long distance resolution to merge the polarities of the z literals.



We can inductively use $z^* \vee t_i, z^* \vee \neg t_i$ to derive $z^* \vee t_{i-1}, z^* \vee \neg t_{i-1}$ for $1 < i < n$.



And finally we can derive the empty clause.



The DAG-like nature of these proofs gives a linear size. □

In order to separate LD-Q-Res and ∀Exp+Res we need a new set of formulas. For this we consider a variant of the parity formulas. Let $\mathrm{xor}_l(o_1, o_2, o, z)$ be the set of clauses $\{z \vee \neg o_1 \vee \neg o_2 \vee \neg o,\ z \vee o_1 \vee o_2 \vee \neg o,\ z \vee \neg o_1 \vee o_2 \vee o,\ z \vee o_1 \vee \neg o_2 \vee o\}$ (xor_l defines o to be equal to $o_1 \oplus o_2$ if $z = 0$). The formulas LQPARITY$_N$ are constructed from QPARITY$_N$ by replacing each occurrence of $\mathrm{xor}(\dots)$ by two copies $\mathrm{xor}_l(\dots, z)$ and $\mathrm{xor}_l(\dots, \neg z)$, yielding

$$\exists x_1, \dots, x_N \forall z \exists t_2, \dots, t_N.\ \mathrm{xor}_l(x_1, x_2, t_2, z) \cup \bigcup_{i=3}^{N} \mathrm{xor}_l(t_{i-1}, x_i, t_i, z)$$
$$\cup\, \mathrm{xor}_l(x_1, x_2, t_2, \neg z) \cup \bigcup_{i=3}^{N} \mathrm{xor}_l(t_{i-1}, x_i, t_i, \neg z) \cup \{z \vee t_N, \neg z \vee \neg t_N\}.$$

It is easy to verify that the same arguments as for QPARITY in Section 5.1 also apply to LQPARITY, yielding:

Proposition 1. *The formulas* LQPARITY$_N$ *have polynomial-size* ∀*Exp+Res refutations, but require exponential-size Q-Res refutations.*

We now want to show that LQPARITY is hard for LD-Q-Res by arguing that long-distance steps do not help to refute these formulas. In the next two lemmas we will show that this actually applies to all QBFs Φ meeting the following condition.

Definition 7. *We say that z is* completely blocked *in a QBF Φ, if all clauses of Φ contain the universal variable z and some existential literal l such that* $\mathrm{lv}(z) < \mathrm{lv}(l)$.

Lemma 7. *Let Φ be a QBF and z be completely blocked in Φ. Let further C be a clause derived from Φ by LD-Q-Res. If C contains some existential literal l such that* $\mathrm{lv}(z) < \mathrm{lv}(l)$*, then $z \in C$, $\neg z \in C$, or $z^* \in C$.*

Proof. We prove the lemma by induction on the derivation depth. The base case is established by the clauses in the matrix of Φ due to the condition that z must be in all matrix clauses and also that all these clauses contain some existential literal that blocks z.

The hypothesis is preserved by $\forall$-reduction because a literal over z cannot be $\forall$-reduced if the clause contains an existential literal l with $\mathrm{lv}(z) < \mathrm{lv}(l)$.

Consider now two clauses $C_1 = D_1 \vee x$ and $C_2 = D_2 \vee \neg x$ resolved into the clause C_3. If C_3 contains some literal l such that $\mathrm{lv}(z) < \mathrm{lv}(l)$, then one of C_1, C_2 must contain l and from induction hypothesis it must also contain the variable z, which then appears in C_3. □

Lemma 8. *Let Φ be a QBF such that z is completely blocked in Φ and let π be a refutation of Φ such that the variable z is $\forall$-reduced as early as possible. Then the derivation of the empty clause in π does not contain z^* in any of its clauses.*

Proof. Assume that we have a clause C in π that contains z^*. We will argue that C is not necessary to derive the empty clause $\bot$, i.e., there is no path in π from C to $\bot$. Since z^* does not appear in any of the matrix clauses, there must be a resolution step where it is introduced. Consider any such two clauses $C_1 = D_1 \vee x \vee z$ and $C_2 = D_2 \vee \neg x \vee \neg z$ resolved into $C = D_3 \vee z^*$. From the assumption that $\forall$-reductions are carried out as soon as possible, in both clauses C_1 and C_2 there must be some literals that block z and $\neg z$, respectively. From the conditions on LD-Q-Res, x or $\neg x$ cannot be the blocking literal (it must be that $\mathrm{lv}(x) < \mathrm{lv}(z)$ upon merging). This means that C contains at least one literal b that blocks z^*.

Now we argue that b cannot be resolved away. For contradiction assume that there is a resolution step of some C' and D on b where there is a path from C to C'. Moreover, let that be the first resolution step on b, i.e., b appears in all clauses on the path between C and C'. From Lemma 7, the clause D must contain a literal on the variable z. But this contradicts the conditions of LD-Q-Res because resolution steps are not permitted on literals b with $\mathrm{lv}(z) < \mathrm{lv}(b)$ if one of the antecedents contains a merged literal z^* and the other contains some literal on z. This means that C does not participate in the derivation of the empty clause $\bot$. □

This enables us to prove the hardness of LQPARITY in LD-Q-Res.

Theorem 10. *Any refutation of* $\mathrm{LQPARITY}_N$ *in LD-Q-Res is exponential in N.*

Proof. Any LD-Q-Res refutation π can be translated in polynomial time into a refutation π' such that $\forall$-reductions are carried out as soon as possible (such a refutation has clauses that are equal to the clauses of π or some universal literals are missing). From Lemma 8, the derivation of $\bot$ in π' contains no occurrences of the merged literal z^*, hence any such clauses can be removed from the refutation. Therefore π' is in fact also a Q-Res refutation. Hence, π must be exponential in N due to Proposition 1. □

Theorem 11. *There are $O(N)$ length LQU$^+$-Res refutations of* $\mathrm{LQPARITY}_N$.

Proof. We simply derive the clauses in $\mathrm{xor}(o_1, o_2, o)$ from $\mathrm{xor}_l(o_1, o_2, o, z)$ and $\mathrm{xor}_l(o_1, o_2, o, \neg z)$ by resolving the clauses on z. In other words we can derive $\mathrm{QPARITY}_N$ from $\mathrm{LQPARITY}_N$ in an $O(N)$ length proof, then we use Theorem 9 and the fact that LQU$^+$-Res simulates LD-Q-Res to complete our short proof. □

Our next goal is to extend the lower bound for the parity formulas to the system LQU$^+$-Res, which enables both long-distance and universal resolution. For this we must again modify the formula QPARITY, using a similar technique as in [11]. The trick is essentially to double the universal literals so they form tautological clauses when resolved. This way resolution on universal variables does not give any advantage.

We define formulas QUPARITY_N from LQPARITY_N as follows: replace the universal quantifier $\forall z$ by two new quantifiers $\forall z_1 \forall z_2$ and replace all occurrences of the literal z by $z_1 \vee z_2$ and likewise of $\neg z$ by $\neg z_1 \vee \neg z_2$. This gives the formulas QUPARITY_N

$$\exists x_1, \ldots, x_N \forall z_1, z_2 \exists t_2, \ldots, t_N . \text{xor}_u(x_1, x_2, t_2, z_1, z_2) \cup \bigcup_{i=3}^{N} \text{xor}_u(t_{i-1}, x_i, t_i, z_1, z_2) \cup \text{xor}_u(x_1, x_2, t_2, \neg z_1, \neg z_2) \cup \bigcup_{i=3}^{N} \text{xor}_u(t_{i-1}, x_i, t_i, \neg z_1, \neg z_2) \cup \{z_1 \vee z_2 \vee t_N, \neg z_1 \vee \neg z_2 \vee \neg t_N\},$$

where $\text{xor}_u(o_1, o_2, o, l_1, l_2)$ is the set of clauses

$$\{l_1 \vee l_2 \vee \neg o_1 \vee \neg o_2 \vee \neg o,\ l_1 \vee l_2 \vee o_1 \vee o_2 \vee \neg o,\ l_1 \vee l_2 \vee \neg o_1 \vee o_2 \vee o,\ l_1 \vee l_2 \vee o_1 \vee \neg o_2 \vee o\}.$$

It is clear that these formulas are false as the universal player should play both z_1 and z_2 as they would z in QPARITY.

We will now assume that in an LQU$^+$-Res refutation we $\forall$-reduce immediately. It is easy to verify that this does not increase proof size (cf. also Proposition 1 in [11]). For QUPARITY we now show an analogous result to Lemma 7.

Lemma 9. *Let C be a clause in an LQU$^+$-Res refutation of QUPARITY_N where $\forall$-reduction steps are performed as soon as possible. If C contains some existential literal l such that $\text{lv}(z_2) < \text{lv}(l)$, then either $z_1, z_2 \in C$, or $\neg z_1, \neg z_2 \in C$, or $z_2^* \in C$.*

Proof. The proof is the same as for Lemma 7, except for the possibility of universal resolution steps. As $\forall$-reductions are required to happen immediately, in our induction hypothesis we know that a z_1 literal can only occur together with the corresponding z_2 literal. Therefore resolving on z_1 removes this variable and merges the complementary z_2 literals; hence we get the z_2^* literal.

The merged literals cannot be pivots, neither can z_2. This holds because we know by induction hypothesis that when z_2 appears unmerged, then also z_1 appears unmerged with the same polarities. Hence resolving with z_2 as the pivot would merge z_1, which is illegal due to the index restriction. □

We now argue that neither long-distance nor universal resolution steps help to refute QUPARITY.

Lemma 10. *Any LQU$^+$-Res refutation of QUPARITY_N does not contain any clauses with z_1^* or z_2^* or any application of resolution on universal pivots.*

Proof. We first argue for z_2^*. Let $z_2^* \in C$. As we assume that $\forall$-reductions are performed immediately, the literal z_2^* is blocked by an existential literal l when z_2^* is created in C by a long-distance resolution step. Then l cannot be removed from C by resolution as any clause with $\neg l$ in it contains a literal over z_2 by Lemma 9. Also z_2^* cannot be removed via universal resolution. So the empty clause can never be derived from any clause containing z_2^*.

Let us now argue for z_1^* and assume $z_1^* \in C$. If z_1^* is introduced into C by resolving clauses D_1 and D_2, the literals z_1 and $\neg z_1$ in D_1 and D_2, respectively, must be blocked by existential literals. Therefore by Lemma 9, the clauses D_1 and D_2 also contain z_2 and $\neg z_2$, respectively. Hence also z_2^* is introduced into C and we get back to the previous case.

Finally, universal resolution steps cannot be performed when deriving the empty clause. For universal resolution on z_1, using again Lemma 9 together with the assumption of performing $\forall$-reductions as early as possible leads to the introduction of z_2^*, and we again get back to the case above.

No resolution on z_2 is possible as from Lemma 9 it would cause both literals of z_1 to be merged, which is illegal due to the index restriction in long-distance resolution over universal variables. □

This immediately implies the hardness of QUPARITY for LQU$^+$-Res because by the previous lemma any LQU$^+$-Res refutation of QUPARITY$_N$ is a Q-Res refutation, which by Theorem 8 is exponential in size.

Theorem 12. QUPARITY$_N$ *require exponential-size refutations in* LQU$^+$-Res.

As QUPARITY$_N$ still remains easy for $\forall$Exp+Res in a proof similar to Lemma 6 we get the following separations.

Corollary 4. *LQU$^+$-Res does not simulate $\forall$Exp+Res, IR-calc, and IRM-calc.*

In Chapter 6 we complete the remaining separations using a counting argument on the formulas from [77], rather than the strategy extraction technique used here. On the other hand, we find that strategy extraction can be used in a much wider context, this is explored in Part III.

Chapter 6

Lower Bounds for Q-Res and Expansion Calculi

While the strategy extraction technique from the last section is very effective for CDCL systems, it does not yield lower bounds in expansion systems. For the strong system IR-calc we know that the strategy extraction method on AC^0 circuits is not directly applicable.

Therefore, to obtain such lower bounds, and separations in turn, we need to use different techniques — and in fact different types of formulas. We obtain an exponential lower bound for the well-known formulas KBKF of Kleine Büning, Karpinski and Flögel [77] in IR-calc. In the same work [77], where Q-Res was introduced, these formulas were suggested as hard formulas for Q-Res. In fact, a number of further separations of QBF proof systems builds on this [11, 53]. Here we show in a technically involved counting argument that the formulas are even hard for IR-calc. As IR-calc simulates Q-Res we obtain as a by-product a formal proof of the hardness of KBKF in Q-Res.

KBKF was shown to be easy for long-distance Q-resolution [53] and thus via simulation does not provide a lower bound for the stronger IRM-calc calculus. Instead we use the modified $\mathrm{KBKF}_{\mathrm{lq}}$ formulas from [11], to show a lower bound for IRM-calc that build upon our result for IR-calc.

This chapter is organised in the following way. We first introduce the KBKF formulas and their previous applications (Section 6.1). We then show a lower bound for IR-calc (Section 6.2) and then extend this bound to even IRM-calc (Section 6.3).

The work in this chapter appeared at the 32nd Symposium on Theoretical Aspects of Computer Science [19].

6.1 The QBFs of Kleine Büning, Karpinski and Flögel

We present a proof complexity analysis of a well-known family of formulas $\mathrm{KBKF}(t)$ first defined by Kleine Büning et al. [77]. In this chapter we prove that the $\mathrm{KBKF}(t)$ formulas are hard for IR-calc, which is stronger than Q-Res (Corollary 2). This provides the first non-trivial lower bound for IR-calc, and further even separates the system from LD-Q-Res.

$\mathrm{KBKF}(t)$ has prefix $\exists y_0, y_{1,0}, y_{1,1} \forall x_1 \exists y_{2,0}, y_{2,1} \forall x_2 \dots \forall x_{t-1} \exists y_{t,0}, y_{t,1} \forall x_t \exists f_1 \dots f_t$ and matrix clauses

$$
\begin{array}{llll}
C_- = \{\neg y_0\} & & C_0 = \{y_0, \neg y_{1,0}, \neg y_{1,1}\} & \\
C_i^0 = \{y_{i,0}, x_i, \neg y_{i+1,0}, \neg y_{i+1,1}\} & & C_i^1 = \{y_{i,1}, \neg x_i, \neg y_{i+1,0}, \neg y_{i+1,1}\} & \text{for } i \in [t-1] \\
C_t^0 = \{y_{t,0}, x_t, \neg f_1, \dots, \neg f_t\} & & C_t^1 = \{y_{t,1}, \neg x_t, \neg f_1, \dots, \neg f_t\} & \\
F_i^0 = \{x_i, f_i\} & & F_i^1 = \{\neg x_i, f_i\} & \text{for } i \in [t].
\end{array}
$$

Let us verify that the $\mathrm{KBKF}(t)$ formulas are indeed false QBFs and — at the same time — provide some intuition about them. The existential player starts by playing $y_0 = 0$ because of clause C_-. Clause C_0 forces the existential player to set one of $y_{1,0}, y_{1,1}$ to 0. Assume the existential chooses $y_{1,0} = 0$ and $y_{1,1} = 1$. If the universal player tries to win, he will counter with $x_1 = 0$, thus forcing the existential player again to set one of $y_{2,0}, y_{2,1}$ to 0. This continues for t rounds, leaving in each round a choice of $y_{i,0} = 0$ or $y_{i,1} = 0$ to the existential player, to which the universal counters by setting x_i accordingly. Finally, the existential player is forced to set one of $f_1, \ldots, f_t$ to 0. This will contradict one of the clauses $F_1^0, F_1^1, \ldots, F_t^0, F_t^1$, and the universal player wins.

It is clear from this explanation, that the existential player has exponentially many choices and the universal player likewise needs to uniquely counter all these choices to win.

Theorem 13 (Kleine Büning, Karpinski, Flögel [77]). *A Q-Res refutation proof of* $\mathrm{KBKF}(t)$ *that requires each resolution step to resolve with a positive existential unit clause (a clause that contains exactly one existential literal which is positive) is at least of size* 2^t.

This kind of refutation is called a Q-unit-resolution refutation. The argument is as follows:

Proof. We take the induction hypothesis on $0 \leq i \leq t$ that any derivable existential unit clause containing $y_{t-i,c}$ for $c \in \{0, 1\}$ must contain a universal literal for every variable x_j when $0 < j < t - i$ and also requires a derivation with 2^i existential unit clauses each containing x_{t-i} if $c = 0$ or $\neg x_{t-i}$ if $c = 1$.

Base case: Positive literal $y_{t,c}$ only appears in clause C_t^c. In order to remove $\neg f_k$ literals they must be resolved with one of the F_k^0 or F_k^1 clauses, either one will introduce a x_i or $\neg x_i$ respectively, which if $i < t$ cannot be removed until $y_{t,c}$ is also gone. The corresponding x_{t-i} is present in the resolvent once all negative literals are removed (it must agree with C_t^c), it can be reduced immediately afterwards.

Inductive step: Any derivation of a positive existential unit clause containing $y_{t-i,c}$ must use the clause $C_{t-i}^c = \{y_{t-i,c}, (\neg)x_{t-i}, \neg y_{t-i+1,0}, \neg y_{t-i+1,1}\}$ as this is the only possible source of the positive $y_{t-i,c}$. Clause C_{t-i}^c must be resolved with an existential unit clause which is required to contain $y_{t-i-1,0}$ or $y_{t-i-1,1}$. Each of these contains every variable x_j for $0 < j < t - i$ and requires derivations the size of 2^{i-1}. Each of these derivations will have a different x_{t-i+1} literal. So when resolution is performed, to remove both negative literals, the size must be at least 2^i. Note that in each of these 2^i unit clauses there must be a x_{t-i} literal and it must always be the one that agrees with the induction hypothesis as that is necessary to resolve with C_{t-i}^c. Afterwards only x_{t-i} can be reduced so the clause must contain a universal literal for every variable x_j for $0 < j < t - i$. □

The essence of this lower bound generalises to an exponential lower bound for Q-Res and eventually stronger calculi (Theorem 16). In contrast the $\mathrm{KBKF}(t)$ formulas have short proofs in QU-Res and LD-Q-Res.

Theorem 14 (Van Gelder [116]). $\mathrm{KBKF}(t)$ *has polynomial-size proofs in QU-Res.*

Proof. We can derive the unit clauses $\{f_i\}$ using universal resolution.

$$\frac{\{x_i, f_i\} \qquad \{\neg x_i, f_i\}}{\{f_i\}}$$

With these we can resolve with C_t^0 or C_t^1 to derive $\{y_{t,0}, x_t\}$ and $\{y_{t,1}, \neg x_t\}$ and by universal reduction we derive $\{y_{t,0}\}$ and $\{y_{t,1}\}$. Inductively we use $\{y_{i+1,0}\}$ and $\{y_{i+1,1}\}$ to derive $\{y_{i,0}\}$ and $\{y_{i,1}\}$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\{y_{i,0}, x_i, \neg y_{i+1,0}, \neg y_{i+1,1}\} \qquad \{y_{i+1,0}\}}{\{y_{i,0}, x_i, \neg y_{i+1,1}\}} \qquad \{y_{i+1,1}\}}{\{y_{i,0}, x_i\}}}{\{y_{i,0}\}} \qquad \dfrac{\dfrac{\dfrac{\{y_{i,1}, \neg x_i, \neg y_{i+1,0}, \neg y_{i+1,1}\} \qquad \{y_{i+1,0}\}}{\{y_{i,1}, \neg x_i, \neg y_{i+1,1}\}} \qquad \{y_{i+1,1}\}}{\{y_{i,1}, \neg x_i\}}}{\{y_{i,1}\}}$$

Finally with $y_{1,0}$ and $y_{1,1}$ we can derive the empty clause.

$$\dfrac{\dfrac{\dfrac{\{y_0, \neg y_{1,0}, \neg y_{1,1}\} \qquad \{y_{1,0}\}}{\{y_0, \neg y_{1,1}\}} \qquad \{y_{1,1}\}}{\{y_0\}} \qquad \{\neg y_0\}}{\bot}$$

□

Theorem 15 (Egly, Widl [53]). $\mathrm{KBKF}(t)$ *has polynomial-size proofs in* LD-Q-Res.

Proof. First we can make the following derivations:

$$\frac{\{y_{t,0}, x_t, \bar{f}_1, \ldots, \bar{f}_t\} \qquad \{x_t, f_t\}}{\{y_{t,0}, x_t, \bar{f}_1, \ldots, \bar{f}_{t-1}\}}$$

$$\frac{\{y_{t,1}, \bar{x}_t, \bar{f}_1, \ldots, \bar{f}_t\} \qquad \{\bar{x}_t, f_t\}}{\{y_{t,1}, \bar{x}_t, \bar{f}_1, \ldots, \bar{f}_{t-1}\}}$$

Next, we inductively derive the clauses

$$\{y_{i,0}, x_i, x_{i+1}^* \ldots x_t^*, \bar{f}_1 \ldots \bar{f}_{i-1}\}, \{y_{i,1}, \bar{x}_i, x_{i+1}^* \ldots x_t^*, \bar{f}_1 \ldots \bar{f}_{i-1}\}$$

for decreasing i.

$$\dfrac{\dfrac{\{y_{i,0}, x_i, \bar{y}_{i+1,0}, \bar{y}_{i+1,1}\} \qquad \{y_{i+1,0}, x_{i+1}, x_{i+2}^* \ldots x_t^*, \bar{f}_1 \ldots \bar{f}_i\}}{\{y_{i,0}, x_i, \neg y_{i+1,1}, x_{i+1}, x_{i+2}^* \ldots x_t^*, \bar{f}_1 \ldots \bar{f}_i\}} \qquad \{y_{i+1,1}, \bar{x}_{i+1}, x_{i+2}^* \ldots x_t^*, \bar{f}_1 \ldots \bar{f}_i\}}{\{y_{i,0}, x_i, x_{i+1}^* \ldots x_t^*, \bar{f}_1 \ldots \bar{f}_i\}}$$

$$\frac{\{y_{i,0}, x_i, x_{i+1}^* \ldots x_t^*, \bar{f}_1 \ldots \bar{f}_i\} \qquad \{x_i, f_i\}}{\{y_{i,0}, x_i, x_{i+1}^* \ldots x_t^*, \bar{f}_1 \ldots \bar{f}_{i-1}\}}$$

$$\dfrac{\dfrac{\{y_{i,1}, \bar{x}_i, \bar{y}_{i+1,0}, \bar{y}_{i+1,1}\} \qquad \{y_{i+1,0}, x_{i+1}, x_{i+2}^* \ldots x_t^*, \bar{f}_1 \ldots \bar{f}_i\}}{\{y_{i,1}, \bar{x}_i, \neg y_{i+1,1}, x_{i+1}, x_{i+2}^* \ldots x_t^*, \bar{f}_1 \ldots \bar{f}_i\}} \qquad \{y_{i+1,1}, \bar{x}_{i+1}, x_{i+2}^* \ldots x_t^*, \bar{f}_1 \ldots \bar{f}_i\}}{\{y_{i,1}, \bar{x}_i, x_{i+1}^* \ldots x_t^*, \bar{f}_1 \ldots \bar{f}_i\}}$$

$$\frac{\{y_{i,1}, \bar{x}_i, x^*_{i+1} \dots x^*_t, \bar{f}_1 \dots \bar{f}_i\} \qquad \{\bar{x}_i, f_i\}}{\{y_{i,1}, \bar{x}_i, x^*_{i+1} \dots x^*_t, \bar{f}_1 \dots \bar{f}_{i-1}\}}$$

And finally derive a contradiction.

$$\dfrac{\dfrac{\dfrac{\{y_0, \neg y_{1,0}, \neg y_{1,1}\} \qquad \dfrac{\{y_{1,0}, x_1, x^*_2 \dots x^*_t\}}{\{y_{1,0}\}}}{\{y_0, \neg y_{1,1}\}} \qquad \dfrac{\{y_{1,1}, \bar{x}_1, x^*_2 \dots x^*_t\}}{\{y_{1,1}\}}}{\{y_0\}} \qquad \{\bar{y}_0\}}{\bot}$$

□

6.2 A Lower Bound in IR-calc for the Formulas of Kleine Büning et al.

We move on to the IR-calc lower bound argument, firstly we use a key property of these formulas. Syntactically, $\mathrm{KBKF}(t)$ are *existential Horn formulas*, i.e., they contain at most one positive existential literal per clause. In fact, they even have a stronger property: C_- is the only clause without a head (a positive existential literal).

We will strengthen this in the next lemma by a simple modification such that now all clauses have a head.

Lemma 11. *We can transform every IR-calc refutation π of $\mathrm{KBKF}(t)$ into an IR-calc proof π' of y_0 from $G(t) = \mathrm{KBKF}(t) \setminus \{\neg y_0\}$. We perform this by:*

1. *deleting every instance of the axiom $\{\neg y_0\}$, and removing the steps where y_0 is a pivot;*
2. *replicating all other steps, using the same rules, same pivots and same annotations when instantiating.*

Proof. The proof is immediate, observing that y_0 cannot gain annotations and the only rules applicable on y_0 are resolution rules between the axiom $\{\neg y_0\}$ and clauses containing y_0. Thus instead of the empty clause we derive $\{y_0\}$. □

After this transformation, which preserves proof length, we can focus on proofs of y_0 from $G(t) = \mathrm{KBKF}(t) \setminus \{\neg y_0\}$. Exploiting that all axioms now contain exactly one positive literal we show a number of invariants, which hold for all clauses in all IR-calc proofs of the formulas.

Let C be an annotated clause in an IR-calc proof of y_0 from $G(t)$. Then the following invariants hold for C.

Invariant 1 *C has exactly one positive literal $y^A_{h,a}$ or f^A_h for $h \leq t$ (or y_0 with no annotation). We call this unique literal the* head *of C and use the indices h and a also in the following invariants to denote its position as well as A for its annotation.*

Proof. Invariant 1 as well as all further invariants is shown by induction on the number of steps to derive C.

Invariant 1 holds in the base case as we no longer have $\{\neg y_0\}$ present in the axioms.

For the inductive step we only need to consider the resolution case as annotations do not affect the polarities of the literals.

Suppose C is derived from resolving D_1 and D_2 with pivot z. Without loss of generality let z be positive in D_2. Then Invariant 1 holds for C as there are two positive literals between D_1 and D_2, but the head z of D_2 gets removed by resolution. □

The next invariant states that when we order the literals in the clause by the prefix, the head, that is the positive literal, appears leftmost. Note that item 3 is for the benefit of Section 6.3.

Invariant 2 *Let $j \in [t]$, $b \in \{0,1\}$, and B be some annotation.*

1. *If $y^A_{h,a}$ is the head of C and $\neg y^B_{j,b} \in C$ then $j > h$.*
2. *If f^A_h is the head of C then there is no $\neg y^B_{j,b} \in C$.*
3. *If f^A_h is the head of C and $\neg f^B_j \in C$ then $j > h$.*

Proof. Invariant 2 holds in all axioms.

For the inductive step we only need to consider the resolution case as annotations do not affect the indices of the literals. Let C be derived from resolving D_1 and D_2.

Assume first that the resolved variable is $y^E_{k,e}$ in D_2. By induction hypothesis we know that $y^A_{h,a}$ or y_0 of C is the head of D_1, but we can ignore the case of y_0. If we have a negative literal $\neg y^B_{j,b} \in C$, then $\neg y^B_{j,b} \in D_1$ or $\neg y^B_{j,b} \in D_2$. If in D_1 then $j > h$ by Invariant 2 for D_1. If in D_2 then $j > k$ by Invariant 2 for D_2. As $\neg y^E_{k,e} \in D_1$ we get $k > h$ again by Invariant 2 for D_1. Therefore $j > h$ and Invariant 2 holds for C.

If the resolved variable is $f^E_k \in D_2$ then by induction hypothesis we only add negative literals $\neg f^B_b$ from D_1 with $b > k$. In order to falsify the invariant we would need $f^A_h \in D_1$ as the head, but then we know $h < k < b$, satisfying the invariant. □

The next two invariants explain how the annotation of the head determines all further annotations in the clause.

Invariant 3 *If positive literal $f_i{}^\tau \in C$ then the following two conditions hold: $x_i \in \mathsf{dom}(\tau)$ and all literals in C have exactly the same annotations.*

Proof. This is true in all axioms, and these annotations cannot be removed in the inductive step. We can only alter the annotations uniformly by instantiation. Suppose we have that C is the resolvent of D_1 and D_2. Using Invariant 2, any resolved variable is $\neg f_j{}^\tau \in D_1$ and must resolve with a clause where all literals have the same annotations as well, hence C has the same annotation in every literal. □

Invariant 4 *If $y^A_{h,a}$ is the head of C and if $\neg y^B_{j,b} \in C$ (or $\neg f^B_j \in C$), then $A \cup \{a/x_h\} \subseteq B$, where all extra annotations in B are of the form c_k/x_k for $k > h$. In other words the head literal $y^A_{h,a}$ determines all annotations up to x_h.*

Proof. For the base case we only need to consider the axioms C^c_i with negative existential literals. The head of C^c_i is $y_{i,c}$ and there are no universal variables in the clause of lower level than the head,

hence its annotation $A = \emptyset$. The axiom is instantiated so that c/x_i is added to the negative literals, hence we satisfy the invariant.

For the inductive step, suppose first that C is derived from instantiation of clause D. By Invariant 2 we know that $y^A_{h,a}$ is the lowest level literal in the clause D. Any annotation involving x_l with $l < h$ is therefore both added to A and to the annotations of all other literals.

Now suppose C is derived from resolving D_1 and D_2. Without loss of generality the pivot appears positively in D_2. We again use the fact that the head $y^A_{h,a}$ of C is also the head of D_1. If negative literal $\neg z^B \in C$ comes from D_1 then $A \cup \{a/x_h\} \subseteq B$ by Invariant 4 for D_1.

If, on the other hand, negative literal $\neg z^B \in C$ comes from D_2, then we consider the nature of the resolved variable. We first let the resolved variable be $y^E_{k,e}$ in D_2. Then $E \cup \{e/x_k\} \subseteq B$ by Invariant 4 for D_2. However, as $\neg y^E_{k,e} \in D_1$ then $A \cup \{a/x_h\} \subseteq E \subseteq E \cup \{e/x_k\} \subseteq B$. Likewise for an annotation in B of level lower than $\mathrm{lv}(y_{h,a})$, it must be in E and hence in A.

If the resolved variable is f^E_k then $E = B$ by Invariant 3, but since $A \cup \{a/x_h\} \subseteq E$ we are done. □

Invariant 5 *If $\neg y^B_{j,b} \in C$ then for all k, $h \leq k < j$ there is $c_k \in \{0, 1\}$ such that $c_k/x_k \in B$.*

Proof. Invariant 5 holds in the axiom case and is unaffected by instantiation.

Now suppose C is derived from resolving D_1 and D_2. Without loss of generality the resolved variable is z^E in D_2. All literals in C that come from D_1 already fulfil Invariant 5. Now we can branch on the nature of variable z.

If $z^E = y^E_{k,e}$ then for the literals coming from D_2 we use Invariant 4. Since $\neg y^E_{k,e} \in D_1$ the annotation E already contains all the necessary assignments for head $y^A_{h,a}$. But since $E \cup \{e/x_k\} \subseteq B$ for any $\neg y^B_{j,b} \in D_2$, then together with Invariant 5 for D_2, these literals have the required annotations for the new head $y^A_{h,a}$.

If the resolved variable is f^E_k, then we do not add or remove any y literals. Hence the induction hypothesis is preserved. □

Invariant 6 *If $\neg y^B_{j,b} \in C$ with $j \leq t$, then there is no $\neg y^D_{k,d} \in C$ such that $B \cup \{b/x_j\} \subseteq D$.*

Proof. We start with the base case for which we consider the axioms. Invariant 6 holds, because C_0 and all C^c_j with $j \leq t$ are the only clauses with negative existential literals, these are all of the same level.

For the inductive step there are two possibilities: either C is derived by instantiation or by resolution. Suppose first that C is derived by instantiation of clause D. We know from Invariants 2 and 5 and induction hypothesis that for any $\neg y^B_{j,b} \in D$ and $\neg y^E_{k,e} \in D$ with $k > j$ there is some $c/x_l \in B \cup \{b/x_j\}$ such that $(1-c)/x_l \in E$ and this conflict does not change by instantiation.

Now suppose C is derived from resolving D_1 and D_2. The important case is when the resolved variable is $y^E_{k,e}$ in D_2. We need to check that if $\neg y^B_{j,b} \in D_1$ and $\neg y^F_{l,f} \in D_2$, then there is some conflict in the annotations (by using Invariants 2 and 5 for C). By Invariant 6 for D_1 we know that

there is some $c/x_l \in B \cup \{b/x_j\}$ such that $(1-c)/x_l \in E$. By Invariant 4 we have $E \cup \{e/x_k\} \subseteq F$, hence $(1-c)/x_l \in F$.

The case where the resolved variable is f_k^E is simpler. We do not add or remove any y literals so the induction hypothesis is preserved. □

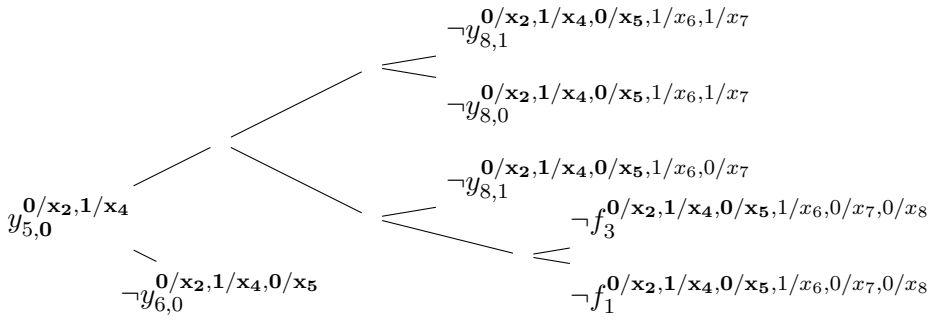


Fig. 23. Structure of an example clause in an IR-calc refutation of $\mathrm{KBKF}(8)$

In order to provide some intuition on the proofs of $\mathrm{KBKF}(t)\backslash\{\neg y_0\}$, we will illustrate how the invariants allow us to study clauses as binary trees.

Example 11. Consider Figure 23. The literals are labelling a tree that branches on the variable index and annotations. By Invariant 1 there is only one head of the clause and we place it at the root of the tree. We branch by increasing the index i of the literals $y_{i,c}$, which will make all negative literals descendants of the root by Invariant 2. In bold we highlight Invariant 4. The remaining annotations must be present by Invariant 5, but we have two choices for each annotation, plus a choice of c for $y_{i,c}$. We use these choices to construct a binary tree. Notice that none of the internal nodes produces a literal: this is prevented by Invariant 6.

Hence the positive literal is the root and the negative literals are the leaves. We can imagine instantiation of clauses to increase the bold part of the annotations and resolution on y literals to merge two different trees on leaf and root to form a new tree.

We will now give the ***overall idea of our lower bound argument***. For a clause C we define a set $\Sigma(C)$ of annotations associated with C. Our lower bound argument then rests on counting the set $\Sigma(C)$ as we progress through the proof. More precisely, we show that axioms have empty Σ (Lemma 12) and that instantiation steps do not add any new elements to Σ at all (Lemma 15). In a resolution step $\frac{D_1 \quad D_2}{C}$, the set $\Sigma(C)$ either equals $\Sigma(D_1) \cup \Sigma(D_2)$ or grows by exactly one new element (Lemma 16). In some sense, we only make progress in the proof in the latter case, and we need exponentially many resolution steps of this kind. Putting everything together, we find that by the end of the proof we must have collected all the exponentially many annotations in $\Sigma(y_0)$, implying an exponential lower bound to the proof length (Theorem 16).

We now give the details of the formal argument. We start with the definition of Σ, which we will first define for annotated variables and then for clauses.

Definition 8 (Σ **for variables).**

1. *We define $\Sigma(y_0)$ as the set of all complete assignments to variables $x_1, \ldots, x_t$.*
2. *For $y^B_{j,b}$ with $\mathsf{dom}(B) = \{x_i \mid i < j\}$ we define $\Sigma(y^B_{j,b}) = \{X \in \Sigma(y_0) \mid B \cup \{b/x_j\} \subseteq X\}$. Invariant 6 ensures that if $\neg y^B_{j,b}$ and $\neg y^D_{k,d}$ appear together in a clause, then $\Sigma(y^B_{j,b}) \cap \Sigma(y^D_{k,d}) = \emptyset$.*
3. *For f^B_j with a complete assignment B, we define $\Sigma(f^B_j) = \{B\}$, otherwise if B is not complete we set $\Sigma(f^B_j) = \emptyset$.*

We now extend this definition to clauses, which we first classify into three types.

Definition 9. *We class C a clause in the proof of $G(t)$ as follows:*

- *Type 1 clause: The head of the clause is f^A_i.*
- *Type 2 clause: The head of the clause is $y^A_{h,a}$ and there is some x_j with $j < h$ where x_j is not given a value in A.*
- *Type 3 clause: Any other clause. These will have head y_0 or $y^A_{h,a}$ where for all $j < h$, $x_j \in \mathsf{dom}(A)$.*

Definition 10 (Σ **for clauses).** *Let C be a clause in an IR-calc proof of y_0 from $G(t)$.*

1. *When C is a type-1 or type-2 clause we set $\Sigma(C) = \emptyset$.*
2. *For a type-3 clause C with head y we set $\Sigma(C) = \Sigma(y) \setminus \left(\bigcup_{\neg l \in C} \Sigma(l)\right)$.*

Remark 2. For a type-3 clause C we have the following properties of $\Sigma(C)$:

- We only remove an annotation when it was originally added from the presence of the head $y^A_{h,a}$ (or y_0). This is true by Invariant 4.
- Unless for some $0 \leq j \leq t$ we have $\neg f^X_j \in C$, we only remove an annotation X at most once from $\Sigma(C)$. This holds by Invariant 6. If $\neg f^X_j \in C$ for $0 \leq j \leq t$, then X can be removed from Σ up to t times in total by $\neg f^X_l \in C$.

An important fact is that for axioms we get empty Σ as we verify in the next lemma.

Lemma 12. *For each clause $C \in \mathrm{KBKF}(t) \setminus \{\neg y_0\}$, instantiated as an IR-calc axiom C^B, we get $\Sigma(C^B) = \emptyset$.*

Proof. For axiom C_0 we first add all annotations to Σ, but then due to the presence of $\neg y_{1,0}$ and $\neg y_{1,0}$ remove all annotations starting with $0/x_1$ and $1/x_1$, respectively. This results in $\Sigma(C_0) = \emptyset$.

Using $C^0_1 = \{y_{1,0}, x_1, \neg y_{2,0}, \neg y_{2,1}\}$ as an IR-calc axiom results in $\{y_{1,0}, \neg y^{0/x_1}_{2,0}, \neg y^{0/x_1}_{2,1}\}$. Computing Σ of this clause, we first add all annotations starting with $0/x_1$, but then remove all annotations starting with $(0/x_1, 0/x_2)$ and $(0/x_1, 1/x_2)$, yielding again empty Σ. Analogous reasoning applies to C^1_1.

When using clauses C^0_i, C^1_i with $2 \leq i \leq t$ as IR-calc axioms, we obtain type-2 clauses, which have empty Σ by definition. The remaining clauses C^0_{t+i}, C^1_{t+i} give rise to type-1 clauses, again with empty Σ. □

It will be crucial for our lower bound argument to understand how $\Sigma(C)$ changes when we go through the clauses C in the proof. For this we first need two technical lemmas on the structure of IR-calc proofs of $\mathrm{KBKF}(t) \setminus \{\neg y_0\}$.

Lemma 13. *In an IR-calc proof from $G(t)$, a type-2 clause cannot be resolved with a type-3 clause.*

Proof. Suppose we have a resolution step between a type-2 and a type-3 clause. We can deduce that the resolved variable is $y^E_{k,e}$, otherwise one of the clauses is a type-1 clause.

The resolved variable has a complete annotation as, by Invariants 4 and 5, type-3 clauses have them for literals $\neg y^E_{j,e}$. However, this means that, by Invariants 4 and 5, the head of the type-2 clause also must have a complete annotation, contradicting our assumption. □

Lemma 14. *Let C be a type-2 or type-3 clause in an IR-calc proof from $G(t)$. If $\neg f^B_j \in C$ and $\neg f^B_l \notin C$ with $j, l > 0$, then there is an annotation of x_l in B.*

Proof. We proceed by induction on the number of lines to derive C. The base case is vacuously true as all $\neg f_j$ literals get introduced in the same axioms.

For the inductive step, we distinguish whether C is derived by instantiation or by resolution. If C is derived by instantiation of D we do not lose any annotation, so the hypothesis remains true. If C is derived by resolving D_1 and D_2, the claim still holds if the resolved variable is different from any f^B_l. If we resolve on f^B_l then D_2 is a type-3 clause and by Invariant 3, x_l appears in B and any other annotation of a literal introduced by D_2. □

The next two lemmas are the key to our lower bound. We show first that the set Σ is not affected by instantiation steps. In Lemma 16 we will then analyse how Σ changes in a resolution step.

Lemma 15. *Suppose in an IR-calc proof from $G(t)$ we instantiate clause D to get clause D'. Then $\Sigma(D) \supseteq \Sigma(D')$.*

Proof. If D is a type-1 clause it remains a type-1 clause under instantiation. Hence $\Sigma(D')$ remains empty.

If D is a type-3 clause it has complete annotations in its head, and this is unchanged by instantiation. Moreover, by Invariant 5, every $\neg y^B_{j,b} \in D$ has a complete annotation, which again is unaffected by instantiation. The only possible effect of the instantiation on type-3 clauses is therefore that negative literals $\neg f^B_j \in D$ receive a complete annotation, which means that the assignment B is deleted from $\Sigma(D')$, thus $\Sigma(D') \subseteq \Sigma(D)$.

If D is a type-2 clause then the only problem arises when instantiation makes the head annotation complete, i.e., the clause turns into a type-3 clause. In this case we will show that $\Sigma(D')$ is empty, hence $\Sigma(D') = \Sigma(D) = \emptyset$. By induction on the number of lines we show:

Induction Hypothesis: Let D be a type-2 clause with head $y^A_{h,a}$ that can be instantiated by σ to get a type-3 clause D'. Then $\Sigma(D')$ is empty.

For the base case, we observe that instantiating any type-2 axiom always gives empty Σ by Lemma 12. For the inductive argument we can ignore the case where D is derived by instantiation as that instantiation could have been incorporated into σ.

Let D now be derived by resolving two clauses D_1 and D_2. Assume without loss of generality that D_1 is a type-2 clause. By Lemma 13 the other clause D_2 must be type-1 or type-2.

