FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

GQBF and Proof Complexity

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Computational Intelligence

eingereicht von

Peter Haberl
Matrikelnummer 0525517

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung
Betreuer: Ao. Univ-Prof. Dr. Uwe Egly

Wien, 11.05.2011

(Unterschrift Verfasser) (Unterschrift Betreuer)

Erklärung zur Verfassung der Arbeit

Peter Haberl

Römerstrasse 3,
3361 Aschbach Markt

"Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit — einschliesslich Tabellen, Karten und Abbildungen —, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe."

Wien, 11.05.2011

_____

Unterschrift

Kurzfassung Das Erfüllbarkeitsproblem für quantifizierte boolesche Formeln — das ist, zu entscheiden ob alle existentiell quantifizierten Variablen einer geschlossenen quantifizierten boolschen Formel in einer Form belegt werden können, so dass die Formel damit wahr wird — ist von besonderem beweistheoretischen und komplexitätstheoretischen Interesse. Es generalisiert das aussagenlogische Erfüllbarkeitsproblem und bietet für jede Stufe der polynomiellen Hierarchie prototypische Probleme. Weiters lassen sich viele Probleme der künstlichen Intelligenz und der Spieltheorie als Erfüllbarkeitsproblem für quantifizierte boolesche Formeln kodieren.

Derzeit existieren mehrere Beweiser für dieses Problem, basierend auf einer Vielzahl unterschiedlicher Lösungsansätze. Ein kürzlich vorgestellter Beweiser konstruiert Beweise mit Hilfe eines Sequenzkalküls. In diesem Papier untersuchen wir die Fähigkeiten dieses Kalküls kurze Beweise erzeugen zu können und vergleichen verschiedene Konstruktionsmerkmale des Kalküls diesbezüglich. Wir vergleichen diesen Kalkül weiters mit einem weit verbreiteten Resolutionskalkül und untersuchen die Auswirkungen von Pränexierung auf die Länge von Beweisen.

Wir zeigen, dass eine strenge Ordnung von Sequenzkalkülen existiert was ihre Fähigkeit betrifft, kurze Beweise erzeugen zu können. Manipulationen im Inneren von Formeln — zusätzlich zu Manipulationen die streng der Struktur von Formeln folgen — können Beweise exponentiell verkürzen. Beweise in Form eines gerichteten, azyklischen Graphen können expontiell kürzer sein als Beweise in Baumform. Weiters zeigen wir obere und untere Schranken für Fähigkeiten, die ein Sequenzkalkül besitzen muss, um ebenso kurze Beweise erzeugen zu können wie Resolutionskalküle. Das Papier beweist auch, dass eine gute Pränexierungsstrategie essentiell ist, um kurze Beweise erzeugen zu können.

Abstact The evaluation problem for quantified boolean formulas — deciding if a truth value can be assigned to each existentially quantified variable of a dedicated quantified boolean formula in a way that the formula evaluates to true — is especially interesting from a proof theoretical and complexity theoretical viewpoint. It generalizes the propositional satisfiability problem and provides natural problems for any level of the polynomial hierarchy. From a more practical perspective, many problems from the field of artificial intelligence and game theory can be encoded as evaluation problems for quantified boolean formulas.

Currently there exists a wide range of solvers for the evaluation problem for quantified boolean formulas, using a variety of different evaluation strategies. A solver presented recently uses a sequent-style calculus to construct proofs for quantified boolean formulas. We research the capabilities of this sequent-style calculus and compare different features of the calculus amongst each other with respect to their power to generate short proofs. We also compare sequent-style calculi to a widely used version of a resolution-like calculus and examine the impact of prenexing on proof size.

In this thesis we establish a strict hierarchy of sequent-style calculi that orders them with respect to their power of creating short proofs. It is proven that adding rules that operate within sub-formulas instead of operating on the outermost connective, respectively quantifier, can exponentially shorten proofs. So can allowing proofs to form directed acyclic graphs rather than trees. We establish an upper and an lower bound of features that are needed for a sequent calculus to generate proofs as short as resolution can. It is also shown that a good choice of prenexing strategy is essential to construct short proofs.

Contents

Part I.

Introduction

1. Introduction

Subsequently we will present an informal introduction to the notions of quantified boolean formulas and the evaluation problem for these formulas. Motivations for dealing with the evaluation problem for quantified boolean formulas will be presented as well as why we are interested in proofs for them. The informal introduction is succeeded by a small summary of the problem that is discussed in this thesis, the results gathered, the systematic approach taken and the state-of-the art. Formal discussion will be presented in part II and all consecutive parts.

The Boolean Satisfiability Problem Evaluating logical formulas has a long tradition in computer science. Most notably, the boolean satisfiability problem is one of the most studied problems in the area of theoretical computer science. Propositional formulas are partitioned into two disjoint classes via the boolean satisfiability problem: into the class of satisfiable formulas and the class of unsatisfiable formulas. Papadimitriou defines satisfiable formulas this way [14]: "We say that a Boolean expression ϕ is satisfiable if there is a truth assignment T appropriate to it such that $T \models \phi$". It is unsatisfiable otherwise. A truth assignment T is a model of a formula ϕ ($T \models \phi$) if it assigns a truth value to each atomic part of the formula such that the formula as a whole is logically equivalent to true. Formal definitions of syntax and semantics will be given in Part II. Further, we call a formula valid if all truth assignments appropriate to it are also models of it.

In general, deciding if a propositional formula is satisfiable is not an easy problem. The Cook-Levin theorem [5] proves that the boolean satisfiability problem is NP-complete. That is, a non-deterministic Turing machine can decide in polynomially many steps — with respect to the size of a formula — if there exists a truth assignment for the formula that is a model of it. From a more mechanical perspective, there is no deterministic algorithm known that solves the problem in polynomially many steps. It is moreover highly doubted that such an algorithm may exists anyhow [7].

The Evaluation Problem for Quantified Boolean Formulas If the language that is used to construct formulas is enriched, e.g. with means to quantify over truth values, complexity rises even higher. In this thesis we will mostly deal with quantified boolean formulas. As the name of this class of formulas already tells, a certain form of quantification is allowed in the construction of formulas.

Propositional formulas are constructed from atomic variables — say from a set $\mathcal{V}$ of variable symbols — by using connectives to form complex formulas. When a propositional formula is then evaluated, a truth value — true, respectively **1** or false, respectively **0** — is assigned to each variable symbol appearing in the formula. The semantics of the connectives used to construct the formula then decide on the basis of the truth values assigned to the formulas atomic parts about the truth value of the formula as a whole. As truth values can be assigned freely to variable symbols, such variables symbols are called free variables.

Quantified boolean formulas may now, in addition to free variables, contain so called bound variables. Each bound variable is bound by either an existential or an universal quantifier. Evaluating formulas that contain bound variables cannot be done as if all variables were free: it is no longer allowed to freely assign a truth value to bound variables during evaluation. We informally[1] define evaluating formulas with quantifiers the following way: (1) If a formula contains free variables only, it is evaluated as if it would be a propositional formula. (2) If a formula is of the form $\forall x\, \phi$, i.e. x is a universally quantified variable symbol, then the formula evaluates to true if ϕ evaluates to true having x assigned to true as well as having x assigned to false. The formula evaluates to false otherwise. (3) If a formula is of the form $\exists x\, \phi$ then it evaluates to true if ϕ evaluates to true having x assigned to true or having x assigned to false. It is false otherwise.

From this evaluation process we can deduce that a formula that only contains bound variables can either be valid or unsatisfiable. If it is satisfiable it is valid too. We call quantified boolean formulas that contain no free variables closed. Deciding whether a closed formula is valid is called evaluation problem for quantified boolean formulas. Assume that ϕ is closed quantified boolean formula that is valid. Then for each existentially quantified variable symbol there exists at least one truth value that guarantees that the corresponding formula can evaluate to true. The collection of these pairings, between variable symbol and truth value, can be seen as a witness for the validity of the formula. Dually, if ϕ was unsatisfiable, there is a collection of pairings, between universally quantified variable symbol and truth value, that can bear witness for the unsatisfiability of the formula.

To solve the evaluation problem for quantified boolean formulas, such witnesses have to be searched and provided. Thus effective means for search and representation of such witnesses are desirable.

The Polynomial Hierarchy For further analysis of the complexity of closed quantified boolean formulas, we define certain subsets of them: Σ_i^q be the set of quantified boolean formulas that have i alternating blocks of quantifies, beginning with a block of existential quantifiers, and a propositional core such that all free variables of the propositional core are bound by the surrounding quan-

[1] A formal definition of the syntax and semantics for quantified boolean formulas follows in part II.

tifiers. Π_i^q be the counterpart having a universal quantifier block as outermost quantifier block. $\forall x_1 \forall x_2 \exists y_1 \forall z_1 \forall z_2 \forall z_3 \, \phi(x_1, x_2, y_1, z_1, z_2, z_3)$ is an example for a formula from Π_3^q.

Each formula of these classes corresponds to a complexity class. It is proven that the evaluation problem for formulas from Σ_i^q is Σ_i^P-complete [22]. Similarly, evaluation of Π_i^q-formulas is Π_i^P-complete. Thus the evaluation problem for quantified boolean formulas offers natural problems for each level of the polynomal hierarchy PH. The generic evaluation problem for quantified boolean formulas is PSPACE-complete [19].

Reducing Problems to the Evaluation Problem for Quantified Boolean Formulas As mentioned before, the evaluation problem for quantified boolean formulas offers a natural problem that is PSPACE-complete. The authors of [8] stress that a "vast number of problems can succinctly be formulated in QBF":

- finite two-player games
- AI planing problems
- modal logic problems
- (un)bounded model checking for finite-state systems
- formal verification problems

These problem are reduced to the evaluation problem for quantified boolean formulas, because it seems easier to solve the original problem via this encoding. Often there do not exist implementations or algorithms that are capable of solving the original problem in an efficient manner. Examples are default reasoning and nested counterfactuals. Hence a fast solver that decides the evaluation problem can provide improvements to many fields of theoretical computer science.

New solvers are developed and tested steadily [15]. Existing solvers use a variety of different evaluation strategies. These strategies are often based on DPLL, BDD, CDCL, resolution and combinations thereof, many times extended by Skolemization. However [8] states that "state-of-the-art QBF solvers are not yet reliable enough." They say, referencing [13], that majority votes had to be used to decide the validity of hard instances as solvers often disagreed about it. This urges to search for certificates that allow to check the correctness of results delivered by a solver in an easy and fast way. In the search of such a unified proof format, the authors of [8] presents a proposal based on extensions — using fresh variables to generate certification code. But as the authors put it, "a purely resolution based proof calculus is not powerful enough to trace the most efficient solvers".

1.1. Problem Description

In the following, the authors of [6] present a sequent-style calculus that allows to prove the validity of quantified boolean formulas. The calculus used in [6] will be the topic of main interest of this thesis. We will look into the capabilities of the sequent calculus presented and identify its main features. These features will be compared amongst each others, especially their capability to generate short proofs. As Egly, Seidl and Woltran show in [6], the choice of a good prenexing strategy is crucial for finding proofs fast. It also has a dramatic impact on the search space that has to be covered by the solver. Thus prenexing strategies will be a topic of interest in this work too.

1.2. Results

We establish a hierarchy of calculi with respect to their power to produce short proofs. We also establish polynomial simulations and exponential separations of the calculi researched: GQBF, GQBF$^+$, GQBF$^+_\diamond$, G and Q-resolution. These results can be seen in Figure 1.1. It is also shown that good prenexing strategies are essential for generating short proofs.

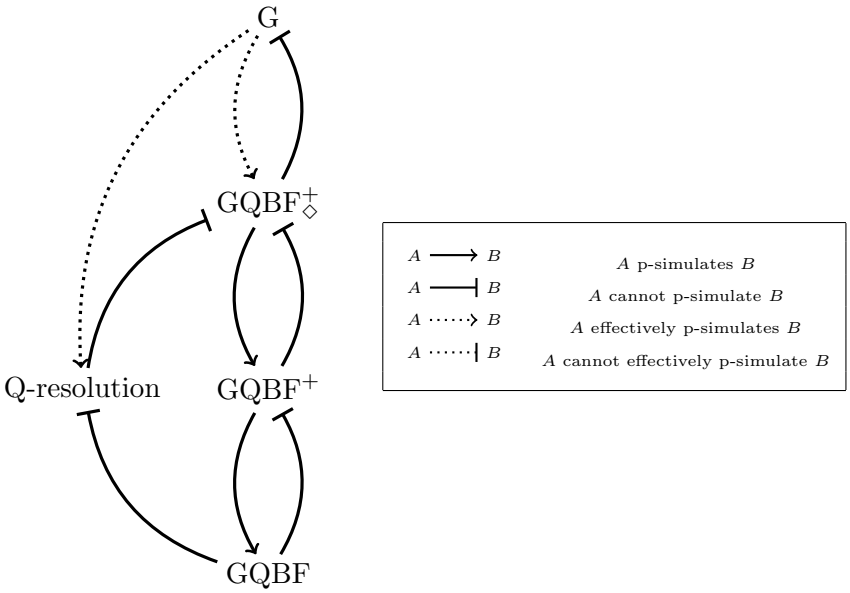


Figure 1.1.: Polynomial simulation results.

1.3. Systematic Approach

The sequent-style quantified boolean calculus presented in [6], is based on six principal derivation rules that operate on the outermost connective or quantifier and several simplification rules that allow manipulations inside formulas. The principal rules resemble the inductive way quantified boolean formulas can be defined while the simplification rules correspond to "tricks" applied in proof search to prune the search space. The authors of [6] prove that the six principal rules suffice to get a sound and complete calculus. This modularity of the calculus allows us to compare various versions of the calculus amongst each other as well as to Q-resolution and G. We show that adding features like the simplification rules to the minimal version of the calculus (built from the six principal rules only) increases the power of GQBF in a way such that it can produce exponentially smaller proofs. Adding further features like directed acyclic graph structured proofs can make proofs even shorter. This thesis step by step adds features to GQBF and researches their capability to generate shorter proofs.

1.4. State-Of-The-Art

Proof theoretic analysis, especially for propositional and first order calculi, has a long tradition and the notion of polynomial simulation is well established. Recently [16] introduced the notion of effectively polynomial simulation and researched simulation relations between calculi of their choice, including various forms of resolution. As the evaluation problem for quantified boolean formulas currently seems to be a topic of special interest [15], there are many calculi that are not covered by [16] or alternative works. Many of the newer calculi defect from input formulas in prenex conjunctive normal form [9, 6, 12], some are based on negation normal form. GQBF is one of them. We currently have no knowledge of any deeper proof theoretic analysis of GQBF, especially with respect to (effectively) polynomial simulation.

1.5. A Guide Through Consecutive Parts

This thesis is basically split into seven parts. The first part contains a short and informal introduction to the topics covered and will end with this guide. What follows is a formal discussion of the main questions: How do features of sequent-style calculi and classes of formulas effect proof size? Part II gives formal definitions of the terms used throughout this thesis. The Parts III and V present various calculi used. IV and VI deal with the power of the calculi presented and the problems that arise with certain families of formulas. Finally VII collects the most important results gathered and presents a final conclusion.

Part II.

Preliminaries

2. Quantified Boolean Formulas

2.1. Syntax

In following we almost exclusively deal with formulas from two languages: formulas from the language of propositional logic and formulas from the language of quantified boolean logic. Thus, before diving deeper into the matter, we first present definitions of these two languages.

Propositional Formula Let $\mathcal{V}$ be a countable set of variable symbols and $\mathcal{C} = \{\top, \bot\}$ be a set of constant symbols. In Addition, let $\neg/1$ (negation), $\wedge/2$ (conjunction), $\vee/2$ (disjunction) be connectives, $\forall$ (universal) and $\exists$ (existential) be quantifiers and (and) be parentheses. Then we define $\mathcal{PL}$, the set of well-formed propositional formulas, inductively as follows:

1. $\mathcal{C} \subseteq \mathcal{PL}$
2. $\mathcal{V} \subseteq \mathcal{PL}$
3. If $\varphi \in \mathcal{PL}$, then $(\neg\varphi) \in \mathcal{PL}$
4. If $\{\varphi, \psi\} \subseteq \mathcal{PL}$ and $\circ \in \{\wedge, \vee\}$, then $(\varphi \circ \psi) \in \mathcal{PL}$

Quantified Boolean Formula Let $\mathcal{V}$ be a countable set of variable symbols and $\mathcal{C} = \{\top, \bot\}$ be a set of constant symbols. In Addition, let $\neg/1$ (negation), $\wedge/2$ (conjunction), $\vee/2$ (disjunction) be connectives, $\forall$ (universal) and $\exists$ (existential) be quantifiers and (and) be parentheses. Then we define $\mathcal{QBF}$, the set of well-formed quantified boolean formulas, inductively as follows:

1. $\mathcal{C} \subseteq \mathcal{QBF}$
2. $\mathcal{V} \subseteq \mathcal{QBF}$
3. If $\varphi \in \mathcal{QBF}$, then $(\neg\varphi) \in \mathcal{QBF}$
4. If $\{\varphi, \psi\} \subseteq \mathcal{QBF}$ and $\circ \in \{\wedge, \vee\}$, then $(\varphi \circ \psi) \in \mathcal{QBF}$
5. If $\varphi \in \mathcal{QBF}$, $Q \in \{\exists, \forall\}$ and $x \in \mathcal{V}$, then $(Qx\,\varphi) \in \mathcal{QBF}$.

From the construction of the sets $\mathcal{PL}$ and $\mathcal{QBF}$ it can be seen that $\mathcal{PL} \subset \mathcal{QBF}$ and thus each propositional formula is a quantified boolean formula. A formula

φ is called quantified boolean formula if $\varphi \in \mathcal{QBF}$ and it is called propositional formula if $\varphi \in \mathcal{PL}$ and $\varphi \notin \mathcal{QBF}$.

We need further notational conventions to be able to fully describe the formulas we use: If $x \in \mathcal{V}$ is a variable symbol then x and $(\neg x)$ are called literals of x. Having the fifth rule of the inductive construction of the quantified boolean formulas in mind we call, with respect to the formula $(Qx\,\varphi)$, φ the scope of the quantifier occurrence Qx. A variable occurrence y is called "free" if it is not in the scope of a quantifier Qy. With respect to $(Qx\,\varphi)$, Qx is the quantifier occurrence binding all free occurrences of x in φ. We will provide a precise and purely syntactic characterization of free variables in the following.

Free Variables of a Formula Let $\varphi \in \mathcal{QBF}$. Then the set of free variables of φ, $\mathit{free}(\varphi)$, is defined as follows:

- If $\varphi = c$ with $c \in \mathcal{C}$, then $\mathit{free}(\varphi) := \emptyset$.
- If $\varphi = x$ with $x \in \mathcal{V}$, then $\mathit{free}(\varphi) := \{x\}$.
- If $\varphi = (\neg\psi)$, then $\mathit{free}(\varphi) := \mathit{free}(\psi)$.
- If $\varphi = (\varphi_1 \circ \varphi_2)$ with $\circ \in \{\wedge, \vee\}$, then $\mathit{free}(\varphi) := \mathit{free}(\varphi_1) \cup \mathit{free}(\varphi_2)$.
- If $\varphi = (Qx\,\psi)$ with $Q \in \{\exists, \forall\}$, then $\mathit{free}(\varphi) := \mathit{free}(\psi) \setminus \{x\}$.

Closed Formulas We call a formula $\varphi \in \mathcal{QBF}$ closed if $\mathit{free}(\varphi) = \emptyset$.

Most of the time we are not interested in formulas that have free variables. Thus we use the term $\mathcal{QBF}$, if not mentioned otherwise, to refer to the following set of formulas: $\{\varphi \in \mathcal{QBF} \mid \varphi \text{ is closed}\}$.

Unitary Formulas We call a quantified boolean formula φ unitary if no two quantifier occurrences in φ bind the same variable symbol and no variable that occurs bound in φ also occurs free in φ.

Note that for each $\varphi \in \mathcal{QBF}$, there exists a $\varphi^{\mathrm{u}} \in \mathcal{QBF}$ (obtained by renaming bound variables) that is unitary and isomorphic to φ. We thus define the set of unitary formulas $\mathcal{QBF}^{\mathrm{u}} := \{\varphi \in \mathcal{QBF} \mid \varphi \text{ is unitary}\}$. We generally assume, if not mentioned otherwise, using a unitary representation instead of the actual formula.

2.2. Simple Syntactic Operations on Formulas

Throughout this thesis we make heavy use of simple syntactic transformations and operations on propositional as well as on quantified boolean formulas. The next few paragraphs describe the most commonly used operations and transformations.

Size of a Formula Let $\varphi \in \mathcal{QBF}$. Then we define $|\varphi|$, the size of the quantified boolean formula φ as follows:

1. If $\varphi = x$ with $x \in (\mathcal{C} \cup \mathcal{V})$, then $|\varphi| := 1$
2. If $\varphi = (\neg\psi)$, then $|\varphi| := 1 + |\psi|$
3. If $\varphi = (\varphi_1 \circ \varphi_2)$, with $\circ \in \{\wedge, \vee\}$, then $|\varphi| := 1 + |\varphi_1| + |\varphi_2|$
4. If $\varphi = (Qx\,\psi)$, with $Q \in \{\exists, \forall\}$, then $|\varphi| := 2 + |\psi|$

Cleansed Formulas We use the definition of [6] and thus call a formula $\varphi \in \mathcal{QBF}$ cleansed if none of the below simplifications can be applied to φ:

$$
\begin{array}{rcl@{\qquad}rcl}
(\neg\bot) & \longmapsto & \top & (\neg\top) & \longmapsto & \bot \\
(\top \wedge \psi) & \longmapsto & \psi & (\psi \wedge \top) & \longmapsto & \psi \\
(\bot \wedge \psi) & \longmapsto & \bot & (\psi \wedge \bot) & \longmapsto & \bot \\
(\top \vee \psi) & \longmapsto & \top & (\psi \vee \top) & \longmapsto & \top \\
(\bot \vee \psi) & \longmapsto & \psi & (\psi \vee \bot) & \longmapsto & \psi \\
(\forall x\,\psi) & \longmapsto & \psi, \text{ if } x \notin \mathit{free}(\psi) & (\exists x\,\psi) & \longmapsto & \psi, \text{ if } x \notin \mathit{free}(\psi)
\end{array}
$$

A cleansed formula thus does not contain truth constants (unless it has size 1 and consist of a truth constant only) and no quantifier that binds a variable symbol that does not appear in its scope.

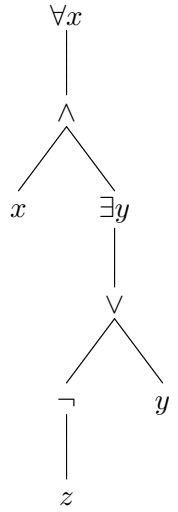
The Instantiation Operator Let $\varphi \in \mathcal{QBF}^{\mathrm{u}}$ be a quantified boolean formula, $x \in \mathcal{V}$ and $\xi \in \mathcal{QBF}$ a quantified boolean formula. Then $\varphi[x\backslash\xi]$ is defined as follows:

- If $\varphi = x$, then $\varphi[x\backslash\xi] := \xi$.
- If $\varphi = y$ with $y \in (\mathcal{C} \cup \mathcal{V})$ and $y \neq x$, then $\varphi[x\backslash\xi] := y$.
- If $\varphi = (\neg\psi)$, then $\varphi[x\backslash\xi] := (\neg\psi[x\backslash\xi])$.
- If $\varphi = (\varphi_1 \circ \varphi_2)$ with $\circ \in \{\wedge, \vee\}$, then $\varphi[x\backslash\xi] := (\varphi_1[x\backslash\xi] \circ \varphi_2[x\backslash\xi])$.
- If $\varphi = (Qx\,\psi)$ with $Q \in \{\exists, \forall\}$, then $\varphi[x\backslash\xi] := \psi[x\backslash\xi]$.
- If $\varphi = (Qy\,\psi)$ with $y \neq x$, $Q \in \{\exists, \forall\}$, then $\varphi[x\backslash\xi] := (Qy\,\psi[x\backslash\xi])$.

It is noteworthy that $\varphi[x\backslash\xi] \in \mathcal{QBF}$ independent of the choices of φ, x and ξ. However, if φ is a closed and unitary quantified boolean formula and $\xi \in \mathcal{C}$ — this will be the most widely used kind of instantiation — $\varphi[x\backslash\xi]$ is closed and unitary too. In general these properties are not preserved.