Suppose first that D_2 is a type-1 clause with head f_j^B. Let D_1' be the instantiation of D_1 by σ. Since σ gives a complete assignment to the annotations of $y_{j,b}^B$ in D it must also do so in D_1. We also observe that $\neg f_j^B \in D_1$ becomes $\neg f_j^X \in D_1'$ under σ. This will only have an effect when X is complete, otherwise we can ignore this literal in terms of Σ. By inductive hypothesis we have $\Sigma(D_1') = \emptyset$. The only result of the resolution step with D_2 is the removal of the literal $\neg f_j^B$ and (by Invariants 2 and 3) potential replacement with other literals $\neg f_k^B$ for $k > j$. In order for this to have any effect on the inductive claim, there can be no other $\neg f_l^B$ (for $1 \leq l \leq t, l \neq j$) in D_1 or D_2, nor can there be $\neg y_{l,u}^G \in D'$ such that $G \subset B$. Hence by Lemma 14, B contains annotations for all x_l with $1 \leq l \leq t, l \neq j$. Further B contains an annotation of x_j as D_2 must contain such an annotation by Invariant 3. Therefore B is a complete assignment to $x_1, \ldots, x_n$. By Invariant 4 this can only happen when D_1 is a type-3 clause rather than a type-2 clause.

Suppose instead, D_2 is a type-2 clause, and without loss of generality the resolved variable y_k^E is positive in D_2, i.e., it is the head of D_2. Let D_2' be the instantiation of D_2 by σ. Under σ the resolved variable is $y_k^{E'}$. Since σ gives a complete assignment to the annotations of the $y_{j,b}$ literals in D it must also do so in D_2, i.e., D_2' is type 3. This holds since D_2 is type 2 and therefore has a negative literal; with Invariant 4 this implies that D_2' is type 3. By inductive hypothesis we therefore have $\Sigma(D_2') = \emptyset$. When computing $\Sigma(D')$, the lack of the negative literal on the resolved variable means we may have additional elements only from $\Sigma(y_k^{E'})$ in $\Sigma(D')$. However, these were exactly the assignments that were added by the head $y_k^{E'}$ in D_2' and so we know — as $\Sigma(D_2')$ is empty — that we have the sufficient literals in D' to remove all elements in $\Sigma(D')$. We do not lose any $\neg f_j^B$ literals in this case, so the inductive claim holds. □

The next lemma is crucially important for the lower bound, it explains the conditions of new elements being added to Σ.

Lemma 16. *Let $\sqcup$ denote disjoint union. Suppose in an IR-calc proof from $G(t)$ we resolve D_1 with D_2 to get clause D, where the resolved variable is positive in D_2.*

If D_1 is a type-3 clause that is resolved with the type-1 clause D_2 with head f_j^B for $j > 0$ and there is no $k > 0$, $k \neq j$ such that $\neg f_k^B \in D_1$ nor $\neg y_{k,e}^E \in D_1$ such that $E \subseteq B$, then $\Sigma(D) = \Sigma(D_1) \sqcup \Sigma(D_2) \sqcup \{B\} = \Sigma(D_1) \sqcup \{B\}$. Otherwise $\Sigma(D) = \Sigma(D_1) \sqcup \Sigma(D_2)$.

Proof. In the first case, D_1 is a type-3 clause and D_2 is a type-1 clause. Then the resolution step removes $\neg f_j^B$ from the clause D_1. The resolvent D must be a type-3 clause as all annotations remain. Because there is no $k > 0$, $k \neq j$ such that $\neg f_k^B \in D_1$, we infer by Lemma 14 that B is complete and $B \in \Sigma(D)$. All other annotations remain in $\Sigma(D)$.

If we otherwise resolve a type-3 clause D_1 with a type-1 clause D_2, but there is another $\neg f_k^B \in D_1$, $\neg y_{k,e}^E \in D_1$ such that $E \subseteq B$, or $\neg f_k^B \in D_2$ (by Invariant 3 this will necessarily occur if D_2 is not a singleton), then the same assignments are added and deleted in $\Sigma(D)$ as in $\Sigma(D_1)$ hence $\Sigma(D) = \Sigma(D_1) = \Sigma(D_1) \sqcup \Sigma(D_2)$.

Consider now the remaining cases. If we resolve a type-1 clause with a type-2 clause, then we obtain a type-2 clause, hence Σ remains empty. Likewise, resolving two type-2 clauses results in a type-2 clause and therefore again empty Σ. By Lemma 13, we cannot resolve type-2 with type-3 clauses.

Therefore the last case is when two type-3 clauses are resolved. Let D_2 provide the positive resolved literal $y^E_{k,e}$. Because $y^E_{k,e}$ is the head of D_2, every annotation $X \in \Sigma(D_2)$ has $E \cup \{e/x_k\} \subseteq X$. As $\neg y^E_{k,e} \in D_1$, the sets of assignments $\Sigma(D_1)$ and $\Sigma(D_2)$ are disjoint. But also there is no annotation $Y \in \Sigma(D_1)$ with $E \cup \{e/x_k\} \subseteq Y$ because of the presence of $\neg y^E_{k,e}$ in D_1 and Invariant 6. Therefore $\Sigma(D)$ is the union of $\Sigma(D_1)$ and $\Sigma(D_2)$ because in our adding/removing process we only get rid of instructions from $y^E_{k,e}$ and $\neg y^E_{k,e}$ which cancel, and keep all other instructions. □

We can now deduce that all proofs of $\mathrm{KBKF}(t)$ in IR-calc are of at least exponential size.

Theorem 16. *All proofs of* $\mathrm{KBKF}(t)$ *in IR-calc have length at least* 2^t.

Proof. We will show that all IR-calc proofs of y_0 from $\mathrm{KBKF}(t) \setminus \{\neg y_0\}$ are of exponential size. By Lemma 11 each refutation of $\mathrm{KBKF}(t)$ can be transformed into one of these in polynomial time. Hence each refutation of $\mathrm{KBKF}(t)$ must be of exponential size.

Consider now an IR-calc proof $\pi = (D_1, D_2, \ldots, D_m)$ of y_0 from $G(t)$ and define $s_i = |\bigcup_{j=1}^{i} \Sigma(D_j)|$. By Lemma 12, the axioms all have empty Σ, hence $s_1 = 0$. By Definition 10, the set $\Sigma(y_0)$ contains all 2^t complete annotations, therefore $s_m = 2^t$. Progressing in the proof from the axioms to y_0, we therefore build up the set Σ from an empty to an exponential-size set. If the clause D_{i+1} is an axiom or derived by instantiation, then $s_i = s_{i+1}$ by Lemmas 12 and 15. For a resolution step, we have $s_{i+1} \leq s_i + 1$ by Lemma 16. Therefore the proof π contains at least 2^t resolution steps. □

Since IR-calc simulates (Theorem 3), we get as a corollary the hardness of $\mathrm{KBKF}(t)$ for Q-Res as already stated in [77].

Corollary 5. *All proofs of* $\mathrm{KBKF}(t)$ *in Q-Res are of at least exponential size.*

As the formulas $\mathrm{KBKF}(t)$ are easy for long-distance and universal resolution [53, 116] we obtain the following exponential separations.

Corollary 6. *IR-calc neither simulates LD-Q-Res nor QU-Res.*

Proof. The formulas $\mathrm{KBKF}(t)$ admit polynomial-size proofs in LD-Q-Res [53] and QU-Res [116], and therefore by the known simulations (including those shown here) also in LQU-Res, LQU$^+$-Res, and IRM-calc. □

6.3 Extending the Lower Bound to IRM-calc

In the previous section we showed that IR-calc neither simulates LD-Q-Res nor QU-Res. The stronger calculus IRM-calc simulates LD-Q-Res by Theorem 5. Therefore, in terms of the simulation order (cf. Figure 22) the only question still open by now is whether IRM-calc also simulates QU-Res. We will show here that this simulation does not hold. To do this we need formulas that are hard for IRM-calc, but easy for QU-Res.

For this we use a modification of the $\mathrm{KBKF}(t)$ formulas from the last section. These modified formulas were introduced in [11], where they are shown to be hard for LD-Q-Res. We will show here that they are indeed hard for the stronger system IRM-calc.

Definition 11 (Balabanov, Widl, Jiang [11]). *The formula* $\mathrm{KBKF}_{\mathrm{lq}}(t)$ *has quantifier prefix* $\exists y_0, y_{1,0}, y_{1,1}\, \forall x_1\, \exists y_{2,0}, y_{2,1}\, \forall x_2 \ldots \forall x_{t-1}\, \exists y_{t,0}, y_{t,1}\, \forall x_t\, \exists f_1 \ldots f_t$ *and matrix clauses*

$$
\begin{aligned}
C_- &= \{\neg y_0\} \\
C_0 &= \{y_0, \neg y_{1,0}, \neg y_{1,1}, \neg f_1, \ldots, \neg f_t\} \\
C_i^0 &= \{y_{i,0}, x_i, \neg y_{i+1,0}, \neg y_{i+1,1}, \neg f_1, \ldots, \neg f_t\} \quad \textit{for } i \in [t-1] \\
C_i^1 &= \{y_{i,1}, \neg x_i, \neg y_{i+1,0}, \neg y_{i+1,1}, \neg f_1, \ldots, \neg f_t\} \ \textit{for } i \in [t-1]
\end{aligned}
$$

$$
\begin{aligned}
&C_t^0 = \{y_{t,0}, x_t, \neg f_1, \ldots, \neg f_t\} \ \ C_t^1 = \{y_{t,1}, \neg x_t, \neg f_1, \ldots, \neg f_t\} \\
&F_i^0 = \{x_i, f_i, \neg f_{i+1}, \ldots, \neg f_t\} \ F_i^1 = \{\neg x_i, f_i, \neg f_{i+1}, \ldots, \neg f_t\} \ \textit{for } i \in [t].
\end{aligned}
$$

We first observe that these formulas remain hard for IR-calc.

Lemma 17. $\mathrm{KBKF}_{\mathrm{lq}}(t)$ *require exponential-size proofs in IR-calc.*

Proof. We define $G(t)$ as $\mathrm{KBKF}_{\mathrm{lq}}(t) \backslash \{\neg y_0\}$ and use exactly the same chain of arguments as in the previous section to show that $\mathrm{KBKF}_{\mathrm{lq}}(t) \backslash \{\neg y_0\}$ require exponential proofs of $\{y_0\}$. We remark that in the previous section we relaxed the invariants for $G(t)$ more than was necessary, but these are important for the $\mathrm{KBKF}_{\mathrm{lq}}(t)$ lower bound argument to go through. This shows that $\mathrm{KBKF}_{\mathrm{lq}}(t)$ requires exponential-size proofs in IR-calc. □

Remark 3. Note that from Section 6.2 Invariant 1 still holds as there are never two positive literals. Invariant 3 holds as merging does not disrupt annotations as they are all identical. Invariant 4 also holds because the role of the head does not change under merging.

We also introduce new invariants specifically for IRM-calc proofs.

Lemma 18. *For any clause* C *in an IRM-calc proof of* $\mathrm{KBKF}_{\mathrm{lq}}(t)$ *the following holds:*

A. If positive literal $y_{i,c}^{\tau} \in C$ *then for all* $j < i$ *either* $x_j \in \mathsf{dom}(\tau)$ *or* $\neg f_j^{\sigma} \in C$ *for some* σ.
B. If $\mathrm{var}(f_i)^{\tau} \in C$ *then for all* $j > i$ *either* $x_j \in \mathsf{dom}(\tau)$ *or* $\neg f_j^{\sigma} \in C$ *with* $\mathsf{dom}(\tau) = \mathsf{dom}(\sigma)$ *and* τ *and* σ *differ only where* $*$ *appears.*

Proof. We prove these claims by induction on the number of lines. For the base cases it is sufficient to observe that all the axioms have these properties.

For the inductive step, we first consider instantiation and merging steps. These keep literals and elements already in the annotation domains, thus preserving Invariant A. For Invariant B, merging may turn a constant value into $*$, but this is allowed and preserves the invariant.

We next consider resolution steps. For Invariant A, assume the head of one of the parent clauses is $y_{i,c}^{\tau}$. We only have to consider the case when f_j with $j < i$ is the pivot. This means that we resolve with a clause that contains a positive literal f_j^{σ}. From Invariant 3 we infer $x_j \in \mathsf{dom}(\sigma)$. During the resolution step the head of the clause $y_{i,c}$ will be instantiated by x_j which will give it an annotation as $j < i$.

For Invariant B, we only need to consider a loss of an f_j literal for $j > i$, which is the pivot when $\mathrm{var}(f_i)^{\tau}$ is otherwise present. In this case we must resolve with a clause that contains a positive f_j^{σ}. By Invariant 3 we get $x_j \in \mathsf{dom}(\sigma)$. During the resolution step $\mathrm{var}(f_i)^{\tau}$ will therefore get annotated with the correct value for x_j. □

Theorem 17. $\mathrm{KBKF}_{\mathrm{lq}}(t)$ *require exponential-size proofs in* IRM-calc.

Proof. We can now show the lower bound for IRM-calc. The proof uses the same technique as in [11]. There they showed that any LD-Q-Res refutation of $\mathrm{KBKF}_{\mathrm{lq}}$ can be transformed in polynomial time into a Q-Res refutation. Instead we show here that any IRM-calc refutation of $\mathrm{KBKF}_{\mathrm{lq}}$ can be transformed in polynomial time into an IR-calc refutation.

To do this we consider the $*$ annotations and the merge rule. Without applications of the merge rule in IRM-calc we essentially have an IR-calc proof (the slight change in the resolution rule will not matter). We therefore now consider exhaustively all the possible literals that can be produced by merging.

1. A positive literal l^{τ} with $*/x_j \in \tau$ is introduced.
2. A negative literal $\neg f_i^{\tau}$ with $*/x_j \in \tau$ is introduced, where $\forall j \geq i, */x_j \notin \tau$.
3. A negative literal $\neg f_i^{\tau}$ is introduced, where $*/x_i \in \tau$.
4. A negative literal $\neg f_i^{\tau}$ is introduced, where $\exists j > i, */x_j \in \tau$.
5. A negative literal $\neg y_{i,c}^{\tau}$ is introduced.

We will either show, that these are impossible or are not essential and can be removed in a polynomial number of IR-calc steps.

1. It is impossible that there is a positive literal l^{τ} with $*/x_j \in \tau$.

It is not possible to introduce a $*$ by merging as there are never two positive literals in a clause. It is also impossible by resolution as we would need a positive pivot with a $*/x_j$ annotation. There must be an earliest point in the proof where this occurs but by the well-ordering principle this is impossible.

2. Literals $\neg f_i^{\tau_1}$ and $\neg f_i^{\tau_2}$ are merged in clause C to get $\neg f_i^{\tau}$ where $\forall j \geq i, */x_j \notin \tau$.

This is possible, but here we show that we can transform this step into a number of steps that do not require merging. If we start and repeat from the first time this appears, we can deal with 2 by resolving $\neg f_i^{\tau_1}$ and $\neg f_i^{\tau_2}$ away before the merging. This is done by resolving twice with clause D (which we will construct below) with $f_i^{\tau_1}$ and $f_i^{\tau_2}$ as pivots. The purpose of D is to resolve while only adding literals that are there already and not introducing any new annotations. To construct D we first observe that by Invariant 3, C has head $y_{h,b}^A$ (or y_0). For any $j < i$ such that $c/x_j \in \tau_1$ and $(1-c)/x_j \in \tau_2$, either $*/x_j \in A$, which is excluded by case 1, or $h \leq j$. Therefore since one $*$ must be created for merging to occur, $h \leq j < i$. Now consider all $j \geq i$: there may exist c_j such that $c_j/x_j \in \tau$, $c_j \in \{0,1\}$ because we are not merging on these annotations. D is created by repeatedly resolving axiom $F_i^{c_i}$ with the set of axioms $\Delta = \{F_j^{c_j} \mid j > i, c_j/x_j \in \tau\}$ in order of increasing j. After this D will be a clause with a f_i^σ head, which will be our pivot and contains some literals $\neg f_j^\sigma$, which will later be merged with literals from C. The fact that $\sigma \subset \tau_1$ and $\sigma \subset \tau_2$ means we can use f_i^σ as a pivot literal resolving with $\neg f_i^{\tau_1}$ and $\neg f_i^{\tau_2}$, and in fact we do both by reusing D.

After resolving twice with D we remove $\neg f_i^{\tau_1}$ and $\neg f_i^{\tau_2}$ without instantiating the clause C. There may be some additional literals $\neg f_j^{\tau_1}$ and $\neg f_j^{\tau_2}$ for $j > i$ from resolving with D, but similar literals must have previously existed in C by Invariant B. These possibly had $*$ annotations in C, but $0, 1$ values from D, so merging can make these the same as they were in C (but we will see below that this is not needed as the remaining cases exclude this possibility). Thus we end up with a subclause of C and thus shorten the remaining proof.

We repeat this until all literals of this form have been removed. We will look at the remaining cases to observe that we actually already have an IR-calc refutation, as all the remaining cases are impossible.

3. There cannot be a negative literal $\neg f_i^\tau$ where $*/x_j \in \tau$ and $j = i$.
4. There cannot be a negative literal $\neg f_i^\tau$ where there is some $j > i$ with $*/x_j \in \tau$.
5. There cannot be a negative literal $\neg y_{i,c}^\tau$ where there is some $j < i$ with $*/x_j \in \tau$.

In all the cases consider that later in the proof we must resolve this negative literal as part of clause C with a clause D to remove that literal. D must now have the positive version of the literal, but it must contain no annotation on x_j.

For case 3, this is impossible as by Invariant 3 it must contain an annotation on x_i. In case 4 we must have $f_j^\sigma \in D$ by Invariant B and so in the resolvent we have $f_j^{\sigma'}$. But since we instantiate when we resolve and $x_j \notin \mathsf{dom}(\sigma)$, we get $*/x_j \in \sigma'$ which takes us to case 3 again. In case 5 we must have $f_j^\sigma \in D$ by Invariant B and so in the resolvent we have $f_j^{\sigma'}$. By Invariant 2 we have that $x_j \notin \mathsf{dom}(\sigma)$, hence $*/x_j \in \sigma'$ which takes us to case 3 again.

Therefore every IRM-calc proof of $\mathrm{KBKF}_{\mathrm{lq}}$ can be transformed in polynomial time to a IR-calc proof. Hence IRM-calc inherits an exponential lower bound from Lemma 17. □

In contrast, $\mathrm{KBKF}_{\mathrm{lq}}$ admits polynomial-size refutations in QU-Res as shown in [11]. Therefore we obtain the following separation.

Corollary 7. *IRM-calc does not simulate QU-Res.*

It is interesting to note that the short QU-Res refutations of KBKF given in [116] (cf. Theorem 14) uses DAG-like structures, reusing derived clauses. In Chapter 7 we explore a new lower bound technique and clarify that DAG-like proofs are required for short QU-Res refutations of KBKF.

Chapter 7

A Game Technique and Lower Bounds in Tree-like QBF Calculi

In the previous sections we looked closely at the different QBF analogues to resolution calculi that were structured as the usual directed acyclic graph (DAG), and explored a new technique– strategy extraction for finding proof size lower bounds. In this section we will now concentrate on tree-like QBF resolution calculi using a lifted version of the game-theoretic technique developed for classical resolution.

This begins an investigation into lifting lower bound techniques from propositional to QBF logic. In Chapters 8 and 9 this is continued, with different techniques. In this chapter we exhibit a positive result, where the lower bound technique works very similarly in QBF as it did before, although we see in Chapter 9 that not every technique can be easily lifted to QBF.

Inspired by this asymmetric Prover-Delayer game of [27–29], we develop here a Prover-Delayer game which tightly characterises the proof size in tree-like Q-resolution. The general idea behind this game is that a Delayer claims to know a model to a false formula, while a Prover asks for values of variables until eventually finding a contradiction. In the course of the game the Delayer scores points proportional to the progress the Prover makes towards reaching a contradiction. By an information-theoretic argument we show that the optimal Delayer will score exactly logarithmically many points in the size of the smallest tree-like Q-resolution proof of the formula. Thus exhibiting clever Delayer strategies automatically gives lower bounds to the proof size, and in principle these bounds are guaranteed to be optimal. In comparison to the game of [27–29], our formulation here needs a somewhat more powerful Prover, who can forget information as well as freely set universal variables. This is necessary as the Prover needs to simulate more complex Q-resolution proofs involving universal variables and rules for them absent in propositional resolution.

In addition, we show that a slight modification of the game also characterises the proof size in tree-like QU-resolution. QU-resolution is a stronger system than Q-resolution [116] (though it is not known whether this also holds for the tree-like versions).

We illustrate this new technique with three examples. The first was used by Janota and Marques-Silva [74] to separate Q-resolution from the system $\forall$Exp+Res defined in [74] (see Chapter 4 Figure 13). We use these separating formulas as an easy first illustration of our technique. Our Delayer strategy as well as the analysis here are quite straightforward; in fact, a simple symmetric game in the spirit of [100] would suffice to get the lower bound.

The second example are QPARITY formulas defined in Chapter 5, where they exemplify the new lower bound technique based on strategy extraction. In this chapter we give a completely

different proof for the hardness of these formulas in tree-like Q-resolution based on our game characterisation. Unlike the proof in Chapter 5 our proof here is direct and does not depend on any circuit lower bounds.

Our third example are the well-known KBKF-formulas of Kleine Büning, Karpinski and Flögel [77]. In the same work [77], where Q-resolution was introduced, these formulas were suggested as hard formulas for the system. In Chapter 6, the formulas KBKF were even shown to be hard for IR-calc, a system stronger than Q-resolution. In fact, a number of further separations of QBF proof systems builds on the hardness of KBKF [11, 53] (cf. also Chapter 6 for further details and the formal proof). Here we use our new technique to show that these formulas require exponential-size proofs in tree-like QU-resolution, which in contrast to the previous two examples provides a new hardness result. This also has the interesting consequence that the formulas of Kleine Büning et al. exponentially separate tree-like and dag-like QU-resolution, as they are known to have short proofs in dag-like QU-resolution [116].

For the $\mathrm{KBKF}(t)$ formulas both the Delayer strategy as well as the scoring analysis are technically involved. It is also interesting to remark that here we indeed need the refined asymmetric game. The formulas $\mathrm{KBKF}(t)$ have very unbalanced proof trees and therefore we cannot use a symmetric Delayer, as symmetric games only yield a lower bound according to the largest full binary tree embeddable into the proof tree (cf. [28]).

Section 7.1 introduces the two-player game and the lower bound characterisation for Q-resolution. Section 7.2 adapts the game and lower bound characterisation for QU-Res. We then introduce our examples in the remaining sections. The hard formulas from Janota and Marques-Silva [74] are used in Section 7.3. The QPARITY formulas from Chapter 5 are used in Section 7.4. Finally a new lower bound using the KBKF formulas from Kleine Büning et al. [77] are used in Section 7.5.

The work in this chapter appeared in the Journal of Computer and System Sciences [26] and at the 9th International Conference on Language and Automata Theory and Applications [25].

7.1 Prover-Delayer Game

We present a two-player game along with a scoring system. The two players will be called Prover and Delayer. The game is played on a fully quantified false QBF ϕ with CNF matrix. The game proceeds in rounds and builds a partial assignment to the variables in the QBF, starting with the empty assignment, i.e. in the beginning all variables are unassigned. In the course of the game the Delayer gets points and tries to score as many points as possible. The Prover tries to win the game by falsifying the matrix and giving the Delayer as small a score as possible.

Each round of the game has the following phases:

1. *Setting universal variables:* The Prover can assign values to any number of universal variables that satisfy the following condition: A universal variable u can be assigned a value if every existential variable with a higher quantification level than u is currently unassigned.

2. *Declare Phase:* The Delayer can choose to assign values to any number of unassigned existential variables of his choice. The Delayer does not score any points for this.
3. *Query Phase:* This phase has three stages, similar to the original game:
   (a) The Prover queries the value of one existential variable x that is currently unassigned.
   (b) The Delayer replies with weights $p_0 \geq 0$ and $p_1 \geq 0$ such that $p_0 + p_1 = 1$.
   (c) The Prover assigns a value for x. If she assigns $x = b$ for some $b \in \{0, 1\}$, the Delayer scores $\lg(\frac{1}{p_b})$ points.
4. *Forget Phase:* The Prover can choose any number of assigned variables (without regard to how they are quantified) in this phase. Every variable chosen by the Prover in this phase will lose its assigned value and hence become an unassigned variable.

The Prover wins the game if any clause in ϕ is falsified. In every round, we check if the Prover has won the game after each phase.

The game only applies to false QBFs, i.e. the falsity of the formula is given as a 'promise'. Intuitively, the Delayer claims to know a model for the false QBF, which the Prover tries to query. Of course, as there is no model, the Prover can always win the game. The crux of the game is therefore not who wins, but how many points the Delayer scores before the Prover finally exposes the Delayer's lie.

Before explaining the connection of the game to Q-resolution, let us try to provide some intuition on the game semantics. The game can be seen as a procedural way of obtaining an assignment that falsifies the matrix of the QBF. At every stage of the game, the Prover maintains a partially filled vector with assignments to the variables in the formula. This vector can be seen by the Delayer as well. Throughout the game, the Prover can never assign values to existential variables without querying them and the Delayer can never assign values to universal variables.

The *'Setting Universal Variables' phase*, where the Prover can assign values to universal variables, mirrors the $\forall$-reduction rule of Q-Res. This intuition will become clear in the proof of Theorem 18.

The Declare and Query phases are used to assign values to the existential variables. The *Declare Phase* merely allows us to express simple strategies for the Delayer that still score sufficiently many points. The Declare Phase is not integral to the characterisation, however it allows lower bound arguments to be made concise by simplifying the states of the game. Note that any lower bound to the score in a strategy that uses the Declare phase non-trivially also holds for an optimal strategy where the Delayer does not use the Declare Phase at all.

The *Query Phase* is the most important phase of the game where the Prover obtains information about existential variables from the Delayer in exchange for points. The Delayer replies with weights so that the Prover is forced to concede points proportional to how much progress she makes in the game towards a contradiction. The intuition behind the scoring system defined as log of the inverse of the weights comes from the Shannon entropy and is made clear in the proof of Theorem 18. Loosely speaking, the Delayer will charge points proportional to the size of the subtree

in the shortest tree-like resolution refutation that the Prover enters by her choice. The query phase therefore corresponds to resolution steps in the Q-Resolution proof.

In the *Forget Phase*, the Prover can choose and delete any variable assignments from the assignment vector obtained so far. This phase is especially useful in preparing a universal variable to be assigned a new value in the next round. To do so, the Prover chooses the universal variable and all the existential variables with a higher quantification level that are currently assigned to lose their assigned values. Once this is done, the universal variable can be assigned a new value in the first phase of the next round of the game. This phase can also be used to prevent the Delayer from abusing the Declare Phase to stop the assignment of universal variables.

The game ends when the assignment vector holds an assignment that falsifies a clause in the matrix.

As a toy example, consider the following formula:

$$\exists e \forall u \exists c_1 \exists c_2 (u \Rightarrow c_1) \wedge (\neg u \Rightarrow c_2) \wedge (e \Rightarrow c_1) \wedge (\neg e \Rightarrow c_2) \wedge (\neg c_1 \vee \neg c_2).$$

We demonstrate a run of the game on the above formula along with the intermediate assignment vectors. Let v denote the assignment vector. The vector v will contain assignments in the order $\langle e, u, c_1, c_2 \rangle$. The game will end when v is an assignment that falsifies one of the matrix clauses. The game starts with all variables unassigned and hence $v = \langle -, -, -, - \rangle$.

- *Round 1:*
  - Setting universal variables: Prover assigns $u = 1$ and hence $v = \langle -, 1, -, - \rangle$.
  - Declare Phase: Delayer declares $c_1 = 1$ and $c_2 = 0$. This satisfies all the clauses that do not involve the variable e in them. $v = \langle -, 1, 1, 0 \rangle$.
  - Query Phase: The Prover queries the variable e. The Delayer replies with $p_0 = 0$ and $p_1 = 1$, thus forcing the Prover to set $e = 1$ and hence $v = \langle 1, 1, 1, 0 \rangle$, for which the Delayer scores $\lg 1 = 0$ points. The assignment does not falsify the formula.
  - Forget Phase: The Prover forgets the value of u, c_1 and c_2 while retaining the value of e. Note that this is possible since e has a lower quantification level than u. Hence we get $v = \langle 1, -, -, - \rangle$.
- *Round 2:*
  - Setting universal variables: Prover assigns $u = 0$ and we have $v = \langle 1, 0, -, - \rangle$.
  - Declare Phase: Delayer declares $c_2 = 1$. The vector $v = \langle 1, 0, -, 1 \rangle$ now satisfies all clauses that do not involve c_1.
  - Query Phase: Prover queries the variable c_1. The Delayer responds with $p_0 = 1/2$ and $p_1 = 1/2$. The Prover wins the game since both $\langle 1, 0, 1, 1 \rangle$ and $\langle 1, 0, 0, 1 \rangle$ falsify the formula. The Delayer scores 1 point.

It is true that the Delayer can score more points by not declaring any assignments in the Declare Phase. However, when showing lower bounds to the score obtained by the Delayer in an optimal strategy, we use the Declare Phase merely to simplify presentation.

We will now show that our game characterises tree-like Q-Resolution.

Theorem 18. *If ϕ has a tree-like Q-Resolution proof of size at most s, then there exists a Prover strategy such that any Delayer scores at most $\lg\lceil\frac{s}{2}\rceil$ points.*

Proof. Let Π be a tree-like Q-resolution refutation of ϕ of size $\leq s$. Informally, the Prover plays according to Π, starting at the empty clause and following a path in the tree to one of the axioms. At a resolution inference the Prover will query the resolved variable and at a universal reduction she will set the universal variable. The Prover will keep the invariant that at each moment in the game, the current assignment α assigns exactly all literals from the current clause C on the path in Π, and moreover α falsifies C. This invariant holds in the beginning at the empty clause, and in the end, Prover wins by falsifying an axiom.

We will now give details and first describe a randomised Prover strategy, i.e. the Prover chooses her answer to Delayer's queries randomly. We will later derandomise the Prover and make her strategy deterministic. Let the Prover be at a vertex in Π labelled with clause C. We describe what the Prover does in the three stages: Setting universal variables, Query phase and the Forget phase.

Setting universal variables: If the current clause C was derived in the proof Π by a $\forall$-reduction $\frac{C\vee z}{C}$, then Prover sets $z = 0$. This is possible as the current assignment contains only variables from C and all existential variables in C have a lower quantification level than z. Prover then moves down to the clause $C \vee z$. The Prover repeats this till arriving at a clause derived by the Resolution rule (or winning the game). Analogous reasoning applies for $\forall$-reduction steps $\frac{C\vee\neg z}{C}$ where Prover sets $z = 1$.

Query phase: Prover is now at a clause in Π that was derived by a Q-resolution step $\frac{C_1\vee x \quad C_2\vee\neg x}{C_1\vee C_2}$. If the Delayer already set the value of x in his Declare phase, then Prover just follows this choice and moves on in the proof tree, possibly setting further universal variables. She does this until she reaches a clause derived by resolution, where the resolved variable x is unassigned. Prover queries x. On Delayer replying with weights w_0 and w_1, the Prover chooses $x = i$ with probability w_i.

If $x = 0$, then Prover defines S to be the set of all variables not in $C_1 \vee x$ and proceeds down to the subtree under that clause. Else, she defines S to be all variables not in $C_2 \vee \neg x$ and proceeds down to the corresponding subtree.

Forget phase: The Prover forgets all variables in the set S.

For a fixed Delayer D, let $q_{D,\ell}$ denote the probability (over all random choices made within the game) that the game ends at leaf ℓ. Let π_D be the corresponding distribution induced on the leaves.

For the Prover strategy described above, we have the following claim:

Claim. If the game ends at a leaf ℓ, then the Delayer scores exactly $\alpha_\ell = \lg\left(\frac{1}{q_{D,\ell}}\right)$ points.

To prove the claim, note that since Π is a tree-like Q-resolution proof, there is exactly one path from the root of Π to ℓ. Let p be the unique path that leads to the leaf ℓ and let the number of random choices made along p be m. Then, we have $q_{D,\ell} = \prod_{i=1}^{m} q_i$ where q_i is the probability for the ith random choice made along p. Since p is the unique path that leads to ℓ, the number of points α_ℓ scored by the Delayer when the game ends at ℓ is exactly the number of points scored when the

game proceeds along the path p. The number of points scored by the Delayer along p is given by:

$$\alpha_\ell = \sum_{i=1}^{m} \lg\left(\frac{1}{q_i}\right) = \lg\left(\prod_i \frac{1}{q_i}\right) = \lg\left(\frac{1}{q_{D,\ell}}\right),$$

which proves the claim.

The Prover strategy we described is randomised. The expected score over all leaves ℓ is the following expression:

$$\sum_{\text{leaves } \ell \in \Pi} q_{D,\ell}\alpha_\ell = \sum_{\text{leaves } \ell \in \Pi} q_{D,\ell} \lg\left(\frac{1}{q_{D,\ell}}\right).$$

By definition, the latter sum is exactly the Shannon entropy $\mathcal{H}(\pi_D)$ of the distribution π_D. Since D is fixed, this entropy will be maximum when π_D is the uniform distribution; i.e., $\mathcal{H}(\pi_D)$ is maximum when, for all leaves ℓ, the probability that the game ends at ℓ is the same. A tree-like Q-resolution proof of size s has at most $\lceil s/2 \rceil$ leaves. So the support of the distribution π_D has size at most $\lceil s/2 \rceil$ and hence $\mathcal{H}(q_{D,\ell}) \leq \lg\lceil s/2 \rceil$.

If the expected score with the randomised Prover is $\leq \lg\lceil s/2 \rceil$, then there is a deterministic Prover who restricts the scores to at most $\lg\lceil s/2 \rceil$. Now we derandomise the Prover by just fixing her random choices accordingly.

We remark that the Delayer actually does not play against the randomised Prover (hence the Delayer cannot exploit that the Prover is randomised), but only against the deterministic Prover, which we know must exist by the argument above. By the probabilistic method, this deterministic Prover prevents every Delayer to earn more than $\lg\lceil s/2 \rceil$ points. □

To obtain the characterisation of Q-resolution we also need to show the opposite direction, exhibiting an optimal Delayer:

Theorem 19. *Let ϕ be an unsatisfiable QBF and let s be the size of a shortest tree-like Q-resolution proof for ϕ. Then there exists a Delayer who scores at least $\lg\lceil s/2 \rceil$ points against any Prover.*

Proof. For any unsatisfiable QBF ϕ, let $L(\phi)$ denote the number of leaves in the shortest tree-like Q-resolution refutation of ϕ. For a partial assignment $\boldsymbol{a}$ Let $L_{\boldsymbol{a}}(\phi)$ denote the number of leaves in the shortest tree-like Q-resolution derivation of any subset of $A = \{l \mid \boldsymbol{a}(l) = 0\}$ from ϕ.

The Delayer starts with an empty partial assignment $\boldsymbol{a}$ and changes $\boldsymbol{a}$ throughout the game. On receiving a query for an existential variable x, the Delayer does the following:

1. Updates $\boldsymbol{a}$ to reflect any changes made by the Prover to any of the variables. These changes include assignments made to both universal variables as well as existential variables.
2. Computes the quantities $\ell_0 = L_{\boldsymbol{a},x=0}(\phi)$ and $\ell_1 = L_{\boldsymbol{a},x=1}(\phi)$.
3. Replies with weights $w_0 = \frac{\ell_0}{\ell_0+\ell_1}$ and $w_1 = \frac{\ell_1}{\ell_0+\ell_1}$.

We will prove the theorem by induction, but since an assignment may change multiple times during a round we have to do induction on individual steps in the game (every time the assignment

potentially changes) rather than rounds. This is fine as we still have a finite number of steps. We show by induction on the number of steps left in the game that the delayer can score at least $\lg(L_{\boldsymbol{a}}(\phi))$ more points when $\boldsymbol{a}$ is the current assignment. Hence he will score $\lg(L_(\phi))$ points in total.

In the base cases, $\boldsymbol{a}$ falsifies a clause so $\lg(L_{\boldsymbol{a}}(\phi)) = 0$ and the delayer can get no more points.

Let n be the number of remaining steps in the game. For the kth case Assume the statement is true for all $n < k$. Now for $n = k$, consider an assignment $\boldsymbol{a}$ in the game where the Delayer can play so that he guarantees himself at most d more points.

If the Prover forgets variables and arrives at $\boldsymbol{a}'$ then $d \geq \lg(L_{\boldsymbol{a}'}(\phi)) \geq \lg(L_{\boldsymbol{a}}(\phi))$.

If the Prover then assigns universal variable u to 0 (without loss of generality) then the delayer still has the same potential score d. The universal variable can only be assigned if $\boldsymbol{a}$ does not block it, so universal reduction can be performed on the clause $\{u\} \cup \{l \mid \boldsymbol{a}(l) = 0\}$ with no change in the number of leaves. Hence $d \geq \lg(L_{\boldsymbol{a},u=0}(\phi)) \geq \lg(L_{\boldsymbol{a}}(\phi))$.

In the Query phase if the Prover chose $x = b$ where $b \in \{0, 1\}$, then the Delayer scores $\lg \frac{1}{w_b}$ for this step alone. We use the induction hypothesis to conclude that the remaining rounds in the game give the Delayer at least $\lg L_{\boldsymbol{a},x=b}(\phi)$. Hence d is at least

$$\lg\left(\frac{1}{w_b}\right) + \lg L_{\boldsymbol{a},x=b}(\phi) = \lg \frac{L_{\boldsymbol{a},x=0}(\phi) + L_{\boldsymbol{a},x=1}(\phi)}{L_{\boldsymbol{a},x=b}(\phi)} + \lg L_{\boldsymbol{a},x=b}(\phi)$$
$$= \lg\left(L_{\boldsymbol{a},x=0}(\phi) + L_{\boldsymbol{a},x=1}(\phi)\right) \geq \lg L_{\boldsymbol{a}}(\phi).$$

The theorem follows since for any binary tree of size s, the number of leaves is $\lceil s/2 \rceil$. □

7.2 Adaptation of the Game Characterisation to QU-Resolution

We extend our characterisation to the stronger system of QU-resolution and show that a small modification to the two-player game tightly characterises the size in tree-like QU-resolution.

The only modification of the game for QU-resolution is in the query phase where the Prover may also query any universal variable u not already assigned. The Delayer replies with weights $p_0 \geq 0$ and $p_1 \geq 0$ such that $p_0 + p_1 = 1$. The Prover then assigns a value for u and if she assigns $u = b$ for some $b \in \{0, 1\}$, the Delayer scores $\lg(\frac{1}{p_b})$ points. The "Setting the Universal Variable" stage still remains with the same restrictions as before, since $\forall$-reduction is also present in QU-resolution.

For this modified game we can show:

Theorem 20. *If ϕ has a tree-like QU-resolution proof π of size at most s, then there exists a Prover strategy such that any Delayer scores at most $\lg\lceil \frac{s}{2} \rceil$ points.*

Proof. We use the same argument as in Theorem 18, i.e., the Prover follows the proof Π in reverse order. Now the only addition is that Π may have resolution steps on universal variables. When this occurs the Prover queries that universal variable as she would for existential variables.

The rest of the argument remains the same: a randomised Prover can choose the value of the query variables according to the weights the Delayer gives, and the Delayer gets an expected score

less than or equal to the Shannon entropy. A de-randomised Prover can therefore always force the Delayer to get less than this score. □

To complete the characterisation we show that the converse holds as well, similarly as in Theorem 19.

Theorem 21. *Let ϕ be an unsatisfiable QBF and let s be the size of a shortest tree-like QU-resolution proof for ϕ. Then there exists a Delayer who scores at least $\lg\lceil s/2\rceil$ points against any Prover.*

Proof. We adapt the proof of Theorem 19 and only list the changes here.

On receiving a query for universal variable u, the Delayer does the following, just as he would for existential variables:

1. Updates $\boldsymbol{a}$ to reflect any changes made by the Prover to any of the variables. These changes include assignments made to both universal variables as well as existential variables.
2. Computes the quantities $\ell_0 = L_{\boldsymbol{a},u=0}(\phi)$ and $\ell_1 = L_{\boldsymbol{a},u=1}(\phi)$.
3. Replies with weights $w_0 = \frac{\ell_0}{\ell_0+\ell_1}$ and $w_1 = \frac{\ell_1}{\ell_0+\ell_1}$.

The induction now proceeds the same. In the inductive step, the same inequality $\lg\left(\frac{1}{w_b}\right) + \lg L_{\boldsymbol{a},x=b}(\phi) \geq \lg L_{\boldsymbol{a}}(\phi)$ is obtained and the characterisation therefore holds. □

7.3 A Game Example on Formulas by Janota and Marques-Silva

We consider the following formulas studied by Janota and Marques-Silva [74]:

$$F_n = \exists e_1 \forall u_1 \exists c_1^1 c_1^2 \cdots \exists e_i \forall u_i \exists c_i^1 c_i^2 \cdots \exists e_n \forall u_n \exists c_n^1 c_n^2 :$$
$$\bigwedge_{i=1}^{n} (e_i \Rightarrow c_i^1) \wedge (u_i \Rightarrow c_i^1) \wedge (\neg e_i \Rightarrow c_i^2) \wedge (\neg u_i \Rightarrow c_i^2) \wedge \bigvee_{i=1}^{n} (\neg c_i^1 \vee \neg c_i^2)$$

These formulas were used in [74] to show that ∀Exp+Res does not simulate Q-resolution, i.e., F_n requires exponential-size proofs in ∀Exp+Res, but has polynomial-size Q-resolution proofs. Janota and Marques-Silva [74] also show that ∀Exp+Res p-simulates tree-like Q-resolution, and hence it follows that F_n is also hard for the latter system.

Consider the original hardness proof of F_n for tree-like Q-resolution (or ∀Exp+Res as it was described originally in [74]). It basically describes that there are exponentially many paths from the axioms to the empty clause, each of which corresponds with an assignment to the universal variables. As it is necessary that all assignments are included, the lower bound follows. We reprove this result using our characterisation.

Let $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$ be the set of all universal variables. In the following, we show a Delayer strategy that scores at least n points against any Prover.

Declare Phase: The Delayer executes the declare routine in Algorithm 5 repeatedly till reaching a fixed point (i.e., until calling the algorithm does not produce any changes to the current assignment).

The intuition here is that the Delayer does not want to falsify any small clause as this can lead to the Prover winning early.

Query Phase: For any variable queried by Prover, Delayer responds with weights $\left(\frac{1}{2}, \frac{1}{2}\right)$.

For $i \in [n]$, let $T_i = \{e_i, c_i^1, c_i^2\}$. Let $\mathcal{C} = \bigvee_{i=1}^{n}(\neg c_i^1 \vee \neg c_i^2)$. Except for $\mathcal{C}$, all other clauses have only two literals. Note that our declare routine in Algorithm 5 simplifies which variable can be queried and avoids having to specify a case-by-case response for the Delayer on the queried variable.

Note that we only need to specify the Delayer strategy, who does not deal with universal variables. Hence we do not have to give details on what happens in phase 1 (setting of universal variables) and phase 4 (forget phase).

Lemma 19. *Algorithm 5 never falsifies a clause that has only two literals.*

Proof. Algorithm 5 declares values for either a variable c_i or an e_i. We look at each of these cases below: Setting either c_i^1 or c_i^2: Note that in the formula F, except for the clause $\mathcal{C}$, the variables c_i^1 and c_i^2 appear as positive literals and on the right hand side of implications. Hence setting either c_i^1 or c_i^2 to 1 does not falsify any clause.

Setting an e_i: Algorithm 5 declares a value for e_i only when at least one of c_i^1 or c_i^2 has value 0. Suppose w.l.o.g. that c_i^2 had value 0 before Algorithm 5 was executed. Then Algorithm 5 assigns e_i to 1. However, note that if e_i was unassigned when Algorithm 5 was called, then it must be the case that c_i^1 is not set to 0 (because otherwise e_i would have been set in some previous execution of Algorithm 5). Hence assigning 1 to e_i does not falsify the clause $(e_i \Rightarrow c_i^1)$ because c_i^1 was either true or unassigned before execution of Algorithm 5. □

Lemma 20. *If the Delayer uses the strategy outlined above, then for any winning Prover strategy, the clause falsified is $\mathcal{C}$.*

Proof. Suppose the clause falsified was D. We will show that if $D \neq \mathcal{C}$, then the Delayer did not use our strategy. In other words we show the Delayer succeeds in delaying the contradiction until all literals in $\mathcal{C}$ are refuted. We consider the following cases:

1. D involves variable u_i for some $i \in [n]$:
   Note that u_i appears in clauses with either c_i^1 or c_i^2. Since both c_i^1 and c_i^2 block u_i, it has to be the case that when u_i was set by the Prover, the variables c_i^1 and c_i^2 were unassigned. Now it is straightforward to see that if the Delayer indeed used the declare routine described in Algorithm 5, then all clauses involving u_i become satisfied after u_i is set by the Prover.

Algorithm 5 Declare Routine

for all clauses $(\ell_1 \Rightarrow \ell_2)$ in F_n **do**
  if $\ell_1 = 1$ **then** Declare $\ell_2 = 1$.
  if $\ell_2 = 0$ and $\mathrm{var}(\ell_1) \notin \mathcal{U}$ **then** Declare $\ell_1 = 0$.

2. D is $(e_i \Rightarrow c_i^1)$ or $(\neg e_i \Rightarrow c_i^2)$:
   Suppose w.l.o.g. that $D = (e_i \Rightarrow c_i^1)$. As a consequence of Lemma 19, it must be the case that D was falsified because of the Prover choosing a value for either e_i or c_i^1. So we have two cases:
   - Prover chose a value for e_i to falsify D: So e_i was unassigned just before the query phase began. But if Algorithm 5 left e_i unassigned, then this means c_i is unassigned or $c_i^1 \neq 0$. Hence if the Delayer indeed used Algorithm 5, D could not have been falsified.
   - Prover chose a value for c_i^1 to falsify D: Following an argument just like the previous case, if the Delayer indeed used Algorithm 5, then c_i would be unassigned at the start of the query phase only if $e_i = 0$ or e_i was unassigned. In both these cases D cannot be falsified by choosing a value for c_i^1. □

Theorem 22. *Delayer scores at least n points against any Prover strategy.*

Proof. From Lemma 20, it is sufficient to show that any Prover strategy that falsifies $\mathcal{C}$ will give the Delayer a score of at least n. $\mathcal{C}$ can be falsified only if all variables c_i^1, c_i^2 have been assigned to 1. We observe that for any $i \in [n]$, the Prover can get at most one of c_i^1 or c_i^2 to be declared for free by setting u_i appropriately. To assign the other c_i to 1, the Prover can either query c_i directly and set it to 1 or query e_i and set it appropriately. Both these ways give the Delayer 1 point. Hence for every $i \in [n]$, the Delayer scores at least 1 point. □

With Theorem 18 this reproves the hardness of F_n for tree-like Q-resolution, already implicitly established in [74]:

Corollary 8. *The formulas F_n require tree-like Q-resolution proofs of size $\Omega(2^n)$.*

Note that this bound is essentially tight as it is easy to construct tree-like Q-resolution refutations of size $O(2^n)$.

7.4 Game Example of QBFs Expressing Parity

We now provide a second example, QPARITY, defined in Section 5. There it demonstrated a weakness of Q-resolution that could be exploited when the Herbrand function of a lone universal variable is not in AC^0. Here that function is $\textsc{Parity}(x_1, \ldots, x_n) = x_1 \oplus \cdots \oplus x_n$.

Before we use a novel lower bound technique based on strategy extraction, which transfers the AC^0 lower bound for PARITY from [67] to Q-resolution. Here we use our game characterisation to prove the lower bound again. This proof is not dependent on any circuit lower bound.

For $n > 1$ define QPARITY_n as follows. Let $\text{xor}(o_1, o_2, o)$ be the set of clauses $\{\neg o_1 \vee \neg o_2 \vee \neg o,\ o_1 \vee o_2 \vee \neg o,\ \neg o_1 \vee o_2 \vee o,\ o_1 \vee \neg o_2 \vee o\}$, which defines o to be $o_1 \oplus o_2$. Define QPARITY_n as

$$\exists x_1, \ldots, x_n \forall z \exists t_2, \ldots, t_n.\ \text{xor}(x_1, x_2, t_2) \cup \textstyle\bigcup_{i=3}^{n} \text{xor}(t_{i-1}, x_i, t_i) \cup \{z \vee t_n, \neg z \vee \neg t_n\}.$$

Intuitively, these formulas express via the universally quantified z that there exists an input $x_1, \ldots, x_n$ for which $x_1 \oplus \cdots \oplus x_n$ is both 0 and 1. Hence, for the universal player the only

Algorithm 6 Declare Routine

if x_1 and x_2 are assigned and t_2 is unassigned **then**
    $t_2 \leftarrow x_1 \oplus x_2$
for $i = 2$ to $i = n - 1$ **do**
    if t_i and x_{i+1} are assigned and t_{i+1} is unassigned **then**
        $t_{i+1} \leftarrow t_i \oplus x_{i+1}$
if z is assigned and t_n is unassigned **then**
    $t_n \leftarrow \neg z$
for $i = n$ to 3 **do**
    if t_i and x_i are assigned and t_{i-1} is unassigned **then**
        $t_{i-1} \leftarrow x_i \oplus t_i$
if x_2 and t_2 are assigned and x_1 is unassigned **then**
    $x_1 \leftarrow t_2 \oplus x_2$
if x_1 and t_2 are assigned and x_2 is unassigned **then**
    $x_2 \leftarrow x_1 \oplus t_2$
for $i = 2$ to $i = n - 1$ **do**
    if t_i and t_{i+1} are assigned and x_{i+1} is unassigned **then**
        $x_{i+1} \leftarrow t_i \oplus t_{i+1}$

way to falsify the formula is to play z as the opposite value of $x_1 \oplus \cdots \oplus x_n$, which means he has to compute the PARITY function. This is crucially exploited in Chapter 5 for the lower bound.

When playing our Prover-Delayer game on these formulas, the Prover queries x_i and t_i variables, or can set the value of z. In setting the value of z she deletes all progress made on the t_i variables, but retains all the information on the x_i variables.

Observe the Delayer has the luxury that if z is set at the beginning of the game he can answer in a way that will never contradict the CNF at least until the value of z is changed. When $z = 0$ the Delayer is trying to build an assignment on the x variables that gives $\text{PARITY}(x_1, \ldots, x_n) = 1$ and tries to make the t variables consistent with that. When $z = 1$ the Delayer is trying to build an assignment on the x variables that gives $\text{PARITY}(x_1, \ldots, x_n) = 0$ and is still trying to make the t variables consistent with that. In fact, as long as the Delayer is playing in this way the Delayer cannot lose.

We formulate this as a strategy below. Like in Section 7.3 we utilise a declare routine (Algorithm 6) for the Delayer to simplify the analysis, with a similar objective: to satisfy a clause that is one existential literal away from unsatisfiability. For this we need to look at all the parity equations $t_i \oplus x_i = t_{i+1}$. Setting one variable might trigger further assignments. A detailed analysis is carried out below.

Observation 23 *Performing Algorithm 6 twice gives the same result as performing Algorithm 6 once.*

Proof. Algorithm 6 breaks down into three parts:

1. the first part is from line 1 to line 5. Each declaration here declares a t_i and the index i increases with each loop;
2. the second part is from line 6 to 10. Each declaration here declares a t_i and the index i decreases with each loop;
3. the final part is from 11 to 17 and declares x_i where the t values are given already.

Observe that because part 1 increases the index in the loops and that t_i is a precondition for t_{i+1}, the t_i being defined propagates in increasing i. Suppose that t_j was not defined when it was checked as a precondition for defining t_{j+1} here. Then it will not be defined for the remainder of this part of the algorithm since the check is passed. The equivalent happens for conditions not met in part 2.

Let us suppose t_j is changed due to part 2, then it must be that t_{j+1} is defined and so no declare happens from applying part 1 again. Therefore a declaration in part 2 cannot affect a condition in part 1. This is likewise true vice versa.

It is impossible for a declared x_j to trigger any condition in any other part of the algorithm as any variable it triggers must be defined as a precondition. □

After the declare routine, the strategy of the Delayer now becomes very simple. When queried on any unassigned existential variable the Delayer sets $p_0 = p_1 = \frac{1}{2}$.

We now turn to the analysis.

Lemma 21. *At most two x_i variables are declared or assigned per turn.*

Proof. Suppose on any given turn the variable x_i is queried and therefore assigned a new value. Then it cannot be the precondition of two different declarations since x_i can only be a precondition for t_i or t_{i-1} in which case the other is required to be defined already as a precondition. As a result of the declaration more t_j variables can be declared, but only with consecutively increasing or decreasing index (not both). It may or may not end with another x_k being declared, but then as t_k and t_{k-1} must be assigned this cannot propagate.

Suppose on any given turn the variable t_i is queried. This can be the precondition to two different declarations, if these are t_j variables these can propagate upwards or downwards. Eventually this results in some x_k being declared, but then as t_k and t_{k-1} must be assigned this cannot propagate. Therefore only two x_k can be declared.

Now suppose the universal player changes the value of z. If there are x_j and x_k that are unassigned with $j < k$ then the declare phase from lines 1 to 5 may set a number of variables t_i increasing in i, satisfying all $\mathrm{xor}(t_{i-1}, x_i, t_i)$. We also notice that this propagation must stop before t_j is reached, as x_j is unassigned. Likewise from Algorithm 6, lines 6 to 10 a downward propagation of t_i variables may occur, satisfying the clauses from $z \vee t_n$, $\neg z \vee \neg t_n$ and $\mathrm{xor}(t_i, x_{i+1}, t_{i+1})$. However this stops before t_{k-1}. No x_i values can now be set as it requires t_{i-1} and t_i to be assigned and x_i to be unassigned. This is impossible as the only unassigned x_i values are for $j \leq i \leq k$ but there are no t variables assigned in between them. □

Lemma 22. *The game cannot end in the query phase.*

Proof. Suppose that the game ends in the query phase. Then there is some clause C in the matrix of QPARITY_n that is falsified by the assignment of literal l to 0 in the query phase. This means that before the query phase all literals except l in C were refuted, but $\text{var}(l)$ was unassigned. Either $C \in \text{xor}(x_1, x_2, t_2)$, $C = \neg z \vee \neg t_n$, $C = z \vee t_n$ or $C \in \text{xor}(t_{i-1}, x_i, t_i)$ for some i, $3 \leq i \leq n$. We use Algorithm 6 to show that if all other literals in C are defined then l cannot be unassigned.

Suppose $C \in \text{xor}(x_1, x_2, t_2)$. We use Lines 12, 14 and 2 to show that l cannot be x_1, x_2 or t_2, respectively. Then suppose $C = \neg z \vee \neg t_n$ then l cannot be $\neg z$ as only existential variables can be queried but cannot be $\neg t_n$ due to Line 7. Now suppose $C = z \vee t_n$. Then l cannot be z as only existential variables can be queried, but it cannot be t_n due to Line 7. Finally suppose that for some i, $3 \leq i \leq n$, $C \in \text{xor}(t_{i-1}, x_i, t_i)$. Then we use Lines 10, 17 and 5 to show that l cannot be t_{i-1}, x_i or t_i, respectively. □

It is also clear that the Prover cannot win simply by setting the universal variables. To do so she must win on the clauses $z \vee t_n$ or $\neg z \vee \neg t_n$, but t_n must be unassigned after the universal variables are set. Therefore the Prover must win in the declare phase.

Lemma 23. *The Prover cannot win until all x_i are assigned.*

Proof. The Prover must win on the declare phase by Lemma 22. Now suppose the Prover wins on a declare phase when some x_j is unassigned. This must be triggered by either setting the universal variables or by querying a variable.

Let us suppose that this change was triggered by setting the universal variable z. Then all t_i variables are unassigned at the start of the declare phase. From Lines 1 to 5 a number of t_i variables may be set, satisfying each corresponding $\text{xor}(t_{i-1}, x_i, t_i)$. We also notice that this propagation must stop before t_j as x_j is unassigned. Likewise from Lines 6 to 10 a downward propagation of t_i variables may occur, satisfying the clauses from $z \vee t_n$, $\neg z \vee \neg t_n$ and $\text{xor}(t_i, x_{i+1}, t_{i+1})$. However this stops before t_{j-1}. So far no clause has been falsified. In the final section of the algorithm in Lines 11 to 17 the only x_i variable that can be declared is x_j, which contradicts our assumption.

Now let us suppose the change was triggered by a queried or declared variable. If x_1 is queried then it cannot immediately contradict a clause but it can trigger a declaration of x_2 or t_2. Likewise if x_2 is queried then it cannot immediately contradict a clause but it can trigger a declaration of x_1 or t_2. If for $i > 1$, x_i is assigned then it cannot immediately contradict a clause but it can trigger a declaration of t_{i-1} or t_i (but not both). If for $i > 1$, t_i is set then it cannot immediately contradict a clause but it can trigger a declaration of t_{i-1} or x_i and x_{i+1} or t_i.

Now suppose that t_{i+1} is declared in Line 5. It cannot cause a contradiction of $\text{xor}(t_i, x_{i+1}, t_{i+1})$ by definition. For $\text{xor}(t_{i+1}, x_{i+2}, t_{i+2})$ observe that if x_{i+2}, t_{i+2} were assigned then they were assigned before the algorithm was executed by the previous declare phase. It now may trigger a declaration of x_{i+1} or t_i.

Now suppose that t_{i-1} is declared by Line 10, it cannot cause a contradiction of $\text{xor}(t_{i-1}, x_i, t_i)$ by definition. For $\text{xor}(t_{i-2}, x_{i-1}, t_{i-1})$ observe that if x_{i-1}, t_{i-2} were assigned then they were either both assigned before the algorithm was executed in the previous declare phase, or t_{i-2} is

assigned by an upwards propagation earlier in the algorithm which means that the queried variable this turn must have index $j < i - 1$, but since this t_{i-1} is declared as a result of downwards propagation this $j \geq i - 1$. This means we cannot get a contradiction here. It now may trigger a declaration of x_{i+1} or t_i.

t_n will always be declared immediately after setting the universal variable.

Now suppose that x_{i+1} is declared by Line 12, 14 or 17, it cannot cause a contradiction in $\text{xor}(t_{i-1}, x_i, t_i)$ (nor $\text{xor}(x_1, x_2, t_2)$) by definition. It cannot trigger any more declarations. □

We can now easily count the score of the Delayer for our strategy and combine the above analysis into the following result:

Theorem 24. *There exists a Delayer strategy that scores at least $\frac{n}{2}$ points against any Prover in the Prover-Delayer game on* QPARITY_n.

Proof. Each turn a variable is queried and by Lemma 21 at most two x_i variables get assigned. The Delayer always gets one point so gets at least $\frac{n}{2}$ points before the game is finished as all the x_i variables must be set (Lemma 23). □

Corollary 9. *Every tree-like Q-resolution refutation of* QPARITY_n *is of size at least* $2^{n/2}$.

We remark that in this example we again use the 'symmetric' Delayer score scheme $(\frac{1}{2}, \frac{1}{2})$, which also corresponds to the information-theoretic intuition behind the game as the Prover learns exactly the same from a parity variable being assigned 0 or 1. This also means that the lower bound argument might be made with just the logical reasoning on the graph level (e.g., show that for any assignment of x variables there is a set-up of t variables which creates a (unique) path from the empty clause to one of $(z \vee t_n)$ or $(\neg z \vee \neg t_n)$). The Prover-Delayer game approach is a smart way of showing that there is an exponential number of such paths.

7.5 Game Example of the Formulas of Kleine Büning et al.

In our third example we look at a family of formulas first defined by Kleine Büning, Karpinski and Flögel [77]. The formulas are known to be hard for Q-resolution and indeed for the stronger system IR-calc (Chapter 6). However, it is known that there exists short dag-like proofs in QU-resolution [116]. In contrast, we use our characterisation to show that these formulas remain hard in tree-like QU-resolution.

We defined the KBKF formulas in Section 6, however, we will use the original notation with an upper and lower index. In the previous section we had to deal with annotations which we do not have here.

$$
\begin{array}{llll}
C_- = \{\neg y_0\} & & & \\
C_0 = \{y_0, \neg y_1^0, \neg y_1^1\} & & & \\
C_i^0 = \{y_i^0, x_i, \neg y_{i+1}^0, \neg y_{i+1}^1\} & C_i^1 = \{y_i^1, \neg x_i, \neg y_{i+1}^0, \neg y_{i+1}^1\} & & \text{for } i \in [t-1] \\
C_t^0 = \{y_t^0, x_t, \neg y_{t+1}, \ldots, \neg y_{t+t}\} & C_t^1 = \{y_t^1, \neg x_t, \neg y_{t+1}, \ldots, \neg y_{t+t}\} & & \\
C_{t+i}^0 = \{x_i, y_{t+i}\} & C_{t+i}^1 = \{\neg x_i, y_{t+i}\} & & \text{for } i \in [t]
\end{array}
$$

The $\mathrm{KBKF}(t)$ formulas are then defined as the union of these clauses under the quantifier prefix $\exists y_0, y_1^0, y_1^1 \, \forall x_1 \, \exists y_2^0, y_2^1 \, \forall x_2, \ldots, \forall x_{t-1} \, \exists y_t^0, y_t^1 \, \forall x_t \, \exists y_{t+1} \ldots y_{t+t}$.

We now want to show an exponential lower bound on proof size for the $\mathrm{KBKF}(t)$ formulas via our game. We will assume throughout that $t > 2$. Intuitively, the strategies here are similar to the strategies described in the game semantics: the Prover is forced to set x_i in increasing i while the Delayer gets a choice of the weights of the values of y_j^0, y_j^1 and declares variables to avoid contradictions. Unlike the semantic game, the variables are not queried in any fixed order. Instead of setting y_j^0, y_j^1 for slowly increasing j and the contradiction being propagated forwards towards the variables in the innermost block like the description above, a large j may be queried and a contradiction may be propagated 'backwards' towards the variables in the outermost blocks. When either $y_j^0 = 0, y_j^1 = 0$ the contradiction is propagated forwards towards the y^{t+i} variables, and when $y_j^0 = 1, y_j^1 = 1$ the contradiction is propagated backwards to latest $y_i^c = 0$ variable. Recall that setting both $y_j^0 = y_j^1 = 1$ only sets one of y_{j-1}^0, y_{j-1}^1 to 1 depending on how x_{i-1} is set so it is useful to the Delayer to make sure that setting y_j^0 or y_j^1 to 1 is worth less points than setting y_{j-1}^0 or y_{j-1}^1 to 1, and likewise the Delayer wants the Prover to make less progress setting high j y_j variables to 0. Taking all of these into consideration a careful Delayer can set the right weights to gain enough points whichever way the Prover makes progress, we give an informal description of such a Delayer strategy.

Delayer strategy – informal description

We think of the existential variables of $\mathrm{KBKF}(t)$ to be arranged as shown in Figure 24.

y_0	y_1^1 y_1^0	y_2^1 y_2^0	$\cdots$	y_t^1 y_t^0	y_{t+1}	y_{t+2}	$\cdots$	y_{2t}

Fig. 24. Variables of $\mathrm{KBKF}(t)$

At any point of time during a run of the game, there is a partial assignment to the variables of the formula that has been constructed by the Prover and Delayer. We define the following:

Definition 12. *For any partial assignment $\boldsymbol{a}$ to the variables, we define $z_{\boldsymbol{a}}$ to be the index of the rightmost column (see Figure 24) where $\boldsymbol{a}$ assigns a 0 to one or more variables in the column. If no such column exists, then $z_{\boldsymbol{a}} = 0$.*

For convenience, we will drop the subscript and just say z when the partial assignment is clear from context. We usually mention the time during a run of the game at which we are referring to z instead of explicitly mentioning the induced partial assignment. z is important for the Delayer strategy and lower bound because it is the main measure of progress of the game. The idea behind the Delayer strategy is the following: We observe that for all $i < t - 2$ and $j \in \{0, 1\}$, to falsify the clause C_i^j,

it is necessary that y_i^j is set to 0, x_i is set to j and both y_{i+1}^0 and y_{i+1}^1 are set to 1. The strategy we design will not let the Prover win on clauses C_i^0 or C_i^1 for any $i < (t-2)$. We do this by declaring either y_{i+1}^0 or y_{i+1}^1 to 0 at a well chosen time. Furthermore, we will show the following statements: (1) When the game ends, $z \geq t$ and (2) After any round in the game, the Delayer has a score of at least αz where $\alpha > 0$ is a global constant. It is easy to see that the lower bound of $\Omega(t)$ for the score of the Delayer follows from statements (1) and (2).

We now give the idea behind the declare routine and the weights. We will give details later.

Declare routine: The importance of the declare routine is to simplify the Delayer strategy for the reader. Since KBKF is the most complicated example we present here, this is where the declare routine benefits us the most. What is gained here is that we can simplify much of the information needed from the current assignment for the Delayer query strategy to just information on z.

We will use the declare routine shown in Algorithm 7. This declare routine is designed specifically to make sure that the game does not end at a clause C_i^b for any $i < (t-2)$ and that statement (1) (at the end of the game $z = t$) holds. Note that line 13 of Algorithm 7 is very similar to the idea behind the declare routine in Section 7.3, i.e., if in any round there is a clause C that has only one existential variable y unassigned and $C|_{y=b}$ is unsatisfiable, then we declare $y = \neg b$ in the immediate declare phase.

We will give away values of variables y_j^0 and y_j^1 for all $j < z$ for free in the declare phase in a way that it neither ends the game, nor make any progress in the game. We first make sure that the Prover cannot exploit an unassigned universal literal by using Lines 8,10 and 17 of Algorithm 7 so that if y_j^0 and y_j^1 are both set to 0 then at least one of y_{j+1}^0 or y_{j+1}^1 is set to zero . This allows the Delayer to answer any query of x_j to satisfy whichever of C_j^0, C_j^1 is not satisfied (but score no points). Giving away the values of the these variables does not prevent the Delayer scoring enough because the points are scored using the variables y_j^0 and y_j^1 for all $j > z$.

There are still some complications for the Delayer strategy; the Prover can set all universal variables to 1 then query y_t^0, y_{t-1}^0, etc. until y_1^0, choosing 1 each time. Subsequently, the Delayer will be forced to set y_1^1 to 0, then y_2^1 to 0 etc. until $y_t^1 = 0$. Then the Prover need only query the variables in C_t^1 to get a contradiction. To counter such strategies, the Delayer declares y_1^0 to 0 instead of allowing it to be queried for the usual score. This is achieved in line 14 of Algorithm 7. It allows the value of z to increase, but in this case only by 1 and when some constant score has already been achieved.

Scoring: At the start of the game, we have $z = 0$, and at the end, we will have $z \geq t$. We will make sure that z increases monotonically. So the higher the value of z, the closer the Prover is to winning the game. Intuitively, the value of z is a mark of progress in the game for the Prover. Hence our scoring is designed so that the Prover is charged for increasing the value of z.

At some intermediate round in the game, if the Prover queries variable y_i^0 or y_i^1 for some $i > z$, our strategy charges a score proportional to $(i - z)$ for letting the Prover set the variable queried to 0. However, in some cases, we will have to adjust this so that the Delayer scores more if the declare phase immediately forces z to an even higher value. If the effect is not immediate the Delayer can

force the Prover to change the universal variables by declaring a 0 at y^1_{i+1} or y^0_{i+1} depending on the universal variables (see line 15 of Algorithm 7).

Delayer strategy – details

We now give full details of the Delayer strategy.

Declare Phase: The Delayer sets y_0 to 0 in the declare phase of the first round.

Let F be the set of all existential variables that were chosen to be forgotten by the Prover in the forget phase of the previous round. The Delayer first does the following "Reset Step": For all variables y in F that had value 0 just before the forget phase of the previous round, the Delayer declares $y = 0$. This Reset Step keeps the state of the game simple.

After the reset step, the Delayer executes Algorithm 7 repeatedly until reaching a fixed point. The notation $y \leftarrow b$ means that the Delayer declares $y = b$ if and only if y is an unassigned variable. Also, we assume that z is updated automatically to be the index of the rightmost column that contains a 0 (see Figure 24).

We observe the following about the reset step:

Observation 25 *The reset step ensures that z always increases monotonically (when z is measured at the beginning of each query phase).*

Line 17 of Algorithm 7 gives us the following observation:

Observation 26 *After the declare phase, for all $i < z$, the existential variables y^0_i and y^1_i have been assigned a value.*

Observation 27 *For all $i > z$ for $c \in \{0, 1\}$, if a line in Algorithm 7 potentially sets y^c_i to 1 and another line potentially sets y^c_i to 0, the line that sets y^c_i to 1 comes first.*

Observation 28 *For all $i < t$, if y^0_i and y^1_i are both set to 0 then at least one of y^0_{i+1} or y^1_{i+1} is set to zero.*

This holds because the Delayer does not allow y^0_i and y^1_i to be both set to 0 during the query phase. This is also avoided in Algorithm 7 by Lines 8 and 10. However since z may change it is possible, in conjunction with Line 14 , that after Line 17 both y^0_{z-1} and y^1_{z-1} equal 0 but by definition some $y^c_z = 0$.

Query Phase: Let the queried variable be y^b_i. From Observation 26, it is easy to see that $i \geq z$. We have the following cases:

- If $i > t$, then the Delayer replies with weights $w_0 = 2^{z-t-1}$ and $w_1 = 1 - w_0$.
- Else $z \leq i \leq t$. We have three cases:
  - If $z = i$ the Delayer replies with weights $w_0 = 0$ and $w_1 = 1$.
  - If x_i is unassigned, then the Delayer replies with weights $w_0 = 2^{z-i}$ and $w_1 = 1 - w_0$.
  - Else x_i holds a value. Then we have the following cases:

Algorithm 7 Declare Routine

$y_z^0 \leftarrow 1, y_z^1 \leftarrow 1$
$z' := z$
if $y_z^{x_z} \neq 0$ or x_z unassigned **then**
    for all $i > z$ **do** $y_i^0 \leftarrow 1; y_i^1 \leftarrow 1$
for $i = 1$ to $z - 1$ **do**
    if x_i is unassigned **then**
        if $y_i^0 = 0$ **then**
            $y_i^1 \leftarrow 1$
        else if $y_i^1 \neq 1$ **then**
            $y_i^0 \leftarrow 1$
for $i = t - 1$ to 1 **do**
    for $j = 0$ to 1 **do**
        if C_i^j is not satisfied with only one literal l that is unassigned **then** satisfy C_i^j with that literal (if existential).
if $z \leq t - 2$, x_{z+1} is assigned and either $y_{z+2}^0 = 1$ or $y_{z+2}^1 = 1$ **then** $y_{z+1}^{1-x_{z+1}} \leftarrow 0$
if $z \neq z'$, x_z assigned and $y_z^{x_z} = 0$ **then**
    if x_{z+1} unassigned **then** $y_{z+1}^0 \leftarrow 0$ **else** $y_{z+1}^{1-x_z} \leftarrow 0$
for all $i < z$ **do** $y_i^0 \leftarrow 0, y_i^1 \leftarrow 0$

- If $b = \neg x_i$, then the Delayer replies with weights $w_0 = 2^{z-i}$ and $w_1 = 1 - w_0$.
- Else $b = x_i$ and Delayer replies with weight $w_0 = 2^{z-j}$, where j is the largest index such that $\forall k : z < k \leq j, x_k$ is assigned and $y_k^{1-x_k} = 1$. Weight $w_1 = 1 - w_0$.

Now suppose the queried variable is x_i

– If $y_i^0 = 1$ or $y_i^1 = 0$ then the Delayer replies with weights $w_0 = 1$ and $w_1 = 0$.
– Else, If $y_i^1 = 1$ or $y_i^0 = 0$ then the Delayer replies with weights $w_0 = 0$ and $w_1 = 1$.
– Otherwise $w_0 = w_1 = 1/2$.

We now analyse the above Delayer strategy. We want to argue that as z increases so does the Delayer score and that z increases sufficiently in total. We start with the following lemma:

Lemma 24. *If the Delayer uses the strategy outlined above, then against any Prover, at the end of the game on* $\mathrm{KBKF}(t)$*, $z \geq t$ (where z is defined as in Definition 12).*

Proof. The Prover cannot win on the clause $\neg y_0$ because the Delayer always sets y_0 to 0. Suppose the Prover wins on the clause C_i^b for some $i \in [t]$ and $b \in \{0, 1\}$. We first show the following claim:

Claim. At the end of the game we have $z = i$.

To prove the claim we note that $i > z$ is impossible because there is a positive literal in every C_i^b clause, so in order to falsify C_i^b some existential literal must be set to 0 and thus $z \geq i$.

To show the claim we now argue for $z \leq i$. The clause C_i^b has positive literal y_i^b. Since C_i^b was falsified, the variable y_i^b must have been set to 0 permanently after some move in the game. We

know that if at least one of y_j^0, y_j^1 (for $j \leq z$) is 0 then both the clauses containing the negative literals are already satisfied. If both are already assigned to 1 it means that if x_{j-1} is assigned, then $y_{j-1}^{x_{j-1}}$ gets set to 1 by the declare phase, and this happens before any instruction to set it to 0 appears by Observation 27. If x_{j-1} is not assigned, note that the universal player cannot declare it without using the forget phase, we will come to this possibility later. If the universal player instead queries x_{j-1} then by Observation 28 the Delayer can always respond without falsifying a clause.

If the universal player uses the forget phase to remove y_j^0 and y_j^1 and change a universal variable x_{j-1} then y_j^0 and y_j^1 will be assigned by the end of the declare phase (because the Delayer redeclares the variables, so z is retained after the forget phase), however at most one of them will be set to 1.

The declare phase sets all other literals before y_z^0 to 0, without a contradiction. Hence also $z \leq i$, which proves the claim.

To continue the proof of the lemma, we note that Algorithm 7 (Line 13) and the Delayer's response on universal variables prevents the clauses from being falsified in the immediate query phase that follows. Hence we only consider the case where a clause is falsified in the declare phase.

We will show that setting $y_i^{x_i}$ to 0 for $i < t$ is not the winning move in a declare phase. Otherwise, immediately before we have a different z and $i = z + 1$, where $y_{z+1}^{x_{z+1}}$ gets set to 0 in the declare phase. This requires that both $y_{z+2}^0 = y_{z+2}^1 = 1$, but then by Observation 27, $y_i^{x_i}$ will be set to 1 immediately before, contradicting it getting set to 0. This means that the winning move can only be done by declaring $y_{i+1}^{x_{i+1}}$ to 1 or $y_{i+1}^{1-x_{i+1}}$ to 1. Only $y_{i+1}^{x_{i+1}}$ can be set in the declare phase (because the universal variable is essential), but this requires $y_{i+1}^{1-x_{i+1}} = 1$ and $y_i^{x_i} = 0$. We can assume that none of these were set in the previous query phase, as what is set in the previous query phase must cause the $y_{i+1}^{x_{i+1}}$ to be set to 1 and, looking at the clauses, must be a higher level. Therefore in the declare phase before that we must have $y_{i+1}^{1-x_{i+1}} = 1$, $y_i^{x_i} = 0$ and $y_i^{x_{i+1}}$ unassigned. So by Line 13, $y_{i+1}^{x_{i+1}}$ is set to 0 in that declare phase (and the Delayer will replace the 0 if the Prover chose to forget it), therefore it cannot be declared to 1 in the next turn. □

Remark 4. If the Prover chooses to assign 1 to an existential variable or assigns a queried universal variable in the query phase on turn k, then by the beginning of the query phase on turn $k + 1$, the value of z (index of the rightmost zero) increments by at most 1. For the increase by 1 it is required that $y_z^{x_z} = 0$ and that for all $c \in \{0, 1\}$, y_{z+1}^c and y_{z+2}^c are unassigned before the query phase on turn k. If the Prover chose to assign 1 to the variable queried and it results in a change of z, then it must cause any of y_{z+1}^0, y_{z+1}^1, y_{z+2}^0 or y_{z+2}^1 to be set to 1, incrementing z by at most one.

For all $i \in [t]$, and $z < t - 1$, let $s_z(y_i^c)$ denote the minimum (over all possible Prover strategies) Delayer score when y_i^c is assigned 1 by the Prover for the first time starting from a partial assignment where the rightmost zero is in column z and every variable to the right of column z is unassigned.

Of note is that $s_z(y_i^c)$ for $i > z + 1$ does not depend on the values of y_j^0, y_j^1 for $j \leq i$ (apart from giving a value to z) when the game is being played as described. This can be seen because the Delayer does not base the scores on these values and these values cannot cause higher index values to be declared to 1.

Combining Observation 25 with the fact that at the start of the game $z = 0$, Lemma 24 implies that the Prover increases z by at least t in the process of winning the game. We will now measure the scores that the Delayer accumulates.

Lemma 25. *For all $z < t - 1$ and $i < t$, each of $s_z(y_i^0)$ and $s_z(y_i^1)$ is at least $2^{t-i} \lg \frac{2^{t-z}}{2^{t-z}-1}$.*

Proof. Suppose in the first round, the Prover sets $x_i = 1$. Since all existential variables of greater level are unassigned she could then somehow set $y_{i+1}^0 = 1$ at cost $s_z(y_{i+1}^1)$. Subsequently, she could still change all universal variables at level greater than $\mathrm{lv}(y_{i+1}^0)$ and delete all existential variables afterwards, and thus can get $y_{i+1}^1 = 1$ at cost $s_z(y_{i+1}^1)$ without deleting y_{z+1}^0. At this point $y_z^1 = 1$ by the declare phase. This means $s_z(y_i^1) \leq 2s_z(y_{i+1}^1)$.

Suppose $s_z(y_i^1) \neq 2s_z(y_{i+1}^1)$. Then it is cheapest for the Prover to query y_i^1 immediately. This gives the Delayer $\lg(\frac{2^{i-z}}{2^{i-z}-1}) = 2i+2-2z-\lg(2^{2i+2-2z}-2^{i+2-z})$ points. Instead the Prover could query both y_{i+1}^0 and y_{i+1}^1 and this gives $2\lg\frac{2^{i+1-z}}{2^{i+1-z}-1} = 2i + 2 - 2z - \lg(2^{2i+2-2z} - 2^{i+2-z} + 1)$, which is slightly cheaper. Hence $s_z(y_i^1) = 2s_z(y_{i+1}^1)$.

Recursively $s_z(y_i^1) = 2^{t-i}s_z(y_t^1)$. Variable y_t^1 can be set to 1 by querying it or by querying all variables in the next existential level. However, asymptotically it will be cheaper to query it directly. Hence $s_z(y_t^1) = \lg\frac{2^{t-z}}{2^{t-z}-1}$. By symmetry, $s_z(y_i^0) = s_z(y_i^1)$ as at the beginning the Prover is free to switch the polarities of all the universal variables with no cost. Note that the Delayer strategy on universal variables prevents the universal player from switching the polarities of x_i during the query phase, so we can assume the Prover has to stick with her choices or use the forget phase. There is no advantage to leaving the universal variables unassigned at the beginning and then querying them later as the score only increases when the Prover chooses 0 for some existential variable on level i and in that case the Delayer is defiant and does not allow the Prover to set x_i to a value useful for the Prover on that query phase. □

We now know that during a run of the game, z increases from 0 to t. Now we show that the Delayer scores $\Omega(z)$ points during any particular run of the game on $\mathrm{KBKF}(t)$ for large enough t:

Lemma 26. *There exist constants $t_0 > 0$ and $\alpha > 0$ such that for all $t > t_0$, at any point of time during a run of the game on $\mathrm{KBKF}(t)$, the Delayer has a score of at least αz.*

Proof. We will take the lemma as an inductive hypothesis on z. On the first turn the Delayer sets $z = 0$ and the Delayer has zero points.

The value of z can change from the Prover picking a 0 in the query phase. In this case the Delayer either scores $j - z$ points when the 0 moves down to $j + 1 - z$ in the declare phase or scores $i - z$ points otherwise. When z does not change in the declare phase, it is the only case where the Prover is not forced to delete all the higher level existential literals and switch the universal variable x_i and so may get the z to be incremented by 1 at a cheaper cost than $s(y_{z+2}^0)$ (which will be our lower bound when 1 is assigned by the Prover to an existential variable to force a change in z). However this is not a problem as we only get this once per time z is changed, hence the Delayer gets at least $\frac{n}{2}$ points if z changes by n.

As remarked earlier, the value of z can change by at most 1 if Prover chooses to assign 1 to a queried variable. This can result from 1 being assigned after a query on y^c_{z+1} or y^{1-c}_{z+1}. In this case, as y^0_{z+2} and y^1_{z+2} are unassigned, the cost of these are 1, so the Prover gets enough points. Now we only need to look at the case where a y^0_{z+2} or y^1_{z+2} gets set to 1 and we start with unassigned existential literals for higher levels than z. Here we know from Lemma 25 that the minimum cost is $\frac{1}{4}2^{t-z}\lg(\frac{2^{t-z}}{2^{t-z}-1})$. Note that t is the only variable in this expression since at any fixed point of time during a run of the game, the value of z is fixed. This quantity can be written as $f(x) = \frac{1}{4}x\lg(\frac{x}{x-1})$ where $x = 2^{t-z}$. It is easy to see that the limit of $f(x)$ as x tends to infinity is the constant $\frac{1}{4\ln 2}$. This implies that $f(x) \in \Omega(1)$. So the Delayer gets $\Omega(1)$ points each time the Prover increments z by 1. More precisely, using the definition of big-Omega, there exists constants $t_0 > 0$ and $\alpha > 0$ such that for all games played on $\mathrm{KBKF}(t)$ for a $t > t_0$, the Delayer scores at least α points each time the Prover increases z by 1. □

Combining Lemma 24 and Lemma 26, we have:

Theorem 29. *There exists a Delayer strategy that scores $\Omega(t)$ against any Prover in the Prover-Delayer game on $\mathrm{KBKF}(t)$.*

Combining Theorem 29 and Theorem 20, we obtain:

Corollary 10. *The formulas $\mathrm{KBKF}(t)$ require tree-like QU-resolution proofs of size $2^{\Omega(t)}$.*

As $\mathrm{KBKF}(t)$ are easy for QU-resolution [116], they therefore provide an exponential separation between tree-like and dag-like QU-resolution by Corollary 10.

We will continue this investigation QBF lower bound techniques in Chapter 8 and 9.

Chapter 8

Feasible Interpolation for QBF Resolution Calculi

In Chapter 5 and 7 we find two lower bounds techniques that work for QBF. In Chapter 7 we introduce a game technique that works for tree-like Q-resolution. The strategy extraction technique from Chapter 5 can be applied to DAG-like QBF resolution systems, but does not apply directly in the expansion based systems such as ∀Exp+Res, IR-calc and IRM-calc. In this chapter, we present a general lower bound technique that works for all QBF resolution systems including expansion based systems.

The technique we present is *feasible interpolation*, which is a present technique in propositional proof systems. Feasible interpolation works on true implication formulas $A(\boldsymbol{p}, \boldsymbol{q}) \rightarrow B(\boldsymbol{p}, \boldsymbol{r})$ (or, equivalently, false conjunctions $A(\boldsymbol{p}, \boldsymbol{q}) \wedge \neg B(\boldsymbol{p}, \boldsymbol{r})$). By Craig's interpolation theorem [47] there is a interpolating formula C in the common variables $\boldsymbol{p}$, such that $A(\boldsymbol{p}, \boldsymbol{q}) \rightarrow C(\boldsymbol{p})$ and $C(\boldsymbol{p}) \rightarrow B(\boldsymbol{p}, \boldsymbol{r})$. While interpolation formulas always exist these may not be of polynomial size [94].

A proof system P is said to admit feasible interpolation, if for any P proof of $A(\boldsymbol{p}, \boldsymbol{q}) \rightarrow B(\boldsymbol{p}, \boldsymbol{r})$ an interpolating circuit for some $C(\boldsymbol{p})$ can be extracted from the proof in polynomial time. This gives us a lower bound technique, if we know that a particular class of formulas does not admit small interpolants (either unconditionally or under suitable assumptions), then there cannot exist small proofs of the formulas in the system P.

In propositional logic, resolution and the cutting planes proof systems are shown to have the feasible interpolation property [99]. In [21] it was shown that all QBF resolution systems have feasible interpolation. Due to the simulations (cf. Figure 18, Section 4.4) it is only necessary to show feasible interpolation for two systems LQU$^+$-Res and IRM-calc. In this thesis we only present the feasible interpolation technique for IRM-calc.

When feasible interpolation is lifted to the QBF setting it provides a greater opportunity for lower bounds. Our second contribution is to use our feasible interpolation for IRM-calc for a new unconditional lower bound. These new formulas $\Phi_n(\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{r})$ are false QBFs that express that a graph cannot both have and not have a k-clique. The clique problem is used as the interpolant precisely because it only has exponential-size monotone circuits [3]. This means that if feasible interpolation can extract monotone circuits in polynomial time, we get an exponential lower bound. We show that IRM-calc has *monotone feasible interpolation* and thus the lower bound applies. This result also applies in the other QBF resolution systems and thus the lower bound is general [21].

Strategy extraction is an important part of QBF proof systems. We see results in strategy extraction for IR-calc and IRM-calc in Chapter 4 and we use strategy extraction as a lower bound technique for QU-Res in Chapter 5. We uncover a tight connection between feasible interpolation and strategy extraction. Both feasible interpolation and strategy extraction transfer hardness from circuit complexity to proof complexity. In this chapter we show that feasible interpolation problems can be transformed into strategy extraction problems, where the interpolant corresponds to the winning strategy of the universal player on the first universal variable. This clarifies that feasible interpolation can be viewed as a special case of strategy extraction.