The Structure Tree of a Formula As the set of quantified boolean formulas is created over a ranked alphabet[1], each quantified boolean formula can easily be treated as a tree. An n-ary symbol constitutes an n-ary node in the structure tree of a formula. The nodes name is the symbol itself, its children are the symbols arguments.

Consider $\varphi = (\forall x\,(x \wedge (\exists y\,((\neg z) \vee y))))$ as an example. Then the structure tree of φ is the following:



Note that for each quantified boolean formula there is a unique structure tree and that for each structure tree there is a unique quantified boolean formula.

Structural Sub-Formulas Let $\varphi \in \mathcal{QBF}$ be a quantified boolean formula. Then ξ is called structural sub-formula of φ if one of the following conditions holds:

- $\varphi = \xi$
- $\varphi = (\neg\psi)$ and ξ is a sub-formula of ψ.
- $\varphi = (\varphi_1 \circ \varphi_2)$ with $\circ \in \{\wedge, \vee\}$ and ξ is a sub-formula of φ_1 or φ_2.
- $\varphi = (Qx\,\psi)$ with $Q \in \{\exists, \forall\}$ and ξ is a sub-formula of ψ.

Note that the structural sub-formulas of φ correspond to sub-trees in the structure tree of φ.

[1] An arity can be assigned to each symbol: e.g. $\wedge$ is a binary functions, constant symbols have arity zero, quantifiers are unary functions.

Sub-Formulas We also want to define sub-formulas. We say that ξ is a sub-formula of a quantified boolean formula φ if it is a structural sub-formula of a substitution instance of φ. Thus, ξ is called structural sub-formula of φ if there exists $n \in \mathbb{N}$, $\mathbf{x} \subseteq \mathcal{V}$ and $\mathbf{c} \subseteq \mathcal{C}$ such that $|x| = |c| = n$ and ξ is a structural sub-formula of $\varphi[x_1 \backslash c_1][\ldots][x_n \backslash c_n]$.

Consider the following example: Let $\varphi = ((\forall x \, \exists y \, (z \vee (x \wedge y))) \wedge z)$ be a quantified boolean formula. Then $(\exists y \, (z \vee (x \wedge y)))$ is a structural sub-formula of φ while $(\exists y \, (\bot \vee (\top \wedge y)))$ is a sub-formula of φ. The formula $(\exists y \, (z \vee (x \vee y)))$ is neither a structural sub-formula nor a sub-formula of φ.

2.3. Semantics

As stated above, we are mostly interested in closed (unitary) quantified boolean formulas. For this subset of the quantified boolean formulas we can define a very simple semantics function: the evaluation function ν. For the general case (arbitrary quantified boolean formulas) we have to provide an additional assignment that prescribes to what value a free variable evaluates to. The set of propositional formulas is a proper subset of the set of quantified boolean formulas. This inclusion enables us to evaluate propositional formulas using the general evaluation function ν_f with adding a proper variable assignment f.

In the following $\mathbf{1}$ denotes true, $\mathbf{0}$ false and $\mathbb{B} = \{\mathbf{1}, \mathbf{0}\}$.

- $inv \colon \mathbb{B} \to \mathbb{B}$ returns $\mathbf{1}$ on input $\mathbf{0}$ and vice versa
- $min \colon \mathbb{B}^2 \to \mathbb{B}$ returns $\mathbf{1}$ on input $(\mathbf{1}, \mathbf{1})$, $\mathbf{0}$ otherwise
- $max \colon \mathbb{B}^2 \to \mathbb{B}$ returns $\mathbf{0}$ on input $(\mathbf{0}, \mathbf{0})$, $\mathbf{1}$ otherwise

The Special Evaluation Function ν We define the evaluation function $\nu \colon \{\varphi \in \mathcal{QBF}^{\mathrm{u}} \mid \varphi \text{ is closed}\} \to \mathbb{B}$ as follows:

- $\nu(\top) = \mathbf{1}$
- $\nu(\bot) = \mathbf{0}$
- $\nu(\neg\psi) = inv(\nu(\psi))$
- $\nu(\varphi_1 \wedge \varphi_2) = min(\nu(\varphi_1), \nu(\varphi_2))$
- $\nu(\varphi_1 \vee \varphi_2) = max(\nu(\varphi_1), \nu(\varphi_2))$
- $\nu(\exists x \, \psi) = max(\nu(\psi[x \backslash \top]), \nu(\psi[x \backslash \bot]))$
- $\nu(\forall x \, \psi) = min(\nu(\psi[x \backslash \top]), \nu(\psi[x \backslash \bot]))$

As a function ν gets a closed (unitary) quantified boolean formula as only input and outputs the formulas truth value. In contrast to the general evaluation function presented in the following, there is no additional input — in the form of an predefined truth assignment to free variables — needed. It follows that a closed quantified boolean formula can either be true or false, valid or unsatisfiable respectively.

We use the symbol $\equiv$ to denote the truth value of a quantified boolean formula. Let $\varphi \in \mathcal{QBF}$ be a closed quantified boolean formula. Then we write $\varphi \equiv \mathbf{1}$ to denote that $\nu(\varphi) = \mathbf{1}$ and $\varphi \equiv \mathbf{0}$ to denote that $\nu(\varphi) = \mathbf{0}$. Moreover, we write $\varphi \equiv \psi$ to express that φ and ψ evaluate to the same truth value (i.e., φ and ψ are logically equivalent).

The General Evaluation Function ν_f Let $\varphi \in \mathcal{QBF}^{\mathrm{u}}$, $f\colon \mathcal{V} \to \mathbb{B}$ be a partial function assigning a truth value to each free variable symbol of φ ($dom(f) = free(\varphi)$). Then we define the evaluation function $\nu_f\colon \mathcal{QBF} \to \mathbb{B}$ as follows:

- $\nu_f(\top) = \mathbf{1}$
- $\nu_f(\bot) = \mathbf{0}$
- $\nu_f(x) = f(x)$
- $\nu_f(\neg\psi) = inv((\nu_f(\psi)))$
- $\nu_f(\varphi_1 \wedge \varphi_2) = min(\nu_f(\varphi_1), \nu_f(\varphi_2))$
- $\nu_f(\varphi_1 \vee \varphi_2) = max(\nu_f(\varphi_1), \nu_f(\varphi_2))$
- $\nu_f(\exists x\, \psi) = max(\nu_{\oplus_{f,x}}(\psi), \nu_{\ominus_{f,x}}(\psi))$
- $\nu_f(\forall x\, \psi) = min(\nu_{\oplus_{f,x}}(\psi), \nu_{\ominus_{f,x}}(\psi))$

where $\oplus$ and $\ominus$ are functionals that extend truth-assigning functions:
$\oplus\colon (\mathcal{V} \to \mathbb{B}) \to (\mathcal{V} \to (\mathcal{V} \to \mathbb{B}))$

$$\oplus_{f,x}(y) := \begin{cases} f(y), & y \in dom(f) \\ \mathbf{1}, & y = x \end{cases}$$

$\ominus\colon (\mathcal{V} \to \mathbb{B}) \to (\mathcal{V} \to (\mathcal{V} \to \mathbb{B}))$

$$\ominus_{f,x}(y) := \begin{cases} f(y), & y \in dom(f) \\ \mathbf{0}, & y = x \end{cases}$$

The following example demonstrates the use of the general evaluation function. Let $\varphi = (\forall x\, (x \wedge (\exists y\, (\neg z \vee y))))$ and $f = \{(z, \mathbf{1})\}$. Then φ is evaluated the following way:

$$
\begin{array}{l}
\nu_f(\varphi) = min(\nu_{f\cup\{(x,\mathbf{1})\}}(x \wedge (\exists y\,(\neg z \vee y))), \nu_{f\cup\{(x,\mathbf{0})\}}(x \wedge (\exists y\,(\neg z \vee y)))) \\
\quad \nu_{f\cup\{(x,\mathbf{1})\}}(x \wedge (\exists y\,((\neg z) \vee y))) = min(\nu_{f\cup\{(x,\mathbf{1})\}}(x), \nu_{f\cup\{(x,\mathbf{1})\}}(\exists y\,(\neg z \vee y)))) \\
\qquad \nu_{f\cup\{(x,\mathbf{1})\}}(x) = \mathbf{1} \\
\qquad \nu_{f\cup\{(x,\mathbf{1})\}}(\exists y\,(\neg z \vee y))) = max(\nu_{f\cup\{(x,\mathbf{1}),(y,\mathbf{1})\}}(\neg z \vee y), \nu_{f\cup\{(x,\mathbf{1}),(y,\mathbf{0})\}}(\neg z \vee y)) \\
\qquad\quad \nu_{f\cup\{(x,\mathbf{1}),(y,\mathbf{1})\}}(\neg z \vee y) = max(\nu_{f\cup\{(x,\mathbf{1}),(y,\mathbf{1})\}}(\neg z), \nu_{f\cup\{(x,\mathbf{1}),(y,\mathbf{1})\}}(y)) \\
\qquad\qquad \nu_{f\cup\{(x,\mathbf{1}),(y,\mathbf{1})\}}(\neg z) = inv(\nu_{f\cup\{(x,\mathbf{1}),(y,\mathbf{1})\}}(z)) \\
\qquad\qquad\quad \nu_{f\cup\{(x,\mathbf{1}),(y,\mathbf{1})\}}(z) = \mathbf{1} \\
\qquad\qquad \nu_{f\cup\{(x,\mathbf{1}),(y,\mathbf{1})\}}(\neg z) = inv(\mathbf{1}) = \mathbf{0} \\
\qquad\qquad \nu_{f\cup\{(x,\mathbf{1}),(y,\mathbf{1})\}}(y) = \mathbf{1} \\
\qquad\quad \nu_{f\cup\{(x,\mathbf{1}),(y,\mathbf{1})\}}(\neg z \vee y) = max(\mathbf{0}, \mathbf{1}) = \mathbf{1} \\
\qquad \nu_{f\cup\{(x,\mathbf{1})\}}(\exists y\,(\neg z \vee y))) = max(\mathbf{1}, ?) = \mathbf{1} \\
\quad \nu_{f\cup\{(x,\mathbf{1})\}}(x \wedge (\exists y\,(\neg z \vee y))) = min(\mathbf{1}, \mathbf{1}) = \mathbf{1} \\
\quad \nu_{f\cup\{(x,\mathbf{0})\}}(x \wedge (\exists y\,(\neg z \vee y))) = min(\nu_{f\cup\{(x,\mathbf{0})\}}(x), \nu_{f\cup\{(x,\mathbf{0})\}}(\exists y\,(\neg z \vee y)))) \\
\qquad \nu_{f\cup\{(x,\mathbf{0})\}}(x) = \mathbf{0} \\
\quad \nu_{f\cup\{(x,\mathbf{0})\}}(x \wedge (\exists y\,(\neg z \vee y))) = min(\mathbf{0}, ?) = \mathbf{0} \\
\nu_f(\varphi) = min(\mathbf{1}, \mathbf{0}) = \mathbf{0}
\end{array}
$$

Thus $\varphi \equiv \mathbf{0}$ holds.

2.4. The Polynomial Hierarchy

In the following we will present a very brief introduction to the polynomial hierarchy and its connection to the evaluation problem for quantified boolean formulas. A basic understanding of complexity theory is assumed. A thorough discussion of the notions introduced can be found amongst others in [14].

Let Σ_i^q and Π_i^q be countably infinite sets of quantified boolean formulas defined inductively the following way:

- $\Sigma_0^q = \Pi_0^q = \mathcal{PL}$
- $\Sigma_i^q = \{\exists x_1\, \exists x_2\, \ldots \exists x_n\, \varphi \mid n \in \mathbb{N}, \varphi \in \Pi_{n-1}^q\}$
- $\Pi_i^q = \{\forall x_1\, \forall x_2\, \ldots \forall x_n\, \varphi \mid n \in \mathbb{N}, \varphi \in \Sigma_{n-1}^q\}$

Thus Σ_i^q is a set of quantified boolean formulas with i alternating quantifier blocks where the outermost quantifier block is existential. Similarly, Π_i^q is a set of quantified boolean formulas with i alternating quantifier blocks where the outermost quantifier block is universal.

Furthermore we define the classes of the polynomial hierarchy, Δ_i^P, Σ_i^P and Π_i^P, inductively as follows:

- $\Delta_0^P = \Sigma_0^P = \Pi_0^P = \mathrm{P}$

- $\Delta_i^P = \mathrm{P}^{\Sigma_{i-1}^P}$
- $\Sigma_i^P = \mathrm{NP}^{\Sigma_{i-1}^P}$
- $\Pi_i^P = \text{co-NP}^{\Sigma_{i-1}^P}$

Then the problem of deciding whether it holds for a closed quantified boolean formula $\varphi \in \Sigma_1^q$ that $\nu(\varphi) = \mathbf{1}$ is in $\Sigma_1^P = \mathrm{NP}$. It is called propositional satisfiability problem. Deciding for a closed quantified boolean formula $\varphi \in \Pi_1^q$ whether $\nu(\varphi) = \mathbf{1}$ is a problem from the class $\Pi_1^P = \text{co-NP}$. It is called propositional validity problem. In general, deciding for a closed $\varphi \in \Sigma_i^q$ whether $\nu(\varphi) = 1$ is Σ_i^P-complete. Similarly, deciding for a closed quantified boolean formula φ from Π_i^q whether $\nu(\varphi) = 1$ is Π_i^P-complete. In other words, the evaluation problem for quantified boolean formulas that are bounded by k alternations of quantifier blocks is either Σ_k^P-complete or Π_k^P-complete. The outermost quantifier decides to which class the problem belongs. The unbounded evaluation problem for quantified boolean formulas is PSPACE-complete.

3. Notational Conventions

If not defined otherwise, we may use:

- $a, b, c, \ldots, x_1, x_2, \ldots$ to denote propositional variables / constants.
- $\mathbf{a}, \mathbf{b}, \mathbf{c}, \ldots, \mathbf{x_1}, \mathbf{x_2}, \ldots$ to denote vectors of propositional variables / constants.
- $\alpha, \beta, \gamma, \ldots, \varphi_1, \varphi_2, \ldots$ to denote quantified boolean formulas (respectively propositional formulas).

For the ease of notation, we may use the following abbreviations:

- We allow to skip unnecessary parentheses as well as to add additional ones: $\varphi \vee \psi$ as well as $((\varphi \vee \psi))$ represent $(\varphi \vee \psi)$.
- $(\varphi \supset \psi)$ is an abbreviation of $((\neg\varphi) \vee \psi)$.
- $\wedge, \vee$ (and thus $\supset$) are treated as right-associative operators: $(\varphi_1 \supset \varphi_2 \supset \varphi_3)$ is an abbreviation of $(\varphi_1 \supset (\varphi_2 \supset \varphi_3))$.
- $\neg$ takes precedence over $\wedge$ which itself takes precedence over $\vee$: $(\varphi_1 \vee \neg\varphi_2 \wedge \varphi_3)$ is an abbreviation of $(\varphi_1 \vee ((\neg\varphi_2) \wedge \varphi_3))$.
- $\exists$ and $\forall$ are preceded by all operators mentioned before: $(\exists x\, \varphi \supset \forall y\, \psi)$ is an abbreviation of $(\exists x\, (\varphi \supset (\forall y\, \psi)))$.
- $(Q\mathbf{x}\, \varphi)$ with $Q \in \{\exists, \forall\}$ and $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ is an abbreviation of the formula $(Qx_1\, (Qx_2\, (\ldots\, (Qx_n\, \varphi)\, \ldots)))$.
- We will use the abbreviation $[\forall\exists]^*_{\mathbf{x}}$ to represent arbitrary quantifier sequences. As an example $[\forall\exists]^*_{\mathbf{x}}$ may stand for $\exists x_1\, \forall x_2\, \forall x_3\, \ldots\, \forall x_n$, or $\forall x_1\, \forall x_2\, \exists x_3\, \exists x_4\, \ldots\, \exists x_m$ or any other finite quantifier sequence of arbitrary length.
- $\bigwedge_{i=j}^{k} \varphi_i$ abbreviates $(\varphi_j \wedge \varphi_{j+1} \wedge \ldots \wedge \varphi_k)$ if $j < k$. If $j > k$ then $\bigwedge_{i=j}^{k} \varphi_i$ abbreviates $(\varphi_j \wedge \varphi_{j-1} \wedge \ldots \wedge \varphi_k)$. Otherwise $\bigwedge_{i=j}^{k} \varphi_i$ denotes $\top$.
- $\bigvee_{i=j}^{k} \varphi_i$ abbreviates $(\varphi_j \vee \varphi_{j+1} \vee \ldots \vee \varphi_k)$ if $j < k$. If $j > k$ then $\bigvee_{i=j}^{k} \varphi_i$ abbreviates $(\varphi_j \vee \varphi_{j-1} \vee \ldots \vee \varphi_k)$. Otherwise $\bigvee_{i=j}^{k} \varphi_i$ denotes $\bot$.

We use the following sets:

- $\mathbb{N} = \{0, 1, 2, \ldots\}$ are the natural numbers.
- $\mathbb{B} = \{\mathbf{1}, \mathbf{0}\}$ is the set of truth values.
- $\mathbf{P}[\mathbb{N}]$ is the set of polynomials over the natural numbers.
- $\mathcal{V}$ is a countable set of variable symbols.
- $\mathcal{C}$ is a countable set of constant symbols (mostly $\{\top, \bot\}$).
- $\mathcal{PL}$ is the set of propositional formulas.
- $\mathcal{QBF}$ is the set of quantified boolean formulas.

Further notation will be introduced when they are needed.

Part III.

GQBF

4. A Sequent Calculus for Quantified Boolean Formulas

In this chapter, we present a sequent calculus for quantified boolean formulas that is introduced in [6]. First we will present a basic, minimal sequent calculus that is able to prove the validity of any true quantified boolean formula in negation normal form. This minimal calculus is then extended by additional simplification rules. In the later sections of this chapter we will ease the constraints on the structure of proofs and will allow proofs in the form of directed acyclic graphs.

4.1. The Calculus

As said in the introductory part of this chapter, the authors of [6] present a sequent-style calculus called GQBF which operates on closed quantified boolean formulas in negation normal form. This means that GQBF cannot prove the validity of arbitrary quantified boolean formulas but of a subset of the quantified boolean formulas.

Negation Normal Form Let $\varphi \in \mathcal{QBF}$ be a quantified boolean formula. Then we say that φ is in negation normal form if none of the below transformations can be applied to φ (respectively its sub-formulas) any more:

$$\neg\neg\psi \longmapsto \psi \tag{4.1}$$

$$\neg(\varphi_1 \wedge \varphi_2) \longmapsto \neg\varphi_1 \vee \neg\varphi_2 \tag{4.2}$$

$$\neg(\varphi_1 \vee \varphi_2) \longmapsto \neg\varphi_1 \wedge \neg\varphi_2 \tag{4.3}$$

$$\neg(\forall x\, \psi) \longmapsto \exists x\, \neg\psi \tag{4.4}$$

$$\neg(\exists x\, \psi) \longmapsto \forall x\, \neg\psi \tag{4.5}$$

Note that as soon as none of the above transformations can be applied any more, negation does only occur on the atomic level (i.e., within literals). Furthermore it is noteworthy that each of these transformation rules preserves the truth value of the formula. Consequently $\varphi \equiv nnf(\varphi)$ holds.

Now let $\psi \in \mathcal{QBF}$ be an arbitrary quantified boolean formula. Applying the above transformations top down on ψ as long as possible results in a quantified boolean formula $\psi' = nnf(\psi)$ which is in negation normal form. At most linearly many (with respect to the size of ψ) transformation steps are performed. All

of them can be done in constant time. Hence the negation normal form of a formula can be calculated in a time polynomial in the size of the source formula. Also note that the rules (3.1) to (3.5) define a canonical rewrite system. Thus, every possible application order of the rules yields the very same irreducible result in negation normal form. Further discussion of this transformation, and of similar ones, can be found in [11]. We denote this transformation by $nnf(\cdot)$.

In the introductory part of this section we stated that GQBF operates on quantified boolean formulas in negation normal form only and that this restricts the calculus to just a subset of all quantified boolean formulas. With the *nnf* function we get on the other hand a very efficient procedure to create an input formula for GQBF. If we want to decide if an arbitrary quantified boolean formula is valid, we just compute its negation normal form and then apply the rules of the GQBF calculus to decide whether the negation normal form is valid. As *nnf* preservers the truth value we can deduce the truth value of the source formula when we know the truth value of the normal form. This step by step procedure allows us to determine the validity respectively unsatisfiability of any arbitrary, closed quantified boolean formula.

Indexing The original calculus [6] includes an indexing of the truth constants ($\top$ and $\bot$) as well as of variable symbols and formulas. Indices are added (bottom up) throughout the derivation process to annotate and track actions performed "in the past". While stemming from a practical point of view — these annotations are used to prune the search space during proof search — these indices may make proofs more readable for human viewers. This is enough reason for us to keep them (in a limited form) and use them for the best. We will employ indices to track truth constants introduced by quantifier elimination rules from their "birth place" upwards, towards the axioms in the proof tree.

A further motivation to keep and use indices is that they enable us to identify the set of variables that, in a way, decide the truth value of a formula. This will be described further later on. As in [6], we will refer to and manipulate indices in a string-like fashion. Also note that some calculi used in later chapters that may produce proofs in directed acyclic graph form cannot use such indices. This is due to the fact that in a directed acyclic graph there may exists many distinct paths from a node to a parent node.

Principal Derivation Rules The minimal sequent-style calculus GQBF contains two axioms and six derivation rules. Derivation rules (inference rules) consist of two parts, namely one or more premises and a single conclusion. Premises as well as conclusions are sequents $\vdash \varphi$ where φ is a closed quantified boolean formula in negation normal form. Depending on the number of premises, we partition the rules into unary and binary derivation rules. Unary rules derive a conclusion from a single premise while binary rules derive a conclusion from two premises. The rules $\vee_1$, $\vee_2$, $\exists_\bot$ and $\exists_\top$ are thus unary derivation

rules, while $\wedge$ and $\forall$ are binary inference rules. The derivation rules are depicted in Figure 4.1.

GQBF is cut-free, sound and complete. A proof of soundness and completeness can be found in the paper introducing GQBF [6]. More general proofs can be found in [20].

Axioms:

$$\vdash \top_\sigma \qquad \vdash \neg\bot_\sigma$$

Derivation Rules:

$$\frac{\vdash \varphi}{\vdash \varphi \vee \psi} \vee_1 \qquad \frac{\vdash \psi}{\vdash \varphi \vee \psi} \vee_2$$

$$\frac{\vdash \varphi \quad \vdash \psi}{\vdash \varphi \wedge \psi} \wedge$$

$$\frac{\vdash \varphi[x\backslash\bot_x]}{\vdash \exists x\, \varphi} \exists_\bot \qquad \frac{\vdash \varphi[x\backslash\top_x]}{\vdash \exists x\, \varphi} \exists_\top$$

$$\frac{\vdash \varphi[x\backslash\bot_x] \quad \vdash \varphi[x\backslash\top_x]}{\vdash \forall x\, \varphi} \forall$$

Whereas $\varphi \in \mathcal{QBF}$ and $\psi \in \mathcal{QBF}$ are closed (unitary) quantified boolean formulas in negation normal form.

Figure 4.1.: The axioms and derivation rules of GQBF.

Application To prove that a closed (unitary) quantified boolean formula φ is valid, the following procedure has to be applied:

1. Calculate the negation normal form $nnf(\varphi)$ of φ.
2. Use $\vdash nnf(\varphi)$ as root of a new GQBF proof.
3. Depending on the outermost quantifier respectively connective of $nnf(\varphi)$, apply a corresponding derivation rule. The application yields a new sequent respectively two new sequents.
4. For each new sequent (open branch), iterate step 3.
5. A branch is closed and not operated on any more if its top-level sequent is an axiom. Note that it may happen that a branch ends in a sequent that cannot be simplified further but is not an axiom. In such a case backtracking to the last "non-deterministic" derivation takes place. When

a sequent contains $\vee$ ($\exists$) as outermost connective (quantifier), the prover has to decide whether to continue with the rule $\vee_1$ or $\vee_2$ ($\exists_\top$ or $\exists_\bot$). If the choice does not yield closed branches only, it may have to be revised to construct a valid proof. This need for backtracking is one of the main sources of the exponential complexity of proof search. Even very short proofs can thus be very hard to find.

6. If all branches end in an axiom, $nnf(\varphi)$ is proven to be valid. Consequently, the validity of φ is proven. Note that proofs may be exponential in the size of the input formula. This is due to the $\forall$-rule. Each application doubles the number of sequents to be proven. This will be proven formally in subsequent chapters.