In Section 8.1 we present the proof for feasible interpolation for IRM-calc. A change needed to adapt this proof for monotone feasible interpolation is given in Section 8.2. We use the monotone feasible interpolation result to prove a lower bound in Section 8.3. Finally in Section 8.4, we discuss the relationship between feasible interpolation and strategy extraction.

The work in this chapter appeared at the 43rd International Colloquium on Automata, Languages, and Programming [21].

8.1 Feasible Interpolation for IRM-calc

In this section we show that feasible interpolation holds in IRM-calc. We adapt the technique first used by [99] to re-prove and generalise the result of [80].

We first use a technical notation for annotated clauses in IRM-calc proofs.

Definition 13. *For clauses C, D we write $C \preceq D$ if for any literal $l \in C$ we have $l \in D$ or $l^* \in D$ and for any $l^* \in C$ we have $l^* \in D$.*

For annotations τ and σ we say that $\tau \preceq \sigma$ if $\mathsf{dom}(\tau) = \mathsf{dom}(\sigma)$ and for any $c/u \in \tau$ we have $c/u \in \sigma$ or $*/u \in \sigma$ and for any $*/u \in \tau$ we have $*/u \in \sigma$.

If C, D are annotated clauses, we write $C \preceq D$ if there is an injective function $f : C \hookrightarrow D$ such that for all $l^\tau \in C$ we have $f(l^\tau) = l^\sigma$ with $\tau \preceq \sigma$.

Consider a false QBF sentence $\mathcal{F}$ of the form

$$\exists \boldsymbol{p} \mathcal{Q} \boldsymbol{q} \mathcal{Q} \boldsymbol{r} \big[A(\boldsymbol{p}, \boldsymbol{q}) \wedge B(\boldsymbol{p}, \boldsymbol{r}) \big],$$

where, $\boldsymbol{p}$, $\boldsymbol{q}$, and $\boldsymbol{r}$ are mutually disjoint sets of propositional variables, $A(\boldsymbol{p}, \boldsymbol{q})$ is a CNF formula on variables $\boldsymbol{p}$ and $\boldsymbol{q}$, and $B(\boldsymbol{p}, \boldsymbol{r})$ is a CNF formula on variables $\boldsymbol{p}$ and $\boldsymbol{r}$. Thus $\boldsymbol{p}$ are the common variables between them. The $\boldsymbol{q}$ and $\boldsymbol{r}$ variables can be quantified arbitrarily, with any number of quantification levels. The sentence is equivalent to the following, not in prenex form

$$\exists \boldsymbol{p} \big[\mathcal{Q} \boldsymbol{q} . A(\boldsymbol{p}, \boldsymbol{q}) \wedge \mathcal{Q} \boldsymbol{r} . B(\boldsymbol{p}, \boldsymbol{r}) \big].$$

Definition 14. *Let $\mathcal{F}$ be a false QBF of the form $\exists \boldsymbol{p}\mathcal{Q}\boldsymbol{q}\mathcal{Q}\boldsymbol{r}.\,[A(\boldsymbol{p},\boldsymbol{q}) \wedge B(\boldsymbol{p},\boldsymbol{r})]$. An* interpolation circuit *for $\mathcal{F}$ is a boolean circuit G such that on every $0,1$ assignment $\boldsymbol{a}$ for $\boldsymbol{p}$ we have*

$$G(\boldsymbol{a}) = 0 \implies \mathcal{Q}\boldsymbol{q}.A(\boldsymbol{a},\boldsymbol{q}) \text{ is false, and}$$
$$G(\boldsymbol{a}) = 1 \implies \mathcal{Q}\boldsymbol{r}.B(\boldsymbol{a},\boldsymbol{r}) \text{ is false.}$$

We say that a QBF proof system S has feasible interpolation *if for any S-proof π of a QBF $\mathcal{F}$ of the form above, we can extract from π an interpolation circuit for $\mathcal{F}$ of size polynomial in the size of π.*

We say that S has monotone feasible interpolation *if the following holds: in the same setting as above, if $\boldsymbol{p}$ appears only positively in $A(\boldsymbol{p},\boldsymbol{q})$, then we can extract from π a monotone interpolation circuit for $\mathcal{F}$.*

As our main result, we show that IRM-calc has feasible interpolation.

Before proving the interpolation theorems, we first outline the general idea:

Proof idea Fix an IRM-calc-proof π of $\mathcal{F}$. Consider the following definition of a $\boldsymbol{q}$-clause and an $\boldsymbol{r}$-clause.

Definition 15. *We call a clause C in π a $\boldsymbol{q}$-clause (resp. $\boldsymbol{r}$-clause), if C contains only variables $\boldsymbol{p},\boldsymbol{q}$ (resp. $\boldsymbol{p},\boldsymbol{r}$). We also call C a $\boldsymbol{q}$-clause (resp. $\boldsymbol{r}$-clause), if C contains only $\boldsymbol{p}$ variables, but all its descendant clauses in the proof π (all clauses with a directed path to C in π) are $\boldsymbol{q}$ (resp. $\boldsymbol{r}$)-clauses. The annotations are not considered and can be from either set.*

From π we construct a circuit C_π with the $\boldsymbol{p}$-variables as inputs. This will be topologically similar to the directed acyclic graph of π: for each node u with clause C_u in the proof π, associate a gate g_u (or a constant-size circuit) in the circuit C_π. In order to check the validity of the circuit we will construct refutations for every assignment to $\boldsymbol{p}$. These will be refutation entirely in $\boldsymbol{q}$-clauses or $\boldsymbol{r}$-clause depending on the outcome of our circuit. We find these refutations in two steps. First, we inductively construct, for any assignment $\boldsymbol{a}$ to the $\boldsymbol{p}$ variables, another proof-like structure $\pi'(\boldsymbol{a})$. Like the circuit C_π these will also be topologically similar to π. For each node u with clause C_u in the proof π, associate a clause $C'_{u,\boldsymbol{a}}$ in the structure $\pi'(\boldsymbol{a})$. The purpose of $\pi'(\boldsymbol{a})$ is to replace rules that rely on $\boldsymbol{p}$ variables. Finally, we obtain $\pi''(\boldsymbol{a})$ from the structure $\pi'(\boldsymbol{a})$ by instantiating $\boldsymbol{p}$ variables to the assignment $\boldsymbol{a}$ and doing some pruning, and show that $\pi''(\boldsymbol{a})$ is a valid proof in S. We then find that if $C_\pi(\boldsymbol{a}) = 0$, then $\pi''(\boldsymbol{a})$ uses only $\boldsymbol{q}$-clauses and thus is a refutation of $\mathcal{Q}\boldsymbol{q}.A(\boldsymbol{a},\boldsymbol{q})$, and if $C_\pi(\boldsymbol{a}) = 1$, then $\pi''(\boldsymbol{a})$ uses only $\boldsymbol{r}$-clauses and thus is a refutation of $\mathcal{Q}\boldsymbol{r}.B(\boldsymbol{a},\boldsymbol{r})$. Thus C_π is the desired interpolant circuit.

More precisely, we show by induction on the height of u in π (that is, the length of the longest path to u from a source node in π) that:

1. $C'_{u,\boldsymbol{a}} \preceq C_u$.
2. $g_u(\boldsymbol{a}) = 0 \implies C''_{u,\boldsymbol{a}}$ is a $\boldsymbol{q}$-clause and can be obtained from the clauses of $A(\boldsymbol{a},\boldsymbol{q})$ alone using the rules of S.

3. $g_u(\boldsymbol{a}) = 1 \implies C''_{u,\boldsymbol{a}}$ is an $\boldsymbol{r}$-clause and can be obtained from the clauses of $B(\boldsymbol{a}, \boldsymbol{r})$ alone using the rules of S.

From the above, we have the following conclusion. Let r be the root of π. Then on any assignment $\boldsymbol{a}$ to the $\boldsymbol{p}$ variables we have:

(1) $C'_{r,\boldsymbol{a}} \preceq C_r = \square$, so $C'_{r,\boldsymbol{a}} = \square$. Therefore, $C''_{r,\boldsymbol{a}} = C'_{r,\boldsymbol{a}}|_{\boldsymbol{a}} = \square$.
(2) $g_r(\boldsymbol{a}) = 0 \implies \square$ is a $\boldsymbol{q}$-clause and can be obtained from the clauses of $A(\boldsymbol{a}, \boldsymbol{q})$ alone using the rules of system S. Hence by soundness of S, $\mathcal{Q}\boldsymbol{q}.A(\boldsymbol{a}, \boldsymbol{q})$ is false.
(3) $g_r(\boldsymbol{a}) = 1 \implies \square$ is an $\boldsymbol{r}$-clause and can be obtained from the clauses of $B(\boldsymbol{a}, \boldsymbol{r})$ alone using the rules of system S. Hence by soundness of S, $\mathcal{Q}\boldsymbol{r}.B(\boldsymbol{a}, \boldsymbol{r})$ is false.

Thus g_r, the output gate of the circuit, computes an interpolant.

When $\mathcal{F}$ has only existential quantification, π is a classical resolution proof, and this is exactly the interpolant computed by Pudlák's method [99]. The challenge here is to construct π' and π'' appropriately when the stronger proof systems are used for general QBF, while maintaining the inductive invariants.

Theorem 30. *IRM-calc has feasible interpolation.*

As mentioned in the proof idea, for an IRM-calc proof π of $\mathcal{F}$ we first describe the circuit C_π with input $\boldsymbol{p}$.

Proof. **Construction of the circuit C_π:** The DAG underlying the circuit is exactly the same as the DAG underlying the proof π. For each node u with clause C_u in π we associate a gate g_u as follows:

u is a leaf node: If $C_u \in A(\boldsymbol{p}, \boldsymbol{q})$ then g_u is a constant 0 gate. If $C_u \in B(\boldsymbol{p}, \boldsymbol{r})$ then g_u is a constant 1 gate.

u is an internal node: We distinguish four cases.

1. u was derived by either
   (a) instantiation,
   (b) merging.
   In this case put a no-operation gate for g_u.
2. u corresponds to a resolution step with an existential variable $x \in \boldsymbol{p}$ as pivot. Nodes v and w are its two children, i.e.
$$\frac{\overbrace{C_1 \vee x}^{\text{node } v} \qquad \overbrace{C_2 \vee \neg x}^{\text{node } w}}{\underbrace{C}_{\text{node } u}}.$$
   In this case, put a selector gate $\mathsf{sel}(x, g_v, g_w)$ for g_u. Here, $\mathsf{sel}(x, a, b) = a$, when $x = 0$ and $\mathsf{sel}(x, a, b) = b$, when $x = 1$. That is, $\mathsf{sel}(x, a, b) = (\neg x \wedge a) \vee (x \wedge b)$. Note that all the variables in $\boldsymbol{p}$ are existential variables without annotations.

3. u corresponds to a resolution step with an existential variable $x \in \boldsymbol{q}$ as pivot. Put an OR gate for g_u.
4. u corresponds to a resolution step with an existential variable $x \in \boldsymbol{r}$ as pivot. Put an AND gate for g_u.

This completes the description of the circuit C_π.

Construction of π' and π'': We construct a proof-like structure $\pi'(\boldsymbol{a})$, which depends on the assignment $\boldsymbol{a}$ to the $\boldsymbol{p}$ variables, the proof π of $\mathcal{F}$, and the circuit C_π. For each node u in π, with clause C_u, we associate a clause $C'_{u,\boldsymbol{a}}$ in $\pi'(\boldsymbol{a})$, and let $C''_{u,\boldsymbol{a}}$ be the instantiation of $C'_{u,\boldsymbol{a}}$ by the assignment $\boldsymbol{a}$. We show (by induction on the height of u in π) that:

1. $C'_{u,\boldsymbol{a}} \preceq C_u$.
2. $g_u(\boldsymbol{a}) = 0 \implies C''_{u,\boldsymbol{a}}$ is a $\boldsymbol{q}$-clause and can be obtained from the clauses of $A(\boldsymbol{a}, \boldsymbol{q})$ alone using the rules of system IRM-calc.
3. $g_u(\boldsymbol{a}) = 1 \implies C''_{u,\boldsymbol{a}}$ is a $\boldsymbol{r}$-clause and can be obtained from the clauses of $B(\boldsymbol{a}, \boldsymbol{r})$ alone using the rules of system IRM-calc.

As described in the proof outline, this suffices to conclude that the circuit C_π computes an interpolant.

At a leaf level: Let node u be a leaf in π. Then $C'_{u,\boldsymbol{a}} = C_u$; that is, copy the clause as it is. Trivially, $C'_{u,\boldsymbol{a}} \preceq C_u$. The gates give the correct values by definition.

At an internal node with instantiation: Let node u be an internal node in π corresponding to an instantiation step by τ. And let node v be its only child. We know $C_u = \mathsf{inst}(\tau, C_v)$.

Suppose $l^{\sigma'} \in \mathsf{inst}(\tau, C'_{v,\boldsymbol{a}})$. Then for some ξ', $l^{\xi'} \in C'_{v,\boldsymbol{a}}$, and $l^{\sigma'} = l^{[\xi' \circ \tau]}$; hence σ' is a subset of ξ' completed with τ. By induction we know that $C'_{v,\boldsymbol{a}} \preceq C_v$. We have an injective function $f : C'_{v,\boldsymbol{a}} \hookrightarrow C_v$ that demonstrates this. Let $f(l^{\xi'}) = l^{\xi}$. Hence $l^{\xi} \in C_v$ for some $\xi' \preceq \xi$. So $l^{\sigma} = l^{[\xi \circ \tau]} \in C_u$. Since the annotations introduced by instantiation match, $\sigma' \preceq \sigma$. We use this to define a function $g : \mathsf{inst}(\tau, C'_{v,\boldsymbol{a}}) \to C_u$ where $g(l^{\sigma'}) = l^{\sigma}$. Now we find any l^{τ_1}, l^{τ_2} where $g(l^{\tau_1}) = g(l^{\tau_2}) = l^{\tau}$ and perform a merging step on l^{τ_1} and l^{τ_2}; note that the resulting literal $l^{\tau'}$ will still satisfy $\tau' \preceq \tau$. Eventually we get a clause which we define as $\mathsf{instmerge}(\tau, C'_{v,\boldsymbol{a}}, C_u) = C'_{u,\boldsymbol{a}}$ where this function is injective. We will use this notation to refer to this process of instantiation and then deliberate merging to get $\preceq C_u$.

Therefore $C'_{u,\boldsymbol{a}} \preceq C_u$.

If the node u is not pruned out in $\pi''(\boldsymbol{a})$, then $C''_{u,\boldsymbol{a}}$ contains no satisfied $\boldsymbol{p}$ literals; hence neither does $C'_{v,\boldsymbol{a}}$. Therefore $C''_{u,\boldsymbol{a}}$ is derived from $C''_{v,\boldsymbol{a}}$; this is a valid step in the proof system.

Because we only use instantiation and merging or a dummy step, $C''_{u,\boldsymbol{a}}$ is a $\boldsymbol{q}$-clause if and only if $C''_{v,\boldsymbol{a}}$ is a $\boldsymbol{q}$-clause. Therefore the no-operation gate g_u gives a valid result by induction.

At an internal node with merging: Let node u be an internal node in π corresponding to a merging step. Let node v be its only child. We have

$$\frac{C_v = D_v \vee b^{\mu} \vee b^{\sigma}}{C_u = D_v \vee b^{\xi}}$$

where $\mathsf{dom}(\mu) = \mathsf{dom}(\sigma)$ and ξ is obtained by merging the annotations μ, σ. That is, $\xi = \mathsf{AMerge}(\mu, \sigma) \triangleq \{c/u \mid c/u \in \mu, c/u \in \sigma\} \cup \{*/u \mid c/u \in \mu, d/u \in \sigma, c \neq d\}$. Note that $\mu, \sigma \preceq \mathsf{AMerge}(\mu, \sigma)$.

Note that from the induction hypothesis, $C'_{v(\boldsymbol{a})} \preceq C_v$, so there is an injective function $f : C'_{v(\boldsymbol{a})} \hookrightarrow C_v$. Suppose $C'_{v(\boldsymbol{a})}$ contains two distinct literals $b^{\mu'}$ and $b^{\sigma'}$ where $f(b^{\mu'}) = b^{\mu}$ and $f(b^{\sigma'}) = b^{\sigma}$. So $C'_{v(\boldsymbol{a})} = D'_v \vee b^{\mu'} \vee b^{\sigma'}$. Then let $C'_{v,\boldsymbol{a}} = D'_v \vee b^{\xi'}$, where $\xi' = \mathsf{AMerge}(\mu', \sigma')$. Otherwise let $C'_{u,\boldsymbol{a}} = C'_{v,\boldsymbol{a}}$.

We first observe whenever we do actual merging, if $c/u \in \xi'$ then one of the following holds:

1. $c/u \in \sigma'$. Then $c/u \in \sigma$ or $*/u \in \sigma$, and so $c/u \in \xi$ or $*/u \in \xi$.
2. $c/u \in \mu'$. Then $c/u \in \mu$ or $*/u \in \mu$, and so $c/u \in \xi$ or $*/u \in \xi$.
3. $e/u \in \mu'$, $d/u \in \sigma'$, $e \neq d$, in which case $*/u \in \xi$.

Since all other annotated literals are unaffected, $C'_{u,\boldsymbol{a}} \preceq C_u$. We never merge $\boldsymbol{p}$ literals as they have no annotations, so if $C''_{u,\boldsymbol{a}}$ is not pruned away, then $C''_{u,\boldsymbol{a}}$ is derived from $C''_{v,\boldsymbol{a}}$ via merging.

In case we do not merge, there might be some $b^{\sigma'} \in C'_{v,\boldsymbol{a}}$ with $\sigma' \preceq \sigma$, which is not removed by merging. However $\sigma' \preceq \sigma \preceq \xi$, so $C'_{u,\boldsymbol{a}} = C'_{v,\boldsymbol{a}} \preceq C_u$. As $C''_{u,\boldsymbol{a}} = C''_{v,\boldsymbol{a}}$, this is a valid inference step (in fact, a dummy step).

Because we only use merging or a dummy step, $C''_{u,\boldsymbol{a}}$ is a $\boldsymbol{q}$-clause if and only if $C''_{v,\boldsymbol{a}}$ is a $\boldsymbol{q}$-clause, therefore the no-operation gate g_u gives a valid result by induction.

At an internal node with $\boldsymbol{p}$-resolution: Let node u in the proof π correspond to a resolution step with pivot $x \in \boldsymbol{p}$. Note that x is existential, as $\boldsymbol{p}$ variables occur only existentially in $\mathcal{F}$. We have

$$\frac{C_v = \overbrace{C_1 \vee x}^{\text{node } v} \qquad \overbrace{C_2 \vee \neg x}^{\text{node } w} = C_w}{C_u = \underbrace{C_1 \vee C_2}_{\text{node } u}}.$$

In the assignment $\boldsymbol{a}$, if $x = 0$, then define $C'_{u,\boldsymbol{a}} = C'_{v,\boldsymbol{a}} \setminus \{x\}$ and if $x = 1$ then define $C'_{u,\boldsymbol{a}} = C'_{w,\boldsymbol{a}} \setminus \{\neg x\}$. By induction, we have $C'_{v,\boldsymbol{a}} \preceq C_v$ and $C'_{w,\boldsymbol{a}} \preceq C_w$. So, if $x = 0$, we have $C'_{u,\boldsymbol{a}} = C'_{v,\boldsymbol{a}} \setminus \{x\} \preceq C_1 \preceq C_u$. If $x = 1$, we have $C'_{u,\boldsymbol{a}} \preceq C'_{w,\boldsymbol{a}} \setminus \{\neg x\} \preceq C_2 \preceq C_u$.

In this case g_u is a selector gate. If $x = 0$ in the assignment $\boldsymbol{a}$, then $g_u(\boldsymbol{a}) = g_v(\boldsymbol{a})$ and $C''_{u,\boldsymbol{a}} = C''_{v,(a)}$. Since the conditions concerning $g_v(\boldsymbol{a})$ and $C''_{v,\boldsymbol{a}}$ are satisfied by induction, the conditions concerning $g_u(\boldsymbol{a})$ and $C''_{u,\boldsymbol{a}}$ are satisfied as well. Similarly, if $x = 1$, then $g_u(\boldsymbol{a}) = g_w(\boldsymbol{a})$ and $C''_{u,\boldsymbol{a}} = C''_{w,\boldsymbol{a}}$, and the statements that are inductively true at w hold at u as well.

At an internal node with $\boldsymbol{q}$-resolution: When we have a resolution step between nodes v and w on a $\boldsymbol{q}$ pivot to get node u, we have

$$\frac{C_v = x^{\tau \cup \xi} \vee D_v \qquad C_w = \neg x^{\tau \cup \sigma} \vee D_w}{C_u = \mathsf{inst}(\sigma, D_v) \cup \mathsf{inst}(\xi, D_w)}$$

where $\mathsf{dom}(\tau)$, $\mathsf{dom}(\xi)$ and $\mathsf{dom}(\sigma)$ are mutually disjoint, and $\mathsf{rng}(\tau) = \{0, 1\}$.

In order to do dummy instantiations we will need to define a $\{0,1\}$ version of ξ and σ. So we define $\xi' = \{c/u \mid c/u \in \xi, c \in \{0,1\}\} \cup \{0/u \mid */u \in \xi\}$, $\sigma' = \{c/u \mid c/u \in \sigma, c \in \{0,1\}\} \cup \{0/u \mid */u \in \sigma\}$. This gives us the desirable property that $\xi' \preceq \xi$, $\sigma' \preceq \sigma$.

Now resuming the construction of C', we use information from the circuit to construct this. If $g_v(\boldsymbol{a}) = 1$, then we define $C'_{u,\boldsymbol{a}} = \mathsf{instmerge}(\sigma', C'_{v,\boldsymbol{a}}, C_u)$. Otherwise, if $g_w(\boldsymbol{a}) = 1$, then we define $C'_{u,\boldsymbol{a}} = \mathsf{instmerge}(\xi', C'_{w,\boldsymbol{a}}, C_u)$. In these cases, we know by the inductive claim that $C'_{u,\boldsymbol{a}}$ does not contain any $\boldsymbol{q}$ literals. Therefore $C'_{u,\boldsymbol{a}}$ is the correct instantiation (as $\xi' \preceq \xi$, $\sigma' \preceq \sigma$) of some subset of D_v or D_w. Hence $C'_{u,\boldsymbol{a}} \preceq C_u$. Furthermore since g_u is an OR gate evaluating to 1 and since $C''_{u,\boldsymbol{a}}$, an $\boldsymbol{r}$-clause, can be obtained by an instantiation step, our inductive claim is true.

Now suppose $g_v(\boldsymbol{a}) = 0$ and $g_w(\boldsymbol{a}) = 0$. If there is no $x^\mu \in C'_{v,\boldsymbol{a}}$ such that $\mu \preceq \tau \cup \xi$, then define $C'_{u,\boldsymbol{a}} = \mathsf{instmerge}(\sigma', C'_{v,\boldsymbol{a}}, C_u)$. Else, if there is no $\neg x^\mu \in C'_{w,\boldsymbol{a}}$ such that $\mu \preceq \tau \cup \sigma$, then define $C'_{u,\boldsymbol{a}} = \mathsf{instmerge}(\xi', C'_{w,\boldsymbol{a}}, C_u)$. In these cases we know that $C'_{u,\boldsymbol{a}}$ is the correct instantiation (as $\xi' \preceq \xi$, $\sigma' \preceq \sigma$) of some subset of D_v or D_w; hence $C'_{u,\boldsymbol{a}} \preceq C_u$. Furthermore, since g_u is an OR gate evaluating to 0, and since $C''_{u,\boldsymbol{a}}$, a $\boldsymbol{q}$-clause, can be obtained by an instantiation step, our inductive claim is true.

The final case is when $g_v(\boldsymbol{a}) = g_w(\boldsymbol{a}) = 0$ and $x^{\tau\cup\xi_1} \in C'_{v,\boldsymbol{a}}$ for some $\xi_1 \preceq \xi$ and $\neg x^{\tau\cup\sigma_1} \in C'_{w,\boldsymbol{a}}$ for some $\sigma_1 \preceq \sigma$. Here, because $\mathsf{dom}(\tau)$, $\mathsf{dom}(\xi)$ and $\mathsf{dom}(\sigma)$ are mutually disjoint, $\mathsf{dom}(\tau)$, $\mathsf{dom}(\xi_1)$ and $\mathsf{dom}(\sigma_1)$ are also mutually disjoint. Thus we can do the resolution step

$$\frac{C'_{v,\boldsymbol{a}} = x^{\tau\cup\xi_1} \vee D'_v \qquad C'_{w,\boldsymbol{a}} = \neg x^{\tau\cup\sigma_1} \vee D'_w}{\mathsf{inst}(\sigma_1, D'_v) \cup \mathsf{inst}(\xi_1, D'_w)}.$$

Since $\mathsf{instmerge}(\sigma_1, D'_v, C_u) \preceq \mathsf{inst}(\sigma, D_v)$ and $\mathsf{instmerge}(\xi_1, D'_w, C_u) \preceq \mathsf{inst}(\xi, D_w)$, we can follow up $\mathsf{inst}(\sigma_1, D'_v) \cup \mathsf{inst}(\xi_1, D'_w)$ with sufficient merging steps to get a clause $C' \preceq C_u$; we define this clause to be the clause $C'_{u,\boldsymbol{a}}$. By the inductive claim, both $C''_{v,\boldsymbol{a}}$ and $C''_{w,\boldsymbol{a}}$ are $\boldsymbol{q}$-clauses; hence $C''_{u,\boldsymbol{a}}$ is also a $\boldsymbol{q}$-clause and is obtained via a valid resolution step.

At an internal node with $\boldsymbol{r}$-resolution: When we have a resolution step between nodes u and v on an $\boldsymbol{r}$-literal, this is the dual of the previous case. □

8.2 Monotone Interpolation

To transfer known circuit lower bounds into size of proof bounds, we need a monotone version of the previous interpolation theorems, which we prove next.

Theorem 31. *IRM-calc has monotone feasible interpolation.*

Proof. In the previous section, we have shown that the circuit $C_\pi(\boldsymbol{p})$ is a correct interpolant for the QBF sentence $\mathcal{F}$. That is, if $C_\pi(\boldsymbol{p}) = 0$ then $\mathcal{Q}\boldsymbol{q}.A(\boldsymbol{a}, \boldsymbol{q})$ is false, and if $C_\pi(\boldsymbol{p}) = 1$ then $\mathcal{Q}\boldsymbol{r}.B(\boldsymbol{a}, \boldsymbol{r})$ is false.

However, if $\boldsymbol{p}$ occurs only positively in $A(\boldsymbol{p}, \boldsymbol{q})$ then we construct a monotone circuit $C_\pi^{mon}(\boldsymbol{p})$ such that, on every $0,1$ assignment $\boldsymbol{a}$ to $\boldsymbol{p}$ we have

$$C_\pi^{mon}(\boldsymbol{a}) = 0 \implies \mathcal{Q}\boldsymbol{q}.A(\boldsymbol{a}, \boldsymbol{q}) \text{ is false, and}$$
$$C_\pi^{mon}(\boldsymbol{a}) = 1 \implies \mathcal{Q}\boldsymbol{r}.B(\boldsymbol{a}, \boldsymbol{r}) \text{ is false.}$$

We obtain $C_\pi^{mon}(\boldsymbol{p})$ from $C_\pi(\boldsymbol{p})$ by replacing all selector gates $g_u = \mathsf{sel}(x, g_v, g_w)$ by the following monotone ternary connective: $g_u = (x \vee g_v) \wedge g_w$ where nodes v and w are the children of u in π.

We also change the proof-like structure $\pi'(\boldsymbol{a})$; the construction is the same as before except that at $\boldsymbol{p}$-resolution nodes, the rule for fixing $C'_{u,\boldsymbol{a}}$ is also changed to reflect the monotone function used instead.

More precisely, the functions $\mathsf{sel}(x, g_v, g_w)$ and $g_u = (x \vee g_v) \wedge g_w$ differ only when $x = 0$, $g_v(\boldsymbol{a}) = 1$, and $g_w(\boldsymbol{a}) = 0$. We set $C'_{u,\boldsymbol{a}}$ to $C'_{w,\boldsymbol{a}} \setminus \{\neg x\}$ if $x = 1$ or if $x = 0$, $g_v(\boldsymbol{a}) = 1$ and $g_w(\boldsymbol{a}) = 0$, and to $C'_{v,\boldsymbol{a}} \setminus \{x\}$ otherwise.

We need to show that at the differing setting, the inductive statements relating the modified $C'_{u,\boldsymbol{a}}$, $g_u(\boldsymbol{a})$ and $C''_{u,\boldsymbol{a}}$ continue to hold. The relation $C'_{u,\boldsymbol{a}} \preceq C_u$ holds by induction. Now consider the gate values.

We know by induction that $g_v(\boldsymbol{a}) = 1$ means that $C''_{v,\boldsymbol{a}}$ is an $\boldsymbol{r}$-clause and can be derived from $B(\boldsymbol{a}, \boldsymbol{r})$ alone. When $x = 0$, $C'_{u,\boldsymbol{a}} = C'_{v,\boldsymbol{a}}$ and the selector gate will output the value of $g_v(\boldsymbol{a})$ which is a 1. Hence $C''_{u,\boldsymbol{a}}$ is an $\boldsymbol{r}$-clause. However observe that at this setting, $g_w(\boldsymbol{a}) = 0$, which means by induction that $C''_{w,\boldsymbol{a}}$ is a $\boldsymbol{q}$-clause and can be derived using $A(\boldsymbol{a}, \boldsymbol{q})$ clauses alone via the appropriate proof system. Thus by our assumption about $\boldsymbol{p}$ variables appearing only positively in A, the clause $C'_{w,\boldsymbol{a}}$ does not contain $\neg x$. Thus we can safely assign $C'_{u,\boldsymbol{a}} = C'_{w,\boldsymbol{a}}$. This completes the proof. □

Notice that IRM-calc being a calculus with (monotone) feasible interpolation means that via a transformation to IRM-calc, every calculus that IRM-calc p-simulates also have feasible (monotone) interpolation. Namely, this means that the expansion calculi IR-calc and ∀Exp+Res have these interpolation results. The same idea can be applied to the CDCL proof systems, it would suffice that the most powerful CDCL system- LQU$^+$-Res (refer to Figure 18 for simulation details) has feasible (monotone) interpolation and indeed it is true as shown in [21], where both feasible interpolation and feasible monotone interpolation was shown to be true in LQU$^+$-Res in much that same fashion as it is shown true for IRM-calc here.

8.3 New Exponential Lower Bounds for IRM-calc

We now apply our interpolation theorems to obtain new exponential lower bounds for a new class of QBFs. The lower bound will be directly transferred from the following monotone circuit lower bound for the problem CLIQUE(n, k), asking whether a given graph with n nodes has a clique of size k.

Theorem 32 (Alon and Boppana 87 [3]). *All monotone circuits that compute* CLIQUE$(n, n/2)$ *are of exponential size.*

We now build the QBF. Fix an integer n (indicating the number of vertices of the graph) and let $\boldsymbol{p}$ be the set of variables $\{p_{uv} \mid 1 \leq u < v \leq n\}$. An assignment to $\boldsymbol{p}$ picks a set of edges, and thus an n-vertex graph. Let $\boldsymbol{q}$ be the set of variables $\{q_{iu} \mid i \in [\frac{n}{2}], u \in [n]\}$. We use the following clauses.

$$\begin{aligned}
C_i &= q_{i1} \vee \cdots \vee q_{in} && \text{for } i \in [\tfrac{n}{2}] \\
D_{i,j,u} &= \neg q_{iu} \vee \neg q_{ju} && \text{for } i, j \in [\tfrac{n}{2}], i < j \text{ and } u \in [n] \\
E_{i,u,v} &= \neg q_{iu} \vee \neg q_{iv} && \text{for } i \in [\tfrac{n}{2}] \text{ and } u, v \in [n], u < v \\
F_{i,j,u,v} &= \neg q_{iu} \vee \neg q_{jv} \vee p_{uv} && \text{for } i, j \in [\tfrac{n}{2}], i < j \text{ and } u \neq v \in [n].
\end{aligned}$$

We can now express CLIQUE$(n, n/2)$ as a polynomial-size QBF $\exists \boldsymbol{q}.A_n(\boldsymbol{p}, \boldsymbol{q})$, where

$$A_n(\boldsymbol{p}, \boldsymbol{q}) = \bigwedge_{i \in [\frac{n}{2}]} C_i \wedge \bigwedge_{i<j, u \in [n]} D_{i,j,u} \wedge \bigwedge_{i \in [\frac{n}{2}], u<v} E_{i,u,v} \wedge \bigwedge_{i<j, u \neq v} F_{i,j,u,v}.$$

Here the edge variables $\boldsymbol{p}$ appear monotone in $A_n(\boldsymbol{p}, \boldsymbol{q})$.

Likewise co-CLIQUE$(n, n/2)$ can be written as a polynomial-size QBF $\forall \boldsymbol{r_1} \exists \boldsymbol{r_2}.B_n(\boldsymbol{p}, \boldsymbol{r_1}, \boldsymbol{r_2})$. To construct this we use a polynomial-size circuit that checks whether the nodes specified by $\boldsymbol{r_1}$ fail to form a clique in the graph given by $\boldsymbol{p}$. We then use existential variables $\boldsymbol{r_2}$ for the gates of the circuit and can then form a CNF $B_n(\boldsymbol{p}, \boldsymbol{r_1}, \boldsymbol{r_2})$ that represents the circuit computation. For $\boldsymbol{r_1}$, we have a variable r_{iu} for every variable q_{iu} and we let the set of variables of $\boldsymbol{r_2}$ be $\{t_K \mid K \in A_{n,k}\} \cup \{t\}$. For each clause K in $A_{n,k}(\boldsymbol{p}, \boldsymbol{q})$, we include an equivalence $t_K \leftrightarrow K[r_{iu}/q_{iu}]$ in $B_{n,k}(\boldsymbol{p}, \boldsymbol{r}, \boldsymbol{t})$, which we represent as a set of clauses. We also introduce clauses for $t \leftrightarrow \bigwedge_{K \in A_{n,k}} t_K$, i.e., t indicates whether the $\boldsymbol{r}$ variables encode a clique. Because we want to represent the co-clique formula we also include $\neg t$ in $B_{n,k}(\boldsymbol{p}, \boldsymbol{r_1}, \boldsymbol{r_2})$, which yields the CNF formula co-CLIQUE$(n, k) = \forall \boldsymbol{r_2} \exists \boldsymbol{r_2}.B_{n,k}(\boldsymbol{p}, \boldsymbol{r_1}, \boldsymbol{r_2})$.

Now we can form a sequence of false QBFs, stating that the graph encoded in $\boldsymbol{p}$ both has a clique of size $n/2$ (as witnessed by $\boldsymbol{q}$) and likewise does not have such a clique as expressed in the B part:

$$\Phi_n(\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{r}) = \exists \boldsymbol{p} \exists \boldsymbol{q} \forall \boldsymbol{r_1} \exists \boldsymbol{r_2}.A_n(\boldsymbol{p}, \boldsymbol{q}) \wedge B_n(\boldsymbol{p}, \boldsymbol{r_1}, \boldsymbol{r_2}).$$

This formula has the unique interpolant CLIQUE$(n, n/2)(\boldsymbol{p})$. But since all monotone circuits for this are of exponential size by Theorem 32 and monotone circuits of size polynomial in IRM-calc and LQU$^+$-Res proofs can be extracted by Theorem 31, all such proofs must be of exponential size, yielding:

Theorem 33. *The QBFs $\Phi_n(\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{r})$ require exponential-size proofs in IRM-calc.*

Theorem 34 (Beyersdorff, Chew, Mahajan, Shukla '15 [21]). *The QBFs $\Phi_n(\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{r})$ require exponential-size proofs in LQU$^+$-Res.*

8.4 Feasible Interpolation vs. Strategy Extraction

Recall the two-player game semantics of a QBF explained in Chapter 3. Every false QBF has a winning strategy for the universal player, where the strategy for each variable depends only on the variables played before. We now explain the relation between strategy extraction and feasible interpolation. In Section 8.1 we studied QBFs of the form $\mathcal{F} = \exists \boldsymbol{p} \mathcal{Q} \boldsymbol{q} \mathcal{Q} \boldsymbol{r}.\, [A(\boldsymbol{p}, \boldsymbol{q}) \wedge B(\boldsymbol{p}, \boldsymbol{r})]$. If we add a common universal variable b we can change it to an equivalent QBF

$$\mathcal{F}^b = \exists \boldsymbol{p} \forall b\, \mathcal{Q} \boldsymbol{q}\, \mathcal{Q} \boldsymbol{r}.\, [(A(\boldsymbol{p}, \boldsymbol{q}) \vee b) \wedge (B(\boldsymbol{p}, \boldsymbol{r}) \vee \neg b)]\,.$$

If $\mathcal{F}$ is false, then also $\mathcal{F}^b$ is false and thus the universal player has a winning strategy, including a strategy for $b = \sigma(\boldsymbol{p})$ for the common universal variable b.

Remark 5. Every winning strategy $\sigma(\boldsymbol{p})$ for b is an interpolant for $\mathcal{F}$, i.e., for every $0, 1$ assignment $\boldsymbol{a}$ of $\boldsymbol{p}$ we have

$$\begin{aligned} \sigma(\boldsymbol{a}) = 0 &\implies \mathcal{Q}\boldsymbol{q}.A(\boldsymbol{a}, \boldsymbol{q}) \text{ is false, and} \\ \sigma(\boldsymbol{a}) = 1 &\implies \mathcal{Q}\boldsymbol{r}.B(\boldsymbol{a}, \boldsymbol{r}) \text{ is false.} \end{aligned}$$

Proof. Suppose not. Then there are two possibilities.

1. There is some $\boldsymbol{a}$ where $\sigma(\boldsymbol{a}) = 0$ and $\mathcal{Q}\boldsymbol{q}.A(\boldsymbol{a}, \boldsymbol{q})$ is true. Then setting $b = 0$ would satisfy $\mathcal{Q}\boldsymbol{r}.B(\boldsymbol{p}, \boldsymbol{r}) \vee \neg b$. But $\mathcal{Q}\boldsymbol{q}.A(\boldsymbol{a}, \boldsymbol{q}) \vee b$ is also satisfied. Hence this cannot be part of the winning strategy for the universal player.
2. There is some $\boldsymbol{a}$ where $\sigma(\boldsymbol{a}) = 1$ and $\mathcal{Q}\boldsymbol{r}.B(\boldsymbol{a}, \boldsymbol{r})$ is true. This is the dual of the above. □

This observation means that every interpolation problem can be reformulated as a strategy extraction problem. We will now show that from proofs of these reformulated interpolation problems we can extract a (monotone) Boolean circuit for the winning strategy on the new universal variable b.

We now prove how to extract strategies for interpolation problems, first for LQU$^+$-Res and then for IRM-calc.

Theorem 35 (Beyersdorff, Chew, Mahajan, Shukla '15 [21]).

1. *From each LQU$^+$-Res refutation π of $\mathcal{F}^b$ we can extract in polynomial time a boolean circuit for $\sigma(\boldsymbol{p})$, i.e., the part of the winning strategy for variable b.*
2. *If in the same setting as above for $\mathcal{F}^b$, the variables $\boldsymbol{p}$ appear only positively in $A(\boldsymbol{p}, \boldsymbol{q})$, then we can extract a monotone boolean circuit for $\sigma(\boldsymbol{p})$ from a LQU$^+$-Res refutation π of $\mathcal{F}^b$ in polynomial time (in the size of π).*

Theorem 36. *1. From each IRM-calc refutation π of $\mathcal{F}^b$ we can extract in polynomial time a boolean circuit for $\sigma(\boldsymbol{p})$, i.e., the part of the winning strategy for variable b.*

2. *If in the same setting as above for $\mathcal{F}^b$, the variables $\boldsymbol{p}$ appear only positively in $A(\boldsymbol{p}, \boldsymbol{q})$, then we can extract a monotone boolean circuit for $\sigma(\boldsymbol{p})$ from a* IRM-calc *refutation π of $\mathcal{F}^b$ in polynomial time (in the size of π).*

Proof. We can use exactly the same constructions as in Theorem 30. The b literals do not affect the argument. □

As a corollary, the versions $\Phi_n^b(\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{r})$ of the formulas from Section 13.4 also need exponential-size proofs in IRM-calc and LQU$^+$-Res.

If we were only concerned about lower bounds we could have used the strategy extraction lower bound, however our approach of first showing the feasible interpolation result illuminates the relationship between feasible interpolation (in both the propositional and QBF setting) and strategy extraction. Another reason that we explored feasible interpolation is to properly compare propositional lower bound techniques to their natural QBF analogues.

We revisit feasible interpolation in Chapter 12 in Part III where we see it work for a QBF adaptation of the Cutting Planes proof system as it does in propositional logic. In Chapter 9 we continue our investigation into lower bound techniques.

Chapter 9

Size, Width and Space for Tree-like QBF Calculi

In previous chapters we have looked at lower bound techniques that apply to QBF-resolution calculi. While strategy extraction used in Chapter 5 is a new technique that works specifically in the QBF-setting, older techniques still apply.

We have found that the game techniques from [27–29] work with slight adjustments (Chapter 7) as do the feasible interpolation techniques from [80,99] (Chapter 8, Chapter 12). The most important technique that works for propositional logic is arguably the size-width relation by Ben-Sasson and Wigderson [13]. It links proof size and proof width in the resolution system, by showing that lower bounds to size can emerge when lower bounds to the width are found.

In this chapter we review the work of Beyersdorff et. al [22], presented at the 33rd Symposium on Theoretical Aspects of Computer Science where the size-width relation in QBF was investigated. It was shown that, in order to understand width-relations in QBF resolution, existential width was better to use. This is due to an example of false formulas that accumulate universal variables in Q-Res proofs, but have short proofs due to reduction.

However existential width is not enough to give us the size-width relation. The crux of the size-width relation proof from [13] fails due to the restrictions in Q-Res. Furthermore, there are examples of formulas in Q-Res and tree-like Q-Res (Q-Res_T) that violate the size-width relation, even for existential width.

In [22] it was shown that tree-like ∀Exp+Res simulates tree-like IR-calc, furthermore the simulation preserves width. Hence, that the two systems are equivalent. Since, after the axiom rules, ∀Exp+Res works the same as proposition resolution it was shown that the size-width relation holds in tree-like ∀Exp+Res and hence tree-like IR-calc.

Section 9.1 covers the basic definitions of width and space and their relations to size and each other in propositional proof complexity. In the remaining chapters we present the work from [22] without proof. Section 9.2 introduces the notion of existential width for QBF and justifies its use. Section 9.3 contains the negative results, that Q-Res does not have the size-width relation. However, Section 9.4 contains a simulation in tree-like systems that make the size-width relation possible in tree-like IR-calc and ∀Exp+Res in Section 9.5.

9.1 Width and Space

Width. The *width* of a clause C is the number of literals in C, denoted $w(C)$. For a CNF F the width is given by $w(F) = \max_{C \in F}(w(C))$. If we take any line-based proof system P that operates

on lines that are clauses and then if we take a proof π, then $w(\pi) = \max(w(L) \ : \ L \text{ is a line in } \pi)$, and the width $w(\vdash_P F) = \min\{w(\pi) \ : \ \pi \vdash_P F\}$.

We can then present the famous result from Ben-Sasson and Wigderson for tree-like resolution and resolution.

Theorem 37 (Ben-Sasson, Wigderson [13]). *For all unsatisfiable CNFs F in n variables the following holds:* $S(\vdash_{\overline{Res_T}} F) \geq 2^{w\left(\vdash_{\overline{Res}} F\right)-w(F)}$ *and* $S(\vdash_{\overline{Res}} F) = \exp\left(\Omega\left(\frac{\left(w\left(\vdash_{\overline{Res}} F\right)-w(F)\right)^2}{n}\right)\right)$.

Note that the width does not change in a tree-like system compared to its dag-like version.

The size-width relation can generate exponential-size lower bounds. An example would be from the famous formulas of Tseitin. The formulas are built on graphs and $\tau(G, f) = \bigwedge_{v \in V(G)} \bigoplus_{v \in e} x_e = f(v)$ where $f(v)$ is an odd-weight boolean function (in other words $\sum_{v \in V(G)} f(v) = 1 \bmod 2$). The formulas $\tau(G, f)$ are always false because for them to be true the sum of the vertex degree would have to be odd.

When the maximal degree of G is constant then the initial width of $\tau(G, f)$ is constant. The width of proof is bounded below by the value of the expansion of G which is given as $\text{expand}(G) = \min |E(V', V \setminus V')| : V' \subseteq V, \frac{|V|}{3} \leq |V'| \leq \frac{2|V|}{3}$. It is known that $\text{expand}(G) = \Omega(|V|)$ for 3-regular graphs, so we get a width lower bound and thus an exponential size lower bound.

Space. Proof space complexity was introduced in [54]. Space, informally, is the minimum amount of computer memory that is simultaneously needed to refute a formula. Just as proof size is related to running time of solvers, proof space has a relation to the memory required for solvers. This is a fairly important relation, many solvers run out of memory instead of timing out and proof space complexity gives us some theoretical framework for understanding how to prevent this.

Space can be measured in different ways. A configurational proof is a sequence of memory configurations $\mathcal{M}_i$ that contain clauses. A configurational proof is subject to the following conditions:

- $\mathcal{M}_0 = \varnothing$
- $\mathcal{M}_{i+1} = \mathcal{M}_i \cup \{C\}$ where C is an axiom, or
- $\mathcal{M}_{i+1} = \mathcal{M}_i \cup \{C\}$ where C is obtainable from an inference on clauses in $\mathcal{M}_i$, or
- $\mathcal{M}_{i+1} = \mathcal{M}_i \backslash \{C\}$.

The Clause Space ($CSpace$) for configurational proof π is given by

$$CSpace(\pi) = \max(|\mathcal{M}_i| \mid \mathcal{M}_i \in \pi).$$

The Clause Space for a resolution for a formula F and proof system P is

$$CSpace\left(\vdash_{\overline{P}} F\right) = \min(CSpace(\pi) | \pi \text{ is a proof in system } P).$$

The Total Space ($TSpace$) for configurational proof π is given by

$$TSpace(\pi) = \max(\sum_{C \in M_i} w(C) \mid \mathcal{M}_i \in \pi).$$

The Total Space for a resolution for a formula F and proof system P is

$$TSpace\left(\vdash_{\mathsf{P}} F\right) = \min(TSpace(\pi) | \pi \text{ is a proof in system } P).$$

Relations between space, size, and width are also explored (cf. also [31, 84]), establishing the size-space relation for tree-like resolution, notice here we are using clause space:

Theorem 38 (Esteban, Torán [55]). *For all unsatisfiable CNFs F the following relation holds:* $S(\vdash_{\mathsf{Res_T}} F) \geq 2^{CSpace\left(\vdash_{\mathsf{Res_T}} F\right)} - 1$.

The fundamental relation between space and width for full resolution was obtained in [6]; a more direct proof was given recently in [56].

Theorem 39 (Atserias, Dalmau [6]). *For all unsatisfiable CNFs F the following relation holds:* $w(\vdash_{\mathsf{Res}} F) \leq CSpace(\vdash_{\mathsf{Res}} F) + w(F) - 1$.

There have also been improvements that allow for relations to total space.

Theorem 40 (Bonacina [35]). *For all unsatisfiable k-CNFs F the following relation holds:* $TSpace(\vdash_{\mathsf{Res}} F) \geq \frac{1}{16}(w(\vdash_{\mathsf{Res}} F) - k - 4)^2$.

9.2 Width versus Existential Width

While width treats all variables equally in propositional logic, the notion of width is more complicated in QBF, due to variables belonging to either universal or existential quantifiers. It was shown in [22] that this notion breaks down any relation between width and space or size.

Proposition 2 (Beyersdorff, Chew, Mahajan, Shukla [22]).
For the false QBFs $\mathcal{F}_n = \forall u_1 \dots u_n \exists e_0 \exists e_1 \dots e_n.(e_0) \wedge \bigwedge_{i \in [n]} (\neg e_{i-1} \vee u_i \vee e_i) \wedge (\neg e_n)$ we have $S(\vdash_{\mathsf{Q\text{-}Res_T}} \mathcal{F}_n) = O(n)$ and $CSpace(\vdash_{\mathsf{Q\text{-}Res_T}} \mathcal{F}_n) = O(1)$, but $w(\vdash_{\mathsf{Q\text{-}Res}} \mathcal{F}_n) = \Omega(n)$.

The width lower bound itself comes from the accumulation of universal literals which cannot be removed until the last step do to the restrictions on $\forall$-Red, notice that even if universal resolution is allowed (as in QU-Res) then everything above still applies because no universal resolution rule cannot be used.

In order to investigate a more robust version of width in QBF it was decided [22] that existential width should be investigated, this counts the number of existential literals in a clause C and is denoted by $w_\exists(C)$. This definition also brings CDCL-based QBF calculi like Q-Res in line with expansion-based QBF calculi like IR-calc, where the only literals present are existential.

9.3 Negative Results: Size-Width, Space-Width Relations Fail in Q-Res

Proposition 3 (Beyersdorff, Chew, Mahajan, Shukla [22]). *There are false sentences ψ_n, with an existential literal b quantified at the innermost level, such that the sentence $\psi_n|_{b=1}$ is false and has a small existential-width proof, but ψ_n itself needs large existential width to refute in Q-Res.*

These ψ_n formulas are based off of formulas from Bonet and Galesi [37] that we denote as BG_n. BG_n is an unsatisfiable CNF over $O(n^2)$ variables with $w(BG_n) = 3$ and $w(\vdash BG_n) = \Omega(n)$. ψ_n is given as $\exists \boldsymbol{x} \exists a \forall u \exists b.\, (a \vee u \vee \neg b) \wedge (\neg a) \wedge BG_n(\boldsymbol{x})$.

The significance of this proposition is that it is a counterexample to a property of resolution that was essential in the propositional DAG-like case of the size-width relation.

Theorem 41 (Beyersdorff, Chew, Mahajan, Shukla [22]). *There exist false QBFs CR'_n over $O(n^2)$ variables, such that $S(\vdash_{\text{Q-Res}_T} CR'_n) = n^{O(1)}$, $w_\exists(CR'_n) = 3$, $CSpace(\vdash_{\text{Q-Res}_T} CR'_n) = O(1)$, and $w_\exists(\vdash_{\text{Q-Res}_T} CR'_n) = \Omega(n)$.*

The formulas CR'_n are Tseitin transformations of a natural completion principle formula CR_n from [74]. Since tree-like space is at least as large as space, Theorem 41 also rules out the space-width relation for general dag-like Q-Res proofs.

However, Theorem 41 cannot be used to show that the size-existential-width relationship for general dag-like proofs fails in Q-Res, because CR'_n have $O(n^2)$ variables.

Instead consider the following formulas ϕ_n, introduced by Janota and Marques-Silva [74]: $\exists e_1 \forall u_1 \exists c_1 c_2 \ldots \exists e_n \forall u_n \exists c_{2n-1} c_{2n}$.

$$\bigwedge_{i \in [n]} \big((\neg e_i \vee c_{2i-1}) \wedge (\neg u_i \vee c_{2i-1}) \wedge (e_i \vee c_{2i}) \wedge (u_i \vee c_{2i})\big) \wedge \bigvee_{i \in [2n]} \neg c_i.$$

These were originally used to show that ∀Exp+Res does not simulate Q-Res; they have short proofs in Q-Res but are hard for ∀Exp+Res and thus Q-Res$_\forall$.

These formulas were used to disprove the size-width relation for Q-Res, but with a modification to make the axiom clauses have constant width. Let ϕ'_n denote the modified formula:

$$\begin{aligned}\phi'_n = {} & \exists e_1 \forall u_1 \exists c_1 c_2 \ldots \exists e_n \forall u_n \exists c_{2n-1} c_{2n} \exists x_0 \ldots x_{2n} \\ & \bigwedge_{i \in [n]} \big((\neg e_i \vee c_{2i-1}) \wedge (\neg u_i \vee c_{2i-1}) \wedge (e_i \vee c_{2i}) \wedge (u_i \vee c_{2i})\big) \wedge \qquad (1) \\ & \neg x_0 \wedge \bigwedge_{i \in [2n]} (x_{i-1} \vee \neg c_i \vee \neg x_i) \wedge x_{2n}. \qquad (2)\end{aligned}$$

Theorem 42 (Beyersdorff, Chew, Mahajan, Shukla [22]). *There is a family of false QBFs ϕ'_n in $O(n)$ variables such that $S(\vdash_{\text{Q-Res}} \phi'_n) = n^{O(1)}$, $w_\exists(\phi'_n) = 3$, and $w_\exists(\vdash_{\text{Q-Res}} \phi'_n) = \Omega(n)$.*

These results demonstrate that the size-width relation fails in Q-Res and the same result can also be applied to QU-Res. This is of great importance as it means that our investigation into proof

complexity in QBF calculi does not have the main techniques from propositional logic. This also highlights the importance of the other techniques that have shown to work in the QBF setting.

9.4 Simulations: Preserving Size, Width, and Space Across Calculi

While it has been shown that strong negative results occur for dag-like Q-Res, we will later see that positive results hold for tree-like calculi. While we have seen investigations into tree-like Q-Res. Other tree-like calculi such as tree-like versions of expansion-based calculi have not been discussed. The most important observation is the following:

Lemma 27 (Beyersdorff, Chew, Mahajan, Shukla [22]). *$\forall$Exp+Res$_T$ p-simulates IR$_T$-calc while preserving width, size, and space.*

Lemma 28 (Beyersdorff, Chew, Mahajan, Shukla [22]). *IR$_T$-calc p-simulates Q-Res$_T$ while preserving space and existential width exactly and size up to a factor of 3.*

Proof (Proof Sketch). This uses the same simulation as given in [18]. This simulation was originally for dag-like proof systems, but here it is checked that it works for tree-like systems.

As a by-product, these simulations enable us to give an easy and elementary proof of the simulation of Q-Res$_\mathsf{T}$ by $\forall$Exp+Res, shown in [74] via a more involved argument.

Corollary 11 (Janota, Marques-Silva [74]). *$\forall$Exp+Res$_T$ p-simulates Q-Res$_T$.*

Using the previous lemmas, the following were shown:

Theorem 43 (Beyersdorff, Chew, Mahajan, Shukla [22]). *For all false QBFs $\mathcal{F}$, the following relations hold:*

1. $\frac{1}{2}S(\vdash_{\text{IR}_T\text{-calc}} \mathcal{F}) \leq S\left(\vdash_{\forall\text{Exp+Res}_T} \mathcal{F}\right) \leq S(\vdash_{\text{IR}_T\text{-calc}} \mathcal{F}) \leq 3S(\vdash_{\text{Q-Res}_T} \mathcal{F})$.
2. $w(\vdash_{\text{IR-calc}} \mathcal{F}) = w(\vdash_{\forall\text{Exp+Res}} \mathcal{F}) \leq w_\exists(\vdash_{\text{Q-Res}} \mathcal{F})$.
3. $CSpace(\vdash_{\forall\text{Exp+Res}_T} \mathcal{F}) = CSpace(\vdash_{\text{IR}_T\text{-calc}} \mathcal{F}) \leq CSpace(\vdash_{\text{Q-Res}_T} \mathcal{F})$.

9.5 Positive Results: Size, Width, and Space in Tree-like QBF Calculi

We first observe that for $\forall$Exp+Res almost the full spectrum of relations from classical resolution remain valid.

Theorem 44 (Beyersdorff, Chew, Mahajan, Shukla [22]). *For all false QBFs $\mathcal{F}$, the following relations hold:*

1. $S\left(\vdash_{\forall\text{Exp+Res}_T} \mathcal{F}\right) \geq 2^{w\left(\vdash_{\forall\text{Exp+Res}} \mathcal{F}\right) - w_\exists(\mathcal{F})}$.
2. $S\left(\vdash_{\forall\text{Exp+Res}_T} \mathcal{F}\right) \geq 2^{CSpace\left(\vdash_{\forall\text{Exp+Res}_T} \mathcal{F}\right)} - 1$.
3. $CSpace\left(\vdash_{\forall\text{Exp+Res}_T} \mathcal{F}\right) \geq CSpace(\vdash_{\forall\text{Exp+Res}} \mathcal{F}) \geq w(\vdash_{\forall\text{Exp+Res}} \mathcal{F}) - w_\exists(\mathcal{F}) + 1$.

This comes quite easily, ∀Exp+Res can be seen as having two parts. A formula is expanded in the axiom cases, and then resolution is performed. These two parts can be separated, which means the resolution part performs exactly as it did in the propositional case, leading to all the previous results.

Since $\forall\mathsf{Exp+Res}_\mathsf{T}$ and IR_T-calc are simultaneously equivalent with respect to size, width, and space (Theorem 43), all results from Theorem 44 transfer to IR_T-calc.

Theorem 45 (Beyersdorff, Chew, Mahajan, Shukla [22]). *For all false QBFs $\mathcal{F}$, the following relations hold:*

1. $S(\vdash_{\mathsf{IR}_\mathsf{T}\text{-calc}} \mathcal{F}) \geq 2^{w\left(\vdash_{\mathsf{IR-calc}} \mathcal{F}\right) - w_\exists(\mathcal{F})}$.
2. $S(\vdash_{\mathsf{IR}_\mathsf{T}\text{-calc}} \mathcal{F}) \geq 2^{CSpace\left(\vdash_{\mathsf{IR}_\mathsf{T}\text{-calc}} \mathcal{F}\right)} - 1$.
3. $CSpace(\vdash_{\mathsf{IR}_\mathsf{T}\text{-calc}} \mathcal{F}) \geq w(\vdash_{\mathsf{IR-calc}} \mathcal{F}) - w_\exists(\mathcal{F}) + 1$.

And finally returning to Q-Res a positive result exists there as well.

Theorem 46 (Beyersdorff, Chew, Mahajan, Shukla [22]). *For a false QBF sentence $\mathcal{F}$,* $S(\vdash_{\mathsf{Q\text{-}Res}_\mathsf{T}} \mathcal{F}) \geq 2^{CSpace\left(\vdash_{\mathsf{Q\text{-}Res}_\mathsf{T}} \mathcal{F}\right)} - 1$.

In summary it has been shown that the size-width relation does work as a lower bound technique when using existential width but only for tree-like calculi. The reason the relation fails in DAG-like Q-resolution can be summarised in the findings of Proposition 3, that the proof width is sensitive to the variable restrictions.

With the importance of the width in propositional lower bounds. The failure perhaps explains some of the difficulty in QBF proof complexity. We continue to look at QBF resolution systems in Chapter 10, looking at applications to the broader DQBF problem.

Chapter 10

Lifting QBF Resolution to DQBF

The logic of dependency quantified Boolean formulas (DQBF) [97] generalises the notion of quantified Boolean formulas that allow Boolean quantifiers over a propositional problem. DQBF is a relaxation of QBF in that the quantifier order is no longer necessarily linear and the dependencies of the quantifiers are completely specified. This is achieved using Henkin quantifiers [68], usually put into a Skolem form. DQBF is NEXPTIME-complete [7], compared to the PSPACE-completeness of QBF [113]. Thus, unless the classes are equal, many problems that are difficult to express in QBF can be succinctly represented in DQBF.

The aim of this chapter is to clarify which of the QBF resolution systems can be lifted to DQBF. This is motivated both by the theoretical quest to understand which QBF resolution paradigms are robust enough to work in the more powerful DQBF setting, as well as from the practical perspective, where recent advances in DQBF solving [57, 58, 62, 117] prompt the question of how to model and analyse these solvers proof-theoretically.

Our starting point is the work of Balabanov, Chiang, and Jiang [9], who show that Q-Res can be naturally adapted to a sound calculus for DQBF, but it is not strong enough and lacks completeness. Using an idea from [11] we extend their result to QU-Res, thus showing that the lifted version of this system to DQBF is not complete either. On the other hand, we present an example showing that the lifted version of LD-Q-Res is not sound, and this transfers to the DQBF analogues of the stronger systems LQU$^+$-Res and IRM-calc.

While this rules out most of the existing QBF resolution calculi already—and in fact all CDCL-based systems (cf. Fig. 22, Chapter 5)— we show that IR-calc, lifted in a natural way to a DQBF calculus D-IR-calc, is indeed sound and complete for DQBF; and in fact this holds as well for the lifted version of the weaker expansion system ∀Exp+Res.

Conceptually, our soundness and completeness arguments use the known correspondence of QBF and DQBF to first-order logic [109], and in particular to the fragment of EPR, also known as the Bernays-Schönfinkel class, which like DQBF is NEXPTIME-complete [86]. In addition to providing soundness and completeness this explains the semantics of both IR-calc and D-IR-calc and identifies these systems as a special case of first-order resolution.

We give a definition of DQBF in Section 10.1. In Section 10.2, we show that QU-Res, LD-Q-Res, LQU$^+$-Res, IRM-calc all fail to be lifted up to DQBF, either via unsoundness or incompleteness. However, in Section 10.3 we show that IR-calc lifted up to DQBF becomes the sound and complete calculus D-IR-calc.

The work in this chapter appeared at the 19th Theory and Applications of Satisfiability Testing International Conference [24].

10.1 Dependency QBF

A *Dependency Quantified Boolean Formula (DQBF)* ϕ in prenex Skolem form consists of a quantifier prefix Π and a propositional matrix ψ. Here we mainly study DQBFs where ψ is in CNF. The propositional variables of ψ are partitioned into sets Y and X. Y is the set of universal variables and X the set of existential variables. For every $y \in Y$, Π contains the quantifier $\forall y$. For every $x \in X$ there is a predefined subset $Y_x \subseteq Y$ and Π contains the quantifier $\exists x(Y_x)$.

The semantics of DQBF are defined in terms of Skolem functions. A Skolem function $f_x : \{0,1\}^{Y_x} \to \{0,1\}$ describes the evaluation of an existential variable x under the possible assignments to its dependencies Y_x. Given a set $F = \{f_x \mid x \in X\}$ of Skolem functions for all the existential variables and an assignment $\alpha : Y \to \{0,1\}$ for the universal variables, the *extension* of α by F is defined as $\alpha_F(x) = f_x(\alpha \restriction Y_x)$ for $x \in X$ and $\alpha_F(y) = \alpha(y)$ for $y \in Y$. A DQBF ϕ is *true* if there exist Skolem functions $F = \{f_x \mid x \in X\}$ for the existential variables such that for every assignment $\alpha : Y \to \{0,1\}$ to the universal variables the matrix ψ propositionally evaluates to 1 under the extension α_F of α by F.

QBF is a special case of DQBF. To see this, we use the sequence from left to right to assign to every variable in the prefix a unique index $\mathrm{ind} : X \cup Y \to \mathbb{N}$, and make every existential variable x depend on all the preceding universal variables by setting $Y_x = \{y \in Y \mid \mathrm{ind}(y) < \mathrm{ind}(x)\}$.

10.2 Problems with Lifting QBF Calculi to DQBF

There is no unique method for lifting calculi from QBF to DQBF. However, we can consider 'natural' generalisations of these calculi, where we interpret index conditions as dependency conditions. This means that when a proof system requires for an existential variable x and a universal variable y with $\mathrm{ind}(y) < \mathrm{ind}(x)$, this should be interpreted as $y \in Y_x$. Analogously $\mathrm{ind}(x) < \mathrm{ind}(y)$ should be interpreted as $y \notin Y_x$. This approach was followed when taking Q-Resolution to D-Q-Resolution in [9]. Balabanov et al. showed there that D-Q-Resolution is not complete for DQBF using the following formula.

$$\forall x_1 \forall x_2 \exists y_1(x_1) \exists y_2(x_2) \tag{3}$$

$$\begin{array}{ll} \{y_1, y_2, x_1\} & \{\neg y_1, \neg y_2, x_1\} \\ \{y_1, y_2, \neg x_1, \neg x_2\} & \{\neg y_1, \neg y_2, \neg x_1, \neg x_2\} \\ \{y_1, \neg y_2, \neg x_1, x_2\} & \{\neg y_1, y_2, \neg x_1, x_2\}. \end{array}$$

These are easily shown to be false, but no steps are possible in D-Q-Resolution, hence D-Q-Resolution is not complete [9]. Consider now the following modification of this formula where the universal variables are doubled:

$$\forall x_1 \forall x_1' \forall x_2 \forall x_2' \exists y_1(x_1, x_1') \exists y_2(x_2, x_2') \tag{4}$$

$$\{y_1, y_2, x_1, x_1'\} \qquad \{\neg y_1, \neg y_2, x_1, x_1'\}$$
$$\{y_1, y_2, \neg x_1, \neg x_1', \neg x_2, \neg x_2'\} \quad \{\neg y_1, \neg y_2, \neg x_1, \neg x_1', \neg x_2, \neg x_2'\}$$
$$\{y_1, \neg y_2, \neg x_1, \neg x_1', x_2, x_2'\} \qquad \{\neg y_1, y_2, \neg x_1, \neg x_1', x_2, x_2'\}.$$

The falsity of Equation 4 follows from the fact that its hypothetical Skolem model would immediately yield a Skolem model for Equation 3 using assignments with $x_1 = x_1'$, $x_2 = x_2'$. But there is no such model because Equation 3 is false. However, since we have doubled the universal literals we cannot perform any generalised QU-Res steps to begin a refutation. This technique of doubling literals was first used in [11].

Now we look at another portion of the calculi from Fig. 22, namely the calculi that utilise merging. As a specific example we consider LD-Q-Res and show that it is not sound when lifted to DQBF in the natural way.

To do this we look at the condition of (L∃R) given in Fig. 10. Here instead of requiring $\mathrm{ind}(x) < \mathrm{ind}(u)$ as a condition for u becoming merged, we require $u \notin Y_x$. This is unsound as we show by the following DQBF:

$$\forall u \forall v \exists x(u) \exists y(v) \exists z(u, v)$$

$$\{x, v, z\} \qquad \{\neg x, \neg v, z\}$$
$$\{y, u, \neg z\} \qquad \{\neg y, \neg u, \neg z\}.$$

Its truth is witnessed by the Skolem functions $x(u) = u$, $y(v) = \neg v$, and $z(u, v) = (u \wedge v) \vee (\neg u \wedge \neg v)$. However, the lifted version of LD-Q-Res admits a refutation:

$$\dfrac{\dfrac{\dfrac{\dfrac{\{x, v, z\} \quad \{\neg x, \neg v, z\}}{\{v^*, z\}} \quad \dfrac{\{y, u, \neg z\} \quad \{\neg y, \neg u, \neg z\}}{\{u^*, \neg z\}}}{\{u^*, v^*\}}}{\{u^*\}}}{\bot}$$

This shows that LD-Q-Res is unsound for DQBF. Likewise, since IRM-calc, LQU-Res and LQU$^+$-Res line-wise simulate LD-Q-Res, this proof would also be available, showing that these are all unsound calculi in the DQBF setting.

10.3 A Sound and Complete Proof System for DQBF

In this section we introduce the D-IR-calc refutation system and prove its soundness and completeness for DQBF. The calculus takes inspiration from IR-calc which we introduce in Chapter 4, which in turn is inspired by first-order translations of QBF. One such translation is to the EPR fragment, i.e., the universal fragment of first-order logic without function symbols of non-zero arity (we only allow constants). We broaden this translation to DQBF and then bring this back down to D-IR-calc in a similar way as in IR-calc.

We adapt annotated literals l^τ to DQBF, such that l is an existential literal and τ is an annotation which is a partial assignment to universal variables in Y_x. In QBF, Y_x contains all universal variables with an index lower than x, and this is exactly the maximal range of the potential annotation to x literals. Thus our definition of annotated literals generalises those used in IR-calc.

Definition 16. *Fix a DQBF $\Pi.\psi$. Let τ be a partial assignment of the universal variables Y to $\{0,1\}$ and let x be an existential variable.* $\mathsf{restrict}_x(\tau)$ *is the assignment where* $\mathsf{dom}(\mathsf{restrict}_x(\tau)) = \mathsf{dom}(\tau) \cap Y_x$ *and* $\mathsf{restrict}_x(\tau)(u) = \tau(u)$.

Definition 17. *We define* $\mathsf{inst}_\tau(C)$ *to be the clause containing all the literals* $l^{\mathsf{restrict}_{\mathrm{var}(l)}(\sigma)}$*, where* $l^\xi \in C$ *and* $\mathsf{dom}(\sigma) = \mathsf{dom}(\xi) \cup \mathsf{dom}(\tau)$ *and* $\sigma(u) = \xi(u)$ *if* $u \in \mathsf{dom}(\xi)$ *and* $\sigma(u) = \tau(u)$ *otherwise.*

With these definitions at hand we can now define the new calculus D-IR-calc. Its rules are exactly the same as the ones for IR-calc stated in Fig. 14, but with the relaxed notions of restriction and instantiation as in Definitions 16 and 17 above.

Before analysing D-IR-calc further we present the translation of DQBF into EPR. We use an adaptation of the translation described for QBF [109], which becomes especially straightforward in the light of the DQBF semantics based on Skolem functions. The key observation is that for the intended two valued Boolean domain the Skolem functions can be represented by predicates.

To translate a DQBF $\Pi \cdot \psi$ we introduce on the first-order side 1) a predicate symbol p of arity one and two constant symbols 0 and 1 to describe the Boolean domain, 2) for every existential variable $x \in X$ a predicate symbol p_x of arity $|Y_x|$, and 3) for every universal variable $y \in Y$ a first-order variable u_y.

Now we can define a translation mapping t_Π. It translates each occurrence of an existential variable x with dependencies $Y_x = \{y_1, \ldots, y_k\}$ to the atom $t_\Pi(x) = p_x(u_{y_1}, \ldots, u_{y_k})$ (we assume an arbitrary but fixed order on the dependencies which dictates their placement as arguments) and each occurrence of a universal variable y to the atom $t_\Pi(y) = p(u_y)$. The mapping is then homomorphically extended to formulas in the obvious way, i.e. $t_\Pi(\neg\psi) = \neg t_\Pi(\psi)$, $t_\Pi(\psi_1 \wedge \psi_2) = t_\Pi(\psi_1) \wedge t_\Pi(\psi_2)$, etc. The mapping satisfies the following.

Lemma 29. *A DQBF $\Pi \cdot \psi$ is true if and only if $t_\Pi(\psi) \wedge p(1) \wedge \neg p(0)$ is satisfiable.*

For the purpose of analysing D-IR-calc, the mapping is further extended to annotated literals. Given an annotation τ, we define a first-order substitution $\bar{\tau} = \{u_y \mapsto \tau(y) \mid y \in \mathsf{dom}(\tau)\}$. Thus $\bar{\tau}$ acts on u_y in the same way as τ does on y. By a standard convention, it acts as the identity mapping elsewhere. The extension of t_Π to annotated literals now additionally applies the substitution $\bar{\tau}$ to the original result: $t_\Pi(x^\tau) = t_\Pi(x)\bar{\tau}$ for an existential variable x.

First-order resolution. We aim to show soundness and completeness of D-IR-calc by relating it via the above translation to a first-order resolution calculus FO-res. This calculus consists of 1) a lazy instantiation rule: given a clause C and a substitution σ derive the instance $C\sigma$, and 2) the resolution rule: given two clauses $C \cup \{L\}$ and $D \cup \{\neg L\}$, where L is a first-order literal, derive the resolvent $C \cup D$. Note that similarly to propositional clauses, we understand first-order clauses as *sets* of first-order literals. Thus we do not need any explicit factoring rule. Also note that we require the pivot literals of the two premises of the resolution rule to be equal (up to the polarity).

Standard first-order resolution, which involves *unification* on the pivot, can be simulated in FO-res by combining the instantiation and the resolution rule.

It is clear that FO-res is sound and complete for first-order logic.

Soundness. Our argument for the soundness of D-IR-calc is the following. Given $\pi = (L_1, L_2, \ldots, L_\ell)$, a D-IR-calc derivation of the empty clause $L_\ell = \bot$ from DQBF $\Pi \cdot \psi$, we show by induction that $t_\Pi(L_n)$ is derivable from $\Psi = t_\Pi(\psi) \wedge p(1) \wedge \neg p(0)$ by FO-res for every $n \leq \ell$. Because $t_\Pi(\bot) = \bot$ is unsatisfiable, so must Ψ be by soundness of FO-res and therefore $\Pi \cdot \psi$ is false by Lemma 29.

We need to consider the three cases by which a clause is derived in D-IR-calc. First, it is easy to verify that D-IR-calc instantiation by an annotation τ corresponds to FO-res instantiation by the substitution $\bar{\tau}$, i.e. $t_\Pi(\mathsf{inst}_\tau(C)) = t_\Pi(C)\bar{\tau}$. Also the D-IR-calc and FO-res resolution rules correspond one to one in an obvious way. Thus the most interesting case concerns the Axiom rule.

Intuitively, the Axiom rule of D-IR-calc removes universal variables from a clause while recording their past presence (and polarity) within the applied annotation τ. We simulate this step in FO-res by first instantiating the translated clause by $\bar{\tau}$ and then resolving the obtained clause with the unit $p(1)$ and/or $\neg p(0)$. Here is an example for a DQBF prefix $\Pi = \forall u \forall v \forall w \exists x(u, v) \exists y(v, w)$:

$$\text{(D-IR-calc)}\ \frac{\{x, y, \neg u, v\}}{\{x^{1/u,0/v}, y^{0/v}\}}\ \text{(FO-res)}\ \frac{\{p_x(u_u, u_v), p_y(u_v, u_w), \neg p(u_u), p(u_v)\}}{\{p_x(1, 0), p_y(0, u_w)\}}$$

Theorem 47. *D-IR-calc is sound.*

Completeness. Let $\Pi \cdot \psi$ be a false DQBF and let us consider $\mathcal{G}(t_\Pi(\psi))$, the set of all ground instances of clauses in $t_\Pi(\psi)$. Here, by a ground instance of a clause C we mean the clause $C\sigma$ for some substitution $\sigma : \mathrm{var}(C) \to \{0, 1\}$. We use an important result on ground instances.

Theorem 48 (Herbrand's theorem [39]). *A first order formula $\forall x_1 \ldots x_n F(x_1 \ldots x_n)$ with $F(x_1 \ldots x_n)$ containing no quantifiers is unsatisfiable if and only if there is a finite set of ground instances whose conjunction is unsatisfiable i.e. t_{ij} for $1 \leq i \leq n$, $1 \leq j \leq m$ such that $\bigwedge_{1 \leq j \leq m} F(t_{1j} \ldots t_{nj})$ unsatisfiable in propositional logic.*

By the combination of Lemma 29 and Herbrand's theorem, $\mathcal{G}(t_\Pi(\psi)) \wedge p(1) \wedge \neg p(0)$ is unsatisfiable and thus it has a FO-res refutation. Moreover, by completeness of *ordered* resolution [8], we can assume that 1) the refutation does not contain clauses subsumed by $p(1)$ or $\neg p(0)$, and 2) any clause containing the predicate p is resolved on a literal containing p. From this it is easy to see that any leaf in the refutation gives rise (in zero, one or two resolution steps with $p(1)$ or $\neg p(0)$) to a clause $D = t_\Pi(C)$ where C can be obtained by D-IR-calc Axiom from a $C_0 \in \psi$. The rest of the refutation consists of FO-res resolution steps which can be simulated by D-IR-calc. Thus we obtain the following.

Theorem 49. *D-IR-calc is refutationally complete for DQBF.*

Although one can lift the above argument with ordered resolution to show that the set $\{t_{\Pi}(C) \mid C$ follows by Axiom from some $C_0 \in \psi\}$ is unsatisfiable for any false DQBF $\Pi \cdot \psi$, we only know how to simulate *ground* FO-res steps by D-IR-calc. That is because a lifted FO-res derivation may contain instantiation steps which *rename variables apart* for which a subsequent resolvent cannot be represented in D-IR-calc. An example is the resolvent $\{p_y(u_v), p_z(u'_v)\}$ of clauses $\{p_x(u_u), p_y(u_v)\}$ and $\{\neg p_x(u_u), p_z(u'_v)\}$ which is obviously weaker than the clause $\{p_y(u_v), p_z(u_v)\}$. However, only the latter has a counterpart in D-IR-calc.

We also remark that in a similar way we can also lift to DQBF the QBF calculus ∀Exp+Res from [74]. It is easily verified that the simulation of ∀Exp+Res by IR-calc shown in Section 4 directly transfers from QBF to DQBF. Hence Theorem 47 immediately implies the soundness of ∀Exp+Res lifted to DQBF. Moreover, because all ground instances are also available in ∀Exp+Res lifted to DQBF, this system is also complete as can be shown by repeating the argument of Theorem 49.

In this chapter we look at how resolution can be implemented beyond QBF. In the remaining chapters we remain in QBF but look beyond the resolution calculus.

Part III

QBF Proof Systems Beyond Resolution

Chapter 11

Extension Variables for QBF Resolution

Using extension variables to abbreviate possibly complex formulas is a well known and powerful concept in proof complexity and solving. In Tseitin transformations, extension variables are used to encode arbitrary propositional formulas in CNF. More generally, allowing the extension rule in proofs is known to shorten proof size drastically for many examples. This makes extension variables also very interesting in the context of solving, and indeed modern proof checking formats such as RAT for SAT-solvers [69] and QRAT for QBF solvers [70] incorporate the use of extension variables.

When augmenting the classical resolution system with the extension rule, allowing it to introduce a new variable v to abbreviate a disjunction $\neg x \vee \neg y$, we arrive at *extended resolution* [115], cf. also [85]. Although resolution itself is considered a weak proof system with many known lower bounds (cf. [108]), extended resolution is an extremely powerful system. Extended resolution is equivalent to extended Frege [45, 82], one of the strongest proof systems considered today. Showing any non-trivial lower bounds for extended Frege constitutes an extremely challenging problem with even hard candidate formulas currently lacking.

In QBF solving and proof complexity, using extension variables was first considered by Jussila et. al. [75], where they augment Q-resolution to an extended Q-resolution system. In comparison to the propositional case, extension variables in QBF present one additional challenge. We cite the relevant passage from [75]:

"In adapting the extension rule to the QBF setting, the crucial question is where to 'put' new variables, e.g., how defined variables are ordered with respect to variables that already occur in the formula. It seems intuitive that new variables can only be existential. It is also clear that they cannot be moved further out than the innermost variable on which they depend. In the experimental section we show that we actually need this freedom to move defined variables as far out as possible. Keeping them in the innermost existential scope, as for instance in the Tseitin encoding of a non-CNF QBF formula, is insufficient."

In this section, we explore this experimental observation with a rigorous theoretical argument. For this we consider two systems: weak extended Q-resolution, where extension variables are quantified at the innermost level, and extended Q-resolution, where extension variables are quantified immediately after the innermost variable on which they depend. Our main result is an exponential separation between these two versions. We show that QPARITY formulas (Chapter 5) have short proofs in extended Q-resolution, but require exponential-size proofs in weak extended Q-resolution.

The lower bound uses the strategy extraction technique (Chapter 5). Here we show that weak extended Q-resolution admits strategy extraction in AC^0, i.e., from each refutation of a false QBF we can extract a winning strategy for the universal player that can be computed by constant-depth

circuits. This allows to transfer Håstad's circuit lower bound for parity [67] to a proof size lower bound for the QPARITY formulas in weak extended Q-resolution.

We divide this chapter as follows; Section 11.1 defines the two ways extension variables can be added to Q-resolution, Section 11.2 shows that short proofs for QPARITY exist in extended Q-resolution, however we see in Section 11.3, via a strategy extraction argument, that short proofs do not exist in weak extended Q-resolution.

The work in this chapter appeared at the AAAI Workshop: Beyond NP 2016 [20].

11.1 Two Versions of Extended Q-resolution

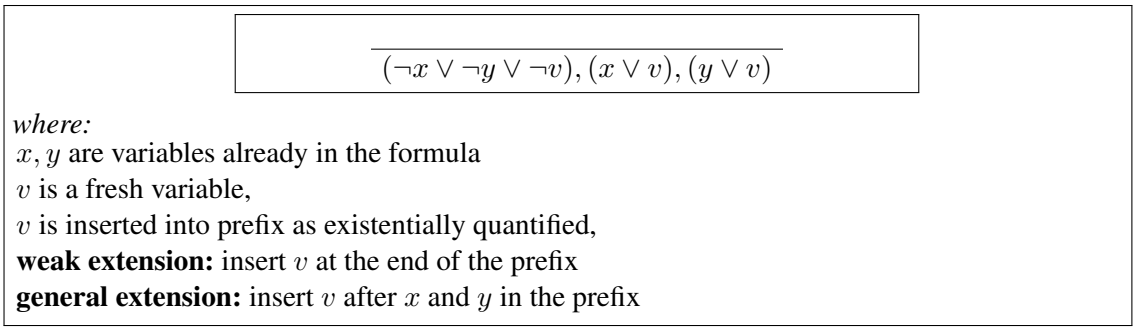


Fig. 25. Two versions of *extension rule*

Extended resolution for propositional resolution [115], enables adding clauses expressing the equality $v \Leftrightarrow (\neg x \vee \neg y)$, for a fresh variable v. We follow this idea in the context of Q-resolution. Here, we need to decide the position of the fresh variable in the prefix. Two versions are considered; a weak one and a general one.

Figure 25 defines the two forms of the extension rule, which gives us two flavours of extended Q-resolution.

Definition 18. Weak extended Q-resolution *is the calculus of Q-resolution enhanced with the extension rule in its weak form.*

Definition 19. Extended Q-resolution *is the calculus of Q-resolution enhanced with the extension rule in its general form.*

Extended resolution and circuits. Semantically, introducing a fresh variable via the extension rule, corresponds to defining $v = x \,\mathsf{nand}\, y$. This enables expressing any Boolean functions in a circuit form by a sequence of extension rules. Consider for instance $x \vee y$. Introduce fresh variables x^{n} and y^{n} to denote the negation of x and y, respectively, by setting $x^{\mathrm{n}} = x \,\mathsf{nand}\, x$. Subsequently, use those to define the final output o by setting $o = x^{\mathrm{n}} \,\mathsf{nand}\, y^{\mathrm{n}}$.

In the following text, whenever it is obvious that a formula is expressible in a circuit form, we omit the intermediate definitions. In particular, in proofs, we enable writing extension clauses where x and y may be literals rather than just variables.

For instance, the extension clauses $x \vee \neg y \vee v$, $\neg x \vee \neg v$, $y \vee \neg v$ are realized by the introduction of a variable x^n corresponding to the negation of x, introduction of v via the extension rule and finally replacing x^n with $\neg x$ by extra resolution steps.

11.2 Short Proofs for QPARITY in Extended Q-resolution

We show that QPARITY (Definition 6, Section 5) is easy for extended Q-resolution.

Theorem 50. *The formulas* QPARITY_n *have linear-size proofs in extended Q-resolution.*

Proof. We define extension variable $s_2 = (x_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2)$. In clausal form this introduces the following clauses:

$$s_2 \vee \bar{x}_1 \vee x_2 \quad s_2 \vee x_1 \vee \bar{x}_2 \quad \bar{s}_2 \vee x_1 \vee x_2 \quad \bar{s}_2 \vee \bar{x}_1 \vee \bar{x}_2$$

For $2 < i \leq n$, we define extension variables $s_i = s_{i-1} \oplus x_i = (s_{i-1} \vee x_i) \wedge (\bar{s}_{i-1} \vee \bar{x}_i)$. In clausal form this introduces the following clauses:

$$s_i \vee \bar{x}_i \vee s_{i-1} \quad s_i \vee x_i \vee \bar{s}_{i-1} \quad \bar{s}_i \vee x_i \vee s_{i-1} \quad \bar{s}_i \vee \bar{x}_i \vee \bar{s}_{i-1}$$

The main part of this short proof is to show that we can easily substitute s_i for t_i. We will show this by induction on i. The shortness comes from the fact that s_n literals are left of z. This allows us to reduce z literals "early" and get a contradiction.

Induction Claim: Clauses $s_i \vee \bar{t}_i$ and $t_i \vee \bar{s}_i$ (that show $s_i = t_i$) are provable in $O(i)$ size proofs.

Base Case: Let $i = 2$,

$$\dfrac{\dfrac{\dfrac{s_2 \vee \bar{x}_1 \vee x_2 \quad \bar{t}_2 \vee \bar{x}_1 \vee \bar{x}_2}{s_2 \vee \bar{t}_2 \vee \bar{x}_1} \quad \dfrac{s_2 \vee x_1 \vee \bar{x}_2 \quad \bar{t}_2 \vee x_1 \vee x_2}{s_2 \vee \bar{t}_2 \vee x_1}}{s_2 \vee \bar{t}_2}$$

$$\dfrac{\dfrac{\dfrac{t_2 \vee \bar{x}_1 \vee x_2 \quad \bar{s}_2 \vee \bar{x}_1 \vee \bar{x}_2}{t_2 \vee \bar{s}_2 \vee \bar{x}_1} \quad \dfrac{t_2 \vee x_1 \vee \bar{x}_2 \quad \bar{s}_2 \vee x_1 \vee x_2}{t_2 \vee \bar{s}_2 \vee x_1}}{t_2 \vee \bar{s}_2}$$

Inductive Step: We assume from the induction hypothesis that we have for some $i \leq n$, $s_{i-1} \vee \bar{t}_{i-1}$ and $t_{i-1} \vee \bar{s}_{i-1}$.