When applying the above procedure, either a GQBF proof in form of a tree is constructed or no GQBF proof can be found. If a proof is found, its root ($\vdash nnf(\varphi)$) is called end-sequent. Note that the structure of a proof of a formula is strongly dependent on the structure of the formula. This is due to the fact that derivation rules can only be applied to the outermost connective or quantifier. Each trace through a GQBF proof (from the root to an axiom) corresponds to a trace through the structure tree of the end-sequent. The $\forall$-and $\wedge$-rules are deterministic while the $\vee$- and $\exists$-rules leave the choice of how to continue upwards in the proof tree to the prover. Note that the premises of the inference rules as presented in Figure 4.1 are always (structural) sub-formulas of their conclusions. Thus Theorem 6.3 from [20] can easily be transferred from LK to GQBF: In a proof in GQBF all the formulas which occur in it are sub-formulas of the formula in the end-sequent.

We write $\pi\colon\ \vdash_{\mathrm{G}} \varphi$ to state that π is a GQBF proof of φ and $\vdash_{\mathrm{G}} \varphi$ to state that there exists a π such that $\pi\colon\ \vdash_{\mathrm{G}} \varphi$.

An Example Derivation Let $\varphi \in \mathcal{QBF}$ with $\varphi = \exists x \forall y\, (\exists z\, y \vee z) \wedge (x \vee (\forall v\, y \wedge \neg v))$ be a closed quantified boolean formula in negation normal form. Moreover, it can be shown by semantic evaluation that $\varphi \equiv \mathbf{1}$ holds. Then we can construct a GQBF proof of φ as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdash \top_y}{\vdash \top_y \vee \bot_z}\vee_1}{\vdash \exists z\, \top_y \vee z}\exists_\bot \quad \dfrac{\vdash \top_x}{\vdash \top_x \vee (\forall v\, \top_y \wedge \neg v)}\vee_1}{\vdash (\exists z\, \top_y \vee z) \wedge (\top_x \vee (\forall v\, \top_y \wedge \neg v))}\wedge \quad \dfrac{\dfrac{\dfrac{\vdash \top_z}{\vdash \bot_y \vee \top_z}\vee_2}{\vdash \exists z\, \bot_y \vee z}\exists_\top \quad \dfrac{\vdash \top_x}{\vdash \top_x \vee (\forall v\, \bot_y \wedge \neg v)}\vee_1}{\vdash (\exists z\, \bot_y \vee z) \wedge (\top_x \vee (\forall v\, \bot_y \wedge \neg v))}\wedge}{\vdash \forall y\, (\exists z\, y \vee z) \wedge (\top_x \vee (\forall v\, y \wedge \neg v))}\forall}{\vdash \exists x \forall y\, (\exists z\, y \vee z) \wedge (x \vee (\forall v\, y \wedge \neg v))}\exists_\top$$

Note that the indices of the truth constants $\top$ and $\bot$ are just "eye-candy". They should make proofs more readable. Thus it holds that $\varphi_\sigma = \varphi$ for any formula φ and any index σ. Contrary to the authors of [6], which introduce them to optimize proof search, we use indices to increase legibility only.

Size of a GQBF Proof Let $\varphi \in \mathcal{QBF}$ be a closed quantified boolean formula in negation normal form and $\pi\colon \vdash_{\mathrm{G}} \varphi$ a GQBF of φ. Then we define the size of a proof π, denoted by $|\pi|$, inductively as follows:

1. φ is an axiom: $|\pi| := 1$
2. π is a proof $\dfrac{\begin{array}{c}\tau\\ \vdots\end{array}}{\vdash \varphi}$ of $\vdash \varphi$ ending at a unary inference rule. Let moreover τ denote the proof of the rule's premise. Then $|\pi| := 1 + |\tau|$.
3. π is a proof $\dfrac{\begin{array}{cc}\tau_1 & \tau_2\\ \vdots & \vdots\end{array}}{\vdash \varphi}$ of $\vdash \varphi$ ending at a binary inference rule and let τ_1 and τ_2 denote the proofs of the rule's premises. Then $|\pi| := 1 + |\tau_1| + |\tau_2|$.

Defined this way, the size of a GQBF proof corresponds to the number of sequents occurring in the proof. It is important to remark that this does not necessary correspond to the number of distinct sequents: Various branches of the proof may contain the same sequent while each branch contains strictly distinct sequents only. This may not be the case for some of the calculi introduced subsequently.

5. Adding Simplification Rules

The previous chapter presented a sound and complete minimal sequent calculus called GQBF which operates on closed quantified boolean formulas. GQBF thus provides a sufficient tool to prove the validity of any possible quantified boolean formula. On one hand GQBF is easy to handle and to implement as the order in which inference rules are applied bottom-up in a branch of a proof strongly depends on the structure tree of the formula to be proven. GQBF always eliminates the outermost quantifier or connective while an inference rule's premises are always (structural) sub-formulas of the conclusion. On the other hand this strict dependency on the structure of the formula to be proven may produce extremely large proofs[1] which have to be searched for in a huge search space [6] — that moreover is hard to search because of the need for iterated backtracking.

These deficiencies in mind, the authors of [6] propose to add additional derivation rules to their minimal calculus. They augmented their minimal GQBF calculus with 19 simplification rules. In contrast to the six principal derivation rules, these simplification rules do not operate necessarily on the outermost connective (respectively quantifier) of the formula to be proven but may apply equivalence-preserving transformations within its sub-formulas. We will call a calculus containing the six principal rules together with the 19 simplification rules GQBF$^+$ or full GQBF.

The main motivation for introducing these 19 simplification rules is to reduce the space that needs to be searched to find proofs. They basically represent "short-cuts" that allow to eliminate dispensable parts of sequents or simplify unnecessary complex sequents. If such parts are not eliminated but stay in the sequent, they may exponentially lengthen proofs. This will be proven in later chapters. Simplification rules can be applied in a fast manner as modifications to formulas are mostly small and local. GQBF$^+$ is still cut-free.

As the minimal calculus is already sound and complete the additional rules of GQBF$^+$ are fully redundant. In [6], Lemma 2, it is shown that whenever there is a GQBF$^+$ proof of a formula, there exists a GQBF proof of the same formula: If π is a proof of φ with simplifications then there exists a proof τ for φ without simplifications.

[1] We will come back to this topic in Part IV and Part VI.

5.1. The Calculus

The Simplification Rules GQBF$^+$ contains two axioms, six principal derivation rules and 19 additional simplification rules[2] The latter ones are presented in Figure 5.1. The axioms and principal derivation rules are carried over from the minimal calculus. Note that for the additional simplification rules, it also holds that the size of their premise — they are all unary inferences — is smaller than the size of their conclusion. The rules are chosen in a way to avoid Ping-Pong effects[3]: The four *LU*-rules and the four *GU*-rules keep the size of the sequent constant and cannot be "undone" by subsequent rule applications. Thus no circular derivations $\vdash \varphi \rightsquigarrow \ldots \rightsquigarrow \vdash \varphi$ are possible. For all remaining rules it holds that their premise is strictly smaller than their conclusion. No circular derivations are possible.

Application GQBF$^+$ is applied in the same way as the minimal calculus is. If the formula to be proven is not in negation normal form, process the input formula to get an equivalent negation normal form. The formula to be proven (respectively the equivalent negation normal form) is taken as end-sequent and thus root of the derivation tree. One of the applicable rules is applied — either one of the newly added simplification rules or the principal derivation rule matching the outermost quantifier or connective. If all branches of the derivation tree can be closed (i.e., end in an axiom), the input formula is proven to be valid.

We write $\pi\colon\ \vdash_{\mathrm{G}^+} \varphi$ to state that π is a GQBF$^+$ proof of φ and $\vdash_{\mathrm{G}^+} \varphi$ to state that there exists a π such that $\pi\colon\ \vdash_{\mathrm{G}^+} \varphi$.

An Example Derivation Let $\varphi \in \mathcal{QBF}$ with $\varphi = \exists x \forall y\, (\exists z\, y \vee z) \wedge (x \vee (\forall v\, y \wedge \neg v))$ be a quantified boolean formula in negation normal form. Moreover $\varphi \equiv \mathbf{1}$. Then we can construct a GQBF$^+$ proof of φ as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdash \top_z}{\vdash \forall y\, \top_z}\, S4}{\vdash \forall y\, (y \vee \top_z)}\, S3a}{\vdash \forall y\, (\exists z\, y \vee z)}\, P1a}{\vdash \forall y\, (\exists z\, y \vee z) \wedge \top_x}\, S2a}{\vdash \forall y\, (\exists z\, y \vee z) \wedge (\top_x \vee (\forall v\, y \wedge \neg v))}\, S3a}{\vdash \exists x \forall y\, (\exists z\, y \vee z) \wedge (x \vee (\forall v\, y \wedge \neg v))}\, \exists_\top$$

[2] We do not present the commutative "twins" of the simplification rules explicitly. We use them in proofs nevertheless. When used, they are named after the twin presented in Figure 5.1. As an example, we may use two commutative "versions" of $S3a$: $\vdash \varphi(\top_\sigma \vee \psi) \rightsquigarrow \vdash \varphi(\top_\sigma)$ and $\vdash \varphi(\psi \vee \top_\sigma) \rightsquigarrow \vdash \varphi(\top_\sigma)$.

[3] It is nevertheless possible to create arbitrary deep branches using the *LU*-rules. An example can be found in the proof of Lemma 8.1.2. These cases on the other hand are easy to avoid in practical implementations and do not occur in short proofs.

Size of a Proof We define the size of a GQBF$^+$ proof in a manner we defined the size of GQBF proofs. We count the number of sequents that appear throughout the derivation tree. An inductive definition can already be found in the definition of the size of GQBF proofs. Again, it is important to stress that the number of sequents in a proof might be much higher than the number of distinct sequents as various branches might contain the very same sequent.

5.2. Soundness and Completeness of Full GQBF

Soundness There are two ways to prove the soundness of GQBF$^+$: The first one is to prove the correctness of the additional rules. The second way is to use Lemma 2 from [6] to prove soundness. Lemma 2 states that for each GQBF$^+$ proof that uses simplification rules, there exists a GQBF$^+$ proof that uses no simplification rules. A GQBF$^+$ proof without simplification rules is a GQBF proof by definition. On the other hand the elimination of the simplification rules does not come for free. It will be shown in subsequent chapters that the elimination of simplification rules may exponentially increase the proof size.

GQBF$^+$ is sound.

Proof. Let π be a GQBF$^+$ proof of a closed quantified boolean formula in negation normal form φ. Now assume that φ is not valid.

Then, by Lemma 2 from [6], we get from the existence of π a GQBF$^+$ proof τ of φ that does not contain any simplification rule inferences. As τ does contain the six principal inference rules only, it is GQBF proof. From the soundness of GQBF and the existence of τ it follows that φ is valid. Clearly, this contradicts the assumption. □

Completeness Completeness can be proven via GQBF too:

GQBF$^+$ is complete.

Proof. Let φ be a closed quantified boolean formula in negation normal form such that φ is valid. Now assume that there does not exists any GQBF$^+$ proof of φ.

As φ is valid and GQBF is complete, there exists a GQBF proof π of φ. Each GQBF proof is a GQBF$^+$ proof by definition. Thus there exists a GQBF$^+$ proof of φ. This contradicts the assumption. □

$$\frac{\vdash \varphi(\bot_\sigma)}{\vdash \varphi(\neg\top_\sigma)}\ S1a \qquad \frac{\vdash \varphi(\top_\sigma)}{\vdash \varphi(\neg\bot_\sigma)}\ S1b$$

$$\frac{\vdash \varphi(\psi_\sigma)}{\vdash \varphi(\top_\sigma \wedge \psi)}\ S2a \qquad \frac{\vdash \varphi(\bot_\sigma)}{\vdash \varphi(\bot_\sigma \wedge \psi)}\ S2b$$

$$\frac{\vdash \varphi(\top_\sigma)}{\vdash \varphi(\top_\sigma \vee \psi)}\ S3a \qquad \frac{\vdash \varphi(\psi_\sigma)}{\vdash \varphi(\bot_\sigma \vee \psi)}\ S3b$$

$$\frac{\vdash \varphi(\psi)}{\vdash \varphi(Qx\,\psi)}\ S4 \text{ (Guard: no occurrence of } x_\sigma \text{ in } \psi)$$

$$\frac{\vdash \varphi(x_\sigma \wedge \psi[x_{\sigma'}\backslash\top_{\sigma\sigma'x}])}{\vdash \varphi(x_\sigma \wedge \psi)}\ LU1a \qquad \frac{\vdash \varphi(\neg x_\sigma \wedge \psi[x_{\sigma'}\backslash\bot_{\sigma\sigma'x}])}{\vdash \varphi(\neg x_\sigma \wedge \psi)}\ LU1b$$

$$\frac{\vdash \varphi(x_\sigma \vee \psi[x_{\sigma'}\backslash\bot_{\sigma\sigma'x}])}{\vdash \varphi(x_\sigma \vee \psi)}\ LU2a \qquad \frac{\vdash \varphi(\neg x_\sigma \vee \psi[x_{\sigma'}\backslash\top_{\sigma\sigma'x}])}{\vdash \varphi(\neg x_\sigma \vee \psi)}\ LU2b$$

$$\frac{\vdash \varphi((x_\sigma \circ \psi)[x_{\sigma'}\backslash\top_{\sigma'x}])}{\vdash \varphi(\exists x\,(x_\sigma \circ \psi))}\ GU1a \qquad \frac{\vdash \varphi((\neg x_\sigma \circ \psi)[x_{\sigma'}\backslash\bot_{\sigma'x}])}{\vdash \varphi(\exists x\,(\neg x_\sigma \circ \psi))}\ GU1b$$

$$\frac{\vdash \varphi((x_\sigma \circ \psi)[x_{\sigma'}\backslash\bot_{\sigma'x}])}{\vdash \varphi(\forall x\,(x_\sigma \circ \psi))}\ GU2a \qquad \frac{\vdash \varphi((\neg x_\sigma \circ \psi)[x_{\sigma'}\backslash\top_{\sigma'x}])}{\vdash \varphi(\forall x\,(\neg x_\sigma \circ \psi))}\ GU2b$$

$$\frac{\vdash \varphi(\psi[x_\sigma\backslash\top_{\sigma x}])}{\vdash \varphi(\exists x\,\psi)}\ P1a \text{ (Guard: no negative occurrence of } x_{\sigma'} \text{ in } \psi)$$

$$\frac{\vdash \varphi(\psi[x_\sigma\backslash\bot_{\sigma x}])}{\vdash \varphi(\exists x\,\psi)}\ P1b \text{ (Guard: no positive occurrence of } x_{\sigma'} \text{ in } \psi)$$

$$\frac{\vdash \varphi(\psi[x_\sigma\backslash\bot_{\sigma x}])}{\vdash \varphi(\forall x\,\psi)}\ P2a \text{ (Guard: no negative occurrence of } x_{\sigma'} \text{ in } \psi)$$

$$\frac{\vdash \varphi(\psi[x_\sigma\backslash\top_{\sigma x}])}{\vdash \varphi(\forall x\,\psi)}\ P2b \text{ (Guard: no positive occurrence of } x_{\sigma'} \text{ in } \psi)$$

Figure 5.1.: Simplification rules augmenting GQBF.

6. From Trees to Directed Acyclic Graphs

6.1. The Calculus

So far, we only allowed GQBF and GQBF$^+$ proofs that have a structure in the form of a tree. In other words: every derivation started with an end-sequent in its root and kept branching with each branch eventually ending in an axiom. When talking about the size of a proof we concluded that the number of sequents in a proof can be much higher than the number of distinct sequents. From a human prover point of view, this reveals an unpleasant redundancy: If, in a closed proof tree, a sequent (which is not an axiom) appears more than once it means that it is proven several times. In such a case it would be preferable to "reuse" an already existing proof when the very same sequent appears once again.

From a mechanical point of view, "reuse" may not come without additional effort. Checking whether a specific sequent already appeared during proof search requires storing and retrieving representations of many sequents. If proof search is performed in a depth-first manner, it requires to store nearly all sequents of all branches closed so far. Storing sequents may have exponential space requirements as proofs may have exponential size.[1] This is in contrast to the fact that the evaluation problem for quantified boolean formulas is the canonical PSPACE problem. Breadth-first does not perform better.

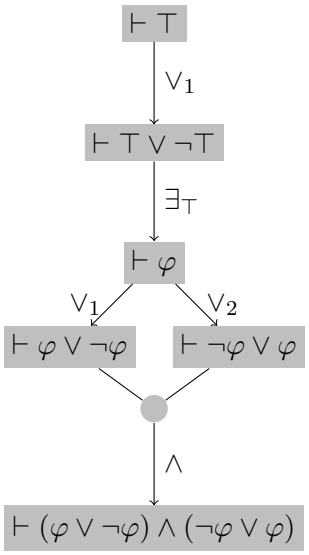
From a proof-theoretical perspective, we can allow the reuse of proofs by easing the restrictions on the form of GQBF$^+$ proofs. Instead of demanding proofs to form a tree we impose a less restrictive constraint on the structure of proofs: they are allowed to be directed acyclic graphs. The following example will demonstrate that allowing proofs to be in directed acyclic graph form instead of tree form can reduce proof size:

A Derivation Example Let $\varphi = (\exists x\, x \vee \neg x)$, $\psi_1 \in \mathcal{QBF}$ and $\psi_2 \in \mathcal{QBF}$ be closed quantified boolean formulas in negation normal form such that $\psi_1 \equiv \psi_2 \equiv \mathbf{0}$. Then we can prove the formula $(\varphi \vee \psi_1) \wedge (\psi_2 \vee \varphi)$ using GQBF$^+$ as follows:

[1] See Part IV for further details.

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdash \top_x}{\vdash (\top_x \vee \neg\top_x)}\,\vee_1}{\vdash \varphi}\,\exists_\top}{\vdash \varphi \vee \psi_1}\,\vee_1 \qquad \dfrac{\dfrac{\dfrac{\vdash \top_x}{\vdash (\top_x \vee \neg\top_x)}\,\vee_1}{\vdash \varphi}\,\exists_\top}{\vdash \psi_2 \vee \varphi}\,\vee_2}{\vdash (\varphi \vee \psi_1) \wedge (\psi_2 \vee \varphi)}\,\wedge$$

The size of the proof is nine. But each of the two branches contains the very same sequent $\vdash \varphi$. Moreover both branches contain exactly the same proof for $\vdash \varphi$. By reusing the proof of $\vdash \varphi$, it is possible to construct a shorter proof of this formula, a directed acyclic graph with six nodes. Two ways to annotate a proof in directed acyclic graph form are presented: Directly as a directed acyclic graph or in linear form using a table. Note that in the table, III appears twice as antecedent.



ID	Rule	Ant.	Consequent
I	-	-	$\vdash \top$
II	$\vee_1$	I	$\vdash \top \vee \neg\top$
III	$\exists_\top$	II	$\vdash \varphi$
IV	$\vee_1$	III	$\vdash \varphi \vee \neg\varphi$
V	$\vee_2$	III	$\vdash \neg\varphi \vee \varphi$
VI	$\wedge$	IV, V	$\vdash (\varphi \vee \neg\varphi) \wedge (\neg\varphi \vee \varphi)$

Application A calculus that contains all derivation rules of GQBF$^+$ and allows proofs to be in directed acyclic graph form is called GQBF$^+_\Diamond$. It is applied in the same way as the other GQBF calculi. If the formula to be proven is not in negation normal form, process the input formula to get an equivalent negation normal form. A sequent corresponding to the input formula (respectively the equivalent negation normal form) is taken as the root of the derivation tree. One of the applicable rules is applied. If in a branch a sequent is reached that has already been seen in some other location during proof search, we do not continue this specific branch. Instead we close it with a reference to the first occurrence of the same sequent. If all branches of the derivation graph can be closed (i.e., end in an axiom or a reference) the input formula is proven to be valid.

We write $\pi\colon\ \vdash_{\mathrm{G}^+_\Diamond} \varphi$ to state that π is a GQBF$^+_\Diamond$ proof of φ and $\vdash_{\mathrm{G}^+_\Diamond} \varphi$ to state that there exists a π such that $\pi\colon\ \vdash_{\mathrm{G}^+_\Diamond} \varphi$.

Size of a Proof We cane define the size of a $\mathrm{GQBF}^{+}_{\Diamond}$ proof in the same manner we defined the size of GQBF^{+} proofs. We count the number of sequents that appear throughout the derivation graph.

Let $\varphi \in \mathcal{QBF}$ be a closed quantified boolean formula in negation normal form and $\pi\colon \vdash_{\mathrm{G}^{+}_{\Diamond}} \varphi$ be a $\mathrm{GQBF}^{+}_{\Diamond}$ proof of φ. Then we define the size of π inductively as follows:

1. φ is an axiom: $|\pi| := 1$
2. π is a proof $\dfrac{R}{\vdash \varphi}$ of $\vdash \varphi$ ending at a unary inference rule. Let moreover R denote a reference to an arbitrary sequent. Then $|\pi| := 1$.
3. π is a proof $\dfrac{R_1 \quad R_2}{\vdash \varphi}$ of $\vdash \varphi$ ending at a binary inference rule. Let R_1 and R_2 denote references to arbitrary sequents. Then $|\pi| := 1$.
4. π is a proof $\dfrac{\begin{array}{c}\tau\\ \vdots\end{array}}{\vdash \varphi}$ of $\vdash \varphi$ ending at a unary inference rule. Let moreover τ denote the proof of the rule's premise. Then $|\pi| := 1 + |\tau|$.
5. π is a proof $\dfrac{\begin{array}{c}\tau\\ \vdots\end{array} \quad R}{\vdash \varphi}$ of $\vdash \varphi$ ending at a binary inference rule. Let τ denote the proof of the rule's first premise and R denote a reference to an arbitrary sequent. Then $|\pi| := 1 + |\tau|$.
6. π is a proof $\dfrac{R \quad \begin{array}{c}\tau\\ \vdots\end{array}}{\vdash \varphi}$ of $\vdash \varphi$ ending at a binary inference rule. Let τ denote the proof of the rule's second premise and R denote a reference to an arbitrary sequent. Then $|\pi| := 1 + |\tau|$.
7. π is a proof $\dfrac{\begin{array}{c}\tau_1\\ \vdots\end{array} \quad \begin{array}{c}\tau_2\\ \vdots\end{array}}{\vdash \varphi}$ of $\vdash \varphi$ ending at a binary inference rule and let τ_1 and τ_2 denote the proofs of the rule's premises. Then $|\pi| := 1 + |\tau_1| + |\tau_2|$.

While in each previous definition of proof size we pointed out the fact that the size of a proof is not necessary equal to the number of distinct sequents in the proofs, we want to point out here that for $\mathrm{GQBF}^{+}_{\Diamond}$ proofs these two numbers are identical. The number of distinct sequents in a proof is the size of the proof.

Soundness and Completeness of Directed Acyclic Graph GQBF $\mathrm{GQBF}^{+}_{\Diamond}$ is sound and complete. Completeness can be proven trivially as each GQBF^{+} proof is a $\mathrm{GQBF}^{+}_{\Diamond}$ proof per definition and GQBF^{+} is complete. A proof of the soundness can be constructed easily as follows:

Assume we have a $\mathrm{GQBF}^{+}_{\Diamond}$ proof π of a formula φ. If π is already in tree form we are finished as it is GQBF^{+} proof which itself is a sound and complete calculus. If not, we start bottom up to find the first position in π where two branches are re-joined. At least one such re-join exists as π is not a tree. At the re-join there is a branch that "normally" continues the proof and at least another one that references the continuation of the normal proof. We replace each reference by a copy of the sub-proof referenced. Clearly, such duplications can drastically increase the size of a proof since we stepwise expand a directed acyclic graph to a tree. If the resulting proof is a tree we are finished. Otherwise we search bottom up for the next join. Finally we end up in a tree proof.

7. Refuting Formulas

Motivation As described in this chapter and the previous chapters, the calculi GQBF, GQBF$^+$ and GQBF$^+_\diamond$ can only prove the validity of formulas directly. Invalidity can be proven indirectly: If a formula is not invalid, there does not exist any GQBF (GQBF$^+$, GQBF$^+_\diamond$) proof for it. As we want to compare GQBF and its extensions to Q-resolution with respect to proof size in the following, this non-constructive approach is not very helpful. While Q-resolution proofs of invalidity have a well-defined size, GQBF proofs of invalidity are purely existential. Hence we further extend GQBF, GQBF$^+$ and GQBF$^+_\diamond$ to be able to produce material proofs of the unsatisfiability of formulas. Such proofs can also be used as certificates for the unsatisfiability of a formula.

The following rule extends the known calculi:

$$\frac{\vdash \mathit{nnf}(\neg\varphi)}{\varphi \vdash}\ \neg$$

Note that this rule cannot introduce Ping-Pong effects as it can only be applied once: If we want to prove the invalidity of a formula φ, we start bottom-up to prove it. $\varphi \vdash$ be the end-sequent. Such an end-sequent does only allow to apply the $\neg$-rule. This results (bottom-up) in a sequent $\vdash \varphi'$. As there is no rule to rule in our calculi to infer $\vdash \psi$ from $\psi' \vdash$, the $\neg$-rule cannot be applied any more. Thus it can only appear in the very root of a proof and it can be applied at most once. We will add $+\neg$ to the name of a calculus to state that it is augmented with the $\neg$-rule. A GQBF$+\neg$, GQBF$^+ +\neg$ or GQBF$^+_\diamond +\neg$ proof tree (directed acyclic graph) ending in an end-sequent $\varphi \vdash$ is a proof of the invalidity of φ.

Correctness of the $\neg$-Rule Assume the $\neg$-rule appears in a proof and its premise is $\vdash \mathit{nnf}(\neg\varphi)$. Then it follows from the soundness of the base calculus (i.e., GQBF, GQBF$^+$ or GQBF$^+_\diamond$) that $\mathit{nnf}(\neg\varphi) \equiv \mathbf{1}$. It can be show semantically that $\mathit{nnf}(\neg\varphi) \equiv \neg\varphi$ and thus $\varphi \equiv \mathbf{0}$. Henceforth the invalidity of φ can safely be derived: $\varphi \vdash$.

An Example Derivation Let $\varphi = \forall x\,(x \wedge \neg x)$ be a closed quantified boolean formula in negation normal form. It can be semantically shown that $\varphi \equiv \mathbf{0}$. This can also be proven using GQBF$+\neg$:

$$\dfrac{\dfrac{\dfrac{\vdash \top_x}{\vdash \neg\top_x \vee x}\ \vee_2}{\vdash \exists x\,(\neg x \vee x)}\ \exists_\top}{\forall x\,(x \wedge \neg x) \vdash}\ \neg$$

Size of a Proof The sizes of GQBF+$\neg$ and GQBF$^+$+$\neg$ proofs are calculated the same way as the size of GQBF proofs are while GQBF$^+_\diamond$+$\neg$ proofs are measured the same way GQBF$^+_\diamond$ proofs are.

Part IV.

Proof Size

8. Features and Proof Size

In the last chapter, we presented three different versions of a sequent calculus for quantified boolean formulas. It starts with presenting a minimal calculus that is already been sound and complete. In the following we introduced a calculus that augments the minimal calculus with 19 additional simplification rules. Two example derivations, one with GQBF, the other with GQBF$^+$ of the same formula already hints that GQBF$^+$ can produce shorter proofs than its minimal predecessor. In this chapter we will show that proofs can even be exponentially shorter. We will continue with proving that the third calculus, which allows proofs in the form of directed acyclic graphs instead of trees, can even produce proofs that are exponentially shorter than GQBF$^+$ proofs. Eventually we will show that GQBF$^+_\diamond$ is not capable of proving all tautologies polynomially. We want to remind the reader of the fact that all calculi that are derived from GQBF are *cut*-free.

8.1. The Power of Simplification Rules

We will start by showing that there is an infinite family of quantified boolean formulas in negation normal form such that no formula from this family can be proven polynomially using GQBF.

Lemma 8.1.1. Closed quantified boolean formulas that are structured in the form of $[\forall\exists]^*_{\mathbf{y}}\,\forall x_1\,\forall x_2 \ldots \forall x_n\,[\forall\exists]^*_{\mathbf{z}}\,\psi$, $n \geq 1$, $(\{x_1, x_2, \ldots x_n\} \cup \mathbf{y} \cup \mathbf{z}) = \mathit{free}(\psi)$, cannot have a GQBF proof of a size smaller than $2^{n+1} - 1$.

Proof. Let $n \geq 1$, $\varphi_n = ([\forall\exists]^*_{\mathbf{y}}\,\forall x_1\,\forall x_2 \ldots \forall x_n\,[\forall\exists]^*_{\mathbf{z}}\,\psi) \in \mathcal{QBF}$ be a closed quantified boolean formula in negation normal form and let $\pi\colon\ \vdash_{\mathrm{G}} \varphi$ be the shortest proof of φ_n.

- A proof of $\forall a_1\,\forall a_2 \ldots \forall a_n\,\psi'$, $\{a_1, a_2, \ldots a_n\} = \mathit{free}(\psi')$, has at least size 2^n: the bottom part of the proof tree[1] contains $2^n - 1$ $\forall$-rules. Each of the 2^n branches that stem from the bottom part of the tree have at least size 1 as ψ' contains all a_i. Thus at least one connective has to be "eliminated" in a proof of $\psi'[a_1 \backslash c_1][\ldots][a_n \backslash c_n]$. We get a minimum size of $2^{n+1} - 1$.
- The quantifier prefix of φ is of a form $[\forall\exists]^*\forall^n[\forall\exists]^*$. The idea presented above can be used to prove that each formula with a quantifier prefix of

[1] When speaking about the bottom part of a proof we refer to the root of the proof tree, the nodes that are connected to the root and all other nodes of a low depth.

the form $\forall^n[\forall\exists]^*$ cannot have a GQBF proof of a size polynomial in n: Simply move the "inner" prefix part into the ψ'-part of the formula.

- The outermost quantifiers ($[\forall\exists]^*_{\mathbf{y}}$) cannot reduce proof size either. 1) Assume the innermost quantifier of this group is a universal one. Then, again, it is just "swallowed" by the above idea and we can iterate with the next one. 2) The innermost quantifier is existential. In this case we know from the first bullet that each proof of $\forall a_1\,\forall a_2 \ldots \forall a_n\,\psi'[y\backslash\top]$ as well as of $\forall a_1\,\forall a_2 \ldots \forall a_n\,\psi'[y\backslash\bot]$ has to have at least size $2^{n+1}-1$. Thus, a proof of $\exists y\,\forall a_1\,\forall a_2 \ldots \forall a_n\,\psi'$ has at least this size too. We can iterate again and eventually prove that a GQBF proof has at least a size of $2^{n+1}-1$.

Thus the shortest proof of φ_n has a size greater or equal to $2^{n+1}-1$. □

Corollary 8.1.1. There are formulas of length $p(n)$, $p \in \mathbf{P}[\mathbb{N}]$, for which the smallest GQBF proof has a size exponential in n.

Proof. Let

$$\varphi_n = \exists a\,\forall x_1\,\forall x_2 \ldots \forall x_n\,\exists b\,((x_1 \vee a \vee b) \wedge (x_2 \vee a \vee b) \wedge \ldots \wedge (x_n \vee a \vee b)) \quad (8.1)$$

be a closed quantified boolean formula in negation normal form. Then φ_n has a length of $(8n+3)$ which is polynomial in n. From Theorem 8.1.1 it follows that the smallest GQBF proof for φ_n has at least size $2^{n+1}-1$ and thus is exponential in n. □

Knowing that there is an infinite family of formulas that cannot be proven polynomially with GQBF, it will be shown now that a special sub-family that is still infinite can be proven polynomially using GQBF$^+$.

Theorem 8.1.1. Let φ_n be Formula (8.1) from Corollary 8.1.1. Then there is a $p \in \mathbf{P}[\mathbb{N}]$ such that there is a GQBF$^+$ proof of length $p(n)$ of φ_n. Moreover there is no GQBF proof of a length polynomial in n.

Proof. Let φ_n be Formula (8.1) from Corollary 8.1.1. Then π: $\vdash_{\mathrm{G}^+} \varphi_n$, a GQBF$^+$ proof of φ_n can be constructed as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdash \top_a}{\vdash \top_a \wedge \top_a \wedge \ldots \wedge \top_a}\,S2a\ ((n-1)\text{ times})}{\vdash \forall x_1\,\forall x_2 \ldots \forall x_n\,\exists b\,(\top_a \wedge \top_a \wedge \ldots \wedge \top_a)}\,S4\ ((n+1)\text{ times})}{\vdash \forall x_1\,\forall x_2 \ldots \forall x_n\,\exists b\,((x_1 \vee \top_a \vee b) \wedge (x_2 \vee \top_a \vee b) \wedge \ldots \wedge (x_n \vee \top_a \vee b))}\,S3a\ (2n\text{ t.})}{\vdash \exists a\,\forall x_1\,\forall x_2 \ldots \forall x_n\,\exists b\,((x_1 \vee a \vee b) \wedge (x_2 \vee a \vee b) \wedge \ldots \wedge (x_n \vee a \vee b))}\,\exists_\top$$

Constructed this way, $|\pi| = 4n+1$ holds. By Corollary 8.1.1 we know that there can be no GQBF proof for φ_n of a size less than $2^{n+1}-1$.

□

So far we have shown that there are formulas that cannot proven polynomially using GQBF. We "defused" some of these formulas by adding simplification rules to our calculus: There are certain formulas that cannot proven polynomially in GQBF but in GQBF$^+$. Subsequently we will prove that there is still an infinite family of quantified boolean formulas that cannot be proven polynomially in GQBF$^+$.

Lemma 8.1.2. Let

$$\varphi_n = \forall x_1 \exists y_1 \ldots \forall x_n \exists y_n \left(\psi(x_1, y_1) \wedge \ldots \wedge \psi(x_n, y_n)\right) \tag{8.2}$$

with $\psi(x, y) = ((x \supset y) \wedge (y \supset x))$ be a closed quantified boolean formula in negation normal form of size $(12n - 1)$. Then there exists no GQBF$^+$ proof of φ_n of a size polynomial in n.

Proof. A proof of φ_n is built bottom up. Assume that φ_n is provable, then the last inference step of the proof has to look like this:

$$\frac{\vdash \varphi'_n}{\vdash \forall x_1 \exists y_1 \ldots \forall x_n \exists y_n \left(\psi(x_1, y_1) \wedge \ldots \wedge \psi(x_n, y_n)\right)} \; ?$$

$\vdash \varphi_n$ is the end-sequent to be proven. Either a principal derivation rule or one of the simplification rules has to be applied to continue the proof. The rule chosen fully defines the antecedent $\vdash \varphi'_n$.

We now determine which rules can be applied, having $\vdash \varphi_n$ as consequent and how φ'_n has to look like. The additional rules that will be first looked upon at are as follows:

- $S1a, S1b, S2a, S2b, S3a, S3b$: These rules cannot be applied since they contain some variant of the truth constants in their consequent. However, φ_n does not contain any.
- $S4$: This rule cannot be applied too. There is no quantifier in φ_n whose bound variable does not occur in the matrix of φ_n.
- $LU1a$ as well as $LU1b$ cannot be applied since φ_n has no sub-formula of the form $(\ell \wedge \Delta)$, with ℓ being a literal and $\Delta \in \mathcal{QBF}$.
- $LU2a$ as well as $LU2b$ can be applied as φ_n contains sub-formulas $(\neg x_i \vee y_i)$ and $(x_i \vee \neg y_i)$. But if we use one of these two rules, the antecedent has to be φ_n again as the applied substitution yields an identical result. Take this as an example:

  $$\frac{\vdash \Delta(x_i \vee (\neg y_i)[x_i \backslash \bot_{x_i}])}{\vdash \Delta(x_i \vee \neg y_i)} \; LU2a$$

  Thus these rule applications are redundant and will not appear in the shortest proofs possible.

- $GU1a, GU1b, GU2a, GU2b$ cannot be applied since φ_n, being in prenex normal form[2], has no sub-formula of the form $Qx\,(\ell \circ \Delta)$, with ℓ being a literal of x, $Q \in \{\exists, \forall\}$, $\circ \in \{\vee, \wedge\}$ and $\Delta \in \mathcal{QBF}$.

- $P1a, P1b, P2a, P2b$ cannot be applied since φ_n is in prenex normal form and the matrix always contains both polarities of a variable.

Because none of the additional rules can be applied, we have to use the $\forall$-rule and thus get the following inference steps at the root of the proof tree ($\mathbf{Q}$ abbreviates $\exists y_1\, \forall x_2\, \exists y_2\, \ldots\, \forall x_n\, \exists y_n$):

$$\dfrac{\dfrac{?}{S_1\colon\ \vdash \mathbf{Q}\,(\psi(\top_{x_1}, y_1) \wedge \ldots \wedge \psi(x_n, y_n))}\ ? \quad \dfrac{?}{S_2\colon\ \vdash \mathbf{Q}\,(\psi(\bot_{x_1}, y_1) \wedge \ldots \wedge \psi(x_n, y_n))}\ ?}{\vdash \forall x_1\, \exists y_1\, \ldots\, \forall x_n\, \exists y_n\,(\psi(x_1, y_1) \wedge \ldots \wedge \psi(x_n, y_n))}\ \forall$$

As the right ending with S_2, is just symmetrical to the left one ending with S_1, we will concentrate on the left branch. Repeating the above evaluation of the simplification rules, we conclude for the left branch that we can continue the proof with these (useful) sequences of rule applications only:

- $(S3a, S2a, P1a \text{ or } \exists_\top, S3a, S2a)$
- $(S3a, P1a \text{ or } \exists_\top, S2a, S3a, S2a)$
- $(S3a, P1a \text{ or } \exists_\top, S3a, S2a, S2a)$
- $(\exists_\top, S3a, S2a, S3a, S2a)$
- $(\exists_\top, S3a, S3a, S2a, S2a)$

Other sequences of rule applications are either not applicable or result, unnecessary increases of proof length (e.g. multiple $LU2a$ and $LU2b$ rule applications that do not change the sequent at all) or do not yield a provable sequent (e.g. $\exists_\bot$)). Since all the above sequences of rule applications have the same length, we use the latter as it has the most straight forward structure. This results in the following inference steps:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdash \varphi_{n-1}}{\vdash \forall x_2\, \exists y_2\, \ldots\, \forall x_n\, \exists y_n\,(\top_{y_1} \wedge \bigwedge_{i=2}^{n} \psi(x_i, y_i))}\ S2a}{\vdash \forall x_2\, \exists y_2\, \ldots\, \forall x_n\, \exists y_n\,(\top_{y_1} \wedge \top_{x_1} \wedge \bigwedge_{i=2}^{n} \psi(x_i, y_i))}\ S2a}{\vdash \forall x_2\, \exists y_2\, \ldots\, \forall x_n\, \exists y_n\,((\neg\top_{x_1} \vee \top_{y_1}) \wedge \top_{x_1} \wedge \bigwedge_{i=2}^{n} \psi(x_i, y_i))}\ S3a}{\vdash \forall x_2\, \exists y_2\, \ldots\, \forall x_n\, \exists y_n\,((\neg\top_{x_1} \vee \top_{y_1}) \wedge (\top_{x_1} \vee \neg\top_{y_1}) \wedge \bigwedge_{i=2}^{n} \psi(x_i, y_i))}\ S3a}{\vdash \exists y_1\, \forall x_2\, \exists y_2\, \ldots\, \forall x_n\, \exists y_n\,((\neg\top_{x_1} \vee y_1) \wedge (\top_{x_1} \vee \neg y_1) \wedge \bigwedge_{i=2}^{n} \psi(x_i, y_i))}\ \exists_\top$$

[2]See Section 9.2 for further details.

Thus, applying this sequence of rules, we get in the left branch a smaller instance of φ_n, $\varphi_{n-1} = \forall x_2 \exists y_2 \ldots \forall x_n \exists y_n (\psi(x_2, y_2) \wedge \ldots \wedge \psi(x_n, y_n))$, as the formula to be proven. For the right open branch we apply a similar procedure and end in $\vdash \varphi_{n-1}$ as open sequent too. We can now iterate the above inference steps in both open branches.

In the end we get 2^n open branches. Each of them can be closed with a short proof. Henceforth we conclude that the minimal proof size is at least 2^n and thus exponential in n. □

Note that although no formula from the family (8.2) can be proven polynomially in n using GQBF$^+$, each such formula can in general be evaluated in a time linear in n. A linear-time algorithm for testing the truth of closed quantified boolean formulas in 2-PCNF can be found in [2]. It will also be presented in Section 11.3.

8.2. The Power of Reusing Sub-Proofs

We established an infinite family of formulas that is not provable polynomially with GQBF$^+$ in Lemma 8.1.2. In the following it is proven that formulas of this family can be proven polynomially using GQBF$^+_\diamond$.

Theorem 8.2.1. Let

$$\varphi_n = \forall x_n \exists y_n \ldots \forall x_1 \exists y_1 (\psi(x_n, y_n) \wedge \ldots \wedge \psi(x_1, y_1)) \tag{8.3}$$

with $\psi(x, y) = ((x \supset y) \wedge (y \supset x))$ be a closed quantified boolean formula in negation normal form of size $(12n - 1)$.[3]

Then there is a GQBF$^+_\diamond$ proof of φ_n with a length of $(11n-1)$ but there exists no GQBF$^+$ proof of φ_n with a length polynomial in n.

Proof. We prove by induction on n that there exists a GQBF$^+_\diamond$ proof of φ_n of a size polynomial in n. Thereby we use the following abbreviations:

- $\mathbf{Q}_n$ is used to abbreviate $\exists y_n \forall x_{n-1} \exists y_{n-1} \ldots \forall x_1 \exists y_1$.
- $\mathbf{Q'}_n$ is used to abbreviate $\forall x_n \exists y_n \ldots \forall x_1 \exists y_1$.
- Δ_n is used to abbreviate $\bigwedge_{i=n}^{2} ((x_i \supset y_i) \wedge (y_i \supset x_i))$.

Base Case: Then $n = 1$ and $\varphi_1 = \forall x_1 \exists y_1 ((\neg x_1 \vee y_1) \wedge (\neg y_1 \vee x_1))$ is a closed quantified boolean formula in negation normal form. A GQBF$^+_\diamond$ proof π_1 of φ_1 can be constructed as follows:

[3] φ_n is identical to Formula (8.2).

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdash \top}{\vdash \top \wedge \top}\,S2a}{\vdash (\neg\top \vee \top) \wedge \top}\,S3a}{\vdash (\neg\top \vee \top) \wedge (\neg\top \vee \top)}\,S3a}{\vdash \exists y_1\,((\neg\top \vee y_1) \wedge (\neg y_1 \vee \top))}\,\exists_\top \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdash \top}{\vdash \top \wedge \top}\,S3a}{\vdash \top \wedge (\top \vee \bot)}\,S3a}{\vdash (\top \vee \bot) \wedge (\top \vee \bot)}\,S1b}{\vdash (\neg\bot \vee \bot) \wedge (\neg\bot \vee \bot)}\,S1b}{\vdash \exists y_1\,((\neg\bot \vee y_1) \wedge (\neg y_1 \vee \bot))}\,\exists_\bot}{\vdash \forall x_1\,\exists y_1\,((\neg x_1 \vee y_1) \wedge (\neg y_1 \vee x_1))}\,\forall$$

The size of π_1 is constant, i.e. $|\pi_1| = 10$. Note that the sequents $\vdash \top \wedge \top$ and $\vdash \top$ are shared by both branches and thus only counted once. They are displayed two times just for a matter of presentation.

Induction hypothesis: Let $n > 0$ and φ_n be Formula (8.2) Then there is a GQBF$^+_\Diamond$ proof of φ_n with a length of $(11n - 1)$.

Induction step $((n-1) \to n)$: Then φ_n is Formula (8.2) and a proof π_n of φ_n can be constructed as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\pi_{n-1}}{\vdash \varphi_{n-1}}}{\vdash \mathbf{Q'}_n\,(\top \wedge \Delta_n)}\,S2a}{\vdash \mathbf{Q'}_n\,(\top \wedge \top \wedge \Delta_n)}\,S2a}{\vdash \mathbf{Q'}_n\,((\neg\top \vee \top) \wedge \top \wedge \Delta_n)}\,S3a}{\vdash \mathbf{Q'}_n\,((\neg\top \vee \top) \wedge (\top \vee \neg\top) \wedge \Delta_n)}\,S3a}{\vdash \mathbf{Q}_n\,((\neg\top \vee y_n) \wedge (\top \vee \neg y_n) \wedge \Delta_n)}\,\exists_\top \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\pi_{n-1}}{\vdash \varphi_{n-1}}}{\vdash \mathbf{Q'}_n\,(\top \wedge \Delta_n)}\,S2a}{\vdash \mathbf{Q'}_n\,(\top \wedge \top \wedge \Delta_n)}\,S2a}{\vdash \mathbf{Q'}_n\,((\top \vee \bot) \wedge \top \wedge \Delta_n)}\,S3a}{\vdash \mathbf{Q'}_n\,((\top \vee \bot) \wedge (\bot \vee \top) \wedge \Delta_n)}\,S3a}{\vdash \mathbf{Q'}_n\,((\top \vee \bot) \wedge (\bot \vee \neg\bot) \wedge \Delta_n)}\,S1b}{\vdash \mathbf{Q'}_n\,((\neg\bot \vee \bot) \wedge (\bot \vee \neg\bot) \wedge \Delta_n)}\,S1b}{\vdash \mathbf{Q}_n\,((\neg\bot \vee y_n) \wedge (\bot \vee \neg y_n) \wedge \Delta_n)}\,\exists_\bot}{\vdash \varphi_n}\,\forall$$

The sequents $\vdash \mathbf{Q'}_n(\top \wedge \top \wedge \Delta_n)$, $\vdash \mathbf{Q'}_n(\top \wedge \Delta_n)$ and $\vdash \varphi_{n-1}$ appear in both branches. As we allow proof to be in the form of directed acyclic graphs, such duplicate sequents refer to the same node of the proof graph and are thus only counted once when the size of π_n is calculated. The size of π_n is calculated from the size of the inference displayed above and the size of the sub-proof π_{n-1}. Henceforth $|\pi_n| = |\pi_{n-1}| + 11$. The sequent $\vdash \varphi_{n-1}$ is not counted as it is already in the size of π_{n-1}. From the induction hypothesis, we know that $|\pi_{n-1}|$ is $(11(n-1)-1)$. Thus $|\pi_n| = ((11(n-1)-1)+11) = (11n-1)$.

A graph that shows the structure of a proof of φ_n as a whole can be found in Figure 8.1. The GQBF$^+_\Diamond$ proof has a size of $(11n-1)$ which is clearly polynomial in n. As outlined by Lemma 8.1.2, there does not exist a GQBF$^+$ proof of φ_n with a size polynomial in n.

□

Still, there is a infinite class of formulas that cannot be proven polynomially using $\mathrm{GQBF}^{+}_{\diamond}$.

Lemma 8.2.1. Let

$$\varphi_n = \forall x_1 \exists y_1 \forall x_2 \exists y_2 \ldots \forall x_n \exists y_n \exists z_1 \exists z'_1 \exists z_2 \exists z'_2 \ldots \exists z_n \exists z'_n \bigwedge_{i=1}^{n} \psi_i \qquad (8.4)$$

be a quantified boolean formula in negation normal form with

$$\psi_i = (\neg x_i \vee y_i) \wedge (\neg y_i \vee x_i) \wedge (x_i \vee z_i \vee z'_i) \wedge (\neg x_i \vee \neg z_i \vee \neg z'_i) \wedge (z_i \vee z'_i) \wedge (\neg z_i \vee \neg z'_i).$$

Then there is no $\mathrm{GQBF}^{+}_{\diamond}$ proof of φ_n with a size polynomial in n.

Proof. The idea of this proof is very similar to the ideas used in the proof of Theorem 8.1.2: It is shown that no simplification rule can be applied to φ_n. Thus the only possibility to start a $\mathrm{GQBF}^{+}_{\diamond}$ proof is to apply the $\forall$-rule. This splits the proof into two separate branches. Now it shown that the two branches are not identical and that no sequence of rule applications (that does not contain the $\forall$-rule) can make their sequents identical. Hence two branches remain open. Each of them then has to be split again using the $\forall$-rule. After some iterations of the same procedure we end up in exponentially many open branches. Now, with eliminating the existential quantifiers, all branches can be re-merged and closed.

We will present the above idea in more details in the following:

1. No simplification rule is applicable. The rules $S[123][ab]$ are not applicable as the formula does not contain any truth constants. The rule $S4$ cannot be applied since there is no quantifier that binds no variable occurrence. There are many variable occurrences that are locally unit. But each of them is locally unit in a trivial context[4] only. So $LU[12][ab]$ will not occur in the shortest proof. No variable occurrence is globally unit. The rules $GU[12][ab]$ cannot be applied. As all variables do appear in both polarities, none of the $P[12][ab]$-rules can be used.