Firstly we use the clauses that express $s_{i-1} = t_{i-1}$ to substitute s_{i-1} for t_{i-1} .

$$\dfrac{\bar{t}_i \vee \bar{x}_i \vee \bar{t}_{i-1} \quad t_{i-1} \vee \bar{s}_{i-1} \quad t_i \vee x_i \vee \bar{t}_{i-1}}{\bar{t}_i \vee \bar{x}_i \vee \bar{s}_{i-1} \quad t_i \vee x_i \vee \bar{s}_{i-1}}$$

$$\dfrac{\bar{t}_i \vee x_i \vee t_{i-1} \quad s_{i-1} \vee \bar{t}_{i-1} \quad t_i \vee \bar{x}_i \vee t_{i-1}}{\bar{t}_i \vee x_i \vee s_{i-1} \quad t_i \vee \bar{x}_i \vee s_{i-1}}$$

We can use that the clauses that define s_i and t_i are structurally the same to derive $s_i = t_i$.

$$\dfrac{\dfrac{s_i \vee \bar{x}_i \vee s_{i-1} \quad \bar{t}_i \vee \bar{x}_i \vee \bar{s}_{i-1}}{s_i \vee \bar{t}_i \vee \bar{x}_i} \quad \dfrac{s_i \vee x_i \vee \bar{s}_{i-1} \quad \bar{t}_i \vee x_i \vee s_{i-1}}{s_i \vee \bar{t}_i \vee x_i}}{s_i \vee \bar{t}_i}$$

$$\dfrac{\dfrac{t_i \vee \bar{x}_i \vee s_{i-1} \quad \bar{s}_i \vee \bar{x}_i \vee \bar{s}_{i-1}}{t_i \vee \bar{s}_i \vee \bar{x}_i} \quad \dfrac{t_i \vee x_i \vee \bar{s}_{i-1} \quad \bar{s}_i \vee x_i \vee s_{i-1}}{t_i \vee \bar{s}_i \vee x_i}}{t_i \vee \bar{s}_i}$$

Since we only add a constant number of clauses, the induction argument allows us to keep a $O(i)$ size proof up until $s_n \vee \bar{t}_n$ and $t_n \vee \bar{s}_n$.

$$\dfrac{s_n \vee \bar{t}_n \quad z \vee t_n}{s_n \vee z} \qquad \dfrac{t_n \vee \bar{s}_n \quad \bar{z} \vee \bar{t}_n}{\bar{s}_n \vee \bar{z}}$$

Because s_n is defined only on x_i variables and so universal reduction is available

$$\dfrac{\dfrac{s_n \vee z}{s_n} \quad \dfrac{\bar{s}_n \vee \bar{z}}{\bar{s}_n}}{\bot}$$

This completes the short refutation in extended Q-resolution. □

11.3 Strategy Extraction and Hardness of QPARITY in Weak Extended Q-resolution

We now complement the upper bound from the previous section with a lower bound for the same formulas in weak extended Q-resolution. The lower bound argument rests on strategy extraction (see Chapter 5). The idea is to show that from refutations of false formulas it is possible to efficiently extract winning strategies for the universal player.

Balabanov and Jiang [10] showed that Q-resolution allows strategy extraction in decision lists. Here we show that the same remains true in weak extended Q-resolution. In comparison to [10] we provide a simplified proof.

Theorem 51. *Given a refutation π of QBF ϕ in weak extended Q-resolution, there exists a winning strategy for the universal player for ϕ such that for each universal variable u of ϕ the winning strategy can be represented as a Boolean function f_u that is expressible as a decision list whose size is polynomial in $|\pi|$.*

Proof. We first review the proof of the strategy extraction theorem for Q-resolution and then explain at the end how it applies to weak extended Q-resolution.

Let $\pi = (L_1, \ldots, L_\ell)$ be a resolution refutation of the false QBF $\mathcal{Q}\phi$ and let

$$\pi_i = \begin{cases} \emptyset & \text{if } i = \ell, \\ (L_{i+1}, \ldots, L_\ell) & \text{otherwise.} \end{cases}$$

We show, by reverse induction on i, that from π_i it is possible to construct in linear time (w.r.t. $|\pi_i|$) a winning strategy σ^i for the universal player for the QBF formula $\mathcal{Q}\phi_i$, where

$$\phi_i = \begin{cases} \phi & \text{if } i = 0, \\ \phi \wedge L_1 \wedge \cdots \wedge L_i & \text{otherwise,} \end{cases}$$

such that for each universal variable u in $\mathcal{Q}\phi$, there exists a decision list D^i_u computing σ^i_u as a function of the variables in $\mathcal{Q}$ left of u, having size $O(|\pi_i|)$.

The statement of the theorem corresponds to the case when $i = 0$. The base case of the induction is for $i = \ell$. In this case σ^ℓ is trivial since ϕ_ℓ contains the line $L_\ell = \bot$, and we can define all the D^ℓ_u as $u \leftarrow 0$.

We show now how to construct σ^{i-1}_u and D^{i-1}_u from σ^i_u and D^i_u:

- If L_i is derived by resolution, then for each universal variable u we set $\sigma^{i-1}_u = \sigma^i_u$ and $D^{i-1}_u = D^i_u$.
- Suppose L_i is the result of an application of a $\forall$-Red rule on clause L_j, that is $L_j = L_i \vee u^c$ with $c \in \{0, 1\}$, where u is the rightmost variable in L_j, and u^0 stands for $\neg u$ and u^1 for u. Let $\boldsymbol{x}_{u'}$ denote the variables on the left of u' in the quantifier prefix of $\mathrm{QBF}\,\mathcal{Q}\phi$. Then we define

$$\sigma^{i-1}_{u'}(\boldsymbol{x}_{u'}) = \begin{cases} \sigma^i_{u'}(\boldsymbol{x}_{u'}) & \text{if } u' \neq u, \\ 1 - c & \text{if } u' = u \text{ and } L_i(\boldsymbol{x}_u) = 0, \\ \sigma^i_u(\boldsymbol{x}_u) & \text{if } u' = u \text{ and } L_i(\boldsymbol{x}_u) = 1. \end{cases}$$

Moreover for each $u' \neq u$ we set $D^{i-1}_{u'} = D^i_{u'}$ and we set D^{i-1}_u as follows:

```
if ¬L_i(x_u) then u ← 1 − c;
else D^u_i(x_u).
```

We now check that for each u', $\sigma^{i-1}_{u'}$ respects all the properties of the inductive claim.

It is clear that $\sigma^{i-1}_{u'}$ and $D^{i-1}_{u'}$ are well defined and constructed in linear time w.r.t. $|\pi_{i-1}|$. Also, by construction $D^{i-1}_{u'}$ computes $\sigma^{i-1}_{u'}$.

To verify that σ^{i-1} is a winning strategy for $\mathcal{Q}\phi_{i-1}$ we fix an assignment ρ to the existential variables of ϕ. Let τ_i be the complete assignment to existential and universal variables, constructed in response to ρ under the strategy σ^i. By induction hypothesis τ_i falsifies ϕ_i. We need to show that τ_{i-1} falsifies ϕ_{i-1}. To show this we distinguish again two cases.

If L_i is derived by the resolution rule, then $\sigma^{i-1} = \sigma^i$ and $\tau_{i-1} = \tau_i$. Hence by induction hypothesis, τ_i falsifies a conjunct from ϕ_i. To argue that τ_{i-1} also falsifies a conjunct from ϕ_{i-1} we only need to look at the case when the falsified conjunct is L_i. As L_i is false under τ_i and L_i is derived by resolution, by soundness of the resolution rule one of the parent formulas of L_i in the application of the resolution rule must be falsified as well. Hence τ_{i-1} falsifies ϕ_{i-1}.

Let now L_i be derived by $\forall$-Red from $L_j = L_i \vee u^c$ for some $j < i$. In this case, our strategy σ^{i-1} changes the assignment τ_i only when τ_i made the universal player win by falsifying L_i. As

we set u to $1 - c$, the modified assignment τ_{i-1} falsifies L_j. Otherwise, if τ_i does not falsify L_i we keep $\tau_{i-1} = \tau_i$ and hence falsify one of the conjuncts of ϕ_{i-1} by induction hypothesis.

Let us now explain how the above applies to weak extended Q-resolution. Consider a refutation π of a QBF $\mathcal{Q}\boldsymbol{x}\phi(\boldsymbol{x})$ in weak extended Q-resolution. We can view π as a Q-resolution refutation of the QBF $\mathcal{Q}\boldsymbol{x}\exists\boldsymbol{y}\phi(\boldsymbol{x}) \wedge \psi(\boldsymbol{x}, \boldsymbol{y})$, where suitable extension variables $\boldsymbol{y}$ together with their definitions $\psi(\boldsymbol{x}, \boldsymbol{y})$ have been added.

We apply the argument above to construct decision lists computing a winning strategy for $\mathrm{QBF}\mathcal{Q}\boldsymbol{x}\exists\boldsymbol{y}\phi(\boldsymbol{x}) \wedge \psi(\boldsymbol{x}, \boldsymbol{y})$. By definition of the $\forall$-Red rule, no $\boldsymbol{y}$ variables are present in $\forall$-Red steps in π. Hence the decision list will also not use any of the extension variables and therefore in fact compute a winning strategy for the original formula $\mathrm{QBF}\mathcal{Q}\boldsymbol{x}\phi(\boldsymbol{x})$. □

This result enables us to show lower bounds for formulas that require hard strategies. An example of such formulas are the QPARITY formulas. As observed earlier, the only winning strategy of the universal player on the QPARITY formulas is to actually compute the PARITY function. However, PARITY is the classic example of a function hard for bounded-depth circuits (and hence by Lemma 5 for decision lists).

Using strategy extraction we can now immediately transfer this circuit lower bound to an exponential lower bound in weak extended Q-resolution.

Theorem 52. *Any refutation of* $\mathrm{QPARITY}_n$ *in weak extended Q-resolution is of exponential size and thus weak extended Q-resolution does not simulate extended Q-resolution.*

Proof. The unique winning strategy for the variable z in $\mathrm{QPARITY}_n$ is to compute $x_1 \oplus \cdots \oplus x_n$. By Theorem 51, there is a polynomial-time algorithm for constructing a decision list D_n from any refutation of $\mathrm{QPARITY}_n$ in weak extended Q-resolution. Such decision list can be converted in polynomial time into a depth-3 circuit by Lemma 5. Hence, the refutation must be of exponential size due to Theorem 7. □

With the upper bound from the previous section we get an exponential separation.

This also has consequences for strategy extraction in extended Q-resolution.

Theorem 53. *Extended Q-resolution has strategy extraction in* P*, but does not admit strategy extraction in* AC^0.

Proof. Inspecting the proof of Theorem 51 it is clear that extended Q-resolution still has strategy extraction in polynomial time, and in fact for the original formula not involving extension variables: for this it suffices to perform the construction of the decision lists as in Theorem 51 and then replace extension variables by their definition. To keep the size polynomial, this requires reusing definitions of extensions variables and hence instead of formulas will lead to polynomial-size circuits. However, the substitutions will increase the depth and not result in AC^0 circuits.

To argue that extended Q-resolution does not admit AC^0 strategy extraction we use again the QPARITY formulas, which have short proofs in extended Q-resolution by Theorem 50, but require exponential-size strategies by Theorem 7. □

Extended resolution is a powerful propositional system which can be lifted to QBF. We look at proof systems of comparible strength and their QBF counterparts in Chapter 13. In Chapter 12 we will look at proof system of intermediate strength between resolution and extended resolution.

Chapter 12

A Cutting Planes Proof System in QBF

In Chapter 3 we presented the Q-resolution proof system from [77], this system takes a propositional system– resolution [34, 107] and adds a simple reduction rule– $\forall$-Red that allows it to be complete for QBF. We find that adding $\forall$-Red to a sound and complete propositional proof system P, gives a sound and refutationally complete QBF proof system $P + \forall\text{red}$. This will be explored further in Chapter 13. In this chapter we focus on when P is the cutting planes proof system.

Cutting planes [46] is a proof system that works on linear inequalities with integer coefficients, it can be considered to work for propositional logic by converting CNF formulas into an equisatisfiable set of linear inequalities. It has been shown that for CNF it is strictly more powerful than propositional resolution and overall it is of intermediate strength between resolution and Frege. Cutting planes and Resolution are the two primary systems that have the feasible interpolation property [80,99]. This means for true implicational formulas an interpolating circuit can be extracted in polynomial time from the proofs. This results can lead to a transfer of circuit complexity lower bounds to proof size lower bounds. Indeed this gives us a lower bound for cutting planes [99].

As resolution can be adapted for the QBF setting, the cutting planes proof system can be adapted for the QBF setting where the CNF matrix gets converted into a set of linear inequalities, this is the CP+$\forall$red system given in [23]. The design of this calculus resembles that of Q-Res and QU-Res where propositional rules are obeyed and the universal reduction rule is added. The soundness and refutational completeness for PCNF was shown in [23]. We can extend this to all quantified boolean linear inequalities, which we do in this chapter.

Beyersdorff et. al. [23] provide a proof complexity investigation into CP+$\forall$red. Firstly, CP+$\forall$red is shown to p-simulate Q-Res and QU-Res. Secondly, feasible interpolation is once again shown to work in CP+$\forall$red, just as it lifts in the resolution case (Chapter 8, [21]). This provides the clique-co clique lower bound from Chapter 8. Thirdly, a strategy extraction technique is also shown for CP+$\forall$red. This is used to show a conditional mutual separation with the expansion based resolution systems. In Chapter 5 we show that strategy extraction admits AC^0 circuits for Q-Res, instead CP+$\forall$red has strategy extraction into a more general class of circuits– TC^0. In this chapter we show that CP+$\forall$red does not have AC^0 strategy extraction.

Section 12.1 introduces the Cutting Planes + $\forall$red proof system from [23]. In this section we also provide a new completeness argument that shows the refutational completeness for all false quantified boolean linear inequalities. We see a selection of proof complexity results for CP+$\forall$red from [23] in Section 12.2, as well as a new result showing that CP+$\forall$red cannot have AC^0 strategy extraction.

This chapter mainly reviews the work of [23] that appeared at the 37th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science.

12.1 Cutting Planes in QBF

Definition 20 (CP+∀red proofs for inequalities [23]). *Consider a set of quantified inequalities $\mathcal{F} \equiv \mathcal{Q}_1 x_1 \dots \mathcal{Q}_n x_n.\ F$, where F also contains the Boolean axioms. A CP+∀red refutation π of $\mathcal{F}$ is a quantified set of linear inequalities $\mathcal{Q}_1 x_1 \dots \mathcal{Q}_n x_n.[I_1, I_2, \dots, I_l]$ where the quantifier prefix is the same as in $\mathcal{F}$, I_l is an inequality of the form $0 \geq C$ for some positive integer C, and for every $j \in \{1, \dots, l\}$,*

- *$I_j \in F$, or*
- *I_j is derived from earlier inequalities in the sequence (for example, I_{k_1}, I_{k_2}, with $k_1, k_2 < j$) via one of the following inference rules:*
  1. **Addition***: From $\sum_k c_k x_k \geq C$ and $\sum_k d_k x_k \geq D$ derive $\sum_k (c_k + d_k) x_k \geq C + D$.*
  2. **Multiplication***: From $\sum_k c_k x_k \geq C$ derive $\sum_k d c_k x_k \geq dC$, where $d \in \mathbb{Z}^+$.*
  3. **Division***: From $\sum_k c_k x_k \geq C$ derive $\sum_k \frac{c_k}{d} x_k \geq \left\lceil \frac{C}{d} \right\rceil$, where $d \in \mathbb{Z}^+$ divides each c_k.*
  4. **∀-red***: From $\sum_{k \in [n] \setminus \{i\}} c_k x_k + h x_i \geq C$ derive $\begin{cases} \sum_{k \in [n] \setminus \{i\}} c_k x_k \geq C & \text{if } h > 0, \\ \sum_{k \in [n] \setminus \{i\}} c_k x_k \geq C - h & \text{if } h < 0. \end{cases}$*

     This rule can be used provided variable x_i is universal, and provided all existential variables with non-zero coefficients in the hypothesis have $\mathrm{ind}(y) < \mathrm{ind}(x_i)$. (That is, if x_j is existential and $c_j \neq 0$, then $j < i$). Observe that when $h > 0$, we are replacing x_i by 0, and when $h < 0$, we are replacing x_i by 1. We say that the universal variable x_i has been reduced.

Each inequality I_j is a line in the proof π. Note that proof lines are always of the form $\sum_k c_k x_k \geq C$ for integer-valued c_k and C. The length of π (denoted $|\pi|$) is equal to the number of lines in it, and the size of π (denoted size(π)) is equal to the bit-size of a representation of the proof (this depends on the number of lines and the binary length of the numbers in the proof).

We start with a set of linear inequalities ϕ and boolean quantifier prefix Π, The aim is to show that $\Pi \bigwedge \phi$ semantically entails $\Pi \bigwedge \phi \wedge L_1 \wedge \dots \wedge L_n$ where $L_1 \dots L_n$ are the inequalities that represent the lines of a proof π in CP+∀red.

The soundness proof [23] of CP+∀red works from an inductive argument on i. The induction hypothesis is $\Pi \bigwedge \phi \vDash \Pi \bigwedge \phi \wedge L_1 \wedge \dots \wedge L_i$.

The base case is simple. In the inductive step, a line can be derived via a cutting planes rule or via universal reduction. If L_{i+1} is derived via a cutting planes rule then clearly $\Pi \bigwedge \phi \wedge L_1 \wedge \dots \wedge L_i \vDash \Pi \bigwedge \phi \wedge L_1 \wedge \dots \wedge L_{i+1}$.

Now suppose that L_{i+1} is derived by reduction, without loss of generality $L_i = L_j[0/u]$ for some $j < i$ and universal variable u.

$$\Pi \bigwedge \phi \wedge L_1 \wedge \dots \wedge L_i$$

$$\equiv \Pi_1 \forall u \Pi_2 \bigwedge \phi \wedge L_1 \wedge \cdots \wedge L_i$$

$$\equiv \Pi_1 \forall u \Pi_2 \bigwedge \phi \wedge L_1 \wedge \cdots \wedge L_i \wedge L_j$$

$$\equiv \Pi_1 (\Pi_2 \bigwedge \phi \wedge L_1 \wedge \cdots \wedge L_i \wedge L_j)[0/u] \wedge (\Pi_2' \bigwedge \phi' \wedge L_1' \wedge \cdots \wedge L_i' \wedge L_j)[1/u]$$

$$\equiv \Pi_1 \Pi_2 \Pi_2' (\bigwedge \phi \wedge L_1 \wedge \cdots \wedge L_i)[0/u] \wedge (\bigwedge \phi' \wedge L_1' \wedge \cdots \wedge L_i')[1/u] \wedge L_j[0/u] \wedge L_j[1/u]$$

Because L_j does not contain any variables in Π_2 (or Π_2'), it can be quantified again over Π_2

$$\Pi_1 \forall u \Pi_2 \bigwedge \phi \wedge L_1 \wedge \cdots \wedge L_i \wedge L_j[0/u] \wedge L_j[1/u]$$

So finally as a semantic consequence

$$\Pi \bigwedge \phi \wedge L_1 \wedge \cdots \wedge L_{i+1}.$$

Refutational completeness was shown for prenex CNF in [23], here we can show that in fact refutational completeness holds for any set of quantified boolean linear inequalities.

Theorem 54. *CP+$\forall$red is refutationally complete for quantified boolean linear inequalities.*

Proof. The key here is that we use the implicational completeness of cutting planes [64] to show that certain lines can be derived that state that the substitution of universal variables with their Herbrand functions will lead to a contradiction with the formula. Reduction can then be performed and eventually we can arrive at the empty clause.

We use the fact that for ϕ a set of inequalities in $x_1, y_1, \ldots x_n, y_n$, when $\forall_b, \exists_b$ are boolean quantifiers and $\forall_b y_1 \exists_b x_1 \ldots \forall_b y_n \exists_b x_n \phi$ is false there is a winning strategy for the universal player in the two-player game for quantified Boolean semantics. In other words for every y_i there is a Boolean formula $C_i(x_1, \ldots x_i, y_1 \ldots y_{i-1})$ that when all $y_i = C_i$, ϕ is false.

We now consider the propositional formula

$$\bigvee_{i=1}^{n} \left(y_i \neq C_i(x_1, \ldots, x_i, y_1, \ldots, y_{i-1})\right).$$

This is a semantic consequence of ϕ, so we can use the implicational completeness of cutting planes, to derive it, however in cutting planes this must be represented as a set of linear inequalities. We can then perform universal reduction on y_n with both 0 and 1 granting us with the eventually semantic consequence of $C_n(x_1, \ldots, x_n, y_1, \ldots, y_{n-1}) \neq 0 \vee \bigvee_{i=1}^{n-1}(y_i \neq C_i(x_1, \ldots, x_i, y_1, \ldots, y_{i-1}))$ and $C_n(x_1, \ldots, x_n, y_1, \ldots, y_{n-1}) \neq 1 \vee \bigvee_{i=1}^{n-1}(y_i \neq C_i(x_1, \ldots, x_i, y_1, \ldots, y_{i-1}))$. Finally taking these two together, the semantic consequence of $\bigvee_{i=1}^{n-1}(y_i \neq C_i(x_1, \ldots, x_i, y_1, \ldots, y_{i-1}))$ can be derived and we repeat this process until we arrive at the empty clause, hence CP+$\forall$red is refutationally complete. □

12.2 Proof Complexity Results of CP+∀red

We now look at the simulation results for the CP+∀red.

Theorem 55 (Beyersdorff, Chew, Mahajan and Shukla [23]). *CP+∀red p-simulates QU-Res.*

This comes from the simulation of resolution by the cutting planes proof system [46]. Like QU-Res, CP+∀red also does not restrict propositional rules on universal variables, so this is why the stronger system is simulated.

Theorem 56 (Beyersdorff, Chew, Mahajan and Shukla [23]). *Q-Res and QU-Res cannot simulate CP+∀red.*

Theorem 57 (Beyersdorff, Chew, Mahajan and Shukla [23]). *The expansion-based system ∀Exp+Res cannot simulate CP+∀red.*

These naturally comes from the propositional separation.

Theorem 58 (Beyersdorff, Chew, Mahajan and Shukla [23]). *If* $\mathsf{P/poly} \not\subseteq \mathsf{TC}^0$ *then CP+∀red cannot simulate ∀Exp+Res.*

The idea here is similar to the strategy extraction technique in Chapter 5. However it uses a conditional lower bound to propose the existence of a circuit. The strategy extraction technique then applies in the CP+∀red system.

Theorem 59 (Beyersdorff, Chew, Mahajan and Shukla [23]). *[Strategy Extraction Theorem] Given a false QBF* $\varphi = \mathcal{Q}.\ \phi$*, with* n *variables, and a CP+∀red refutation* π *of* φ *of size* m*, it is possible to extract from* π *a winning strategy* σ_u *for each universal variable* $u \in \varphi$*, such that each* σ_u *can be computed by Boolean circuits of* $(m+n)^{O(1)}$ *size, constant depth, with unbounded fan-in AND, OR, NOT gates as well as threshold gates.*

In particular, if φ *can be refuted in CP+∀red in* $n^{O(1)}$ *size, then the winning strategies can be computed in* TC^0*.*

We will see in Chapter 13 that ∀-Red rule and strategy extraction are tightly connected, this is why we can get strategy extraction here.

We do not get polynomial time strategy extraction, however, in the weak model of AC^0. To show this, we define the family of false QBFs QMAJORITY_n. The QBF expresses the false sentence

$$\exists x_1, \ldots, x_{2n+1}\, \forall z\, \text{MAJORITY}(x_1, \ldots, x_{2n+1}) \neq z.$$

To express this compactly with a CNF matrix, we use auxiliary variables t_k^i for $i \in [2n+1]$ and $0 \leq k \leq i$, and inductively define $t_k^i = \text{THRESHOLD}_k(x_1, \ldots, x_i)$ by using $t_k^i = t_k^{i-1} \vee (t_{k-1}^{i-1} \wedge x_i)$. Therefore $t_{n+1}^{2n+1} = \text{MAJORITY}(x_1, \ldots, x_{2n+1})$.

We define QMAJORITY_n as the QBF with the prefix

$$\exists x_1, \ldots, x_{2n+1}\, \forall z\, \exists t_0^1, t_1^1,\ \exists t_0^2, t_1^2, t_2^2,\ \ldots \exists t_0^{2n+1}, t_1^{2n+1}, \ldots, t_{2n+1}^{2n+1}$$

and the CNF matrix

$$\begin{array}{llll}
\{t_0^i\} & & & i \in [2n+1] \\
\{x_1, \neg t_1^1\} & \{\neg x_1, t_1^1\} & & \\
\{t_{i-1}^{i-1}, \neg t_i^i\} & \{x_i, \neg t_i^i\} & \{\neg x_i, \neg t_{i-1}^{i-1}, t_i^i\} & 2 \le i \le 2n+1 \\
\{x_i, \neg t_k^i, t_k^{i-1}\} & \{\neg t_k^i, t_k^{i-1}, t_{k-1}^{i-1}\} & & 2 \le i \le 2n+1, k \in [i-1] \\
\{\neg t_k^{i-1}, t_k^i, \} & \{\neg x_i, t_k^i, \neg t_{k-1}^{i-1}\} & & 2 \le i \le 2n+1, k \in [i-1] \\
\{z, t_{n+1}^{2n+1}\} & \{\neg z, \neg t_{n+1}^{2n+1}\}. & &
\end{array}$$

Note that this is a false prenex QBF with CNF matrix, and is of size $\Theta(n^2)$.

Theorem 60. *1. Any proof system with polynomial time* AC^0 *strategy extraction requires exponential size proofs of* QMAJORITY

2. QMAJORITY *has polynomial-sized proofs in* CP+$\forall$red.

Proof. For the lower bound, note that the only winning strategy for the single universal variable z is the function $\textsc{Majority}(x_1, \ldots, x_{2n+1})$ itself. By the results of [67], constant-depth circuits for PARITY, and hence for MAJORITY, must be of size exponential in the number of variables. On the other hand, if we have a proof of size S, and one can extract a winning strategy for the universal player as an AC^0 circuit of size polynomial in S. Therefore, if QMAJORITY has proof of size S, then the winning strategy for the universal player, and hence MAJORITY, can be computed by a polynomial size AC^0-circuit in S. It follows that S must be exponential in n.

We now describe a CP+$\forall$red proof for QMAJORITY_n of length $\Theta(n^2)$.

We aim to first derive inductively that $t_k^i = \textsc{Threshold}_k(x_1, \ldots, x_i)$, for each $i \in [2n+1]$ and $0 \le k \le i$, in a cutting planes derivation. To simplify the expressions, we use the notation PSUM_i to denote the partial sum $\sum_{j \le i} x_j$. We write the implications and inequalities as follows:

	forward direction	backward direction
Implication	$t_k^i \rightarrow \text{PSUM}_i \ge k$	$t_k^i \leftarrow \text{PSUM}_i \ge k$
Inequality	$-kt_k^i + \text{PSUM}_i \ge 0$	$(i-k+1)t_k^i - \text{PSUM}_i \ge 1-k$

We proceed by induction on i.

Base case: For $i = 1$, k is 0 or 1. At $k = 0$, the forward direction is the Boolean axiom $x_1 \ge 0$, and the backward direction inequality $2t_0^1 - x_1 \ge 1$ is obtained by adding the Boolean axiom $-x_1 \ge -1$ and twice the unit clause axiom $t_0^1 \ge 1$. For $k = 1$, both directions are the inequalities corresponding to the axioms $t_1^1 \leftrightarrow x_1$.

Inductive Step: Now assume that $i \ge 2$. The extreme values of k, namely $k = 0$ and $k = i$, are easy and we deal with them first.

$\mathbf{k = 0}$: For the forward direction, we simply add all Boolean axioms $x_j \ge 0$ for $j \le i$ together to get $\text{PSUM}_i \ge 0$. For the backward direction, similarly, we add all Boolean axioms $-x_j \ge -1$ for $j \le i$ together to get $-\text{PSUM}_i \ge -i$, and then add $(i+1)$ times the unit clause axiom $t_0^i \ge 1$.

$\mathbf{k = i}$: The forward inequality is derived as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\{\neg t_i^i, t_{i-1}^{i-1}\}}{-t_i^i + t_{i-1}^{i-1} \geq 0}}{-(i-1)t_i^i + (i-1)t_{i-1}^{i-1} \geq 0} \quad -(i-1)t_{i-1}^{i-1} + \text{PSUM}_{i-1} \geq 0}{-(i-1)t_i^i + \text{PSUM}_{i-1} \geq 0} \quad \dfrac{\{\neg t_i^i, x_i\}}{-t_i^i + x_{i-1} \geq 0}}{-it_i^i + \text{PSUM}_i \geq 0}$$

The backward inequality is derived as follows:

$$\dfrac{\dfrac{\{t_i^i, \neg t_{i-1}^{i-1}, \neg x_i\}}{t_i^i - t_{i-1}^{i-1} - x_i \geq -1} \quad t_{i-1}^{i-1} - \text{PSUM}_{i-1} \geq 2 - i}{t_i^i - \text{PSUM}_i \geq 1 - i}$$

$\mathbf{1 \leq k < i}$: Now we consider the intermediate values. The backward direction is a bit easier and we do it first. It uses the inductively derived backward direction for $i - 1$.

We first derive inequalities for $\text{PSUM}_{i-1} \geq k \rightarrow t_k^i$ and $\text{PSUM}_{i-1} \geq k - 1 \wedge x_i \rightarrow t_k^i$ and then derive the inequality for $\text{PSUM}_i \geq k \rightarrow t_k^i$.

The derivation of an inequality for $\text{PSUM}_{i-1} \geq k \rightarrow t_k^i$ is as follows.

$$\dfrac{\dfrac{\dfrac{\{t_k^i, \neg t_k^{i-1}\}}{t_k^i - t_k^{i-1} \geq 0}}{(i-k)t_k^i - (i-k)t_k^{i-1} \geq 0} \quad (i-k)t_k^{i-1} - \text{PSUM}_{i-1} \geq 1 - k}{(i-k)t_k^i - \text{PSUM}_{i-1} \geq 1 - k}$$

The derivation of the inequality for $\text{PSUM}_{i-1} \geq k - 1 \wedge x_i \rightarrow t_k^i$ is as follows.

$$\dfrac{\dfrac{\dfrac{\{t_k^i, \neg t_{k-1}^{i-1}, \neg x_i\}}{t_k^i - t_{k-1}^{i-1} - x_i \geq -1}}{(i-k+1)(t_k^i - t_{k-1}^{i-1} - x_i) \geq k - i - 1} \quad (i-k+1)t_{k-1}^{i-1} - \text{PSUM}_{i-1} \geq 2 - k}{(i-k+1)t_k^i - \text{PSUM}_{i-1} - (i-k+1)x_i \geq 1 - i}$$

We can conclude with the following derivations.

$$\dfrac{\dfrac{(i-k)t_k^i - \text{PSUM}_{i-1} \geq 1 - k}{(i-k)^2 t_k^i - (i-k)\text{PSUM}_{i-1} \geq (i-k)(1-k)} \quad (i-k+1)t_k^i - \text{PSUM}_{i-1} - (i-k+1)x_i \geq 1 - i}{((i-k+1)(i-k)+1)t_k^i - (i-k+1)\text{PSUM}_i \geq 1 - k(i+1-k)}$$

$$\dfrac{\dfrac{\dfrac{t_k^i \geq 0}{(i-k)t_k^i \geq 0} \quad ((i-k+1)(i-k)+1)t_k^i - (i-k+1)\text{PSUM}_i \geq 1 - k(i+1-k)}{(i-k+1)^2 t_k^i - (i-k+1)\text{PSUM}_i \geq 1 - k(i+1-k)}}{(i-k+1)t_k^i - \text{PSUM}_i \geq 1 - k}$$

The forward direction uses both the directions of the inductively derived inequalities for $i-1$. Recall that $i \geq 2$ and $1 \leq k \leq i-1$. We need to derive $t_k^i \to \text{PSUM}_i \geq k$. The key to this is to derive and then combine inequalities for $t_k^i \to \text{PSUM}_{i-1} \geq k-1$ and $t_k^i \to x_i \vee \text{PSUM}_{i-1} \geq k$.

In order to show $t_k^i \to \text{PSUM}_{i-1} \geq k-1$ from our inductive hypothesis and the clause $\neg t_k^i \vee t_k^{i-1} \vee t_{k-1}^{i-1}$, we use the fact that $t_k^{i-1} \to t_{k-1}^{i-1}$ is true. But first we must derive this fact from the induction hypothesis. We start the derivation as follows:

$$\frac{(i+1-k)t_{k-1}^{i-1} - \text{PSUM}_{i-1} \geq 2-k \qquad -kt_k^{i-1} + \text{PSUM}_{i-1} \geq 0}{-kt_k^{i-1} + (i+1-k)t_{k-1}^{i-1} \geq 2-k}$$

Now we equalise the coefficients on the left-hand-side by adding a multiple of an appropriate Boolean axiom, and then a division rule yields $-t_k^{i-1} + t_{k-1}^{i-1} \geq 0$.

If $i+1-2k > 0$, then we proceed as follows:

$$\frac{\dfrac{-t_k^{i-1} \geq -1}{-(i+1-2k)t_k^{i-1} \geq -(i+1-2k)} \qquad -kt_k^{i-1} + (i+1-k)t_{k-1}^{i-1} \geq 2-k}{\dfrac{-(i+1-k)t_k^{i-1} + (i+1-k)t_{k-1}^{i-1} \geq 2-(i+1-k)}{-t_k^{i-1} + t_{k-1}^{i-1} \geq 0}}$$

Alternatively, if $i+1-2k \leq 0$,

$$\frac{\dfrac{t_{k-1}^{i-1} \geq 0}{-(i+1-2k)t_{k-1}^{i-1} \geq 0} \qquad -kt_k^{i-1} + (i+1-k)t_{k-1}^{i-1} \geq 2-k}{\dfrac{-kt_k^{i-1} + kt_{k-1}^{i-1} \geq 2-k}{-t_k^{i-1} + t_{k-1}^{i-1} \geq 0}}$$

(At the last step, we may obtain 1 on the right hand side if $k=1$. In that case, we further add $0 \geq -1$, which may be considered an axiom or may be derived by adding the two Boolean axioms for any variable.)

Next we use the derived inequality $-t_k^{i-1} + t_{k-1}^{i-1} \geq 0$ to derive $-t_k^i + t_{k-1}^{i-1} \geq 0$.

$$\frac{\dfrac{-t_k^{i-1} + t_{k-1}^{i-1} \geq 0 \qquad \dfrac{\{-t_k^i, t_k^{i-1}, t_{k-1}^{i-1}\}}{-t_k^i + t_k^{i-1} + t_{k-1}^{i-1} \geq 0}}{-t_k^i + 2t_{k-1}^{i-1} \geq 0} \qquad -t_k^i \geq -1}{\dfrac{-2t_k^i + 2t_{k-1}^{i-1} \geq -1}{-t_k^i + t_{k-1}^{i-1} \geq 0}}$$

This, with the inductive hypothesis, lets us derive $t_k^i \to \text{PSUM}_{i-1} \geq k-1$.

$$\frac{\dfrac{-t_k^i + t_{k-1}^{i-1} \geq 0}{-(k-1)t_k^i + (k-1)t_{k-1}^{i-1} \geq 0} \qquad -(k-1)t_{k-1}^{i-1} + \text{PSUM}_{i-1} \geq 0}{-(k-1)t_k^i + \text{PSUM}_{i-1} \geq 0}$$

As described earlier, we also need an inequality for $t_k^i \to x_i \vee \sum_{j<i} x_j \geq k$. Under Boolean conditions, the inequality $-kt_k^{i-1} + \sum_{j<i} x_j + kx_i \geq 0$ suffices. We use an axiom clause along with the inductive hypothesis to derive it.

$$\dfrac{\dfrac{\dfrac{\{\neg t_k^i, t_k^{i-1}, x_i\}}{-t_k^i + t_k^{i-1} + x_i \geq 0}}{-kt_k^i + kt_k^{i-1} + kx_i \geq 0} \qquad -kt_k^{i-1} + \text{PSUM}_{i-1} \geq 0}{-kt_k^i + \text{PSUM}_{i-1} + kx_i \geq 0}$$

Now we combine the derived inequalities to obtain the inequality for the forward direction.

$$\dfrac{\dfrac{-kt_k^i + \text{PSUM}_{i-1} + kx_i \geq 0 \qquad \dfrac{-t_k^i \geq -1}{(1-k)t_k^i \geq 1-k}}{(1-2k)t_k^i + \text{PSUM}_{i-1} + kx_i \geq 1-k} \qquad \dfrac{-(k-1)t_k^i + \text{PSUM}_{i-1} \geq 0}{-(k-1)^2 t_k^i + (k-1)\text{PSUM}_{i-1} \geq 0}}{\dfrac{-k^2 t_k^i + k\text{PSUM}_i \geq 1-k}{-kt_k^i + \text{PSUM}_i \geq 0}}$$

After Induction: With the induction part of the proof completed, we have shown that $(n+1)t_{n+1}^{2n+1} - \text{PSUM}_{2n+1} \geq -n$ and $-(n+1)t_{n+1}^{2n+1} + \text{PSUM}_{2n+1} \geq 0$ can be derived in a short proof of length $\Theta(n^2)$. (We use $\Theta(1)$ additional steps for both directions of each (i,k) pair.) We now complete the refutation via universal reduction, which can be applied after eliminating the t variables; the partial sums do not block the reduction. The first fragment below reduces z by setting $z = 0$, the second one sets $z = 1$.

$$\dfrac{\dfrac{\dfrac{\{z, t^{2n+1}\}}{z + t_{n+1}^{2n+1} \geq 1}}{(n+1)z + (n+1)t_{n+1}^{2n+1} \geq n+1} \qquad -(n+1)t_{n+1}^{2n+1} + \text{PSUM}_{2n+1} \geq 0}{\dfrac{(n+1)z + \text{PSUM}_{2n+1} \geq n+1}{\text{PSUM}_{2n+1} \geq n+1}}$$

$$\dfrac{\dfrac{\dfrac{\{\neg z, \neg t^{2n+1}\}}{-z - t_{n+1}^{2n+1} \geq -1}}{-(n+1)z - (n+1)t_{n+1}^{2n+1} \geq -(n+1)} \qquad (n+1)t_{n+1}^{2n+1} - \text{PSUM}_{2n+1} \geq -n}{\dfrac{-(n+1)z - \text{PSUM}_{2n+1} \geq -2n-1}{-\text{PSUM}_{2n+1} \geq -n}}$$

$$\dfrac{-\text{PSUM}_{2n+1} \geq -n \qquad \text{PSUM}_{2n+1} \geq n+1}{0 \geq 1}$$

□

It was shown in [99] that monotone feasible interpolation holds for cutting planes. This result was extended to CP+$\forall$red in the same way that the resolution result was extended to Q-Res.

Theorem 61 (Beyersdorff, Chew, Mahajan and Shukla [23]). *CP+$\forall$red for inequalities admits monotone real feasible interpolation. That is, let $\mathcal{F}$ be any false quantified boolean set of inequalities of the form $\exists \boldsymbol{p}\mathcal{Q}\boldsymbol{q}\mathcal{Q}\boldsymbol{r}.\big[A(\boldsymbol{p},\boldsymbol{q}) \wedge B(\boldsymbol{p},\boldsymbol{r})\big]$ where $A \cup B$ also includes all Boolean axioms, and where the coefficients of $\boldsymbol{p}$ are either all non-negative in A or are all non-positive in B. For any S-proof π of $\mathcal{F}$, we can extract from π a monotone real circuit C of size polynomial in the length l of π and the number n of $\boldsymbol{p}$ variables in $\mathcal{F}$, such that C computes a Boolean function, and on every $0,1$ assignment $\boldsymbol{a}$ for $\boldsymbol{p}$,*

$$C(\boldsymbol{a}) = 0 \implies \mathcal{Q}\boldsymbol{q}.A(\boldsymbol{a},\boldsymbol{q}) \text{ is false, and}$$
$$C(\boldsymbol{a}) = 1 \implies \mathcal{Q}\boldsymbol{r}.B(\boldsymbol{a},\boldsymbol{r}) \text{ is false.}$$

Such a C is called a monotone real interpolating circuit for $\mathcal{F}$.

In Chapter 13 we apply the same reduction rule to Frege systems as we have done with cutting planes and show further strategy extraction results.

Chapter 13

Frege Systems for QBF

In Chapter 12 we saw that the cutting planes proof system can be augmented with a reduction rule in a similar way to how Q-resolution [77] builds on resolution. A strategy extraction theorem is present for both Q-Res and CP+$\forall$red, in the complexity classes AC^0 and TC^0 respectively. In this chapter we continue these ideas and provide a general method of transforming a propositional proof system into a QBF proof system, by adding the universal reduction rule.

A. From Propositional to QBF: New QBF Proof Systems. We exhibit a general method how to transform a propositional proof system to a QBF proof system. Our method is both conceptually simple and elegant. Starting from a propositional proof system P comprised of axioms and rules, we design a system P + $\forall$red for closed prenex QBFs (Definition 21). Throughout the proof, the quantifier prefix is fixed, and lines in the system P + $\forall$red are conceptually the same as lines in P, i.e. clauses in resolution, circuits from C in C-Frege, or inequalities in cutting planes. Our new system P + $\forall$red uses all the rules from P, and can apply those on arbitrary lines, irrespective of whether the variables are existentially or universally quantified. To make the system complete, we introduce a universal reduction rule that allows us to replace universal variables by simple Herbrand functions, which can be represented as lines in P. The link to Herbrand functions provides a clear semantic meaning for the $\forall$red rule, resulting in a natural and robust system P + $\forall$red. Our new systems P + $\forall$red are inspired by the approach taken in the definition of Q-Res [77]; and indeed when choosing resolution as the base system P, our system P + $\forall$red coincides with the previously studied QU-Res [116]. While our definitions are quite general and yield for example previously missing QBF versions of polynomial calculus or cutting planes, we concentrate here on exploring the hierarchy C-Frege + $\forall$red of new QBF Frege systems.

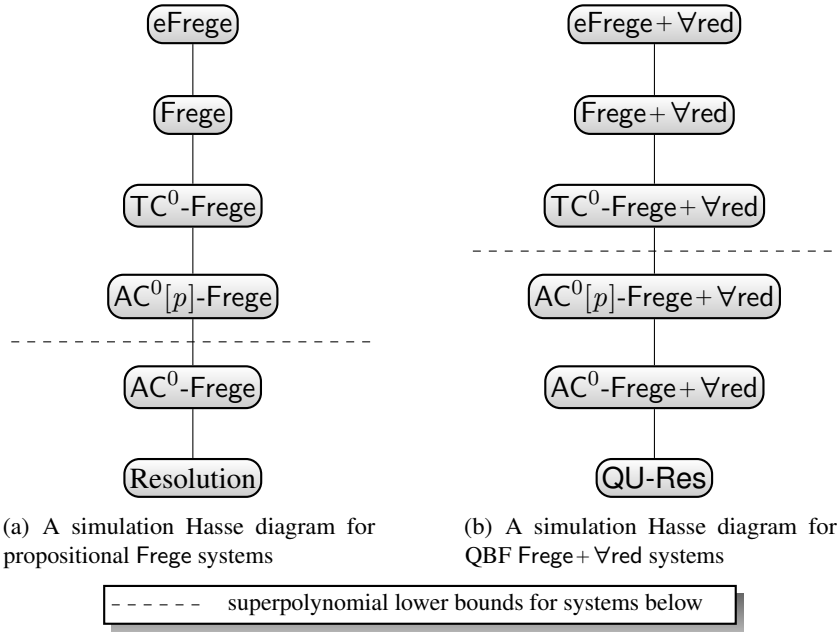
B. From Circuit to QBF Lower Bounds: a General Technique. It is a long-standing belief that circuit lower bounds correspond to proof size lower bounds, and clearly some of the strongest lower bounds in proof complexity as those for AC^0-Frege are inspired by proof techniques in circuit complexity, cf. the survey of [12]. Here we give a precise and formal account on how *any* circuit lower bound for C can be directly lifted to a proof size lower bound in C-Frege + $\forall$red.

Conceptually, our lower bound method uses the idea of *strategy extraction*, an important paradigm in QBF (Theorem 63). Semantically, a QBF can be understood as a game between a universal and an existential player, where the universal player wins if and only if the QBF is false. Winning strategies for the universal player can be very complex. We also find that the strategy extraction theorem is implied by the existence of the reduction rule and is present in all our C-Frege + $\forall$red systems. Precisely, we show that from each refutation of a false QBF in a system

C-Frege + ∀red we can efficiently extract a winning strategy for the universal player in a simple computational model we call C-decision lists. We observe that C-decision lists are easy to transform into C circuits itself, with only a slight increase in complexity.

A direct transfer of circuit complexity to proof complexity is conjectured for propositional proof complexity, but such a transfer has not be proved. In C-Frege + ∀red we get an exact transfer. To obtain a proof-size lower bound we need a function f that is hard for C. By strategy extraction, refutations of $\mathcal{Q}$-f in C-Frege + ∀red yield C-circuits for f; hence all such refutations must be long. In fact, we even show the converse implication to hold, i.e. from small C-circuits for f we construct short proofs of $\mathcal{Q}$-f in C-Frege + ∀red. Our lower bound technique widely generalises ideas recently used in Chapter 5 to show lower bounds for Q-Res and QU-Res for formulas originating from the PARITY function.

C. Lower Bounds and Separations: Applying our Framework. We apply our proof technique to a number of famous circuit lower bounds, thus obtaining lower bounds and separations for C-Frege + ∀red systems that are yet unparalleled in propositional proof complexity. In Figure 26 we summarise the QBF Frege systems. Essentially we show three new important lower bound results.



(a) A simulation Hasse diagram for propositional Frege systems

(b) A simulation Hasse diagram for QBF Frege + ∀red systems

Fig. 26. A comparison of known lower bounds in Frege and Frege + ∀red systems

C.(i) Lower Bounds and Separations for the QBF Proof System AC$^0[p]$-Frege + ∀red. We observe the current picture for Frege systems in Figure 26(a). We see that the best lower bounds are for the bounded depth Frege system– AC^0-Frege. However AC$^0[p]$-Frege which allows mod p

gates has no known lower bounds. In QBF we can move this dotted line slightly higher to a lower bound for $\mathsf{AC}^0[p]$-$\mathsf{Frege}+\forall\mathsf{red}$ as shown in Figure 26(b).

The seminal results of [102, 112] showed that PARITY and more generally MOD_q are the classic examples for functions that require exponential-size bounded-depth circuits with MOD_p gates, where p and q are different primes. Using these functions, we define families of QBFs that require exponential-size proofs in $\mathsf{AC}^0[p]$-$\mathsf{Frege}+\forall\mathsf{red}$ by strategy extraction.

C.(ii) $\mathsf{AC}^0[p]$-$\mathsf{Frege}+\forall\mathsf{red}$ and TC^0-$\mathsf{Frege}+\forall\mathsf{red}$ are Separated. MAJORITY is another classic function in circuit complexity, for which exponential lower bounds are known for constant-depth circuits with MOD_p gates for each prime p [102, 112]. Using our technique, we transfer these to lower bounds in $\mathsf{AC}^0[p]$-$\mathsf{Frege}+\forall\mathsf{red}$ for all primes p. Carefully choosing the QBF encoding of MAJORITY, we obtain polynomial upper bounds for the MAJORITY formulas in TC^0-$\mathsf{Frege}+\forall\mathsf{red}$, thus proving an exponential separation between the two QBF proof systems $\mathsf{AC}^0[p]$-$\mathsf{Frege}+\forall\mathsf{red}$ and TC^0-$\mathsf{Frege}+\forall\mathsf{red}$. Again, such a separation is wide open in propositional proof complexity.

To obtain separations of these proof systems, the exact formulation of the QBFs matters. When defining the PARITY or MOD_q formulas directly from (arbitrary) NC^1-circuits computing these functions, we obtain polynomial-size upper bounds in $\mathsf{Frege}+\forall\mathsf{red}$. However, when carefully choosing specific and indeed very natural encodings, we can prove upper bounds for the MOD_q formulas even in $\mathsf{AC}^0[q]$-$\mathsf{Frege}+\forall\mathsf{red}$, thus obtaining exponential separations of all the $\mathsf{AC}^0[p]$-$\mathsf{Frege}+\forall\mathsf{red}$ systems for distinct primes p.

As mentioned before, lower bounds for $\mathsf{AC}^0[p]$-Frege (as well as their separations) are major open problems in propositional proof complexity.

C.(iii) CNFs Separating the AC^0_d-$\mathsf{Frege}+\forall\mathsf{red}$ Hierarchy. As a third example for our approach we investigate the fine structure of AC^0-$\mathsf{Frege}+\forall\mathsf{red}$, comprising all AC^0_d-$\mathsf{Frege}+\forall\mathsf{red}$ systems, where all formulas in proofs are required to have at most depth d for a fixed constant d. In circuit complexity the SIPSER$_d$ functions from [38] provide an exponential separation of depth-$(d-1)$ from depth-d circuits [66]. With our technique, this separation translates into a separation of AC^0_{d-3}-$\mathsf{Frege}+\forall\mathsf{red}$ from AC^0_d-$\mathsf{Frege}+\forall\mathsf{red}$, where the increased gap of size 3 comes from our transformation of C-decision lists into C-circuits.

The SIPSER$_d$ formulas achieving these separations are prenexed CNFs, i.e. the formulas have depth 2. While in propositional proof complexity the hierarchy of AC^0_d-Frege systems is exponentially separated [2, 83, 98], such a separation by formulas of depth *independent of* d is a major open problem.

D. Characterisation of $\mathsf{Frege}+\forall\mathsf{red}$ Systems with Strategy Extraction. In addition to the unconditional bounds, we can yield conditional lower bounds based on complexity assumptions. Indeed we find using strategy extraction that $\mathsf{PSPACE} \not\subseteq \mathsf{NC}^1$ would show that $\mathsf{Frege}+\forall\mathsf{red}$ has a lower bound. In fact this connection is two-way, we show that $\mathsf{Frege}+\forall\mathsf{red}$ can only be found from

either lower bounds in Frege or lower bounds of NC^1 in PSPACE. This is perhaps one of the most important aspects of Frege + ∀red since it essentially shows that strong QBF calculi will be difficult to show lower bounds for.

In Section 13.1 we define the C-Frege + ∀red systems and justify their soundness and completeness. In Section 13.2 we show the *strategy extraction theorem* for C-Frege + ∀red, a formalised strategy extraction exists in C-Frege + ∀red which we present in Section 13.3. In Section 13.4 we show the lower bounds and separations for $\mathsf{AC}^0[p]$-Frege, in Section 13.5 we show the separations of AC^0_d-Frege + ∀red systems. Finally in Section 13.6 we characterise hardness in Frege + ∀red and eFrege + ∀red system.

The work in this chapter appeared at the 7th Annual Innovations in Theoretical Computer Science conference [17].

13.1 Defining QBF Frege Systems

For the following we fix a circuit class C with some natural properties, e.g. closure under restrictions.

Definition 21 (C-Frege + ∀red)**.** *A refutation of a false QBF $\mathcal{Q}\phi$ in the system* C-Frege + ∀red *is a sequence of lines $L_1, \ldots, L_\ell$ where each line is a circuit from the class* C*, ϕ is a conjunction of* C *circuits, $\mathcal{Q}$ is a quantifier prefix of all variables in ϕ, $L_\ell = \bot$ and each L_i is either a conjunct in ϕ, inferred from previous lines L_j using the inference rules of* C-Frege *or using the following rule*

$$\frac{L_j}{L_j[u/B]}\ (\forall\mathsf{red}),$$

where $L_j[u/B]$ belongs to the class C*, u is the innermost variable among the variables of L_j and B is a circuit from the class* C *containing only variables left of u.*

The formal justification why C-Frege + ∀red is a sound and complete QBF proof system is given in Theorem 62 below. However, let us pause a moment to see why adding the ∀red rule results in a natural proof system C-Frege + ∀red. Recall that we consider C-Frege + ∀red as a refutation system; hence we aim to refute false quantified C formulas. A standard approach to witness the falsity of quantified formulas is through *Herbrand functions*, which replace a universal variable u by a function in the existential variables left of u. These functions can be viewed as 'counterexample functions'. In Definition 21, B plays the role of the Herbrand function. Clearly, when restricting formulas to a class C we should also restrict B to that class, and substituting the Herbrand function into the formula should again preserve C.

Note that we are even allowed to choose different Herbrand functions B for the same variable u in different parts of the proof. In general, this will be unsound (unless variables right of u are renamed). However, it is safe to do if the line L_j does not contain any variables right of u.

It is illustrative to see how our construction compares to previously studied QBF resolution systems. Choosing Res as our propositional proof system, which is an OR-Frege system, we obtain Res + ∀red. In Res + ∀red the ∀red rule can substitute a universal u by either another variable or

by a constant 0/1. In the former case, we simply obtain a weakening step. In the latter case, if u appears positively in the clause then substituting u by 0 precisely corresponds to an application of the $\forall$red rule in Q-Res, whereas substituting u by 1 results in the useless tautology $\top$.[2] As $Res + \forall$red can resolve on existential and universal variables, our system $Res + \forall$red is exactly the well-known QU-Res (with weakening).

We now proceed to show soundness and completeness of the new QBF systems.

Theorem 62. *For every circuit complexity class* C*, the system* C-Frege+$\forall$red *is a refutational QBF proof system.*

Proof. $Res + \forall$red is complete as it p-simulates Q-Res, which is complete for QBF [77]. To obtain the completeness for C-Frege+$\forall$red we first use de Morgan's rules to expand the formula into a CNF. This is possible as, by definition, C-Frege is implicationally complete. Now we can refute the CNF by $Res + \forall$red. C-Frege+$\forall$red p-simulates $Res + \forall$red and hence C-Frege+$\forall$red is complete.

Regarding the soundness of C-Frege+$\forall$red, let $(L_1, \ldots, L_\ell)$ be a refutation of $\mathcal{Q}\phi$ in the system C-Frege+$\forall$red and let

$$\phi_i = \begin{cases} \phi & \text{if } i = 0, \\ \phi \wedge L_1 \wedge \cdots \wedge L_i & \text{otherwise.} \end{cases}$$

By induction on i we prove that $\mathcal{Q}\phi$ semantically entails $\mathcal{Q}\phi_i$, i.e. $\mathcal{Q}\phi \models \mathcal{Q}\phi_i$. Hence, at step $i = \ell$ we will immediately obtain that $\mathcal{Q}\phi$ is false, since $L_\ell = \{\bot\}$ and $\mathcal{Q}\phi_\ell \equiv \bot$.

Since $\mathcal{Q}\phi = \mathcal{Q}\phi_0$ the base case of the induction holds.

We show now that $\mathcal{Q}\phi \models \mathcal{Q}\phi_i$ implies $\mathcal{Q}\phi \models \mathcal{Q}\phi_{i+1}$. By definition, $\phi_{i+1} = (\phi_i \wedge L_{i+1})$ and L_{i+1} was either introduced by a C-Frege rule or by the $\forall$red rule. If L_{i+1} was introduced by a C-Frege rule then $\phi_i \models L_{i+1}$, so $\phi_i \models \phi_{i+1}$ and clearly $\mathcal{Q}\phi \models \mathcal{Q}\phi_i \models \mathcal{Q}\phi_{i+1}$.

Suppose now that L_{i+1} was introduced by the $\forall$red rule, say $L_{i+1} = L_j[u/B]$ with $j \leq i$, u the innermost variable among the ones in L_j and B relying only on the variables left of u. Moreover suppose that $\mathcal{Q}\phi_i = \mathcal{Q}_1\boldsymbol{x}\, \forall u\, \mathcal{Q}_2\boldsymbol{y}\phi_i$, then we have the following chain of equivalences

$$\begin{aligned}
\mathcal{Q}\phi_i &= \mathcal{Q}_1\boldsymbol{x}\, \forall u\, \mathcal{Q}_2\boldsymbol{y}\phi_i \\
&\equiv \mathcal{Q}_1\boldsymbol{x}\, \forall u\, \mathcal{Q}_2\boldsymbol{y}\phi_i \wedge L_j \\
&\equiv \mathcal{Q}_1\boldsymbol{x}\Big(\big(\mathcal{Q}_2\boldsymbol{y}\phi_i[u/0] \wedge L_j[u/0]\big) \wedge \big(\mathcal{Q}_2\boldsymbol{y}\phi_i[u/1] \wedge L_j[u/1]\big)\Big) \\
&\equiv \mathcal{Q}_1\boldsymbol{x}\Big(L_j[u/0] \wedge L_j[u/1] \wedge \big(\mathcal{Q}_2\boldsymbol{y}\phi_i[u/0]\big) \wedge \big(\mathcal{Q}_2\boldsymbol{y}\phi_i[u/1]\big)\Big) \\
&\equiv \mathcal{Q}_1\boldsymbol{x}\Big(L_j[u/0] \wedge L_j[u/1] \wedge \forall u \mathcal{Q}_2\boldsymbol{y}\phi_i\Big) \\
&\equiv \mathcal{Q}_1\boldsymbol{x}\Big(L_j[u/0] \wedge L_j[u/1] \wedge L_j[u/B] \wedge \forall u \mathcal{Q}_2\boldsymbol{y}\phi_i\Big)
\end{aligned}$$

[2] Note that, contrasting the usual setting of Q-Res [77], our definition of $Res + \forall$red does not need to disallow tautologous resolvents as these will always be reduced to $\top$.

$$\equiv \mathcal{Q}_1 \boldsymbol{x} \forall u \mathcal{Q}_2 \boldsymbol{y} \phi_i \wedge L_j[u/0] \wedge L_j[u/1] \wedge L_j[u/B].$$

From this follows, by weakening, that

$$\mathcal{Q}\phi_i \models \mathcal{Q}_1 \boldsymbol{x} \forall u \mathcal{Q}_2 \boldsymbol{y} \phi_i \wedge L_j[u/B],$$

hence $\mathcal{Q}\phi \models \mathcal{Q}\phi_{i+1}$. □

Clearly lower bounds on the complexity of C-Frege + ∀red follow from lower bounds on C-Frege. The lower bounds we show later will be of a different kind as they will be 'purely for QBF proof systems' in the sense that they will lower bound the number of occurrences of the ∀red rule in refutations.

13.2 Strategy Extraction

We introduce now the simple computational model of C-decision lists.

Definition 22 (C-decision list). *A* C-decision list *is a program of the following form*

```
if C_1(x) then u ← B_1(x);
  else if C_2(x) then u ← B_2(x);
        ⋮
    else if C_{ℓ−1}(x) then u ← B_{ℓ−1}(x);
      else u ← B_ℓ(x),
```

where $C_1, \ldots, C_{\ell-1}$ and $B_1, \ldots, B_\ell$ are circuits in the class C*. Hence a decision list as above computes a Boolean function $u = g(\boldsymbol{x})$.*

This definition generalises decision lists from [105], where the conditions $C_i(\boldsymbol{x})$ are expressible as terms and the decisions are expressed as constants. These changes are to reflect our new calculi. We note that for many cases C-decision lists can be easily transformed into C-circuits.

Proposition 4. *Let D be a* C*-decision list using circuits $C_1, \ldots, C_{\ell-1}$ and $B_1, \ldots, B_\ell$, such that D computes the Boolean function g. Then there exists a circuit $D' \in$ C computing the same function g, such that the size of D' is linear in the size of D and*

$$\mathrm{depth}(D') \leq \max\left\{ \max_{1 \leq i \leq \ell-1} \{\mathrm{depth}(C_i)\}, \max_{1 \leq i \leq \ell} \{\mathrm{depth}(B_i)\} \right\} + 2.$$

Proof. We have that

$$u \equiv \bigvee_{j=1}^{\ell} \left(C_j(\boldsymbol{x}) \wedge B_j(\boldsymbol{x}) \wedge \bigwedge_{k<j} \neg C_k(\boldsymbol{x}) \right),$$

where C_ℓ is a circuit computing the constant 1. □

Balabanov and Jiang [10] proved a strategy extraction result for QU-Res. Here we generalise that result to the full hierarchy of C-Frege+ ∀red QBF proof systems. This result is the main tool we use to prove size lower bounds in such systems.

Theorem 63 (Strategy Extraction). *Given a false QBF $\mathcal{Q}\phi$ and a refutation π of $\mathcal{Q}\phi$ in* C-Frege + ∀red*, it is possible to extract in linear time (w.r.t. $|\pi|$) a collection of* C*-decision lists D computing a winning strategy on the universal variables of ϕ.*

Proof. Let $\pi = (L_1, \ldots, L_s)$ be a refutation of the false QBF $\mathcal{Q}\phi$ and let

$$\pi_i = \begin{cases} \emptyset & \text{if } i = s, \\ (L_{i+1}, \ldots, L_s) & \text{otherwise.} \end{cases}$$

We show, by downward induction on i, that from π_i it is possible to construct in linear time (w.r.t. $|\pi_i|$) a winning strategy σ^i for the universal player for the QBF formula $\mathcal{Q}\phi_i$, where

$$\phi_i = \begin{cases} \phi & \text{if } i = 0, \\ \phi \wedge L_1 \wedge \cdots \wedge L_i & \text{otherwise,} \end{cases}$$

such that for each universal variable u in $\mathcal{Q}\phi$, there exists a C-decision list D^i_u computing σ^i_u as a function of the variables in $\mathcal{Q}$ left of u, having size $O(|\pi_i|)$.

The statement of the Strategy Extraction Theorem corresponds to the case when $i = 0$. The base case of the induction is for $i = s$. In this case σ^s is trivial since ϕ_s contains the line $L_s = \bot$, and we can define all the D^s_u as $u \leftarrow 0$.

We show now how to construct σ^{i-1}_u and D^{i-1}_u from σ^i_u and D^i_u:

• If L_i is derived by some Frege rule, then for each universal variable u we set $\sigma^{i-1}_u = \sigma^i_u$ and $D^{i-1}_u = D^i_u$.

• If L_i is the result of an application of a ∀red rule, that is $\dfrac{L_j}{L_j[u/B]}$, where u is the rightmost variable in L_j, $L_j[u/B]$ is a circuit in C using only variables on the left of u, and $L_j[u/B] = L_i$. Let $\boldsymbol{x}_{u'}$ denote the variables on the left of u' in the quantifier prefix of $\mathcal{Q}\phi$. Then we define

$$\sigma^{i-1}_{u'}(\boldsymbol{x}_{u'}) = \begin{cases} \sigma^i_{u'}(\boldsymbol{x}_{u'}) & \text{if } u' \neq u, \\ B(\boldsymbol{x}_u) & \text{if } u' = u \text{ and } L_j[u/B](\boldsymbol{x}_u) = 0, \\ \sigma^i_u(\boldsymbol{x}_u) & \text{if } u' = u \text{ and } L_j[u/B](\boldsymbol{x}_u) = 1. \end{cases}$$

Moreover for each $u' \neq u$ we set $D^{i-1}_{u'} = D^i_{u'}$ and we set D^{i-1}_u as follows:

```
if ¬L_j[u/B](x_u) then u ← B(x_u);
   else D^u_i(x_u).
```

We now check that for each u', $\sigma^{i-1}_{u'}$ respects all the properties of the inductive claim.

▶ *$\sigma^{i-1}_{u'}$ and $D^{i-1}_{u'}$ are well defined.* By construction $L_j[u/B]$ is a formula in the variables $\boldsymbol{x}$ left of u. This immediately implies that, for each universal variable u', the strategy $\sigma^{i-1}_{u'}$ is well defined and D^{i-1}_u is also well defined. By induction hypothesis D^i_u is a C-decision list, so D^{i-1}_u is also a C-decision list.

▶ σ^{i-1} *and* $D^{i-1}_{u'}$ *are constructed in linear time w.r.t.* $|\pi_{i-1}|$. This holds by inductive hypothesis and the fact that computing $\neg L_j(u/B)$ is linear in $|\pi_{i-1}|$.

▶ $D^{i-1}_{u'}$ *computes* $\sigma^{i-1}_{u'}$. For $u' \neq u$, by induction hypothesis, $D^{i-1}_{u'}$ computes $\sigma^{i}_{u'}$. The same happens, by construction, for $u' = u$.

▶ σ^{i-1} *is a winning strategy for* $\mathcal{Q}\phi_{i-1}$. Fix an assignment ρ to the existential variables of ϕ. Let τ_i be the complete assignment to existential and universal variables, constructed in response to ρ under the strategy σ^i. By induction hypothesis τ_i falsifies ϕ_i. We need to show that τ_{i-1} falsifies ϕ_{i-1}. To show this we distinguish again two cases.

If L_i is derived by some Frege rule, then $\sigma^{i-1} = \sigma^i$ and $\tau_{i-1} = \tau_i$. Hence by induction hypothesis, τ_i falsifies a conjunct from ϕ_i. To argue that τ_{i-1} also falsifies a conjunct from ϕ_{i-1} we only need to look at the case when the falsified conjunct is L_i. As L_i is false under τ_i and L_i is derived by a sound Frege rule, one of the parent formulas of L_i in the application of the Frege rule must be falsified as well. Hence τ_{i-1} falsifies ϕ_{i-1}.

Let now $L_i = L_j[u/B]$ for some $j < i$. In this case, our strategy σ^{i-1} changes the assignment τ_i only when τ_i made the universal player win by falsifying L_i. As we set u to $B(\tau_i(\boldsymbol{x}))$, the modified assignment τ_{i-1} falsifies L_j. Otherwise, if τ_i does not falsify L_i we keep $\tau_{i-1} = \tau_i$ and hence falsify one of the conjuncts of ϕ_{i-1} by induction hypothesis. □

From the proof of the Strategy Extraction Theorem it is clear that the size of the C-decision list computing the winning strategy extracted from the refutation π has size that is actually linear in the number of applications of the $\forall$red rule in π. More precisely, the size of the C-decision list computing the winning strategy for variable u corresponds exactly to the number of $\forall$red rules on u in π.

13.3 Formalised Strategy Extraction

It can be shown [32] that each strategy extraction theorem (Theorem 63) for C-Frege+$\forall$red can be formalised in C-Frege itself.

Theorem 64 (Beyersdorff, Pich [32]). *Let* C *be* AC^0, $\mathsf{AC}^0[p]$, TC^0, NC^1, *or* P/poly. *Given a* C-Frege+$\forall$red *refutation* π *of a QBF* $\exists x_1 \forall y_2 \ldots \exists x_n \forall y_n\, \phi(x_1, \ldots, x_n, y_1, \ldots, y_n)$ *where* $\phi \in \Sigma^q_0$, *we can construct in time* $|\pi|^{O(1)}$ *a* C-Frege *proof of*

$$\bigwedge_{i=1}^{n} (y_i \leftrightarrow C_i(x_1, \ldots, x_i, y_1, \ldots, y_{i-1})) \rightarrow \neg\phi(x_1, \ldots, x_n, y_1, \ldots, y_n)$$

for some circuits $C_i \in$ C. *(The depth of the* C-Frege *proof increases by a constant compared to the depth of the* C-Frege+$\forall$red *proof.)*

We demonstrate an application from [32] of the Strategy Extraction Theorem to obtain normal forms for C-Frege + $\forall$red proofs. The authors show that any C-Frege + $\forall$red refutation can be

efficiently rewritten as a C-Frege derivation followed essentially just by ∀red rules. This result is then used to show that in the ∀red rule it is sufficient to only substitute constants.

Theorem 65 (Beyersdorff, Pich [32]). *Let* C *be* AC^0, $\mathsf{AC}^0[p]$, TC^0, NC^1, *or* P/poly. *For any* C-Frege + ∀red *refutation* π *of a QBF* ψ *of the form*

$$\exists x_1 \forall y_2 \cdots \exists x_n \forall y_n \, \phi(x_1, \ldots, x_n, y_1, \ldots, y_n)$$

where $\phi \in \Sigma_0^q$, *there is a* $|\pi|^{O(1)}$*-size* C-Frege + ∀red *refutation of* ψ *starting with a* C-Frege *derivation of*

$$\bigvee_{i=1}^{n} (y_i \neq C_i(x_1, \ldots, x_i, y_1, \ldots, y_{i-1})),$$

from ϕ *for some circuits* $C_i \in$ C, *followed by* n *applications of the* ∀red *rule, gradually replacing the rightmost variable* y_i *by circuit* $C_i(x_1, \ldots, x_i, y_1, \ldots, y_{i-1})$ *and cutting the inequality* $y_i \neq C_i(x_1, \ldots, x_i, y_1, \ldots, y_{i-1})$ *out of the disjunction* $\bigvee_{i=1}^{n} (y_i \neq C_i(x_1, \ldots, x_i, y_1, \ldots, y_{i-1}))$.

An immediate consequence of Theorem 65 is the p-equivalence of C-Frege + ∀red and its tree-like version.

Corollary 12 (Beyersdorff, Pich [32]). *Let* C *be* AC^0, $\mathsf{AC}^0[p]$, TC^0, NC^1, *or* P/poly. *Then* C-Frege + ∀red *is p-equivalent to tree-like* C-Frege + ∀red.

Finally we further simplify C-Frege + ∀red so that every application of the ∀red rule only substitutes constants 0/1 instead of general circuits. We denote the resulting system as C-Frege + $\forall\mathsf{red}_{0,1}$. This shows that C-Frege + ∀red systems are indeed very robustly defined.

Theorem 66 (Beyersdorff, Pich [32]). *Let* C *be* AC^0, $\mathsf{AC}^0[p]$, TC^0, NC^1, *or* P/poly. *Then,* C-Frege + ∀red *and* C-Frege + $\forall\mathsf{red}_{0,1}$ *are p-equivalent.*

13.4 Separations and Lower Bounds via Circuit Complexity

We now introduce a class of QBFs defined from some circuits C_n computing a function f. Choosing different functions f, these formulas will form the basis of our lower bounds.

Definition 23 ($\mathcal{Q}$-C_n). *Let* n *be an integer and* C_n *be a circuit with inputs* $x_1, \ldots, x_n$. *Let* $t_1, \ldots, t_{m-1}$ *be a topological ordering of the internal gates of* C_n, *and let the output gate of* C_n *be* t_m. *We define*

$$\mathcal{Q}\text{-}C_n = \exists x_1 \cdots \exists x_n \forall u \exists t_1 \cdots \exists t_m (u \leftrightarrow \neg t_m) \wedge \bigwedge_{i=1}^{m} G_i,$$

where $u \leftrightarrow \neg t_m \equiv (u \vee t_m) \wedge (\neg u \vee \neg t_m)$ *and* G_i *expresses as a CNF the function computed in the circuit* C_n *at gate* i, *e.g. if node* t_i *computes the* $\wedge$ *of* t_j *and* t_k *then*

$$G_i = t_i \leftrightarrow (t_j \wedge t_k) \equiv (\neg t_i \vee t_j) \wedge (\neg t_i \vee t_k) \wedge (t_i \vee \neg t_j \vee \neg t_k),$$

similarly if gate i computes $\neg$, $\vee$, $\oplus$, MOD_p, T_k or some other Boolean function.

Informally, the QBF $\mathcal{Q}$-C_n expresses that there exists an input $\boldsymbol{x}$ such that $C_n(\boldsymbol{x})$ neither evaluates to 0 nor 1, an obvious contradiction as C_n computes a total function on $\{0,1\}^n$. Using these formulas together with the Strategy Extraction Theorem, we now establish a tight connection between the circuit class C and C-Frege$+\forall$red.

Theorem 67. *Let C be one of the circuit classes AC^0, $\mathsf{AC}^0[p]$, TC^0, NC^1, $\mathsf{P/poly}$ and let $(C_n)_{n\in\mathbb{N}}$ be a non-uniform family of circuits where C_n is a circuit with n inputs. Then the following implications hold:*

1. *if the QBFs $\mathcal{Q}$-C_n have C-Frege$+\forall$red refutations of size bounded by a function $q(n)$, then for each n, C_n is equivalent to a circuit C'_n where C'_n is of size $O(q(n))$ and uses the gates and depth allowed in C;*
2. *if $(C_n)_{n\in\mathbb{N}}$ is a polynomial-size circuit family from C then the QBFs $\mathcal{Q}$-C_n have polynomial-size refutations in C-Frege$+\forall$red.*

Proof. Regarding (i), by the Strategy Extraction Theorem and Proposition 4, if the QBF $\mathcal{Q}$-C_n has a refutation in C-Frege$+\forall$red of size S then a winning strategy for the universal player can be computed by a circuit $C'_n \in \mathsf{C}$ of size $O(S)$. We have that in $\mathcal{Q}$-C_n the quantifier prefix looks like $\exists x_1 \cdots \exists x_n \forall u \exists \boldsymbol{t}$. Now, by construction, $u \not\equiv C_n(x_1,\ldots,x_n)$, hence a winning strategy for the universal player must consist of playing $u = C_n(x_1,\ldots,x_n)$. This means that the circuit C'_n computing the winning strategy for the universal player is equivalent to the circuit C_n and the size bound follows.

Regarding (ii), let

$$\mathcal{Q}\text{-}C_n = \exists x_1 \cdots \exists x_n \forall u \exists t_1 \cdots \exists t_m (u \leftrightarrow \neg t_m) \wedge \phi_n,$$

where ϕ_n is a formula depending on the circuit C_n. By definition, the t_i are indexed w.r.t. a topological ordering of the nodes of C_n.

We prove, by induction on i, that there exists a circuit $D_i \in \mathsf{C}$ such that $t_i \leftrightarrow D_i$ is derivable in C-Frege with size polynomial in $|D_i|$. Suppose that t_i corresponds to a gate $\odot(t_{j_1},\ldots,t_{j_\ell})$ with fan-in ℓ, where $\odot$ could be an $\wedge, \vee, \neg, \oplus, MOD_p, T_k, \ldots$ from the gates allowed in the class C. By the inductive property we know that $t_{j_k} \leftrightarrow D_{j_k}$ is provable in C-Frege with proofs of size polynomial in $|D_{j_k}|$. Moreover, C-Frege is able to prove

$$\frac{t_{j_1} \leftrightarrow D_{j_1} \qquad \cdots \qquad t_{j_\ell} \leftrightarrow D_{j_\ell} \qquad t_i \leftrightarrow \odot(t_{j_1},\ldots,t_{j_\ell})}{t_i \leftrightarrow \odot(D_{j_1},\ldots,D_{j_\ell})} \ .$$

Let then $D_i = \odot(D_{j_1},\ldots,D_{j_\ell})$. At the m-th step C-Frege proves that $t_m \leftrightarrow D_m$, from which follows that

$$\frac{t_m \leftrightarrow D_m \qquad u \leftrightarrow \neg t_m}{u \leftrightarrow \neg D_m} \ .$$

Since now u is universal and the innermost variable of $u \leftrightarrow \neg D_m$, we can apply the $\forall$red rule and get $0 \leftrightarrow \neg D_m$, $1 \leftrightarrow \neg D_m$, which leads to an immediate contradiction in the QBF proof system C-Frege + $\forall$red.

In particular, a Boolean function f is computable by polynomial-size C circuits if and only if $\mathcal{Q}$-C_n have polynomial-size C-Frege refutations for each choice of Boolean circuits $(C_n)_{n\in\mathbb{N}}$ computing f. Note that the circuits C_n are not necessarily circuits from the class C.

In the remainder of this section we apply Theorem 67 to a number of circuit classes and transfer circuit lower bounds to proof size lower bounds. □

Lower Bounds for Bounded-depth QBF Frege Systems. PARITY is one of the best-studied functions in terms of its circuit complexity. With Theorem 67 we can immediately transfer circuit lower bounds for PARITY to $\mathsf{AC}^0[p]$-Frege + $\forall$red, regardless of the encoding for PARITY.

Corollary 13 ($\mathcal{Q}$-PARITY lower bounds). *Let C_n be a family of polynomial-size circuits computing* PARITY. *For each odd prime p the QBFs $\mathcal{Q}$-C_n require proofs of exponential size in* $\mathsf{AC}^0[p]$-Frege + $\forall$red.

Proof. The exponential lower bound for the proof size in $\mathsf{AC}^0[p]$-Frege + $\forall$red follows from Theorem 67 and the fact that for each odd prime p any family of bounded-depth circuits with MOD_p gates computing PARITY must be of exponential size [102, 112].

We highlight that non-trivial lower bounds for $\mathsf{AC}^0[p]$-Frege are one of the major open problems in propositional proof complexity. We complement the lower bound in Corollary 13 with an upper bound for arbitrary NC^1 encodings of PARITY in Frege + $\forall$red. □

Corollary 14 ($\mathcal{Q}$-PARITY upper bounds). *Let C_n be a family of* NC^1 *circuits computing* PARITY. *Then the QBFs $\mathcal{Q}$-C_n have polynomial-size proofs in* Frege + $\forall$red.

Proof. By a result of [93], PARITY can be computed by circuits in NC^1. Hence if we consider a family C_n of NC^1 circuits computing PARITY then the polynomial upper bound in Frege + $\forall$red follows immediately from Theorem 67. □

In fact, this upper bound can be improved to the QBF proof system $\mathsf{AC}^0[2]$-Frege + $\forall$red, albeit not for arbitrary NC^1-encodings of PARITY, as it is not clear how these could be handled in bounded depth. For this purpose, we consider explicit QBFs for PARITY, which can be built from its inductive definition $\text{PARITY}(x_1, \ldots, x_n) = \text{PARITY}(x_1, \ldots, x_{n-1}) \oplus x_n$. This leads to the QBFs

$$\Phi_n = \exists x_1 \cdots \exists x_n \forall u \exists t_2 \cdots \exists t_n \, (t_2 \leftrightarrow (x_1 \oplus x_2)) \wedge \bigwedge_{i=3}^{n} (t_i \leftrightarrow (t_{i-1} \oplus x_i)) \wedge (u \leftrightarrow \neg t_n),$$

where $a \leftrightarrow (b \oplus c) \equiv (\neg a \vee \neg b \vee \neg c) \wedge (\neg a \vee b \vee c) \wedge (a \vee \neg b \vee c) \wedge (a \vee b \vee \neg c)$. This formulation of $\mathcal{Q}$-PARITY was considered in Chapter 5.

Corollary 15. *The* PARITY*-formulas Φ_n require refutations of exponential size in* $\mathsf{AC}^0[p]$-Frege $+\forall$red *for each odd prime p, but it have polynomial-size* $\mathsf{AC}^0[2]$-Frege$+\forall$red *refutations.*

Proof. The lower bound follows as in Corollary 13. For the upper bound we cannot use Theorem 67, but need to give a more direct proof. Without loss of generality we can assume that our $\mathsf{AC}^0[2]$-Frege $+\forall$red system uses the connectives $\{\wedge, \vee, \neg, \leftrightarrow, \oplus\}$.

Then it is easy to see, by induction on i, that Frege proves $t_i \leftrightarrow \oplus(x_1, x_2, \ldots, x_i)$ with a proof of size linear in i. Hence, similarly to what was done in Theorem 67, we get

$$u \leftrightarrow \neg \oplus (x_1, x_2, \ldots, x_n).$$

Then u is the rightmost variable; hence by the $\forall$red rule we have

$$1 \leftrightarrow \neg \oplus (x_1, x_2, \ldots, x_n) \quad \text{and} \quad 0 \leftrightarrow \neg \oplus (x_1, x_2, \ldots, x_n),$$

which gives an immediate contradiction. □

In fact, we can further strengthen Corollary 15 and use Smolensky's circuit lower bounds for an even more ambitious separation of *all* $\mathsf{AC}^0[p]$-Frege$+\forall$red systems. For this we consider the function

$$MOD_p(x_1, \ldots, x_n) = \begin{cases} 1 \text{ if } \sum_{i=1}^n x_i \equiv 0 \pmod p \\ 0 \text{ otherwise.} \end{cases}$$

For $r \leq p-1$ let

$$MOD_{p,r}(x_1, \ldots, x_n) = \begin{cases} 1 \text{ if } \sum_{i=1}^n x_i \equiv r \pmod p \\ 0 \text{ otherwise.} \end{cases}$$

If we want to use MOD_p for a separation of $\mathsf{AC}^0[p]$-Frege$+\forall$red and $\mathsf{AC}^0[q]$-Frege$+\forall$red for different primes p, q, then MOD_p has to be encoded as a QBF in the language common to both proof systems, which means that we cannot use MOD_p or MOD_q gates. As for PARITY, an arbitrary NC^1 encoding as in Corollary 13 will also not work (this would just give upper bounds in Frege $+\forall$red), so we need to devise again explicit QBF encodings for MOD_p. Such QBFs can be built using the fact that MOD_p, that is $MOD_{p,0}$, can be defined for $r \neq 0$ by

$$MOD_{p,r}(x_1, \ldots, x_i) = (MOD_{p,r}(x_1, \ldots, x_{i-1}) \wedge \neg x_i) \vee (MOD_{p,r-1}(x_1, \ldots, x_{i-1}) \wedge x_i),$$

and for $r = 0$ by

$$MOD_{p,0}(x_1, \ldots, x_i) = (MOD_{p,0}(x_1, \ldots, x_{i-1}) \wedge \neg x_i) \vee (MOD_{p,p-1}(x_1, \ldots, x_{i-1}) \wedge x_i).$$

Using variables s_i^r for $MOD_{p,r}(x_1,\dots,x_i)$ this leads to the QBFs

$$\Theta_n^p = \exists x_1 \cdots \exists x_n \forall u \exists s_1^0 \exists s_1^1 \exists s_2^0 \exists s_2^1 \exists s_2^2 \cdots \exists s_n^0 \cdots \exists s_n^{p-1} (u \leftrightarrow \neg s_n^0) \wedge (s_1^1 \leftrightarrow x_1) \wedge (s_1^0 \leftrightarrow \neg x_1) \wedge$$

$$\bigwedge_{\substack{1<i\le n\\ 0<r\le p-1}} \Big(s_i^r \leftrightarrow (s_{i-1}^r \wedge \neg x_i) \vee (s_{i-1}^{r-1} \wedge x_i)\Big) \wedge \bigwedge_{1<i\le n} \Big(s_i^0 \leftrightarrow (s_{i-1}^0 \wedge \neg x_i) \vee (s_{i-1}^{p-1} \wedge x_i)\Big).$$

Corollary 16. *For each pair p, q of distinct primes the MOD_p-formulas Θ_n^p require proofs of exponential size in* $\mathsf{AC}^0[q]$-$\mathsf{Frege}+\forall\mathsf{red}$*, but have polynomial-size proofs in* $\mathsf{AC}^0[p]$-$\mathsf{Frege}+\forall\mathsf{red}$.

Proof. The exponential lower bound for the QBF proof system $\mathsf{AC}^0[q]$-$\mathsf{Frege}+\forall\mathsf{red}$ follows from Theorem 67 together with the result from [102, 112] that for distinct primes p, q any family of bounded-depth circuits with MOD_q gates computing MOD_p must be of exponential size.

Regarding the upper bound, without loss of generality we can assume that our $\mathsf{AC}^0[p]$-Frege system uses the connectives $\{\wedge,\vee,\neg,\leftrightarrow,MOD_p\}$. Then it is easy to see, by induction on i, that $\mathsf{AC}^0[p]$-Frege proves

$$s_i^r \leftrightarrow MOD_p(x_1,\dots,x_i,\underbrace{1,1,\dots,1}_{p-r}),$$

with a proof of size linear in i. Hence, similarly to what was done in Theorem 67 and Corollary 15, we get

$$u \leftrightarrow \neg MOD_p(x_1,\dots,x_n,\underbrace{1,1,\dots,1}_{p}).$$

Then u is the rightmost variable; hence by the $\forall\mathsf{red}$ rule we have

$$1 \leftrightarrow \neg MOD_p(x_1,\dots,x_n,\underbrace{1,1,\dots,1}_{p}) \quad\text{and}\quad 0 \leftrightarrow \neg MOD_p(x_1,\dots,x_n,\underbrace{1,1,\dots,1}_{p}),$$

which gives an immediate contradiction. □

Another notorious function in circuit complexity is MAJORITY. Again we can transform circuit lower bounds to proof size lower bounds for arbitrary encodings of MAJORITY.

Corollary 17 (lower bounds for $\mathcal{Q}$-MAJORITY). *Let C_n be a family of polynomial-size circuits computing* $\textsc{Majority}(x_1,\dots,x_n)$*. Then for every prime p, the QBFs $\mathcal{Q}$-C_n require proofs of exponential size in* $\mathsf{AC}^0[p]$-$\mathsf{Frege}+\forall\mathsf{red}$.

Proof. The lower bound follows again applying Theorem 67 and the fact that MAJORITY requires exponential-size bounded-depth circuits with MOD_p gates [102, 112]. □

For general encodings, we can again show $\mathsf{Frege}+\forall\mathsf{red}$ upper bounds.

Corollary 18 ($\mathcal{Q}$-MAJORITY upper bounds). *Let C_n be a family of* NC^1 *circuits computing* $\textsc{Majority}(x_1,\dots,x_n)$*. Then the QBFs $\mathcal{Q}$-C_n have polynomial-size proofs in the QBF proof system* $\mathsf{Frege}+\forall\mathsf{red}$.