2. Thus the $\forall$-rule has to be applied and the proof is split into two open branches.

   - The positive branch: In this branch x_1 is replaced by $\top$. Changes in the sequent appear only in the term ψ_1. We apply $\exists_\top$ and a sequence of simplification rules to eliminate the first two conjuncts of

[4] We call a variable occurrence trivially unit if applying a $LU[12][ab]$-rule to it will not change the formula at all. This happens if the formula it is connected to via $\vee$ or $\wedge$ does not contain any occurrences of the variable symbol. This can be demonstrated with the following example: Let $\xi(x \vee (a \wedge b))$ be a quantified boolean formula. Then x is locally unit in the disjunction $(x \vee (a \wedge b))$. But x does not appear in the second disjunct $(a \wedge b)$. Thus applying $LU2a$ will yield the same formula $\xi(x \vee (a \wedge b))$ as $(a \vee b)[x \backslash \bot] = (a \vee b)$.

ψ_1. Simplification rules are used also to eliminate the truth constants from the third and fourth conjunct. Thus, three conjuncts of ψ_1 remain: $(\neg z_1 \vee \neg z_1')$, $(z_1 \vee z_1')$ and $(\neg z_1 \vee \neg z_1')$.

- The negative branch: We proceed the same way as in the positive branch: $\exists_\bot$ and simplification rules trim ψ_1 to a conjunction of the following three disjuncts: $(z_1 \vee z_1')$, $(z_1 \vee z_1')$ and $(\neg z_1 \vee \neg z_1')$.

3. The remainder of ψ_1 cannot be simplified further in neither branch. The remainder of ψ_1 does not contain truth constants, no non-trivial locally unit variable occurrences and no globally unit variable occurrences. All its variables are present in both polarities and the formula as a whole does not contain "unnecessary" quantifiers. Hence no simplification rule can be applied in neither branch. The left and the right branch are strictly different. Thus they cannot be merged too. Two branches remain open.

4. The $\forall$-rule has to be applied in each branch to eliminate the second universal quantifier. Again, changes are local only and are limited to ψ_2. The above argumentation can be reused and the procedure can be iterated in every open branch.

5. After applying the $\forall$-rule n times (in each open branch), 2^n branches are open. Each one can be closed with a short proof.

□

This formula family demonstrates that, although proofs can get exponentially shorter through eliminating duplicate sequents with directed acyclic graph proofs, duplicate sub-formulas in sequents can still exponentially enlarge proofs.

We have now established a kind of hierarchy between the three GQBF calculi. The findings of this chapter will be used later on in the Chapter 9 and Part VI. Chapter 9 will investigate in more detail where the complexity of certain families of formulas stems from and whether there are efficient means to reduce it. Part VI will deal with the relations between different calculi in more detail.

At the end of this chapter we want to stress that we speak about proof sizes when we talk about proof complexity. Furthermore we are always interested in the shortest proof possible for an actual formula (with respect to a specific calculus). In general, given a formula φ of size n, a calculus A, and the smallest A-proof of φ with a size of $f(n)$, the run-time complexity of searching a proof for φ in A is $\Omega(f(n))$. Note that additional optimization strategies that are widely used in proof search (like dependency-directed backtracking [6]) can be used to keep the search complexity as close as possible to the lower bound.

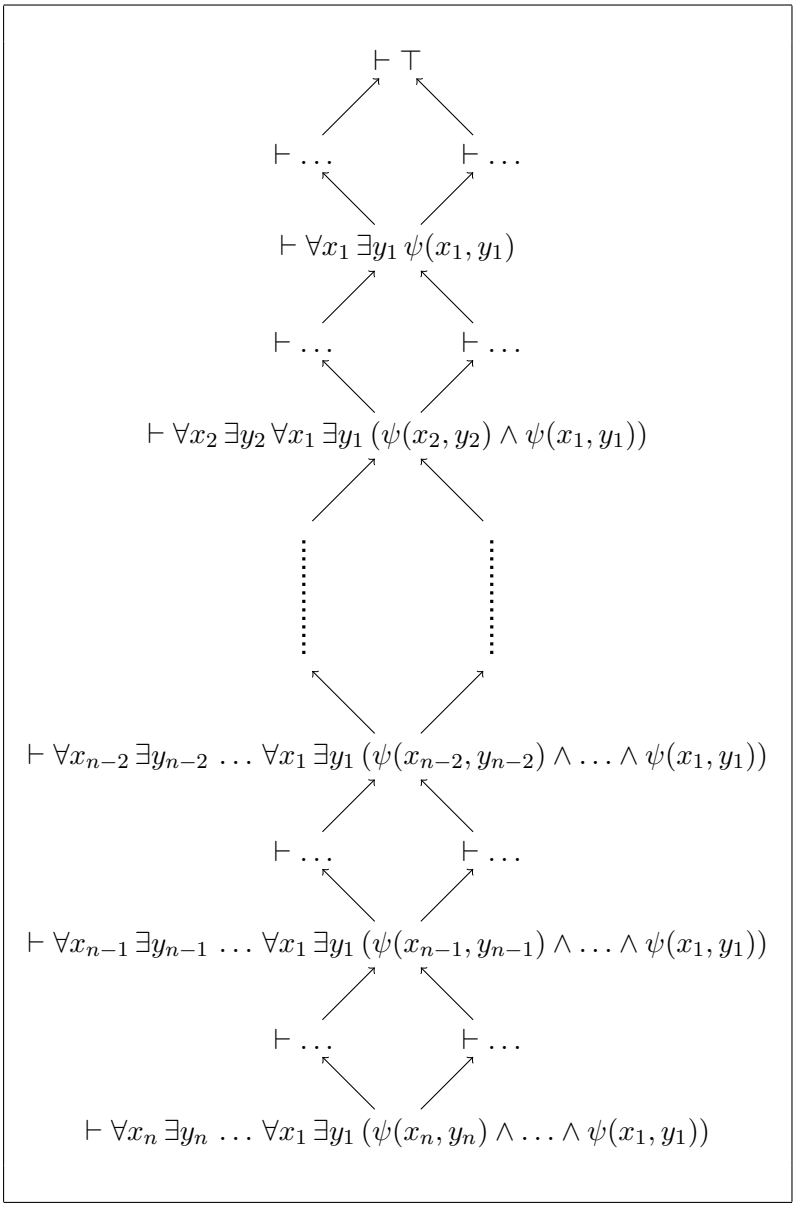


Figure 8.1.: The Structure of the GQBF$^{+}_{\diamond}$ Proof.

9. Transformations and Proof Size

In this part we will discuss how a change within a formula affects the size of its smallest GQBF (GQBF$^+$, GQBF$^+_\Diamond$) proof. We will show amongst others that prenexing — i.e., moving quantifiers up in the structure tree of a formula — can exponentially enlarge GQBF proofs. We start with an introduction of additional normal forms used in this part and in Part V.

9.1. Equivalent Replacement

Theorem 9.1.1 (Equivalent replacement for quantified boolean formulas)**.** Let φ_1, φ_2 be two quantified boolean formulas such that $\varphi_1 \equiv \varphi_2$ holds. Furthermore let $\omega(\varphi_1)$ be a quantified boolean formula[1] and $\omega(\varphi_2)$ be a copy of $\omega(\varphi_1)$ where each sub-formula occurrence of φ_1 is replaced by φ_2. Then it holds that $\omega(\varphi_1) \equiv \omega(\varphi_2)$.

Proof. Let φ_1, φ_2, $\omega(\varphi_1)$ and $\omega(\varphi_2)$ be quantified boolean formulas such that $\varphi_1 \equiv \varphi_2$ holds. Then we prove by induction that $\omega(\varphi_1) \equiv \omega(\varphi_2)$.

Induction Base:

- If $\omega(\varphi_1) = x$ with $x \in (\mathcal{C} \cup \mathcal{V})$, then $\omega(\varphi_2) = x$. Thus $\omega(\varphi_1) \equiv \omega(\varphi_2)$ holds as $x \equiv x$ holds.
- If $\omega(\varphi_1) = \varphi_1$, then $\omega(\varphi_2) = \varphi_2$. Hence $\omega(\varphi_1) \equiv \omega(\varphi_2)$ holds as $\varphi_1 \equiv \varphi_2$ holds.

Induction Hypothesis: Let φ_1, φ_2, $\omega(\varphi_1)$ and $\omega(\varphi_2)$ be quantified boolean formulas such that $\varphi_1 \equiv \varphi_2$ holds. Then $\omega(\varphi_1) \equiv \omega(\varphi_2)$ holds.

Induction Step:

- If $\omega(\varphi_1) = \neg\psi(\varphi_1)$, then $\omega(\varphi_2) = \neg\psi(\varphi_2)$. By the induction hypothesis $\psi(\varphi_1) \equiv \psi(\varphi_2)$ holds. Thus $\neg\psi(\varphi_1) \equiv \neg\psi(\varphi_2)$ and $\omega(\varphi_1) \equiv \omega(\varphi_2)$ hold too.
- If $\omega(\varphi_1) = \psi_1(\varphi_1) \vee \psi_2(\varphi_1)$, then $\omega(\varphi_2) = \psi_1(\varphi_2) \vee \psi_2(\varphi_2)$. We apply the induction hypothesis twice and get $\psi_1(\varphi_1) \equiv \psi_1(\varphi_2)$ and $\psi_2(\varphi_1) \equiv \psi_2(\varphi_2)$. It follows that $(\psi_1(\varphi_1) \vee \psi_2(\varphi_1)) \equiv (\psi_1(\varphi_2) \vee \psi_2(\varphi_2))$ holds and thus that $\omega(\varphi_1) \equiv \omega(\varphi_2)$ is true.

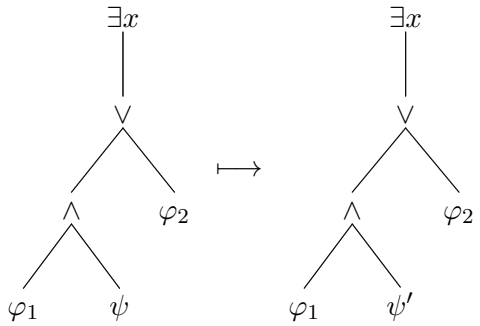
[1] It is assumed that $(\omega(\top) \vee \varphi_1)$ and $(\omega(\top) \vee \varphi_2)$ are both unitary formulas.

- If $\omega(\varphi_1) = \psi_1(\varphi_1) \wedge \psi_2(\varphi_1)$, then $\omega(\varphi_2) = \psi_1(\varphi_2) \wedge \psi_2(\varphi_2)$. We apply the induction hypothesis twice and get $\psi_1(\varphi_1) \equiv \psi_1(\varphi_2)$ and $\psi_2(\varphi_1) \equiv \psi_2(\varphi_2)$. It follows that $(\psi_1(\varphi_1) \wedge \psi_2(\varphi_1)) \equiv (\psi_1(\varphi_2) \wedge \psi_2(\varphi_2))$ holds and thus that $\omega(\varphi_1) \equiv \omega(\varphi_2)$ is true.
- If $\omega(\varphi_1) = \exists x\, \psi(\varphi_1)$ then $\omega(\varphi_2) = \exists x\, \psi(\varphi_2)$. We know that $\varphi_1[x\backslash\xi] = \varphi_1$ and that $\varphi_2[x\backslash\xi] = \varphi_2$ for any ξ as x does neither occur in φ_1 nor in φ_2. The induction hypothesis is applied twice and we get $\psi[x\backslash c](\varphi_1) \equiv \psi[x\backslash c](\varphi_2)$ for $c \in \mathcal{C}$. It follows that $(\exists x\, \psi(\varphi_1)) \equiv (\exists x\, \psi(\varphi_2))$ and thus $\omega(\varphi_1) \equiv \omega(\varphi_2)$ holds.
- If $\omega(\varphi_1) = \forall x\, \psi(\varphi_1)$ then $\omega(\varphi_2) = \forall x\, \psi(\varphi_2)$. We know that $\varphi_1[x\backslash\xi] = \varphi_1$ and that $\varphi_2[x\backslash\xi] = \varphi_2$ for any ξ as x does neither occur in φ_1 nor in φ_2. The induction hypothesis is applied twice and we get $\psi[x\backslash c](\varphi_1) \equiv \psi[x\backslash c](\varphi_2)$ for $c \in \mathcal{C}$. It follows that $(\forall x\, \psi(\varphi_1)) \equiv (\forall x\, \psi(\varphi_2))$ and thus $\omega(\varphi_1) \equiv \omega(\varphi_2)$ holds.

□

What happens if we exchange a sub-formula of a quantified boolean formula with another one with a proof of the original formula? Before presenting normal forms, we will have a short look in the impact of exchanging sub-formulas on proofs. The following example shall present an introduction to the matter that follows.

Example Let $\omega(\psi) = \exists x\, (\varphi_1 \wedge \psi) \vee \varphi_2)$ be a valid quantified boolean formula. Moreover let ψ' be a quantified boolean formula such that ψ' is logically equivalent to ψ. Let $\omega(\psi')$ be the formula resulting by replacing ψ by ψ' in $\omega(\psi)$.



Let π be the following GQBF-proof of $\omega(\psi)$ where c is a truth constant:

$$\dfrac{\dfrac{\dfrac{\begin{matrix}\pi_{\varphi_1}\\ \vdots\\ \vdash \varphi_1[x\backslash c]\end{matrix} \quad \begin{matrix}\pi_{\psi}\\ \vdots\\ \vdash \psi[x\backslash c]\end{matrix}}{\vdash \varphi_1[x\backslash c] \wedge \psi[x\backslash c]}\,\wedge}{\vdash (\varphi_1[x\backslash c] \wedge \psi[x\backslash c]) \vee \varphi_2[x\backslash c]}\,\vee_1}{\vdash \exists x\, (\varphi_1 \wedge \psi) \vee \varphi_2)}\,\exists_c$$

How can a proof of $\omega(\psi')$ be constructed when a proof of $\omega(\psi)$ is given? A proof π' of $\omega(\psi')$ can be constructed with a structure identical with the structure of π. However, π and π' will differ structurally at places where π_ψ, a sub-proof of π, is replaced by $\pi_{\psi'}$, a proof of ψ'. These replacements will appear in π at each occurrence of the sequent $\vdash \psi[x\backslash c]$. π' can now constructed as follows:

$$\dfrac{\dfrac{\dfrac{\begin{matrix}\pi_{\varphi_1}\\ \vdots\\ \vdash \varphi_1[x\backslash c]\end{matrix} \quad \begin{matrix}\pi_{\psi'}\\ \vdots\\ \vdash \psi'[x\backslash c]\end{matrix}}{\vdash \varphi_1[x\backslash c] \wedge \psi'[x\backslash c]}\wedge}{\vdash (\varphi_1[x\backslash c] \wedge \psi'[x\backslash c]) \vee \varphi_2[x\backslash c]}\vee_1}{\vdash \exists x\,(\varphi_1 \wedge \psi') \vee \varphi_2)}\exists_c$$

From the equivalent replacement theorem for quantified boolean formulas and the soundness and the completeness of GQBF if follows that for every GQBF proof π of a closed quantified boolean formula $\varphi(\psi)$ in negation normal form there exists a GQBF proof π' of $\varphi(\psi')$ if $\psi \equiv \psi'$ holds, ψ' is closed and in negation normal form.

We will prove in the forthcoming paragraphs that equivalent replacement in formulas will correspond to strictly local changes (i.e., sub-proofs are replaced by others) in their GQBF proofs. Furthermore we will prove that if a formula is replaced by a formula that can be proven at least as short as the replaced one, the size of the proof cannot increase.

Theorem 9.1.2. Let $\varphi_x \in \mathcal{QBF}$ and $\varphi_y \in \mathcal{QBF}$ be two valid closed quantified boolean formulas in negation normal form. Moreover let π_x be the shortest GQBF proof of φ_x and π_y be the shortest GQBF proof of φ_y. Let $\omega(\varphi_x)$ be a closed quantified boolean formula in negation normal formula having φ_x as sub-formula, π_x^ω the shortest GQBF proof of $\omega(\varphi_x)$ and k the number of $\vdash \varphi_x$ sequents in π_x^ω.

Then a GQBF proof of $\omega(\varphi_y)$ exists. Furthermore, the shortest proof GQBF proof π_y^ω of $\omega(\varphi_y)$ — a duplicate of $\omega(\varphi_x)$ having all occurrences of φ_x replaced by φ_y — has a at most a length of $|\pi_x^\omega| + k(|\pi_y| - |\pi_x|)$.

Proof. The existence of π_y^ω follows from the soundness and completeness of GQBF. The size of π_y^ω is proven by induction on the structure of π_x^ω. We will show the base case and some exemplary steps. The remaining cases are very similar.

Induction Base:

- π_x^ω is a proof of a formula that does not contain φ_x at all. Then $k = 0$ and π_y^ω is identical to π_x^ω. As $k = 0$, $|\pi_y^\omega| = |\pi_x^\omega| + 0 = |\pi_x^\omega| + k(|\pi_y| - |\pi_x|)$ holds.

- π_x^ω is a proof of φ_x. Then $\omega(\varphi_x) = \varphi_x$ and $k = 1$. We choose π_y^ω to be identical to π_x^ω. Again, $|\pi_y^\omega| = |\pi_y| = |\pi_x| + (|\pi_y| - |\pi_x|) = |\pi_x^\omega| + k(|\pi_y| - |\pi_x|)$ holds.

Induction Hypothesis:

- Let φ_x, φ_y, $\omega(\varphi_x)$ and $\omega(\varphi_y)$ be valid, closed quantified boolean formulas in negation normal form.
- Let π_x: $\vdash_{\mathrm{G}} \varphi_x$, π_y: $\vdash_{\mathrm{G}} \varphi_y$ and π_x^ω: $\vdash_{\mathrm{G}} \omega(\varphi_x)$ be minimal proofs.
- Let k be the number of $\vdash \varphi_x$ sequents in π_x^ω.

▷ Then a GQBF proof of $\omega(\varphi_y)$ exists with $|\pi_y^\omega| \leq |\pi_x^\omega| + k(|\pi_y| - |\pi_x|)$.

Induction Step (Expemplary):

- The very last rule applied in π_x^ω is a $\vee_i$-rule and $\vdash \varphi_x$ appears k' times in π_x^ω.

$$\dfrac{\begin{array}{c}\tau_x^\omega \\ \vdots \\ \vdash \omega_i(\varphi_x)\end{array}}{\vdash \omega_1(\varphi_x) \vee \omega_2(\varphi_x)} \vee_i \quad \longmapsto \quad \dfrac{\begin{array}{c}\tau_y^\omega \\ \vdots \\ \vdash \omega_i(\varphi_y)\end{array}}{\vdash \omega_1(\varphi_y) \vee \omega_2(\varphi_y)} \vee_i$$

  By the induction hypothesis we get that there exists a proof τ_y^ω of $\omega_i(\varphi_y)$ such that the size of τ_y^ω is at most $|\tau_x^\omega| + k'(|\pi_y| - |\pi_x|)$. The application of the $\vee_i$ does not add additional instances of $\vdash \varphi_x$ and thus $k = k'$ holds. The size of π_x^ω is $1+|\tau_x^\omega|$. It holds that $|\pi_y^\omega| = |\tau_y^\omega|+1 = |\tau_x^\omega|+k'(|\pi_y|-|\pi_x|)+1 = |\pi_x^\omega| + k(|\pi_y| - |\pi_x|)$. Note that even though τ_x^ω may be a minimal proof of $\vdash \omega_1(\varphi_x) \vee \omega_2(\varphi_x)$, there may exist a proof of $\vdash \omega_1(\varphi_y) \vee \omega_2(\varphi_y)$ that is even shorter than τ_y^ω. Nevertheless $|\pi_y^\omega| \leq |\pi_x^\omega| + k(|\pi_y| - |\pi_x|)$ holds.

- The very last rule applied in π_x^ω is a $\forall$-rule and $\vdash \varphi_x$ appears k' times in π_x^ω.

$$\dfrac{\begin{array}{cc}\tau_{x,\top}^{\omega'} & \tau_{x,\bot}^{\omega'} \\ \vdots & \vdots \\ \vdash \omega'[a\backslash\top](\varphi_x) & \vdash \omega'[a\backslash\bot](\varphi_x)\end{array}}{\vdash \forall a\, \omega'(\varphi_x)} \forall$$

  As φ_x is a closed formula, $\varphi_x = \varphi_x[a\backslash c]$ for $c \in \mathcal{C}$ (assumed that $\omega(\varphi_x)$ is unitary). Without loss of generality, assume that $k' = (k^\top + k^\bot)$, that $\vdash \varphi_x$ occurs $k^\top$ times in $\tau_{x,\top}^\omega$ and that $\vdash \varphi_x$ occurs $k^\bot$ times in $\tau_{x,\bot}^\omega$. We apply the induction hypothesis two times:

– $\tau^{\omega'}_{y,\top}$ is a proof of $\omega[a\backslash\top](\varphi_y)$ with $|\tau^{\omega'}_{y,\top}| = |\tau^{\omega'}_{x,\top}| + k^{\top}(|\pi_y| - |\pi_x|)$.

– $\tau^{\omega'}_{y,\bot}$ is a proof of $\omega[a\backslash\bot](\varphi_y)$ with $|\tau^{\omega'}_{y,\bot}| = |\tau^{\omega'}_{x,\bot}| + k^{\bot}(|\pi_y| - |\pi_x|)$.

The $\forall$-rule does not add new instances of $\vdash \varphi_x$ and thus $k = k' = k^{\top} + k^{\bot}$. We now construct π^{ω}_y as follows:

$$\dfrac{\begin{array}{cc} \tau^{\omega'}_{y,\top} & \tau^{\omega'}_{y,\bot} \\ \vdots & \vdots \\ \vdash \omega'[a\backslash\top](\varphi_y) & \vdash \omega'[a\backslash\bot](\varphi_y) \end{array}}{\vdash \forall a\, \omega'(\varphi_y)}\ \forall$$

The size of π^{ω}_y is $1 + |\tau^{\omega'}_{y,\top}| + |\tau^{\omega'}_{y,\bot}| = 1 + |\tau^{\omega'}_{x,\top}| + k^{\top}(|\pi_y| - |\pi_x|) + |\tau^{\omega'}_{x,\bot}| + k^{\bot}(|\pi_y| - |\pi_x|) = |\pi^{\omega}_x| + k(|\pi_y| - |\pi_x|)$.

□

Note that we demanded that φ_x and φ_y are valid. It is easy to extend the above theorem and ease the $\varphi_x \equiv \varphi_y \equiv \mathbf{1}$ condition to φ_x and φ_y being logically equivalent only. This results in two cases. In the first one where both formulas are valid the above theorem can be used to prove that a proof of $\omega(\varphi_y)$ of the size specified in Theorem 9.1.2 indeed exists. In the second case where both formulas are invalid, the sequent $\vdash \varphi_x$ cannot appear in π^{ω}_x anyway and thus $k = 0$. Consequently, $|\pi^{\omega}_y| = |\pi^{\omega}_x|$ holds. We can also extend the theorem to allow free variables in formulas:

Theorem 9.1.3. Let

- Let $n \in \mathbb{N}$ be an integer.
- Let $\varphi_x \in \mathcal{QBF}$ and $\varphi_y \in \mathcal{QBF}$ be two logically equivalent quantified boolean formulas in negation normal form with $\mathit{free}(\varphi_x) = \mathit{free}(\varphi_y) = \mathcal{F}$ and $|\mathcal{F}| = n$ such that for every function $e\colon \mathcal{F} \to \mathbb{B}$ holds that $\nu_e(\varphi_x) = \nu_e(\varphi_y)$.
- Let $\omega(\varphi_x)$ be a closed quantified boolean formula in negation normal formula having φ_x as sub-formula.
- Let π^{ω}_x the shortest GQBF proof of $\omega(\varphi_x)$.
- Let $s_x\colon \mathcal{C}^n \to \mathbb{N}$ be a (partial) function that maps an instantiation vector $(c_1, c_2, \ldots, c_n)$ to the size of the shortest GQBF proof of the formula $\varphi_x[f_1\backslash c_1][f_2\backslash c_2][\ldots][f_n\backslash c_n]$, whereas $\{f_i \mid 1 \leq i \leq n\} = \mathcal{F}$. The value of s_x is only defined for valid instances.