Proof. By a result of [93], the function MAJORITY is computable in NC^1 and hence $\mathcal{Q}$-C_n are well defined. The upper bound then follows from Theorem 67. □

As for the MOD_p functions, we can improve on this upper bound by considering explicit QBF encodings of MAJORITY, thereby even obtaining a separation of $\mathsf{AC}^0[p]$-Frege+$\forall$red systems from TC^0-Frege+$\forall$red.[3] Explicit QBFs for MAJORITY can be defined using the following property of the k-threshold function.

$$T_k(x_1,\dots,x_i) \equiv T_k(x_1,\dots,x_{i-1}) \vee (T_{k-1}(x_1,\dots,x_{i-1}) \wedge x_i). \tag{5}$$

Using variables t_k^i for $T_k(x_1,\dots,x_i)$ this gives rise to the QBFs Ψ_n which is given by

$$\exists x_1 \cdots \exists x_n \forall u \exists t_1^1 \cdots \exists t_{n/2}^n (u \leftrightarrow \neg t_{n/2}^n) \wedge \bigwedge_{i\le n} t_0^i \wedge (t_1^1 \leftrightarrow x_1) \wedge \bigwedge_{\substack{k\le n/2\\ i\le n}} \left(t_k^i \leftrightarrow t_k^{i-1} \vee (t_{k-1}^{i-1} \wedge x_i)\right).$$

Corollary 19. *For each prime p the* MAJORITY*-based formulas Ψ_n require proofs of exponential-size in the QBF proof system* $\mathsf{AC}^0[p]$-Frege+$\forall$red*, but have polynomial-size proofs in* TC^0-Frege +$\forall$red.

Proof. The exponential lower bound from [102, 112] will give us the exponential lower bound w.r.t. the size of Ψ_n in $\mathsf{AC}^0[p]$-Frege+$\forall$red, since the size of Ψ_n is $O(n^2)$.

Regarding the polynomial-size proof of the QBF formula Ψ_n in TC^0-Frege+$\forall$red we can proceed similarly as for PARITY in Frege. The crucial feature here is that T_k are, by definition of TC^0, in the language of TC^0-Frege. Hence $T(x_1,\dots,x_i) \equiv T_k(x_1,\dots,x_{i-1}) \vee (T_{k-1}(x_1,\dots,x_{i-1}) \wedge x_i)$ can be used to prove $t_k^j \leftrightarrow T_k(x_1,\dots,x_j)$ and we can easily refute Ψ_n in TC^0-Frege+$\forall$red. □

We note that a separation of $\mathsf{AC}^0[p]$-Frege from TC^0-Frege constitutes a major open problem in propositional proof complexity as we are currently lacking lower bounds for $\mathsf{AC}^0[p]$-Frege.

13.5 Lower Bounds for Constant-depth QBF Frege Systems

We now aim at a fine-grained analysis of AC^0-Frege by studying its subsystems AC^0_d-Frege. Our next result is a version of Theorem 67, however, we need to be a bit more careful for circuits of fixed depth d.

Theorem 68. *Let $(C_n)_{n\in\mathbb{N}}$ be a non-uniform family of circuits where C_n is a circuit with n inputs. Then the following implications hold:*

(i) if the QBFs $\mathcal{Q}$-C_n have AC^0_d-Frege+$\forall$red *refutations of size bounded by a function $q(n)$, then for each n, C_n is equivalent to a depth-$(d+2)$ circuit C'_n of size $O(q(n))$;*

[3] Clearly, such a separation already follows from Corollary 16 together with the simulation of $\mathsf{AC}^0[p]$-Frege+$\forall$red by TC^0-Frege+$\forall$red. Here we will prove the stronger result that all these systems are separated by *one* natural principle, namely MAJORITY.

(ii) if $(C_n)_{n\in\mathbb{N}}$ is a family of polynomial-size depth-d circuits, then the QBFs $\mathcal{Q}$-C_n have polynomial-size refutations in AC^0_d-Frege$+\forall$red.

Proof. The proof of (i) follows the proof of the analogous statement of Theorem 67. The Strategy Extraction Theorem in this case tell us that from refutations of $\mathcal{Q}$-C_n in AC^0_d-Frege$+\forall$red of size S we can extract a winning strategy for the universal player that can be computed by AC^0_d-decision lists of size $O(S)$. By Proposition 4, this means that the winning strategy can be also computed by AC^0_{d+2} circuits and the size upper bound follows.

The proof of point (ii) follows the proof of the analogous statement of Theorem 67. That proof will give us that $\mathcal{Q}$-C_n has polynomial-size refutations in AC^0_{d+2}-Frege$+\forall$red. Here we want to prove that $\mathcal{Q}$-C_n has actually polynomial-size proofs in AC^0_d-Frege$+\forall$red. Without loss of generality suppose that the last gate t_m of C_n is an $\bigwedge$, that is

$$\mathcal{Q}\text{-}C_n = \exists x_1 \cdots \exists x_n \forall u \exists t_1 \cdots \exists t_m (u \leftrightarrow \neg t_m) \wedge (t_m \leftrightarrow \bigwedge_{j\leq \ell} t_{i_j}) \wedge \phi_n,$$

where each t_{i_j} is an $\bigvee$ gate and ϕ_n is the encoding of the rest of the circuit C_n. We clearly have that

$$\frac{u \leftrightarrow \neg t_m \qquad t_m \leftrightarrow \bigwedge_{j\leq \ell} t_{i_j}}{u \leftrightarrow \bigvee_{j\leq \ell} \neg t_{i_j}}$$

From which we obtain both

$$u \vee \bigwedge_{j\leq \ell} t_{i_j},$$

$$\neg u \vee \bigvee_{j\leq \ell} \neg t_{i_j}.$$

Now we can proceed, similarly as in Theorem 67. By induction (on the depth of C_n) AC^0_d-Frege is able to substitute t_{i_j} with D_{i_j} where D_{i_j} is an AC^0_{d-1}-formula over the $x_1, \ldots, x_n$ variables starting with an $\bigvee$. More precisely by induction we can prove that AC^0_d-Frege proves both

$$t_{i_j} \vee \neg D_{i_j},$$

$$\neg t_{i_j} \vee D_{i_j}.$$

Hence from our derivations follows that $\neg u \vee \bigvee_{j\leq \ell} \neg D_{i_j}$, which is an AC^0_d-formula only over the variables $u, x_1, \ldots, x_n$. Hence by the $\forall$red rule we get

$$\bigvee_{j\leq \ell} \neg D_{i_j}.$$

We get first that $\bigwedge_{j\leq \ell}(u \vee t_{i_j})$ and then we get $\bigwedge_{j\leq \ell}(u \vee D_{i_j})$, which, again, is an AC^0_d-formula over the variables $u, x_1, \ldots, x_n$. By the $\forall$red rule we get

$$\bigwedge_{j\leq \ell} D_{i_j}.$$

From these follows immediately a contradiction. □

From Theorem 67 we obtain a wealth of lower bounds for $Res + \forall\mathsf{red}$.

Corollary 20. *Let $f(x_1, \ldots, x_n)$ be a Boolean function requiring exponential-size depth-3 circuits and let $(C_n)_{n\in\mathbb{N}}$ be polynomial-size circuits (of unbounded depth) computing f. Then the QBFs $\mathcal{Q}$-C_n require exponential-size refutations in AC^0_1-$\mathsf{Frege}+\forall\mathsf{red}$ and hence, in particular, in $Res + \forall\mathsf{red}$.*

We now prove a separation of constant-depth $\mathsf{Frege}+\forall\mathsf{red}$ systems. For this we employ the Sipser functions separating the hierarchy of constant-depth circuits. We quote the definition of the $\textsc{Sipser}_d$ function from [38]:

$$\textsc{Sipser}_d = \bigwedge_{i_1\le m_1} \bigvee_{i_2\le m_2} \bigwedge_{i_3\le m_3} \cdots \bigodot_{i_d\le m_d} x_{i_1 i_2 i_3 \ldots i_d},$$

where $\bigodot = \bigvee$ or $\bigwedge$ depending on the parity of d. The variables $x_1, \ldots, x_n$ appear as $x_{i_1 i_2 i_3 \ldots i_d}$ for $i_j \le m_j$, where $m_1 = \sqrt{m/\log m}$, $m_2 = m_3 = \cdots = m_{d-1} = m$, $m_d = \sqrt{dm \log m/2}$ and $m = (n\sqrt{2/d})^{1/(d-1)}$.

Corollary 21. *Fix an integer $d \ge 2$. Let $(C^n_d)_{n\in\mathbb{N}}$ be a family of polynomial-size depth-$(d+3)$ circuits computing the function $\textsc{Sipser}_{d+3}(x_1, \ldots, x_n)$. Then the QBFs $\mathcal{Q}$-C^n_d need exponential-size proofs in AC^0_d-$\mathsf{Frege}+\forall\mathsf{red}$, but have polynomial-size proofs in AC^0_{d+3}-$\mathsf{Frege}+\forall\mathsf{red}$.*

Proof. The lower bound follows from Theorem 68 and from the result that for every d, $\textsc{Sipser}_{d+3}$ needs exponential-size depth-$(d+2)$ circuits [66]. Regarding the upper bound, by construction C^n_d has depth $d+3$ and polynomial-size. Hence, by Theorem 68, the family $\mathcal{Q}$-C^n_d has polynomial-size proofs in AC^0_{d+3}-$\mathsf{Frege}+\forall\mathsf{red}$. □

Note that the gap of size 1 in the circuit separation of [66] increases to a gap of size 3 in our proof system separation, due to the transformation in Proposition 4. We highlight that in contrast to Corollary 21 where our separating formulas are CNFs, a separation of the depth-d Frege hierarchy with formulas of depth independent of d is a major open problem in propositional proof complexity.

13.6 Characterizing QBF Frege and Extended Frege Lower Bounds

We finally address the question of lower bounds for $\mathsf{Frege}+\forall\mathsf{red}$ or even $\mathsf{eFrege}+\forall\mathsf{red}$. Our next result states that achieving such lower bounds unconditionally will either imply a major breakthrough in circuit complexity or a major breakthrough in classical proof complexity and vice versa. (Notice that it might be much easier to obtain the disjunction than any of the disjuncts.)

Theorem 69. *Let C be either $\mathsf{P/poly}$ or NC^1. C-$\mathsf{Frege}+\forall\mathsf{red}$ is not polynomially bounded if and only if $\mathsf{PSPACE} \not\subseteq \mathsf{C}$ or C-Frege is not polynomially bounded.*[4]

[4] By NC^1 we mean *non-uniform* NC^1. Note that by the space hierarchy theorem it is known that $\mathsf{PSPACE} \not\subseteq$ uniform NC^1, but this does not suffice for $\mathsf{Frege}+\forall\mathsf{red}$ lower bounds.

Proof. Clearly if C-Frege is not polynomially bounded then C-Frege + ∀red is not polynomially bounded. If PSPACE $\not\subseteq$ C then let f be a Boolean function in PSPACE but not in C. Since QBF is PSPACE-complete there exists a QBF $\mathcal{Q}\boldsymbol{w}\phi(\boldsymbol{w}, x_1, \ldots, x_n)$ with a CNF ϕ such that

$$f(x_1, \ldots, x_n) \equiv \mathcal{Q}\boldsymbol{w}\phi(\boldsymbol{w}, x_1, \ldots, x_n).$$

We define

$$\mathcal{Q}\text{-}f_n = \exists x_1 \cdots \exists x_n \forall u (u \leftrightarrow \mathcal{Q}\boldsymbol{w}\phi(\boldsymbol{w}, x_1, \ldots, x_n)),$$

which can be rewritten into formulas Θ_n in prenex form. Notice that the only winning strategy for the universal player on both $\mathcal{Q}\text{-}f_n$ and Θ_n is to compute $u = f(x_1, \ldots, x_n)$. Therefore, the Strategy Extraction Theorem together with $f \notin$ C immediately implies super-polynomial lower bounds for Θ_n in C-Frege + ∀red.

The opposite direction was shown first in [32] and the argument is as follows. We assume that C-Frege + ∀red is not polynomially bounded. Then there is a sequence of true QBFs $Q\,\psi_n$ such that $\neg Q\,\psi_n$ do not have polynomial-size refutations in C-Frege + ∀red. Let $Q\,\psi_n$ have the form

$$\forall x_1 \exists y_1 \ldots \forall x_n \exists y_n\, \psi_n(x_1, \ldots, x_n, y_1, \ldots, y_n).$$

If PSPACE $\not\subseteq$ C, we are done. Otherwise, there are polynomial-size circuits C_i witnessing the existential quantifiers in $Q\,\psi_n$. That is, for any $x_1, \ldots, x_n, y_1, \ldots, y_n$

$$\bigwedge_{i=1}^{n} (y_i \leftrightarrow C_i(x_1, \ldots, x_i, y_1, \ldots, y_{i-1})) \rightarrow \psi_n(x_1, \ldots, x_n, y_1, \ldots, y_n).$$

We claim that these are a sequence of tautologies without polynomial-size C-Frege proofs. Otherwise, having $\neg\psi_n$, C-Frege can derive $\bigvee_i y_i \neq C_i(x_1, \ldots, x_i, y_1, \ldots, y_{i-1})$ by a polynomial-size proof, and so as in Theorem 65, C-Frege + ∀red can efficiently refute $\neg Q\,\psi_n$. □

We remark that we do have a separation between *uniform* NC^1 and PSPACE, because uniform $\mathsf{NC}^1 \subseteq$ LOGSPACE and LOGSPACE $\neq$ PSPACE by the space hierarchy theorem. Therefore, choosing $f \in$ PSPACE $\setminus$ uniform NC^1 and considering the prenex formulas Θ_n arising from $\mathcal{Q}\text{-}f_n$ we can infer the weaker result that Frege + ∀red has no uniform short proofs of Θ_n.

Chapter 14

Conclusion

In this thesis we introduce improvements to QBF proof complexity in the following areas:

1. We introduce new proof systems to better understand and characterise solving and complexity in QBF and DQBF.
2. We introduce the most current QBF lower bounds and we complete the simulation/separation structure of the main QBF directed acyclic graph (DAG) resolution systems.
3. We lift lower bound techniques from propositional to QBF logic, where we have particular success in tree-like calculi. However for DAG like systems and QBF proof complexity in general we find that a very different picture is emerging.
4. We exhibit a new interesting lower bound technique for QBF calculi of varying power– strategy extraction.

New Proof Systems. Prior to this work, a handful of proof systems existed for QBF. The QCDCL style proof systems Q-Res and each of its improvements LD-Q-Res, QU-Res and LQU$^+$-Res were defined. The foundational expansion-based calculus– $\forall$Exp+Res was developed due to its relation to RAReQS and stronger calculi such as the sequent calculus G and the first order resolution EPR were also known.

In Chapter 4 we built natural expansion based QBF proof systems– IR-calc and IRM-calc that hybridised the quantifier handling in QCDCL and expansion based solving. Some progress towards bringing the solvers together can be seen in the QESTOS solver [33] which uses DPLL conflicts to generate the strategies used in CEGAR solvers. However, in IR-calc we added partial expansion instead of full expansion, this is not yet fully implemented in practical solving.

IR-calc is also naturally applicable to DQBF and forms a sound and complete resolution calculus D-IR-calc as seen in Chapter 10. There have been recent advances in DQBF solving [57,58,62,117]. We believe that D-IR-calc will be an important calculus when relating solvers to proof systems in this area. DQBF is also related to dependency-schemes, which while not explored in this thesis are increasingly prevalent in solving with solvers such as DepQBF. Indeed dependency schemes have received recent interest from a theoretical point of view. Slivovsky and Szeider showed a sound QBF resolution calculus that incorporates dependency schemes [111]. This was further adapted to long-distance resolution [96]. The work by Beyersdorff and Blinkhorn [16] details the conditions on making dependency schemes sound in proof systems.

In addition we introduced the P + $\forall$red system (Chapter 13) which could also be seen as a calculus that is of practical importance in the future. If we suppose SAT solving advances beyond the resolution system to system P, then QCDCL solvers may correlate to P + $\forall$red. In particular, we

see CP+∀red in Chapter 12, where cutting planes is a useful system related to integer programming. We also introduce a hierarchy of C-Frege + ∀red systems, for the powerful Frege systems.

From C-Frege + ∀red systems, we get a tighter grasp on the connections between QBF proof complexity and circuit complexity. C-Frege + ∀red systems naturally match to complexity classes C both in description and in lower bounds, which is a connection not yet established in propositional proof systems. Furthermore, formalisations of strategy extraction show that these systems characterise hardness in Frege and circuit complexity, showing that lower bounds can only be obtained via circuit complexity or propositional proof complexity.

New Lower Bounds. The states of lower bounds has improved significantly in QBF, previously only a few lower bounds were known for QBF calculi:

- the KBKF formulas from [77] that show a lower bound for input Q-Res, two modifications of the KBKF formulas were presented in [11] that mutually separate QU-Res and LD-Q-Res.
- the formulas from [73] that show a lower bound for ∀Exp+Res but are easy for Q-Res.
- any propositional lower bounds lifted as purely existential QBF.

We add a significant number of new lower bounds which are as follow:

- the QPARITY lower bounds for Q-Res, QU-Res (Chapter 5) and weak extended Q-Res (Chapter 11), as well as their modifications for hardness in LD-Q-Res and LQU$^+$-Res.
- more generally, the $\mathcal{Q}$-C_n lower bounds which work for a range of C-Frege + ∀red proof systems (Chapter 13).
- the clique co-clique formulas which are genuine QBFs for the QBF calculi with the monotone feasible interpolation property (Chapters 8 and 12).
- We also show that the KBKF formulas do indeed provide a lower bound to Q-Res because they provide a lower bound to IR-calc (Chapter 6). KBKF also provides a lower bound to tree-like QU-Res (Chapter 7). The modification of KBKF in [11] provide a lower bound to IRM-calc.

DAG versus Tree-like Calculi. In Chapters 4, 5 and 6 we show that in QBF resolution systems we completely separate the simulation structure for DAG-like QBF proof systems. To show the separations we used new techniques for QBF such as the strategy extraction technique (Chapter 5) along with the examples of [77] (Chapter 6).

It is quite important to note that for the DAG like QBF systems the size-width relation fails. This is a major technique in propositional proof complexity. Instead the stand-out success that led us to many new lower bounds comes from connections to hardness in circuit complexity. This allows us to look only at higher level properties of the calculi, which we believe is more illuminating on sources of difficulty than the previous counting techniques in QBF proof complexity.

In addition to strategy extraction, the other technique that works for DAG-like QBF calculi is the feasible interpolation technique, although we see that this can somewhat be seen as a special case of

strategy extraction (Chapter 8). This so far is the sole case of a general lower bound technique that lifts from propositional proof complexity to DAG-like QBF systems.

For tree-like QBF systems, we find that lower bound techniques generally do transfer from propositional proof complexity. We see this in two examples: 1.) The game technique for Q-Res and QU-Res (Chapter 7) 2.) The size-width relation for ∀Exp+Res and IR-calc. A similar game technique for QBF was shown in [40], this was used on an ensemble of QU-Res systems with oracles in the polynomial hierarchy.

Just as the DAG like systems have simulation and separations, these relations are also possible in the tree-like cases. It is known that tree-like ∀Exp+Res and IR-calc are equivalent despite the fact that they are strictly separated in DAG-like resolution. We also know all the simulations that are present in the DAG-like case remain. However all other potential separations are unknown. It is even possible that all the QBF resolution systems are equivalent on tree-like proofs. This is important as the distinction between tree-like and DAG-like has practical relevance. Due to the nature of DPLL solving, hard examples are likely to have tree-like traces. Therefore there are still some important questions about how this affects expansion versus QCDCL solving. Since in the tree-like calculi, ∀Exp+Res p-simulates IR-calc it also p-simulates Q-Res, and the converse is still unknown, despite the fact DAG Q-Res does not simulate ∀Exp+Res.

Strategy Extraction. The most successful technique in QBF proof complexity is the strategy extraction technique (Chapters 5,11,12 and 13). For false QBFs, the universal player is guaranteed a winning strategy and this can be represented by a circuit. In strategy extraction for proofs this circuit has to be extracted in polynomial time from the proof.

Strategy extraction is also possible in solvers. In some sense, strategy extraction is desirable in solvers. Not only can the correctness of a solving instance be verified by a strategy, for QBF such as those that relate to games the winning strategy may be just as important as the result. On the other hand, we show that strategy extraction can lead to lower bounds, we show this in the case of proof systems but the same is true for solvers.

Strategy extraction is possible in polynomial time for all the QBF resolution calculi (cf. Chapter 4 and [11]). The key to lower bounds in Q-Res and QU-Res is that it can be done in restrictive AC^0 circuits of which super-polynomial lower bounds are known.

Strategy extraction has a strong relationship with the universal reduction rule and our new P + ∀red calculi. We find that even extended Frege systems augmented with the reduction rule have strategy extraction in polynomial time (Chapter 13). This property can create conditional and unconditional lower bounds.

Comparisons of strong proof systems for QBFs were made in [52], including comparisons to IR-calc and IRM-calc. In [32] it was shown that strategy extraction characterises hardness for strong systems such as Frege + ∀red systems and beyond. This means that lower bounds for stronger systems are limited by either propositional proof complexity or circuit complexity. The

work by Beyersdorff et. al [30] shows further characterisations for even weaker systems, with a third possibility of hardness by quantifier alternations.

References

1. M. AJTAI, *Σ^1_1-formulae on finite structures*, Annals of Pure and Applied Logic, 24 (1983), pp. 1–48.
2. M. AJTAI, *The complexity of the pigeonhole-principle*, Combinatorica, 14 (1994), pp. 417–433.
3. N. ALON AND R. B. BOPPANA, *The monotone circuit complexity of boolean functions*, Combinatorica, 7 (1987), pp. 1–22.
4. C. ANSOTEGUI, C. P. GOMES, AND B. SELMAN, *The achilles' heel of QBF*, in Association for the Advancement of Artificial Intelligence (AAAI), vol. 2, 2005, pp. 2–1.
5. S. ARORA AND B. BARAK, *Computational Complexity – A Modern Approach*, Cambridge University Press, 2009.
6. A. ATSERIAS AND V. DALMAU, *A combinatorial characterization of resolution width*, Journal of Computer and System Sciences, 74 (2008), pp. 323–334.
7. S. AZHAR, G. PETERSON, AND J. REIF, *Lower bounds for multiplayer non-cooperative games of incomplete information*, Journal of Computers and Mathematics with Applications, 41 (2001), pp. 957–992.
8. L. BACHMAIR AND H. GANZINGER, *Resolution theorem proving*, in Handbook of Automated Reasoning, J. A. Robinson and A. Voronkov, eds., Elsevier and MIT Press, 2001, pp. 19–99.
9. V. BALABANOV, H.-J. K. CHIANG, AND J.-H. R. JIANG, *Henkin quantifiers and boolean formulae: A certification perspective of DQBF*, Theoretical Computer Science, 523 (2014), pp. 86–100.
10. V. BALABANOV AND J.-H. R. JIANG, *Unified QBF certification and its applications*, Formal Methods in System Design, 41 (2012), pp. 45–65.
11. V. BALABANOV, M. WIDL, AND J.-H. R. JIANG, *QBF resolution systems and their proof complexities*, in Proc. 17th International Conference on Theory and Applications of Satisfiability Testing, 2014, pp. 154–169.
12. P. BEAME AND T. PITASSI, *Propositional proof complexity: Past, present, and future*, in Current Trends in Theoretical Computer Science: Entering the 21st Century, G. Paun, G. Rozenberg, and A. Salomaa, eds., World Scientific Publishing, 2001, pp. 42–70.
13. E. BEN-SASSON AND A. WIGDERSON, *Short proofs are narrow - resolution made simple*, Journal of the ACM, 48 (2001), pp. 149–169.
14. M. BENEDETTI AND H. MANGASSARIAN, *QBF-based formal verification: Experience and perspectives*, JSAT, 5 (2008), pp. 133–191.
15. O. BEYERSDORFF, *On the correspondence between arithmetic theories and propositional proof systems a survey*, Mathematical Logic Quarterly, 55 (2009), pp. 116–137.
16. O. BEYERSDORFF AND J. BLINKHORN, *Dependency schemes in QBF calculi: Semantics and soundness*, in Principles and Practice of Constraint Programming - 22nd International Conference (CP), 2016, pp. 96–112.
17. O. BEYERSDORFF, I. BONACINA, AND L. CHEW, *Lower bounds: From circuits to QBF proof systems*, in Proc. ACM Conference on Innovations in Theoretical Computer Science (ITCS'16), ACM, 2016, pp. 249–260.
18. O. BEYERSDORFF, L. CHEW, AND M. JANOTA, *On unification of QBF resolution-based calculi*, in MFCS, II, 2014, pp. 81–93.
19. ———, *Proof complexity of resolution-based QBF calculi*, in Proc. Symposium on Theoretical Aspects of Computer Science, LIPIcs series, 2015, pp. 76–89.
20. ———, *Extension variables in QBF resolution*, in Beyond NP, Papers from the 2016 AAAI Workshop, 2016.
21. O. BEYERSDORFF, L. CHEW, M. MAHAJAN, AND A. SHUKLA, *Feasible interpolation for QBF resolution calculi*, in Proc. International Colloquium on Automata, Languages, and Programming (ICALP'15), Springer, 2015, pp. 180–192.
22. ———, *Are short proofs narrow? QBF resolution is not simple.*, in Proc. Symposium on Theoretical Aspects of Computer Science (STACS'16), 2016.
23. ———, *Understanding Cutting Planes for QBFs*, in 36th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2016), vol. 65 of Leibniz International Proceedings in Informatics (LIPIcs), 2016, pp. 40:1–40:15.

24. O. BEYERSDORFF, L. CHEW, R. A. SCHMIDT, AND M. SUDA, *Lifting QBF resolution calculi to DQBF*, in Proc. 19th International Conference on Theory and Applications of Satisfiability Testing, 2016, pp. 490–499.
25. O. BEYERSDORFF, L. CHEW, AND K. SREENIVASAIAH, *A game characterisation of tree-like Q-resolution size*, in LATA, Springer, 2015, pp. 486–498.
26. ———, *A game characterisation of tree-like q-resolution size*, Journal of Computer and System Sciences, (2017), pp. –.
27. O. BEYERSDORFF, N. GALESI, AND M. LAURIA, *A lower bound for the pigeonhole principle in tree-like resolution by asymmetric prover-delayer games*, Information Processing Letters, 110 (2010), pp. 1074–1077.
28. ———, *A characterization of tree-like resolution size*, Information Processing Letters, 113 (2013), pp. 666–671.
29. ———, *Parameterized complexity of DPLL search procedures*, ACM Transactions on Computational Logic, 14 (2013).
30. O. BEYERSDORFF, L. HINDE, AND J. PICH, *Reasons for hardness in QBF proof systems*, Electronic Colloquium on Computational Complexity (ECCC), 24 (2017), p. 44.
31. O. BEYERSDORFF AND O. KULLMANN, *Unified characterisations of resolution hardness measures*, in Proc. 17th International Conference on Theory and Applications of Satisfiability Testing, 2014, pp. 170–187.
32. O. BEYERSDORFF AND J. PICH, *Understanding Gentzen and Frege systems for QBF*, in Proc. ACM/IEEE Symposium on Logic in Computer Science (LICS'16), 2016.
33. N. BJØRNER, M. JANOTA, AND W. KLIEBER, *On conflicts and strategies in QBF.*, in LPAR (short papers), 2015, pp. 28–41.
34. A. BLAKE, *Canonical expressions in boolean algebra*, PhD thesis, University of Chicago, 1937.
35. I. BONACINA, *Total space in resolution is at least width squared*, in 43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016), 2016, pp. 56:1–56:13.
36. M. L. BONET, J. L. ESTEBAN, N. GALESI, AND J. JOHANNSEN, *On the relative complexity of resolution refinements and cutting planes proof systems*, SIAM J. Comput., 30 (2000), pp. 1462–1484.
37. M. L. BONET AND N. GALESI, *Optimality of size-width tradeoffs for resolution*, Computational Complexity, 10 (2001), pp. 261–276.
38. R. B. BOPPANA AND M. SIPSER, *The complexity of finite functions*, vol. 99, Laboratory for Computer Science, Massachusetts Institute of Technology, 1989.
39. S. R. BUSS, *On herbrand's theorem*, in Logic and Computational Complexity, Springer, 1995, pp. 195–209.
40. H. CHEN, *Proof complexity modulo the polynomial hierarchy: Understanding alternation as a source of hardness*, in ICALP, 2016, pp. 94:1–94:14.
41. E. CLARKE, O. GRUMBERG, S. JHA, Y. LU, AND H. VEITH, *Counterexample-guided abstraction refinement for symbolic model checking*, Journal of the ACM, 50 (2003), pp. 752–794.
42. S. A. COOK, *The complexity of theorem-proving procedures*, in Proceedings of the Third Annual ACM Symposium on Theory of Computing, STOC '71, New York, NY, USA, 1971, ACM, pp. 151–158.
43. S. A. COOK AND T. MORIOKA, *Quantified propositional calculus and a second-order theory for NC1*, Arch. Math. Log., 44 (2005), pp. 711–749.
44. S. A. COOK AND P. NGUYEN, *Logical Foundations of Proof Complexity*, Cambridge University Press, 2010.
45. S. A. COOK AND R. A. RECKHOW, *The relative efficiency of propositional proof systems*, The Journal of Symbolic Logic, 44 (1979), pp. 36–50.
46. W. J. COOK, C. R. COULLARD, AND G. TURÁN, *On the complexity of cutting-plane proofs*, Discrete Applied Mathematics, 18 (1987), pp. 25–38.
47. W. CRAIG, *Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory*, The Journal of Symbolic Logic, 22 (1957), pp. 269–285.
48. M. DAVIS, G. LOGEMANN, AND D. W. LOVELAND, *A machine program for theorem-proving*, Commun. ACM, 5 (1962), pp. 394–397.
49. M. DAVIS AND H. PUTNAM, *A computing procedure for quantification theory*, Journal of the ACM, 7 (1960), pp. 210–215.

50. R. DECHTER, *Enhancement schemes for constraint processing: Backjumping, learning, and cutset decomposition*, Artificial Intelligence, 41 (1990), pp. 273 – 312.
51. N. EÉN AND N. SÖRENSSON, *An extensible SAT-solver*, in International conference on theory and applications of satisfiability testing, Springer, 2003, pp. 502–518.
52. U. EGLY, *On stronger calculi for QBFs*, in Proc. 19th International Conference on Theory and Applications of Satisfiability Testing, 2016.
53. U. EGLY, F. LONSING, AND M. WIDL, *Long-distance resolution: Proof generation and strategy extraction in search-based QBF solving*, in LPAR, K. L. McMillan, A. Middeldorp, and A. Voronkov, eds., Springer, 2013, pp. 291–308.
54. U. EGLY AND H. TOMPITS, *Proof-complexity results for nonmonotonic reasoning*, ACM Transactions on Computational Logic, 2 (2001), pp. 340–387.
55. J. L. ESTEBAN AND J. TORÁN, *Space bounds for resolution*, Information and Computation, 171 (2001), pp. 84–97.
56. Y. FILMUS, M. LAURIA, M. MIKSA, J. NORDSTRÖM, AND M. VINYALS, *From small space to small width in resolution*, ACM Trans. Comput. Log., 16 (2015), pp. 28:1–28:15.
57. B. FINKBEINER AND L. TENTRUP, *Fast DQBF refutation*, in Sinz and Egly [110], pp. 243–251.
58. A. FRÖHLICH, G. KOVÁSZNAI, A. BIERE, AND H. VEITH, *idq: Instantiation-based DQBF solving*, in Sinz and Egly [110], pp. 103–116.
59. M. L. FURST, J. B. SAXE, AND M. SIPSER, *Parity, circuits, and the polynomial-time hierarchy*, Mathematical Systems Theory, 17 (1984), pp. 13–27.
60. I. P. GENT AND A. G. ROWLEY, *Encoding connect-4 using quantified Boolean formulae*, 2nd Intl. Work. Modelling and Reform. CSP, (2003), pp. 78–93.
61. G. GENTZEN, *Untersuchungen über das logische Schließen*, Mathematische Zeitschrift, 39 (1935), pp. 68–131.
62. K. GITINA, R. WIMMER, S. REIMER, M. SAUER, C. SCHOLL, AND B. BECKER, *Solving DQBF through quantifier elimination*, in Proc. Data, Automata and Test in Europe (DATE), ACM, 2015, pp. 1617–1622.
63. C. P. GOMES, B. SELMAN, AND H. KAUTZ, *Boosting combinatorial search through randomization*, in Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence, AAAI '98/IAAI '98, Menlo Park, CA, USA, 1998, American Association for Artificial Intelligence, pp. 431–437.
64. R. E. GOMORY, *An algorithm for integer solutions to linear programs*, Recent advances in mathematical programming, 64 (1963), pp. 260–302.
65. A. GOULTIAEVA, A. VAN GELDER, AND F. BACCHUS, *A uniform approach for generating proofs and strategies for both true and false QBF formulas*, in International Joint Conference on Artificial Intelligence IJCAI, T. Walsh, ed., IJCAI/AAAI, 2011, pp. 546–553.
66. J. HÅSTAD, *Almost optimal lower bounds for small depth circuits*, in Proc. 18th ACM Symposium on Theory of Computing, ACM Press, 1986, pp. 6–20.
67. J. HÅSTAD, *Computational Limitations of Small-depth Circuits*, MIT Press, 1987.
68. L. HENKIN, *Some remarks on infinitely long formulas*, in Infinitistic Methods, Proceedings of Symposium on Foundations of Mathematics, 1961, pp. 167–183.
69. M. HEULE, W. A. H. JR., AND N. WETZLER, *Verifying refutations with extended resolution*, in 24th International Conference on Automated Deduction (CADE), 2013, pp. 345–359.
70. M. HEULE, M. SEIDL, AND A. BIERE, *A unified proof system for QBF preprocessing*, in 7th International Joint Conference on Automated Reasoning (IJCAR), 2014, pp. 91–106.
71. M. JANOTA, R. GRIGORE, AND J. MARQUES-SILVA, *Counterexample guided abstraction refinement algorithm for propositional circumscription*, in 14th European Conference on Logics in Artificial Intelligence (JELIA), Sept. 2010, pp. 195–207.
72. M. JANOTA, W. KLIEBER, J. MARQUES-SILVA, AND E. M. CLARKE, *Solving QBF with counterexample guided refinement*, in Proc. 15th International Conference on Theory and Applications of Satisfiability Testing, A. Cimatti and R. Sebastiani, eds., vol. 7317, Springer, 2012, pp. 114–128.

73. M. JANOTA AND J. MARQUES-SILVA, *On propositional QBF expansions and Q-resolution*, in SAT, M. Järvisalo and A. Van Gelder, eds., Springer, 2013, pp. 67–82.
74. M. JANOTA AND J. MARQUES-SILVA, *Expansion-based QBF solving versus Q-resolution*, Theor. Comput. Sci., 577 (2015), pp. 25–42.
75. T. JUSSILA, A. BIERE, C. SINZ, D. KRÖNING, AND C. M. WINTERSTEIGER, *A first step towards a unified proof checker for QBF*, in SAT, J. Marques-Silva and K. A. Sakallah, eds., vol. 4501, Springer, 2007, pp. 201–214.
76. H. KLEINE BÜNING AND U. BUBECK, *Theory of quantified Boolean formulas*, in Handbook of Satisfiability, A. Biere, M. Heule, H. van Maaren, and T. Walsh, eds., vol. 185 of Frontiers in Artificial Intelligence and Applications, IOS Press, 2009, pp. 735–760.
77. H. KLEINE BÜNING, M. KARPINSKI, AND A. FLÖGEL, *Resolution for quantified Boolean formulas*, Inf. Comput., 117 (1995), pp. 12–18.
78. W. KLIEBER, S. SAPRA, S. GAO, AND E. M. CLARKE, *A non-prenex, non-clausal QBF solver with game-state learning*, in Proc. 13th International Conference on Theory and Applications of Satisfiability Testing, O. Strichman and S. Szeider, eds., vol. 6175, Springer, 2010, pp. 128–142.
79. J. KRAJÍČEK, *Bounded Arithmetic, Propositional Logic, and Complexity Theory*, vol. 60 of Encyclopedia of Mathematics and Its Applications, Cambridge University Press, Cambridge, 1995.
80. ———, *Interpolation theorems, lower bounds for proof systems and independence results for bounded arithmetic*, The Journal of Symbolic Logic, 62 (1997), pp. 457–486.
81. J. KRAJÍČEK AND P. PUDLÁK, *Quantified propositional calculi and fragments of bounded arithmetic*, Zeitschrift für mathematische Logik und Grundlagen der Mathematik, 36 (1990), pp. 29–46.
82. ———, *Some consequences of cryptographical conjectures for S_2^1 and EF*, Information and Computation, 140 (1998), pp. 82–94.
83. J. KRAJÍČEK, P. PUDLÁK, AND A. WOODS, *Exponential lower bounds to the size of bounded depth Frege proofs of the pigeonhole principle*, Random Structures and Algorithms, 7 (1995), pp. 15–39.
84. O. KULLMANN, *Investigating a general hierarchy of polynomially decidable classes of CNF's based on short tree-like resolution proofs*, Electronic Colloquium on Computational Complexity (ECCC), 99 (1999).
85. ———, *On a generalization of extended resolution*, Discrete Applied Mathematics, 96-97 (1999), pp. 149–176.
86. H. R. LEWIS, *Complexity results for classes of quantificational formulas*, Journal of Computer and System Sciences, 21 (1980), pp. 317 – 353.
87. J. H. LIANG, C. OH, V. GANESH, K. CZARNECKI, AND P. POUPART, *Maplecomsps, maplecomsps lrb, maplecomsps chb*, SAT Competition, (2016), p. 52.
88. F. LONSING AND A. BIERE, *DepQBF: A dependency-aware QBF solver*, JSAT, 7 (2010), pp. 71–76.
89. N. MANTHEY, *Solver submission of riss 1.0 to the sat competition 2011*, SAT Competition, (2011).
90. J. MARQUES-SILVA, *The Impact of Branching Heuristics in Propositional Satisfiability Algorithms*, Springer Berlin Heidelberg, Berlin, Heidelberg, 1999, pp. 62–74.
91. J. P. MARQUES SILVA, I. LYNCE, AND S. MALIK, *Conflict-driven clause learning SAT solvers*, in Handbook of Satisfiability, IOS Press, 2009.
92. M. W. MOSKEWICZ, C. F. MADIGAN, Y. ZHAO, L. ZHANG, AND S. MALIK, *Chaff: Engineering an efficient sat solver*, in Proceedings of the 38th annual Design Automation Conference, ACM, 2001, pp. 530–535.
93. D. E. MULLER AND F. P. PREPARATA, *Bounds to complexities of networks for sorting and for switching*, Journal of the ACM, 22 (1975), pp. 195–201.
94. D. MUNDICI, *Tautologies with a unique Craig interpolant, uniform vs. nonuniform complexity*, Annals of Pure and Applied Logic, 27 (1984), pp. 265–273.
95. C. H. PAPADIMITRIOU, *Computational Complexity*, Addison-Wesley, 1994.
96. T. PEITL, F. SLIVOVSKY, AND S. SZEIDER, *Long Distance Q-Resolution with Dependency Schemes*, Springer International Publishing, Cham, 2016, pp. 500–518.
97. G. L. PETERSON AND J. REIF, *Multiple-person alternation*, in Proc. 20th Annual Symposium on Foundations of Computer Science, IEEE, 1979, pp. 348–363.

98. T. PITASSI, P. BEAME, AND R. IMPAGLIAZZO, *Exponential lower bounds for the pigeonhole principle*, Computational Complexity, 3 (1993), pp. 97–140.
99. P. PUDLÁK, *Lower bounds for resolution and cutting planes proofs and monotone computations*, The Journal of Symbolic Logic, 62 (1997), pp. 981–998.
100. P. PUDLÁK AND R. IMPAGLIAZZO, *A lower bound for DLL algorithms for SAT*, in Proc. 11th Symposium on Discrete Algorithms, 2000, pp. 128–136.
101. M. N. RABE AND L. TENTRUP, *CAQE: A certifying QBF solver*, in Proceedings of the 15th Conference on Formal Methods in Computer-Aided Design, FMCAD Inc, 2015, pp. 136–143.
102. A. A. RAZBOROV, *Lower bounds for the size of circuits of bounded depth with basis* $\{\wedge, \oplus\}$, Math. Notes Acad. Sci. USSR, 41 (1987), pp. 333–338.
103. R. A. RECKHOW, *On the lengths of proofs in the propositional calculus*, PhD thesis, University of Toronto, 1976.
104. J. RINTANEN, *Asymptotically optimal encodings of conformant planning in QBF*, in AAAI, AAAI Press, 2007, pp. 1045–1050.
105. R. L. RIVEST, *Learning decision lists*, Machine Learning, 2 (1987), pp. 229–246.
106. J. A. ROBINSON, *Theorem-proving on the computer*, Journal of the ACM, 10 (1963), pp. 163–174.
107. ———, *A machine-oriented logic based on the resolution principle*, Journal of the ACM, 12 (1965), pp. 23–41.
108. N. SEGERLIND, *The complexity of propositional proofs*, Bulletin of Symbolic Logic, 13 (2007), pp. 417–481.
109. M. SEIDL, F. LONSING, AND A. BIERE, *qbf2epr: A tool for generating EPR formulas from QBF*, in Third Workshop on Practical Aspects of Automated Reasoning (PAAR), P. Fontaine, R. A. Schmidt, and S. Schulz, eds., vol. 21 of EPiC Series in Computing, EasyChair, 2013, pp. 139–148.
110. C. SINZ AND U. EGLY, eds., *Theory and Applications of Satisfiability Testing - SAT*, vol. 8561, Springer, 2014.
111. F. SLIVOVSKY AND S. SZEIDER, *Variable dependencies and Q-resolution*, in Sinz and Egly [110], pp. 269–284.
112. R. SMOLENSKY, *Algebraic methods in the theory of lower bounds for Boolean circuit complexity*, in Proc. 19th ACM Symposium on Theory of Computing, ACM Press, 1987, pp. 77–82.
113. L. J. STOCKMEYER AND A. R. MEYER, *Word problems requiring exponential time: Preliminary report*, in Proceedings of the 5th Annual ACM Symposium on Theory of Computing, 1973, pp. 1–9.
114. A. TARSKI, *The semantic conception of truth: and the foundations of semantics*, Philosophy and phenomenological research, 4 (1944), pp. 341–376.
115. G. S. TSEITIN, *On the complexity of proof in prepositional calculus*, Zapiski Nauchnykh Seminarov POMI, 8 (1968), pp. 234–259.
116. A. VAN GELDER, *Contributions to the theory of practical quantified boolean formula solving*, in Principles and Practice of Constraint Programming, Springer, 2012, pp. 647–663.
117. R. WIMMER, K. GITINA, J. NIST, C. SCHOLL, AND B. BECKER, *Preprocessing for DQBF*, in Proc. 18th International Conference on Theory and Applications of Satisfiability Testing, 2015, pp. 173–190.
118. H. ZHANG, *Sato: An efficient propositional prover*, in Proceedings of the 14th International Conference on Automated Deduction, (CADE), Springer-Verlag, 1997, pp. 272–275.
119. L. ZHANG AND S. MALIK, *Conflict driven learning in a quantified Boolean satisfiability solver*, in ICCAD, 2002, pp. 442–449.