- Let $s_y \colon \mathcal{C}^n \to \mathbb{N}$ be a (partial) function that maps an instantiation vector $(c_1, c_2, \ldots, c_n)$ to the size of the shortest GQBF proof of the formula $\varphi_y[f_1 \backslash c_1][f_2 \backslash c_2][\ldots][f_n \backslash c_n]$, whereas $\{f_i \mid 1 \leq i \leq n\} = \mathcal{F}$. The value of s_y is only defined for valid instances.

- $k \colon \mathcal{C}^n \to \mathbb{N}$ be a (total) function that maps an n-dimensional instantiation vector $(c_1, c_2, \ldots, c_n)$ to the number of occurrences of the sequent $\vdash \varphi_x[f_1 \backslash c_1][f_2 \backslash c_2][\ldots][f_n \backslash c_n]$ in π_x^ω. Again, $\{f_i \mid 1 \leq i \leq n\}$ be $\mathcal{F}$.

- $\omega(\varphi_y)$ be a copy of $\omega(\varphi_x)$ having all occurrences of φ_x replaced by φ_y.

Then a GQBF proof of $\omega(\varphi_y)$ exists. Furthermore, the shortest proof GQBF proof π_y^ω of $\omega(\varphi_y)$ has a length equal or less to $|\pi_x^\omega| + \sum_{\mathbf{c} \in \mathcal{C}^n} k(\mathbf{c})\big(s_x(\mathbf{c}) - s_y(\mathbf{c})\big)$.

Proof. Analog to the proof of Theorem 9.1.2. □

Note that if we use Theorem 9.1.2 (Theorem 9.1.3) to replace sub-formulas by sub-formulas that (thats instances) can be proven in an equivalently short manner, then it holds that $|\pi_y^\omega| = |\pi_x^\omega|$. We can prove this by applying Theorem 9.1.2 (Theorem 9.1.3) two times. The first time we use it to proven that $|\pi_y^\omega| \leq |\pi_x^\omega|$. Then we swap the two formulas and prove $|\pi_x^\omega| \leq |\pi_y^\omega|$.

9.2. Normal Forms

When we add quantifiers to the language of propositional logic, we get — depending on the type of quantifiers added — either quantified boolean logic, first order logic or even higher (ramified) types when adding several different types of quantifiers. For first order logic it is well known that for each formula of first order logic there exists a logically equivalent formula in prenex normal form (see, e.g., Theorem 5.2 in [14]). Prenex normal forms are used to establish the arithmetical as well as the analytical hierarchy. Also some proof calculi for first order logics depend on an input in prenex normal form. Similarly, some proof calculi for quantified boolean logics require their input to be in prenex normal form. One of them, Q-resolution will be presented in subsequent chapters.

Prenex Normal Form A quantified boolean formula φ is in prenex normal form if it is, for some n, isomorphic to a formula $F_{n,p} \in \mathcal{QBF}$ from the following family:

$$F_{n,p} := Q_1 x_1 \, Q_2 x_2 \, \ldots \, Q_n x_n \, \varphi_p$$

where $\varphi_p \in \mathcal{PL}$, $Q_i \in \{\exists, \forall\}$, $1 \leq i \leq n$. We call φ_p the propositional core (or matrix) of φ and the sequence of Q_is the quantifier prefix of φ. Note that for each quantified boolean formula, there exists at least one equivalent formula in prenex normal form. In general there may be even more such normal forms

for one formula. We can use the algorithm *pnf* to transform formulas into equivalent prenex normal form.

Let $\varphi \in \mathcal{QBF}$ be a quantified boolean formula. Then $\mathit{pnf}(\varphi)$ is defined through the rules (1) to (5) below. These rules are actually equivalences which are directed. They are applied to φ and its sub-formulas until no rule can be applied any more.

$$\neg\exists x\, \psi \longmapsto \forall x\, \neg\psi \tag{1}$$
$$\neg\forall x\, \psi \longmapsto \exists x\, \neg\psi \tag{2}$$
$$(Qx\, \psi) \circ \psi^* \longmapsto Qx\, (\psi \circ \psi^*) \tag{3}$$
$$\psi^* \circ (Qx\, \psi) \longmapsto Qx\, (\psi^* \circ \psi) \tag{4}$$
$$Qx\, \psi^* \longmapsto \psi^* \tag{5}$$

In the rules (1) to (5), $x \notin \mathit{free}(\psi^*)$, $Q \in \{\exists, \forall\}$ and $\circ \in \{\wedge, \vee\}$ hold. Note that there is some don't-care-indeterminism introduced by the rules (3) and (4). As an example, the formula $(\forall p\, \varphi) \wedge (\exists q\, \psi)$ has two distinct prenex normal forms, depending on the application order of the rules (3) and (4): The order (3)(4) yields $\forall p\, \exists q\, \varphi \wedge \psi$ whereas the order (4)(3) yields $\exists q\, \forall p\, \varphi \wedge \psi$. Nevertheless, we mostly use *pnf* as if it were an actual function $\mathit{pnf}\colon \mathcal{QBF} \to \mathcal{QBF}$. Each of the rules (1) to (5) is strictly equivalence preserving. In other words, $\longmapsto$ can be replaced by $\equiv$ in the definitions of the rules (1) to (5). Thus applications of the rules (1) to (5) can be seen as equivalent replacements. A consequence of this is that $\varphi \equiv \mathit{pnf}(\varphi)$ holds for any quantified boolean formula. Moreover, if the replacements are directed (performed from left to right), $\mathit{pnf}(\varphi)$ can be calculated in a time polynomial in the size of φ.

We may also want to add additional rules that allow to "contract" quantifiers:

$$\forall x_1\, \forall x_2\, \big(\psi_1(x_1) \wedge \psi_2(x_2)\big) \longmapsto \forall x\, \big(\psi_1(x) \wedge \psi_2(x)\big) \tag{6}$$
$$\exists x_1\, \exists x_2\, \big(\psi_1(x_1) \vee \psi_2(x_2)\big) \longmapsto \exists x\, \big(\psi_1(x) \vee \psi_2(x)\big) \tag{7}$$

Again, $x_1 \notin \mathit{free}(\psi_2)$ and $x_2 \notin \mathit{free}(\psi_1)$. Transformations containing the rules (1) to (5) as well as the rules (6) to (7) will be called pnf^+. The formula $\mathit{pnf}^+(\varphi)$ can still be calculated in a time polynomial in the size of φ.

Keep in mind that, e.g., $\forall x_1\, \forall x_2\, \big(\psi_1(x_1) \vee \psi_2(x_2)\big)$ is generally not equivalent to $\forall x\, \big(\psi_1(x) \vee \psi_2(x)\big)$, which is demonstrated in the following example:

$$\begin{aligned} \bot &\equiv \exists y\, \big(\forall x\, (x \supset y \wedge y \supset x)\big) \vee \big(\forall x\, (\neg x \supset y \wedge y \supset \neg x)\big) \\ &\not\equiv \\ \top &\equiv \exists y \forall x\, (x \supset y \wedge y \supset x) \vee (\neg x \supset y \wedge y \supset \neg x) \end{aligned}$$

Conjunctive Normal Form A propositional formula $\varphi \in \mathcal{PL}$ is in conjunctive normal form if it is a conjunction of disjunctions of literals. That is, if it is of a form $((\ell_{1,1} \vee \ell_{1,2} \vee \ldots \vee \ell_{1,k_1}) \wedge (\ell_{2,1} \vee \ell_{2,2} \vee \ldots \vee \ell_{2,k_2}) \wedge \ldots \wedge (\ell_{n,1} \vee \ell_{n,2} \vee \ldots \vee \ell_{n,k_n}))$ where $\ell_{i,j}$ denotes a literal.

There are three principle ways to transform a formula into a corresponding conjunctive normal form. Each transformations has advantages as well as disadvantages. We will present them in the following.

1. Transform φ to a formula ψ that is equivalent to φ, i.e., $\varphi \equiv \psi$ holds. This procedure does not change the set of variables of φ. Generally this transformation can be implemented in two steps: First bring φ to a corresponding negation normal form and then iteratively apply distributivity laws to $nnf(\varphi)$:

$$(\varphi_1 \wedge \varphi_2) \vee \varphi_3 \longmapsto (\varphi_1 \vee \varphi_3) \wedge (\varphi_2 \vee \varphi_3) \tag{9.1}$$
$$\varphi_1 \vee (\varphi_2 \wedge \varphi_3) \longmapsto (\varphi_1 \vee \varphi_2) \wedge (\varphi_1 \vee \varphi_3) \tag{9.2}$$

   Note that the application of these rules duplicates the formula φ_3 (respectively φ_1). Thus the resulting formula $cnf(\varphi)$ may be exponentially larger in size than the source formula φ. It follows that cnf, though being easily mechanizable, is not guaranteed to terminate in a time polynomial in the size of the input formula.

2. There is a way to conquer the exponential blow up of the first transformation. The idea is to introduce globally new variables that "encode" the structure of the input formula. With introducing new variables logical equivalence cannot preserved any more. φ is thus transformed to a formula ψ such that $\varphi \overset{\text{sat}}{\equiv} \psi$ holds. Again this can be done in two steps. First bring φ to negation normal form, then iteratively apply the following rewrite rules to $nnf(\varphi)$:

$$(\varphi_1 \wedge \varphi_2) \vee \varphi_3 \longmapsto (\varphi_1 \vee x) \wedge (\varphi_2 \vee x) \wedge (\varphi_3 \vee \neg x) \tag{9.3}$$
$$\varphi_1 \vee (\varphi_2 \wedge \varphi_3) \longmapsto (\varphi_1 \vee \neg x) \wedge (\varphi_2 \vee x) \wedge (\varphi_3 \vee x) \tag{9.4}$$

   Clearly, the resulting formula is satisfiable if and only if the input formula is. A satisfying assignment to the input can easily be extended to a satisfying assignment to the resulting formula in conjunctive normal form. While the first transformation can be applied to quantified boolean formulas without any problem, the second one has to be treated carefully. It introduces new variables that are free by definition. They are globally new and no new quantifier is introduced. Henceforth the second procedure cannot be applied without changes to closed quantified boolean formulas if there is the condition that the result has to be closed too.

3. A third method is to introduce a label for each sub-formula of φ and encode the structure of φ using the new labels [21, 17]. This technique has two advantages: First, the structure of the original can be restored fully from the informations stored in the labels and their relations in the conjunctive normal form formula. Further, the resulting formula is at polynomial

(linear) in the size of the input formula. As new labels are introduced, $cnf(\varphi)$ is not logically equivalent to φ. Still, it can be shown that indeed each model of $cnf(\varphi)$ does also satisfy φ. Note that this approach can be applied to any propositional formula whereas approach number 2 depends on its input formula to already be in negation normal form.

The transformation can formally be defined the following way:

- Let $\mathcal{S}$ be the set of all structural sub-formulas of φ.
- Let $\mathcal{L} = \{d_\xi \mid \xi \in \mathcal{S}\}$ be a set of variable symbols that do not occur in φ ($\mathcal{L} \subseteq \mathcal{V}$, $\mathcal{L} \cap free(\varphi) = \emptyset$). d_ξ is used as a label for the sub-formula ξ of φ.
- Then $cnf(\varphi) := \bigwedge_{\psi \in \mathcal{S}} (\oplus(\psi) \wedge \ominus(\psi))$.
- $\oplus(\psi)$ and $\ominus(\psi)$ are defined as follows:
  - If $\psi = p$ with $p \in (\mathcal{C} \cup \mathcal{V})$ then $\oplus(\psi) = (d_\psi \vee \neg p)$ and $\ominus(\psi) = (\neg d_\psi \vee p)$.
  - If $\psi = \neg\psi'$ then $\oplus(\psi) = (d_\psi \vee d_{\psi'})$ and $\ominus(\psi) = (\neg d_\psi \vee \neg d_{\psi'})$.
  - If $\psi = (\psi_1 \wedge \psi_2)$ then $\oplus(\psi) = (d_\psi \vee \neg d_{\psi_1} \vee \neg d_{\psi_2})$ and $\ominus(\psi) = ((\neg d_\psi \vee d_{\psi_1}) \wedge (\neg d_\psi \vee d_{\psi_2}))$.
  - If $\psi = (\psi_1 \vee \psi_2)$ then $\oplus(\psi) = ((d_\psi \vee \neg d_{\psi_1}) \wedge (d_\psi \vee \neg d_{\psi_2}))$ and $\ominus(\psi) = (\neg d_\psi \vee d_{\psi_1} \vee d_{\psi_2})$.

Special care has to be taken when this transformation is applied to quantified boolean formulas. As in transformation 2, new labels are introduced. These labels correspond to free variables. Thus additional quantifiers have to be added.

There are various improvements to this transformation:

- Do not use fresh labels for atomic formulas. Use atomic formulas as labels for themselves instead.
- Do not use labels for negated sub-formulas.
- Use the same label for identical sub-formulas.

Clause Notation We also write formulas in conjunctive normal form as clause sets:

$$\{\{\ell_{1,1}, \ell_{1,2}, \ldots, \ell_{1,k_1}\}, \{\ell_{2,1}, \ell_{2,2}, \ldots, \ell_{2,k_2}\}, \ldots, \{\ell_{n,1}, \ell_{n,2}, \ldots, \ell_{n,k_n}\}\}$$

is the clause form of

$$((\ell_{1,1} \vee \ell_{1,2} \vee \ldots \vee \ell_{1,k_1}) \wedge (\ell_{2,1} \vee \ell_{2,2} \vee \ldots \vee \ell_{2,k_2}) \wedge \ldots \wedge (\ell_{n,1} \vee \ell_{n,2} \vee \ldots \vee \ell_{n,k_n}))$$

Prenex Conjunctive Normal Form A quantified boolean formula $\varphi \in \mathcal{QBF}$ is in prenex conjunctive normal form if it is in prenex normal form and its propositional core ($core(\varphi)$) is in conjunctive normal form. It can be calculated in two steps: Bring the source formula into prenex normal form and then apply a transformation to conjunctive normal to its core that is suitable. Keep in mind that indeterminism is introduced through prenexing and size (computation time) hazards are hidden in the transformation of the core. The impacts of these transformations on proofs are discussed in the following.

9.3. GQBF-Proofs and Prenexing

Subsequently we will have a look into the complexity of prenexing, with a special focus on proof transformations. In other words, we will examine the impact of prenexing operations on the structure of GQBF proofs.

For the next part, assume we have a GQBF proof π of a closed quantified boolean formula $\varphi \in \mathcal{QBF}$ that is not in prenex normal form but in negation normal form. As quantifiers and connectives are eliminated from the outermost to the innermost one, each trace through the proof tree of π resembles a trace through the structure tree of φ.

Now imagine that we perform a prenexing operation on φ, e.g., moving a quantifier up one level towards the root) in the structure tree of φ. Clearly, π does not prove the adapted formula φ'. But there is an easy way to adapt π such that it is a proof of the new φ': The rule that treated the quantifier shifted upwards in φ' is also shifted upwards (towards the root) in π. Transformations on proofs that resemble prenexing operations on formulas will be discussed in the following.

To be more precise, we use the following mechanical procedure to transform a proof of φ into a proof of $pnf(\varphi)$:

1. If φ is in prenex normal form: Stop.
2. On the formula level: Move the outermost quantifier up one level. Use one of the seven prenexing rules for it. If there is more than one quantifier on the outermost non-prenex level, choose one according to the quantifier prefix that should be reached eventually. A thorough discussion of the impact of different prenexing strategies can be found in [3].
3. On the proof level: Locally adapt the proof to prove the new formula. In general this means to bubble a quantifier rule up over another rule.
4. Continue with 1.

Prenexing was defined as an inductive operation on the input formula and its sub-formulas. We make use of the very same induction on the structure of φ, to fully describe the impact of prenexing to a proof.

1. $\neg\exists x\,\psi \longmapsto \forall x\,\neg\psi$ and $\neg\forall x\,\psi \longmapsto \exists x\,\neg\psi$.
   As π is a proof of φ, φ has to be in negation normal form. These cases cannot emerge.

2. $(\exists x\,\psi) \circ \psi^* \longmapsto \exists x\,(\psi \circ \psi^*)$ where x does not occur in ψ^*.
   Here we distinguish two cases: The first one is that $x \notin \mathit{free}(\psi)$. This case just reduces to case 5. The second case where $x \in \mathit{free}(\psi)$ is the more interesting one and will be discussed in the following.

   Assume that $\circ$ is $\wedge$. Then π, the proof of φ, starts with $\vdash \varphi$ as end-sequent. The root of the proof may, depending on the structure of φ, split into various branches. Some of them do not contain $\vdash (\exists x\,\psi) \circ \psi^*$ while others do. We use the Equivalent Replacement Theorem to transform π to a proof τ of φ'. By using Theorem 9.1.3 (for replacing $(\exists x\,\psi) \circ \psi^*$ by $\exists x\,(\psi \circ \psi^*)$), we can calculate a maximal size for the new proof. The most interesting changes occur as soon as we reach $\vdash (\exists x\,\psi) \circ \psi^*$ in a branch:

$$\dfrac{\dfrac{\dfrac{\pi'}{\vdots}\;\vdash \psi[x\backslash c]}{\vdash \exists x\,\psi}\,\exists_c \qquad \dfrac{\pi''}{\vdots}\;\vdash \psi^*}{\vdash (\exists x\,\psi) \wedge \psi^*}\,\wedge}{\vdots}\;\vdash \varphi \qquad \longmapsto \qquad \dfrac{\dfrac{\dfrac{\pi'}{\vdots}\;\vdash \psi[x\backslash c] \qquad \dfrac{\pi''}{\vdots}\;\vdash \psi^*}{\vdash \psi[x\backslash c] \wedge \psi^*}\,\wedge}{\vdash \exists x\,(\psi \wedge \psi^*)}\,\exists_c}{\vdots}\;\vdash \varphi'$$

   The size of the branch stays constant. In other words: We replace sub-proofs by new sub-proofs of an equal size. Let s_x denote the size of the sub-proof to be replaced, s_y the size of the replacement and k the number of occurrences of the sub-proof to be replaced. Then the maximal size of τ is $|\pi| + k(s_x - s_y)$. As s_x is equal to s_y, $|\tau|$ is at most $|\pi|$. As stated in the paragraph below the proof of Theorem 9.1.3, it even holds that $|\pi| = |\tau|$.

   Similarly, we construct proof for the cases where $\circ$ is $\vee_i$. If i is one we get the following schema:

$$\dfrac{\dfrac{\dfrac{\pi'}{\vdots}\;\vdash \psi[x\backslash c]}{\vdash \exists x\,\psi}\,\exists_c}{\vdash (\exists x\,\psi) \vee \psi^*}\,\vee_1}{\vdots}\;\vdash \varphi \qquad \longmapsto \qquad \dfrac{\dfrac{\dfrac{\pi'}{\vdots}\;\vdash \psi[x\backslash c]}{\vdash \psi[x\backslash c] \vee \psi^*}\,\vee_1}{\vdash \exists x\,(\psi \vee \psi^*)}\,\exists_c}{\vdots}\;\vdash \varphi'$$

   If i is two, we get the following:

$$\dfrac{\begin{array}{c}\pi'\\ \vdots\\ \vdash \psi^*\end{array}}{\vdash (\exists x\, \psi) \vee \psi^*}\ \vee_2 \quad \longmapsto \quad \dfrac{\dfrac{\begin{array}{c}\pi'\\ \vdots\\ \vdash \psi^*\end{array}}{\vdash \psi[x\backslash c] \vee \psi^*}\ \vee_2}{\vdash \exists x\, (\psi \vee \psi^*)}\ \exists_c$$

Again, let k denote the number of occurrences of the sub-proof to be replaced in π. In the first case ($\vee_1$), we replace sub-proofs by sub-proofs of equivalent sizes. Thus, by Theorem 9.1.3, τ is exactly as big as π is. In the second case ($\vee_2$), sub-proofs are replaced by bigger sub-proofs. We get a maximal size of τ of $|\pi| + k$.

3. $(\forall x\, \psi) \circ \psi^* \longmapsto \forall x\, (\psi \circ \psi^*)$
   The case where $x \notin \mathit{free}(\psi)$ reduces to case 5. The case where x appears free in ψ again splits into three distinct cases. Note that if ψ^* contains a universal quantifier, case 6 can be used in subsequent prenexing steps to undo previous increases in proof size.

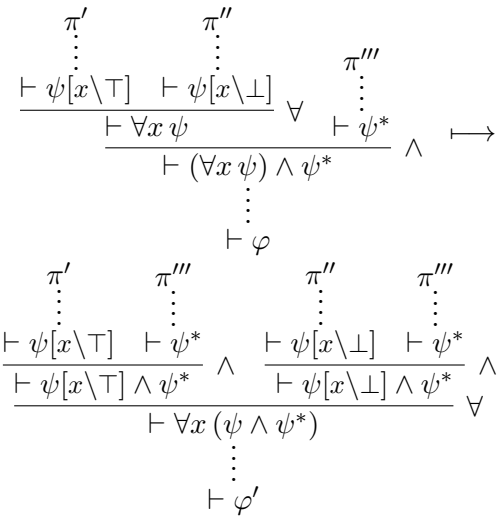
$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\pi'\\ \vdots\\ \vdash \psi[x\backslash\top]\end{array} \quad \begin{array}{c}\pi''\\ \vdots\\ \vdash \psi[x\backslash\bot]\end{array}}{\vdash \forall x\, \psi}\ \forall}{\vdash (\forall x\, \psi) \vee \psi^*}\ \vee_1}{\vdash \varphi} \quad \longmapsto \quad \dfrac{\dfrac{\begin{array}{c}\pi'\\ \vdots\\ \vdash \psi[x\backslash\top]\end{array}}{\vdash \psi[x\backslash\top] \vee \psi^*}\ \vee_1 \quad \dfrac{\begin{array}{c}\pi''\\ \vdots\\ \vdash \psi[x\backslash\top]\end{array}}{\vdash \psi[x\backslash\bot] \vee \psi^*}\ \vee_1}{\vdash \forall x\, (\psi \vee \psi^*)}\ \forall}{\vdash \varphi'}$$

If the connective ($\circ$) is $\vee$ and $\vee_1$ was used to produce the minimal proof π, sub-proofs are replaced by sub-proofs of a bigger size. The $\vee_1$ rule has to be applied twice in τ.

$$\dfrac{\dfrac{\begin{array}{c}\pi'\\ \vdots\\ \psi^*\end{array}}{\vdash (\forall x\, \psi) \vee \psi^*}\ \vee_2}{\vdash \varphi} \quad \longmapsto \quad \dfrac{\dfrac{\begin{array}{c}\pi'\\ \vdots\\ \vdash \psi^*\end{array}}{\vdash \psi[x\backslash\top] \vee \psi^*}\ \vee_2 \quad \dfrac{\begin{array}{c}\pi'\\ \vdots\\ \vdash \psi^*\end{array}}{\vdash \psi[x\backslash\bot] \vee \psi^*}\ \vee_2}{\vdash \forall x\, (\psi \vee \psi^*)}\ \forall}{\vdash \varphi'}$$

If the connective ($\circ$) is $\vee$ and $\vee_2$ was used to produce the minimal proof π, sub-proofs are replaced by sub-proofs of a bigger size. The $\vee_2$ rule has to be applied two times in τ. Additionally, the sub-proof π' appears two times in

each replacement, whereas only one instance is found in each replaced sub-proof. This can dramatically increase the size of the proof. The maximal size increase thus depends on the size of π': $|\tau| \leq |\pi| + k(1 + |\pi'|)$ whereas k is the number of occurrences of the sequent $\vdash (\forall x\, \psi) \vee \psi^*$ in π. Note that proofs in directed acyclic graph form can help to reduce the increase in proof size. Subsequent chapters will discuss this in more detail. Also note that the increased complexity of the proof may not be relevant in proof search. Implementations like [6] may avoid to fully traverse both branches during proof search with the use of dependency directed backtracking.

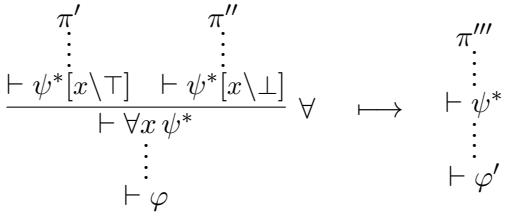


As with the previous case, a sub-proof has to be copied. The size of τ is a most $|\pi| + k(1 + |\pi'''|)$.

4. $\psi^* \circ (\exists x\, \psi) \longmapsto \exists x\, (\psi^* \circ \psi)$
   Symmetric to cases 2.

5. $\psi^* \circ (\forall x\, \psi) \longmapsto \forall x\, (\psi^* \circ \psi)$
   Symmetric to cases 3.

6. $Qx\, \psi^* \longmapsto \psi^*$ where x does not appear in ψ^*.



Choose π''' as the shorter one of π' and π'' and let $c := \top$ if $\pi''' = \pi'$ or $c := \bot$ otherwise. Note that $\psi^*[x\backslash\xi] = \psi^*$ for arbitrary $\xi \in \mathcal{QBF}$. By Theorem 9.1.3 we know that the size of τ is at most $|\pi| + k(|\pi'''| - (1 + |\pi'| + |\pi''|)) \leq |\pi|$. Again, k be the number of occurrences of $\vdash \forall x\, \psi^*$ in π.

7. $\forall x_1\, \forall x_2\, \big(\psi_1(x_1) \wedge \psi_2(x_2)\big) \longmapsto \forall x\, \big(\psi_1(x) \wedge \psi_2(x)\big)$ whereas x_2 does not appear in ψ_1 and x_1 does not appear in ψ_2.
The resulting proof is strictly shorter than the source proof.

$$\dfrac{\dfrac{\dfrac{\begin{matrix}\pi_1'\\ \vdots\\ \vdash \psi_1(\top) \wedge \psi_2(\top)\end{matrix} \quad \begin{matrix}\pi_2'\\ \vdots\\ \vdash \psi_1(\top) \wedge \psi_2(\bot)\end{matrix}}{\vdash \forall x_2\, \psi_1(\top) \wedge \psi_2(x_2)}\,\forall \quad \dfrac{\begin{matrix}\pi_1''\\ \vdots\\ \vdash \psi_1(\bot) \wedge \psi_2(\top)\end{matrix} \quad \begin{matrix}\pi_2''\\ \vdots\\ \vdash \psi_1(\bot) \wedge \psi_2(\bot)\end{matrix}}{\vdash \forall x_2\, \psi_1(\bot) \wedge \psi_2(x_2)}\,\forall}{\vdash \forall x_1\, \forall x_2\, \psi_1(x_1) \wedge \psi_2(x_2)}\,\forall}{\begin{matrix}\vdots\\ \vdash \varphi\end{matrix}}$$

is transformed into:

$$\dfrac{\dfrac{\begin{matrix}\pi_1'\\ \vdots\\ \vdash \psi_1(\top) \wedge \psi_2(\top)\end{matrix} \quad \begin{matrix}\pi_2''\\ \vdots\\ \vdash \psi_1(\bot) \wedge \psi_2(\bot)\end{matrix}}{\vdash \forall x\, \psi_1(x) \wedge \psi_2(x)}\,\forall}{\begin{matrix}\vdots\\ \vdash \varphi'\end{matrix}}$$

Thus, $|\tau|$ is at most $|\pi| + k((\pi_1' + \pi_2'') - (1 + \pi_1' + \pi_2' + \pi_1'' + \pi_2'')) \leq |\pi|$.

8. $\exists x_1\, \exists x_2\, \big(\psi_1(x_1) \vee \psi_2(x_2)\big) \longmapsto \exists x\, \big(\psi_1(x) \vee \psi_2(x)\big)$ whereas x_2 does not appear in ψ_1 and x_1 does not appear in ψ_2.
The resulting proof is strictly shorter than the source proof.

$$\dfrac{\dfrac{\dfrac{\begin{matrix}\pi_i'\\ \vdots\\ \vdash \psi_i(c_i)\end{matrix}}{\vdash \psi_1(c_1) \vee \psi_2(c_2)}\,\vee_i}{\vdash \exists x_2\, \psi_1(c_1) \vee \psi_2(x_2)}\,\exists_{c_2}}{\vdash \exists x_1\, \exists x_2\, \psi_1(x_1) \vee \psi_2(x_2)}\,\exists_{c_1} \quad \longmapsto \quad \dfrac{\dfrac{\begin{matrix}\pi_i'\\ \vdots\\ \vdash \psi_i(c_i)\end{matrix}}{\vdash \psi_1(c_i) \vee \psi_2(c_i)}\,\vee_i}{\vdash \exists x\, \psi_1(x) \vee \psi_2(x)}\,\exists_{c_i}$$

$$\begin{matrix}\vdots\\ \vdash \varphi\end{matrix} \qquad\qquad \begin{matrix}\vdots\\ \vdash \varphi'\end{matrix}$$

Hence $|\tau|$ is at most $|\pi| - k \leq |\pi|$.

On basis of the findings about prenexing we can formulate and prove the following theorem:

Theorem 9.3.1. [Prenexing can exponentially enlarge GQBF proofs] Let $\varphi_n \in \mathcal{QBF}$ be a closed quantifed boolean formula in negation normal form defined recursively as follows:

- $\psi(x, y) = (x \vee \neg y) \wedge (\neg x \vee y)$
- $\varphi_1 = \forall x_1 \, \exists y_1 \, \psi(x_1, y_1)$
- $\varphi_n = \forall x_n \, \exists y_n \, (\psi(x_n, y_n) \vee \varphi_{n-1})$

Then there is a GQBF proof π: $\vdash_{\mathrm{G}} \varphi_n$ with $|\pi| = 11$ whereas there is no GQBF proof τ such that τ is a proof of a prenexed form of φ_n with a size polynomial in n.

Proof. See Figure 9.1 for the structure tree of φ_n.

Surely, there exists a short GQBF proof of φ_n with $|\varphi_n| = 11$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdash \top_{x_n}}{\vdash \top_{x_n} \vee \neg \top_{y_n}} \vee_1 \quad \dfrac{\vdash \top_{y_n}}{\vdash \neg \top_{x_n} \vee \top_{y_n}} \vee_2}{\vdash (\top_{x_n} \vee \neg \top_{y_n}) \wedge (\neg \top_{x_n} \vee \top_{y_n})} \wedge}{\vdash \psi(\top_{x_n}, \top_{y_n}) \vee \varphi_{n-1}} \vee_1}{\vdash \exists y_n \, (\psi(\top_{x_n}, y_n) \vee \varphi_{n-1})} \exists_\top \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdash \neg \bot_{y_n}}{\vdash \bot_{x_n} \vee \neg \bot_{y_n}} \vee_2 \quad \dfrac{\vdash \neg \bot_{x_n}}{\vdash \neg \bot_{x_n} \vee \bot_{y_n}} \vee_1}{\vdash (\bot_{x_n} \vee \neg \bot_{y_n}) \wedge (\neg \bot_{x_n} \vee \bot_{y_n})} \wedge}{\vdash \psi(\bot_{x_n}, \bot_{y_n}) \vee \varphi_{n-1}} \vee_1}{\vdash \exists y_n \, (\psi(\bot_{x_n}, y_n) \vee \varphi_{n-1})} \exists_\bot}{\vdash \forall x_n \, \exists y_n \, (\psi(x_n, y_n) \vee \forall x_{n-1} \, \exists y_{n-1} \, (\psi(x_{n-1}, y_{n-1}) \vee \varphi_{n-2}))} \forall$$

The prenex normal form of φ_n is unique: It is the quantified bollean formula $\varphi_n^{pnf} = \forall x_n \, \exists y_n \, \ldots \, \forall x_1 \, \exists y_1 \, (\psi(x_n, y_n) \vee \ldots \vee \psi(x_1, y_1))$. A proof of φ_n^{pnf} starts in its root with an application of the $\forall$-rule, splitting the proof in two subproofs. Each of them starts with an $\exists$-rule assigning the value from the previous $\forall$-rule to its variable. The next $\forall$-rule starts the very process again. After elimination all quantifiers, the proof has 2^n open branches, each of them already containing $2n$ rule applications. Each of these branches can be closed using 3 steps, similar to the branches of the proof of φ_n. Thus, the size of a proof for φ_n^{pnf} is far beyond 2^n. □

Formulas like φ_n^{pnf} are used in game theory to describe certain characteristics of games. A formula with a similar structure is used in [1] to describe mutual exclusion patterns for the Evader/Persuer game.

We can provide a slightly weaker theorem for GQBF$^+$:

Theorem 9.3.2. [Prenexing can exponentially enlarge GQBF$^+$ proofs] Let φ_n be a closed quantified boolean formula in negation normal form defined inductively as follows:

- $\psi(x, y) = (x \vee \neg y) \wedge (\neg x \vee y)$
- $\varphi_1 = \forall x_1 \exists y_1 \, \psi(x_1, y_1)$
- $\varphi_n = (\forall x_n \exists y_n \, (\psi(x_n, y_n)) \wedge (\varphi_{n-1})$

Then there is a GQBF$^+$ proof π: $\vdash_{\mathrm{G}} \varphi_n$ of a size polynomial in n whereas there exists na prenexed from of φ_n which cannot be proven polynomially using GQBF$^+$. GQBF$^+$ proof.

Proof. A polynomially sized GQBF$^+$ proof π_n of φ_n can be constructed as follows:

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\tau_\top \\ \vdots \\ \vdash \psi(\top_{x_n}, \top_{y_n})\end{array}}{\vdash \exists y_n \, \psi(\top_{x_n}, y_n)} \exists_\top \quad \dfrac{\begin{array}{c}\tau_\bot \\ \vdots \\ \vdash \psi(\bot_{x_n}, \bot_{y_n})\end{array}}{\vdash \exists y_n \, \psi(\bot_{x_n}, y_n)} \exists_\bot}{\vdash \forall x_n \exists y_n \, \psi(x_n, y_n)} \forall \quad \begin{array}{c}\pi_{n-1} \\ \vdots \\ \vdash \varphi_{n-1}\end{array}}{\vdash \varphi_n} \wedge$$

The proof π_n first (bottom-up) applies the $\wedge$ rule to split φ_n into two parts. The first part with the end-sequent $\vdash \forall x_n \exists y_n \, \psi(x_n, y_n)$ can be proven with a short polynomimal proof. As there exist proofs $\tau_\top$ and $\tau_\bot$ with $|\tau_\top| = 5$ and $|\tau_\bot| = 7$, $\forall x_n \exists y_n \, \psi(x_n, y_n)$ can be proven with a short GQBF$^+$ proof of size of 15. In the right branch $\vdash \varphi_{n-1}$ remains to be proven. A GQBF$^+$ proof of φ_{n-1} can be constructed the same way as a the above proof of φ_n is. The size of π_n is thus $16n - 1$.

As prenex form of φ_n we choose $\varphi_n^{pnf} = \forall x_n \exists y_n \ldots \forall x_1 \exists y_1 \, (\psi(x_n, y_n) \wedge \ldots \wedge \psi(x_1, y_1))$. Then we know from Lemma 8.1.2 that no polynomial GQBF$^+$ proof of φ_n^{pnf} exists.

□

It is noteworthy that φ_n^{pnf} can be evaluated in a time linear in n using algortihms presented in [2]. From theorem 9.3.2 we conclude that the choice of the prenexing strategy is essentially a proof size (and thus proof search) issue. It is easy to find "better" prenex forms for φ_n that can provide exponentially shorter proofs. A better prenex normal form can be constructed this way: 1) Move all universal quantifiers up to the topmost level using the prenexing rules (4) and (5). This yields $\forall x_n \ldots \forall x_1 \bigwedge_{i=n}^{1} \exists y_i \, \psi(x_i, y_i)$. Now rule (6) is applied $(n-1)$ times to get $\forall x \bigwedge_{i=n}^{1} \exists y_i \, \psi(x, y_i)$ which has a polynomial GQBF$^+$ proof. Experimental results with a reference implementation of a GQBF$^+$ solver from [6] underline this point.

In the analysis presented in the very beginning of this section we concluded that there is one main factor that drives the enlargement of proofs when a prenex form is required: the duplication of sub-proofs. Apart from that, there only appear constant increases in proof size. It is thus tempting to think that GQBF$^+_\diamond$

is only affected polynomially by prenexing as duplication can be eliminated effectively. But there is a hazard that contradict this assumption: Deferring "decisions" might drastically increase proof size. This hazard is used to prove the following theorem:

Theorem 9.3.3. [Prenexing can exponentially enlarge $\mathrm{GQBF}^+_\Diamond$ proofs]
Let $\psi^{short} = \exists a_1\, \exists a_2((a_1 \vee a_2) \wedge (\neg a_1 \vee \neg a_2))$ and

$$\psi_n^{long} = \forall x_1\, \exists y_1\, \forall x_2\, \exists y_2\, \ldots\, \forall x_n\, \exists y_n\, \exists z_1\, \exists z'_1\, \exists z_2\, \exists z'_2\, \ldots\, \exists z_n\, \exists z'_n \bigwedge_{i=1}^{n} \omega_i$$

be closed quantified boolean formula in negation normal form with

$$\omega_i = (\neg x_i \vee y_i) \wedge (\neg y_i \vee x_i) \wedge (x_i \vee z_i \vee z'_i) \wedge (\neg x_i \vee \neg z_i \vee \neg z'_i) \wedge (z_i \vee z'_i) \wedge (\neg z_i \vee \neg z'_i).$$

Moreover let $\varphi_n = \psi^{short} \vee \psi_n^{long}$ be a closed quantified boolean formula in negation normal form. Then there is a $\mathrm{GQBF}^+_\Diamond$ proof π: $\vdash_{\mathrm{G}} \varphi_n$ of a size polynomial in n whereas there exists a prenexed from of φ_n for which there is no $\mathrm{GQBF}^+_\Diamond$ proof polynomial in n.

Proof. See figure 9.2 for the structure tree of φ_n. As ψ_n^{long} is formula (8.4), we know from Lemma 8.2.1 that there is no polynomial $\mathrm{GQBF}^+_\Diamond$-proof of ψ_n^{long}. On the other hand ψ^{short} can be proven polynomially with $\mathrm{GQBF}^+_\Diamond$ (e.g., with a simple $\mathrm{GQBF}^+_\Diamond$ proof of size 7).

Clearly, there is a polynomial $\mathrm{GQBF}^+_\Diamond$ proof of φ_n: Apply (bottom-up) $\vee_1$ to get a single open branch with the sequent $\vdash \varphi^{short}$ which can be proven with a $\mathrm{GQBF}^+_\Diamond$ proof of size 7.

As prenex form φ_n^{pnf} we choose the formula

$$\begin{aligned}\varphi_n^{pnf} = \forall x_1\, \exists y_1\, \forall x_2\, \exists y_2\, \ldots\, \forall x_n\, \exists y_n\, \exists z_1\, \exists z'_1\, \exists z_2\, \exists z'_2\, \ldots\, \exists z_n\, \exists z'_n\, \exists a_1\, \exists a_2 \\ (((a_1 \vee a_2) \wedge (\neg a_1 \vee \neg a_2)) \vee (\omega_i \wedge \omega_i \wedge \ldots \wedge \omega_i)).\end{aligned}$$

Then a proof nearly identical to the proof of Lemma 8.2.1 suffices to prove that there cannot exist any polynomial $\mathrm{GQBF}^+_\Diamond$-proof of φ_n^{pnf}: As the existential quantifiers for a_1 and a_2 cannot be elimintated using a simplification rule (a_1 and a_2 both occur in both polarities and do not occur globally unit), all quantifiers that precede them have to be eliminated before they can be elimintated themselves. When all preceding quantifiers are eliminated, the proof is already exponential in n as Lemma 8.2.1 proves.

□

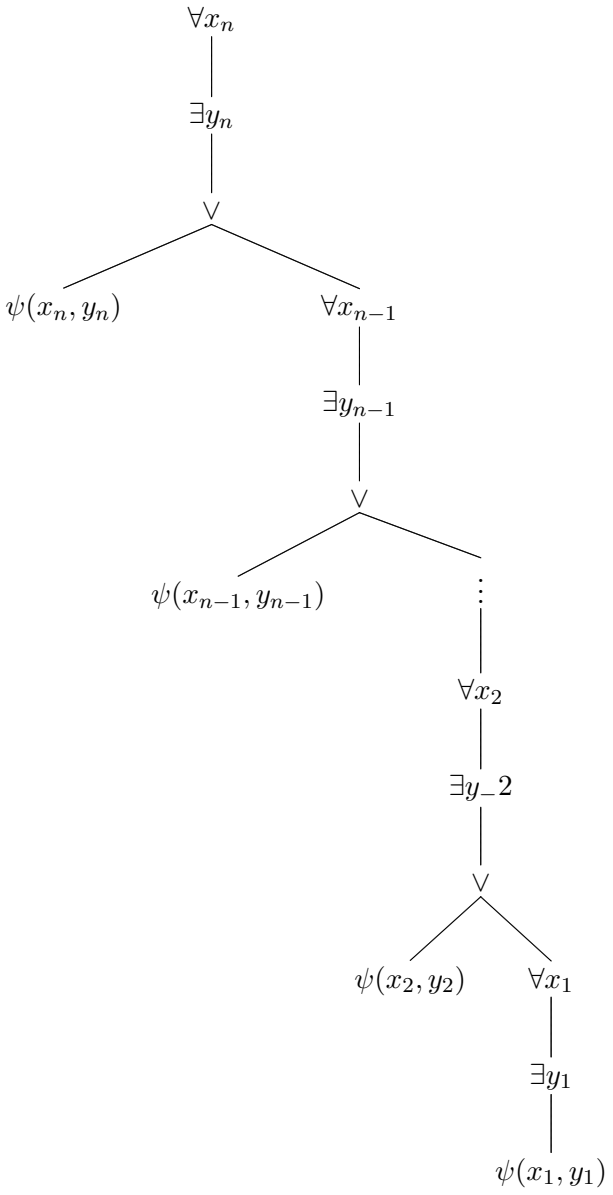


Figure 9.1.: The Structure of φ_n from the proof of Theorem 9.3.1.

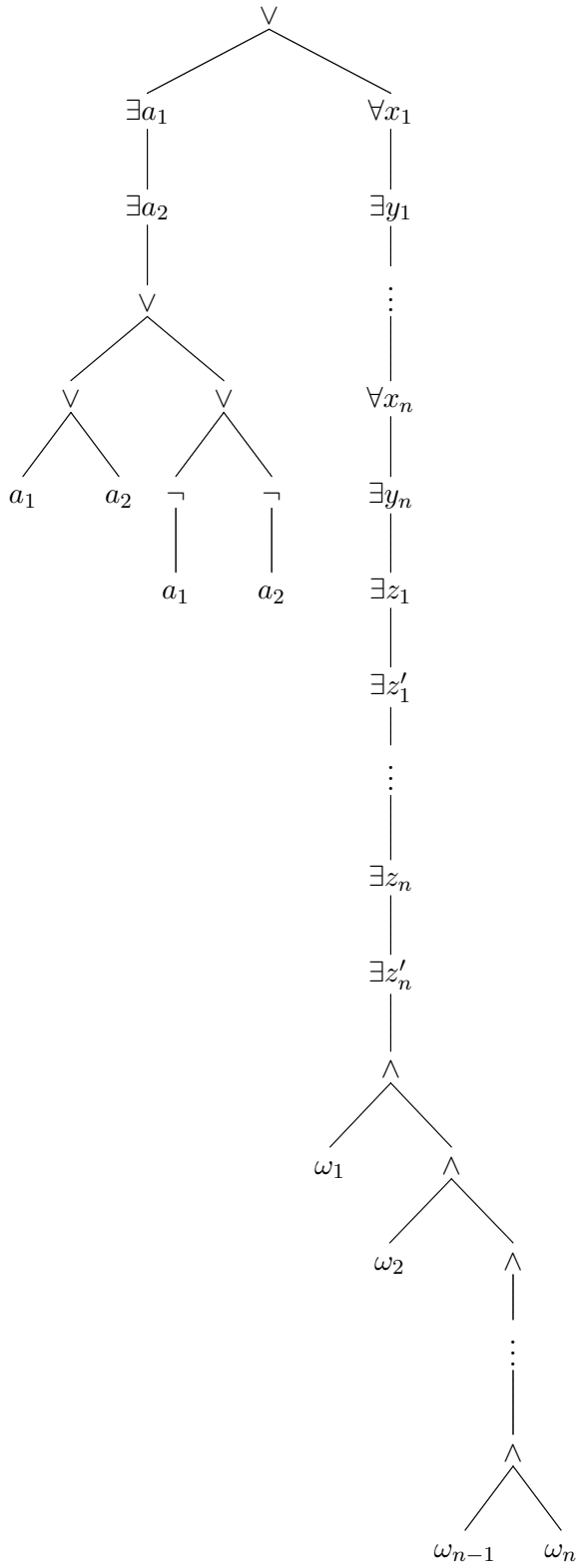


Figure 9.2.: The Structure of φ_n from the proof of Theorem 9.3.3.

Part V.

Q-resolution

10. Resolution for Quantified Boolean Formulas

In this chapter, we describe a resolution-style proof system for quantified boolean formulas that has been first presented in [4]. As for GQBF, we will first provide a firm definition of the calculus called Q-resolution and will provide further details and findings in the subsequent chapters.

10.1. The Calculus

Q-resolution is a sound and complete resolution-like calculus for quantified boolean formulas. In contrast to GQBF, Q-resolution cannot operate on quantified boolean formulas in negation normal form. Like resolution for propositional logic, it applies to formulas in conjunctive normal form. As quantified boolean formulas contain quantifiers, they have to be treated specially. They are shifted as far as possible "outwards" yielding input formulas in prenex normal form. Formulas that are not in prenex conjunctive normal form have to be brought into an equivalent normal form before being operated on by Q-resolution. Various impacts of this necessity are discussed in Chapter 9 and in Part VI. Throughout this chapter — and in the following ones — we will use both, formula and clause notation for quantified boolean formulas in prenex conjunctive normal form synonymously. Thereby we may use clause notation for the propositional core of formulas and annotate the quantifier prefix separately. For a short reminder on clause forms, see Section 9.2. Before we describe the derivation rules of Q-resolution, we introduce some notational forms.

Formulas in Prenex Conjunctive Normal From As Clauses Q-resolution operates on input formulas in prenex conjunctive normal form only. The order of literals inside a clause is of no relevance in general as such formulas that differ in the ordering of their variables only are isomorphic. For the sake of simpler definitions, Q-resolution imposes a ordering constraint on its input formulas. It will be presented in the following:

Let $\varphi \in \mathcal{QBF}^{\mathrm{u}}$ be a quantified boolean formula in prenex conjunctive normal form and $V = \mathit{free}(\mathit{core}(\varphi))$ be the free variables of the propositinal core of φ. Then we define the relation $<_\varphi \colon V \times V \to \mathbb{B}$ as follows: (x, y) is in the relation $<_\varphi$ if and only if x precedes y in the quantifier prefix of φ. In other words: $x <_\varphi y$ if Qy appears within a sub-tree of Qx in the structure tree of φ.

Without loss of generality, we always choose a clause notation for a disjunction such that the order within the clause respects the $<_\varphi$ relation.

Let us consider an example: Let $\varphi = \forall a \exists b \forall c \forall d\,(b \vee c \vee a) \wedge (\neg a \vee c \vee d) \vee (b \vee \neg d)$ be a closed quantified boolean formula in prenex conjunctive normal form. Then φ induces the clauses $\{a, b, c\}$, $\{\neg a, c, d\}$ and $\{b, \neg d\}$. The clause $\{a, b, c\}$ corresponds to the disjunction $(b \vee c \vee a)$ whereas the following ordering constraints — derived from the quantifier prefix of φ — are satisfied within the clause: $a <_\varphi b <_\varphi c$. The set $\{\neg a, c, d\}$ corresponds to $(\neg a \vee c \vee d)$ and the set $\{b, \neg d\}$ represents $(b \vee \neg d)$.

As one can see in the example, the three clauses do not describe the formula entirely — it cannot be deduced from the clauses if a certain variable is universally or existentially quantified. To deal with this deficiency we introduce a function $quan/1$ that returns the quantifier type for a variable symbol. To continue the example: $quan(a) = quan(c) = quan(d) = \forall$, and $quan(b) = \exists$.

We implicitly treat clauses as sets. If a clause contains multiple copies of a literal, they are contracted immediately, i.e. $\{a, a, a, b, \neg c, \neg c, d, d, \neg d\}$ is contracted to $\{a, b, c, d, \neg d\}$. This has impacts on the calculation of proof sizes and will be discussed in more detail subsequently.

Pure Clauses The authors of [4] define a pure $\forall$-clause as "a non-tautological clause consisting exclusively of $\forall$-literals." Note that the empty clause $\varnothing$ thus is a pure $\forall$-clause.

Derivation Rules Q-resolution has two principal derivation rules.

- Trailing Universals Elimination:

$$\frac{\{\ell_1, \ell_2, \ldots, \ell_n, \ell_{n+1}, \ell_{n+2}, \ldots, \ell_{n+m}\}}{\{\ell_1, \ell_2, \ldots, \ell_n\}}\ p$$

  The rule may only be applied if $\{\ell_{n+1}, \ell_{n+2}, \ldots, \ell_{n+m}\}$ is a non-tautological and pure $\forall$-clause and if ℓ_1 to ℓ_n are either existentially or universally quantified. It thus removes all universally quantified literals that do not precede any existentially quantified literals occurring in the clause. Note that the literals with the clauses are ordered with respect the quantifier prefix of the formula to be proven! We call the set $\{\ell_1, \ell_2, \ldots, \ell_n\}$ consequent of the p-rule and we call the set $\{\ell_1, \ell_2, \ldots, \ell_n, \ell_{n+1}, \ell_{n+2}, \ldots, \ell_{n+m}\}$ antecedent of the p-rule.

- Resolution:

$$\frac{A_1\colon \{\ell_1, \ldots, \ell_n, x, \ell_{n+1}, \ldots, \ell_{n+m}\} \qquad A_2\colon \{\ell'_1, \ldots, \ell'_{n'}, \neg x, \ell'_{n'+1}, \ldots, \ell'_{n'+m'}\}}{C\colon \{\ell_1, \ldots, \ell_n, \ell_{n+1}, \ldots, \ell_{n+m}, \ell'_1, \ldots, \ell'_{n'}, \ell_{n'+1}, \ldots, \ell_{n'+m'}\}}\ r$$

In the above instance of the r-rule, the set C is called consequent while the sets A_1 and A_2 are called antecedents. The resolution rule may only be applied if the following three conditions are satisfied.

1. Rule p is not applicable to neither of the antecedents.
2. $quan(x) = quan(\neg x) = \exists$.
3. The antecedents do not contain complementary literals (other than x).

Implicit Assumptions Q-resolution as presented above has very strong rules. They implicitly involve the following derivations and assumptions:

- Clauses are treated as sets: Multiple occurrences of literals in a clause are impossible. Literals are contracted to a single literal if they appear more than one time in the same polarity within a clause. Also a literal cannot appear in both polarities in a clause as non-tautological clauses are allowed only.

- It is important to note that the authors of the original paper [4] do not use the p-rule directly and explicitly. Instead they use it as a kind of preprocessing stage for the r-rule. That change in notation does not change the set of formulas that can be proved but the calculation of proof sizes might differ. While we incoorporate each application of the p-rule explicitly in the caluluation of proof sizes, [4] does not. Thus, proof sizes stated in [4] have to be treated as lower bounds rather than exact values when compared to proof sizes calculated for e.g. GQBF.

Application We use Q-resolution to prove the invalidity of quantified boolean formulas. As been stated before, Q-resolution relies on the formula to be refuted to be in prenex conjunctive normal form. If the formala to be worked with is not, it has to be translated into a corresponding prenex conjunctive normal form. It is then represented as a set of clauses and a global quantifier prefix.

A Q-resolution is a derivation in form of a directed acylic graph whereas the consequents of r-rules are called Q-resolvents and the consequents of p rules are called clean Q-resolvents. If we refer to clauses, we refer to Q-resolvents as well as clean Q-resolvents. The axioms of such a Q-resolution proof are the clauses that represent the formula to be refuted. In the course of the proof, new clauses are derived, first from the axioms, then from the axioms and previously derived clauses. If a Q-resolution proof derives the empty clause $\varnothing$, then the formula that provided the axioms is said to be disproved or refuted.

We write $\pi\colon\ \vdash_{\mathrm{R}} \neg\varphi$ to state that π is a Q-resolution proof of the invalidity of φ. Hence, π a proof of the validity of $\neg\varphi$. Furthermore we write $\vdash_{\mathrm{R}} \neg\varphi$ to state that there exists a Q-resolution proof π of the in validity of $\neg\varphi$.

Q-resolution is sound and complete. A proof can be found in the paper introducing Q-resolution [4]. A quantified boolean formula is false if and only if the empty clause can be derived from the formula via Q-resolution.

A Derivation Example Let $\varphi = \forall x_1 \forall x_2 \exists y_1 \exists y_2 \forall x_3((x_1 \vee \neg x_2 \vee \neg y_1 \neg \vee \neg y_2 \vee x_3) \wedge (\neg x_2 \vee y_2) \wedge (y_1 \vee x_3)$ be a quantified boolean formulas in prenex conjunctive normal form that is unsatisfiable. Then there exists a Q-resolution proof π of the invalidity of φ. It can be constructed the following way:

$$\dfrac{\dfrac{\dfrac{\{x_1, \neg x_2, \neg y_1, \neg y_2, x_3\}}{\{x_1, \neg x_2, \neg y_1, \neg y_2\}}\, p \qquad \{\neg x_2, y_2\}}{\{x_1, \neg x_2, \neg y_1\}}\, r \qquad \dfrac{\{y_1, x_3\}}{\{y_1\}}\, p}{\dfrac{\{x_1, \neg x_2\}}{\varnothing}\, p}\, r$$

Size of Q-resolution Proofs Let $\varphi \in \mathcal{QBF}$ be a quantified boolean formulas in prenex conjunctive normal form and π: $\vdash_{\mathrm{R}} \neg\varphi$ be a Q-resolution proof of the invalidity of φ. Then we define the size of the refuation π, denoted $|\pi|$ inductively as follows:

1. π is an axiom (i.e.: a clause from φ): $|\pi| := 1$
2. π is a proof $\dfrac{R}{\vdash \varphi}\, p$ of $\vdash \varphi$ ending at a p-inference. Let moreover R denote a reference to a clause. Then $|\pi| := 1$.
3. π is a proof $\dfrac{R_1 \quad R_2}{\vdash \varphi}\, r$ of $\vdash \varphi$ ending at a resolution step. Let R_1 and R_2 denote references to clauses. Then $|\pi| := 1$.
4. π is a proof $\dfrac{\begin{array}{c}\tau\\ \vdots\end{array}}{\vdash \varphi}\, p$ of $\vdash \varphi$ ending at a p-inference. Let moreover τ denote the proof of the rule's antecedent. Then $|\pi| := 1 + |\tau|$.
5. π is a proof $\dfrac{\begin{array}{c}\tau\\ \vdots\end{array} \quad R}{\vdash \varphi}\, r$ of $\vdash \varphi$ ending at a resolution step. Let τ denote the proof of the rule's first premise and R denote a reference to a clause. Then $|\pi| := 1 + |\tau|$.
6. π is a proof $\dfrac{R \quad \begin{array}{c}\tau\\ \vdots\end{array}}{\vdash \varphi}\, r$ of $\vdash \varphi$ ending at an application of r. Let τ denote the proof of the rule's second premise and R denote a reference to a clause. Then $|\pi| := 1 + |\tau|$.

7. π is a proof $\dfrac{\begin{array}{cc}\tau_1 & \tau_2\\ \vdots & \vdots\end{array}}{\vdash \varphi}\, r$ of $\vdash \varphi$ ending at a resolution step and let τ_1 and τ_2 denote the proofs of the rule's antecedents. Then $|\pi| := 1 + |\tau_1| + |\tau_2|$.

Thus the size of Q-resolution refutation corresponds to the number of distinct clauses, including axioms, in it. This stems from the fact that we allow proofs to be in directed acyclic graph form. It shall also be noted that we count clauses created by the p-rule whereas the original paper [4] does not. Thus proof sizes from the original paper always constitute lower bounds to our notion of proof size. Also note that implicit contraction is used within clauses. Making contraction explicit can further increase proof size. Again, the sizes used can be seen as lower bounds. Also, the clauses that represents a formula can contain duplicated clauses. Q-resolution does not take this into account. As it operates on a set of clauses, duplicate clauses are "contracted" implicitly. Note that contracting duplicate sub-formulas can exponentially shorten $\mathrm{GQBF}^{+}_{\diamond}$ proofs. See the proof of Theorem 8.2.1 for a demonstrating example.

Part VI.

Simulation

11. Simulation

This part deals with the notions of polynomial simulation and effectively polynomial simulation. We will define these notions in detail and constitute a hierarchy for GQBF-style calculi. Furthermore we will discuss relations between GQBF-style calculi and Q-resolution with respect to these notions. Finally we will compare all calculi mentioned to G_0 from [10], another sequent-style calculus for quantified boolean formulas. First we will define the notions of polynomial simulation (p-simulation), strong polynomial simulation (strong p-simulation) and effectively polynomial simulation (effectively p-simulation) [16][1].

Polynomial Simulation In [16] p-simulation is defined the following way.

Definition 11.0.1. Let A and B be two proof systems. Then A p-simulates B if for all formulas φ, the shortest A-proof for φ is at most polynomially longer than the shortest B-proof of φ.

Strong Polynomial Simulation Again, we use the definition given in [16]:

Definition 11.0.2. Let A and B be two proof systems. Then A strongly p-simulates B if A p-simulates B and moreover, there is a polynomial-time computable function f that transforms B-proofs of φ into A-proofs of φ.

Effectively Polynomial Simulation The authors of [16] introduce the notion of effectively p-simulation as follows:

Definition 11.0.3. Let A and B be two proof systems. Then A effectively p-simulates B if there is a polynomial-time in m truth-preserving transformation from (encodings of) boolean formulas to (encodings of) boolean formulas, $R(\varphi, m)$ such that when m is at least the size of the shortest B-proof of φ , $R(\varphi, m)$ has an A proof of size polynomial in $|\varphi| + m$. If there also exists a polynomial-time function (again polytime in $|\varphi| + m$) that maps B-proofs of φ to A-proofs of $R(\varphi, m)$, then we say that A strongly effectively p-simulates B.

Additionally m truth-preserving is defined that way:

Definition 11.0.4. Let R be a function $\mathcal{QBF} \times \mathbb{N} \to \mathcal{QBF}$. Then R is called polynomial-time, in m truth-preserving transformation if the following conditions hold for all pairs $(\varphi, m) \in (\mathcal{QBF} \times \mathbb{N})$:

[1] The authors of [16] abbreviate "effectively polynomial simulations" by the term "effectively-p simulations".

- $\varphi \equiv R(\varphi, m)$.
- $R(\varphi, m)$ can be calculated in a time polynomial in $|\varphi| + m$ where $|\varphi|$ is the number of connectives in φ and m is an auxiliary parameter.

Now we will discuss the relations between different calculi presented so far. We will often use results gathered in the previous parts and chapters of this thesis. First we will constitute a hierarchy of GQBF based calculi.

11.1. A Hierarchy for Sequent-Style Calculi

Theorem 11.1.1 (GQBF cannot p-simulate GQBF$^+$)**.** There is a countably infinite family of formulas $\varphi_1, \varphi_2, \ldots$, such that for members φ_i of this family there exists a GQBF$^+$ proof of polynomial size with respect to $|\varphi_i|$ whereas there is no GQBF proof of φ_i of polynomial size.

Proof. By Theorem 8.1.1.

□

Theorem 11.1.2 (GQBF$^+$ can strongly p-simulate GQBF)**.** There exists a function f that is computable in polynomial time and that transform GQBF proofs into GQBF$^+$ proofs such that the following holds: Let π be a GQBF proof. Then a polynomial $p \in \mathbf{P}[\mathbb{N}]$ exists such that $|f(\pi)| = p(|\pi|)$.

Proof. Let f be the identity function id. Choose p to be the identity function too, regardless of π. Then we get:

$$|f(\pi)| = |id(\pi)| = |\pi| = id(|\pi|) = p(|\pi|)$$

□

Theorem 11.1.3 (GQBF$^+$ cannot p-simulate GQBF$^+_\Diamond$)**.** There is a countably infinite family of formulas such that for members of this family there exists a GQBF$^+_\Diamond$ proof of polynomial size whereas there is no GQBF$^+$ proof of polynomial size.

Proof. By Theorem 8.2.1.

□

Theorem 11.1.4 (GQBF$^+_\Diamond$ can strongly p-simulate GQBF$^+$)**.** There exists a polynomial-time computable function f that transform GQBF$^+$ proofs into GQBF$^+_\Diamond$ proofs such that the following holds: Let π be a GQBF$^+$ proof. Then a polynomial $p \in \mathbf{P}[\mathbb{N}]$ exists such that $|f(\pi)| = p(|\pi|)$.

Proof. Identical to the proof of Theorem 11.1.2: Let f be the identity function id. Choose p to be the identity function too, regardless of π. Then we get:

$$|f(\pi)| = |id(\pi)| = |\pi| = id(|\pi|) = p(|\pi|)$$

□

Transitivity of p-Simulation Clearly, p-simulation as defined before is a transitive relation. Assume that A p-simulates B and B p-simulates C. Take a formula φ and the smallest C-proof π_C of φ. Then a proof π_B of φ and a polynomial p_{BC} exist such that $|\pi_B| = p_{BC}(|\pi_C|)$. Further a proof π_A of φ and a polynomial p_{AB} exist such that $|\pi_A| = p_{AB}(|\pi_B|)$. It follows that $|\pi_A| = p_{AB}(|\pi_B|) = p_{AB}(p_{BC}(|\pi_C|)) = p_{AC}(|\pi_C|)$ with $p_{AC} \in \mathbf{P}[\mathbb{N}]$.

The same holds for strong p-simulation. If A strongly p-simulates B via $f\colon \mathcal{P}_B \to \mathcal{P}_A$ and B strongly p-simulates C via $g\colon \mathcal{P}_C \to \mathcal{P}_B$ then A strongly p-simulates C via $h = (f \circ g)\colon \mathcal{P}_C \to \mathcal{P}_A$.

From p-Simulation to Effectively p-Simulation Effectively p-simulation is implied by p-simulation. If A p-simulates B then A effectively p-simulates B via the polynomial-time 0 truth-preserving transformation $R\colon \mathcal{QBF} \times \mathbb{N} \to \mathcal{QBF} = (\lambda xy\,.\,x)$. In other words we do not transform the input formula as polynomial simulation is given anyway. In the other direction, if A cannot effectively p-simulate B then A surely cannot p-simulate B. This can be proven this way: Let A and B be two calculi such that A cannot effectively p-simulate B. Now assume that A can instead p-simulate B via f. Then A effectively p-simulates B via $(\lambda xy\,.\,x)$ which yields a contradiction.

Transitivity can be established for effectively p-simulation in the same way as for p-simulation and strong p-simulation.

G Can Effectively p-Simulate Directed Acyclic Graph GQBF A fully-fledged sequent calculus for quantified boolean formulas, named G, is presented in [10]. Subsequently we will introduce this calculus and discuss its position in the hierarchy of GQBF-style calculi.

The authors of [10] construct their calculus as an extension to propositional LK from [20]. In [10] G-proofs are not restricted to trees. Instead they are allowed to be in directed (acyclic) graph form. Thus sequents may be premisses to multiple inferences like in $\mathrm{GQBF}^{+}_{\diamond}$. Note that the sequents of G are, in contrast to GQBF, of the form $\Gamma \to \Delta$ with Γ and Δ being finite (multi-) sets of quantified boolean formulas. The authors of [10] state that G includes the following derivation rules:

- Initial Sequents: They correspond to axioms in GQBF-style calculi.
  - $x \to x$, where $x \in \mathcal{V}$ is a variable symbol.
  - $\bot \to$
  - $\to \top$
- Structural rules, *cut*-rule, logical rules, quantifier rules: See Figure 11.1.

The power of G to provide short proofs stems from the quantifier rules. None of them requires to split the proof into two branches. Eigenvariables are introduced instead. A eigenvariable is a variable symbol that does not occur in

Structural Rules:

$$\frac{\Gamma \to \Delta}{D, \Gamma \to \Delta}\ w_l \qquad \frac{\Gamma \to \Delta}{\Gamma \to \Delta, D}\ w_r$$

$$\frac{D, D, \Gamma \to \Delta}{D, \Gamma \to \Delta}\ c_l \qquad \frac{\Gamma \to \Delta, D, D}{\Gamma \to \Delta, D}\ c_r$$

Cut:

$$\frac{\Gamma \to \Delta, D \quad D, \Pi \to \Lambda}{\Gamma, \Pi \to \Delta, \Lambda}\ cut$$

Logical Rules:

$$\frac{\Gamma \to \Delta, D}{\neg D, \Gamma \to \Delta}\ \neg_l \qquad \frac{D, \Gamma \to \Delta}{\Gamma \to \Delta, \neg D}\ \neg_r$$

$$\frac{\Gamma \to \Delta, C_1 \quad \Gamma \to \Delta, C_2}{\Gamma \to \Delta, C_1 \wedge C_2}\ \wedge_r \qquad \frac{C_1, \Gamma \to \Delta}{C_1 \wedge C_2, \Gamma \to \Delta}\ \wedge_l^1 \qquad \frac{C_2, \Gamma \to \Delta}{C_1 \wedge C_2, \Gamma \to \Delta}\ \wedge_l^2$$

$$\frac{C_1, \Gamma \to \Delta \quad C_2, \Gamma \to \Delta}{C_1 \vee C_2 \Gamma \to \Delta}\ \vee_l \qquad \frac{\Gamma \to \Delta, C_1}{\Gamma \to \Delta, C_1 \vee C_2}\ \vee_r^1 \qquad \frac{\Gamma \to \Delta, C_2}{\Gamma \to \Delta, C_1 \vee C_2}\ \vee_r^1$$

Quantifier Rules:

$$\frac{\varphi[x\backslash\psi], \Gamma \to \Delta}{\forall x\, \varphi, \Gamma \to \Delta}\ \forall_l \qquad \frac{\Gamma \to \Delta, \varphi[x\backslash e]}{\Gamma \to \Delta, \forall x\, \varphi}\ \forall_r$$

e is a eigenvariable. I.e., it does not occur in the lower sequent.

$$\frac{\varphi[x\backslash e], \Gamma \to \Delta}{\exists x\, \varphi, \Gamma \to \Delta}\ \exists_l \qquad \frac{\Gamma \to \Delta, \varphi[x\backslash\psi]}{\Gamma \to \Delta, \exists x\, \varphi}\ \exists_r$$

e is a eigenvariable. I.e., it does not occur in the lower sequent.

Figure 11.1.: The derivation rules of G.

any sequent of any trace from the end-sequent to the position it is introduced. Thus, from a bottom-up perspective, eigenvariables are always fresh variable symbols. Inroducing eigenvariables when quantifiers are eliminated allows to

"defer" the decision of choosing a truth value for the variable symbol that is instantiated. Derivations that are further down the derivation tree have then to take the eigenvariables into account. The benefit of avoiding branching is gained at the cost of ordering constraints on the sequence of derivations. This can be seen at the following example:

$$\dfrac{\dfrac{e \to t}{\exists x\, x \to t}\,\exists_l}{\exists x\, x \to \exists x\, x}\,\exists_r \qquad \text{or} \qquad \dfrac{\dfrac{e \to e}{e \to \exists x\, x}\,\exists_r}{\exists x\, x \to \exists x\, x}\,\exists_l$$

The sequence $(\exists_r, \exists_l)$ does not produce a proof while the sequence $(\exists_l, \exists_r)$ of derivation rules yields a valid G proof of the sequent $\exists x\, x \to \exists x\, x$.

According to [16], G_i "is a subsystem of G obtained by restricting the cut rule to Σ_i^q QBF formulas only". In this manner, G_0 is the *G*-style proof system that allows *cut* to be used only on propositional formulas.

Now [16] gives and proves the following, quite interesting, Theorem (Theorem 3.8 in the paper):

Theorem 11.1.5 (The Effective Power of G_0)**.** For any i, G_0 can effectively p-simulate any proof system for Σ_i^q quantified boolean formulas.

This is quite remarkable as it basically states that there exists a calculus that is effectively optimal. Especially as [18] proved that from the existence of an optimal calculus for quantified boolean formulas — the existence of a calculus that polynomially simulates all other calculi — follows that (NP ∩ co-NP) has complete languages. Currently it is considered as highly unlikely that such complete languages indeed exist.

As a corollary from Theorem 11.1.5 we can conclude that G can effectively p-simulate $\mathrm{GQBF}^+_\diamond$.

Corollary 11.1.1. G can effectively p-simulate $\mathrm{GQBF}^+_\diamond$. By transitivity it can also p-simulate GQBF and GQBF^+ effectively.

Theorem 11.1.6 ($\mathrm{GQBF}^+_\diamond$ cannot p-simulate G)**.** There is a countably infinite family of formulas such that for members of this family there exists a G proof of polynomial size whereas there is no $\mathrm{GQBF}^+_\diamond$ proof of polynomial size.

Proof. Let

$$\varphi_n = \forall x_1\, \exists y_1\, \forall x_2\, \exists y_2\, \ldots\, \forall x_n\, \exists y_n\, \exists z_1\, \exists z'_1\, \exists z_2\, \exists z'_2\, \ldots\, \exists z_n\, \exists z'_n \bigwedge_{i=1}^{n} \psi_i$$

be a quantified boolean formula in negation normal form with

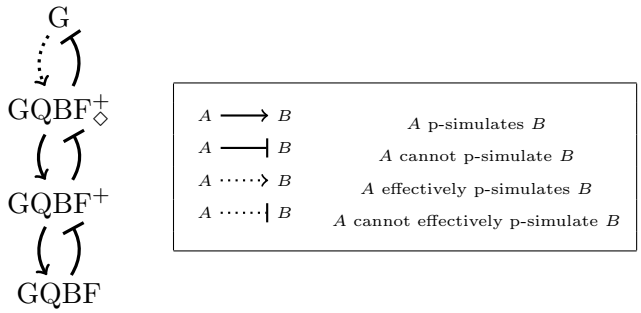
$$\psi_i = (\neg x_i \vee y_i) \wedge (\neg y_i \vee x_i) \wedge (x_i \vee z_i \vee z'_i) \wedge (\neg x_i \vee \neg z_i \vee \neg z'_i) \wedge (z_i \vee z'_i) \wedge (\neg z_i \vee \neg z'_i).$$

Then we know by Lemma 8.2.1 that φ_n is not polynomially provable using $\mathrm{GQBF}^+_\diamond$. It remains to show that φ_n is indeed polynomially provable with G. A *cut*-free proof with a mid-sequent $\to \omega$ can be constructed (bottom up) as follows:

- Let $\to \varphi_n$ be the end-sequent. We apply $4n$ quantifier rules (bottom up) to construct the mid-sequent $\to \omega$: x_i is replaced by e_i which always satisfies the eigenvariable condition. y_i is instantiated with e_i while z_i and z_i' are instantiated by $\top$.
- All quantifiers are eliminated and $\to \bigwedge_{i=1}^{n} \psi_i[x_i \backslash e_i][y_i \backslash e_i][z_i \backslash \top][z_i' \backslash \top]$ is the mid-sequent $\to \omega$.
- Now we apply the $\wedge_r$-rule to the mid-sequent $\to \omega$ to separate the first conjunct $\psi_1[x_1 \backslash e_1][y_1 \backslash e_1][z_1 \backslash \top][z_1' \backslash \top]$ from the conjunction. The left branch can be proven using a proof of constant size while the right branch contains a strictly smaller instance of the mid-sequent.
- Iterate the above step in the left branch. Again, the newly created left branch can be closed easily while the right contains a strictly smaller instance.

The size of the overall proof is $(19n - 1)$ and thus polynomial in n. Also note that it is not even necessary to use *cut* or weakening and that a proof in form of a tree suffices to constitute a proof of polynomial size. □

We can thus establish a complete hierarchy of sequent-style calculi for quantified boolean formulas:



11.2. Simulation between GQBF and Q-resolution

In the last section we established a hierarchy for sequent-style calculi. We now try to augment this hierarchy by adding relations to resolution-style calculi. For this purpose, we research the relations to Q-resolution in more detail. At the beginning we compare Q-resolution to a weak sequent calculus and continue with comparing it to stronger sequent calculi.

Theorem 11.2.1 (GQBF cannot p-simulate Q-resolution)**.** For every $n \geq 1$ there exists a quantified boolean formula φ_n such that there exists a Q-resolution proof π: $\vdash_{\mathrm{R}} \neg\varphi_n$ with $|\pi| \leq 5$ whereas the shortest GQBF proof τ: $\vdash_{\mathrm{G}} \varphi_n$ has $|\tau| \geq 2^n$.

Proof. Let $\varphi_n = \exists x_1 \exists x_2 \ldots \exists x_n \forall a \, (\psi(x_1, a) \wedge \psi(x_2, a) \wedge \ldots \wedge \psi(x_n, a))$ with $\psi(x, a) := (a \vee x) \wedge (a \vee \neg x) \wedge (\neg a \vee x) \wedge (\neg a \vee \neg x)$. Clearly, $|\varphi_n| = 14n + 1$.

Then there exists a Q-resolution proof π: $\vdash_{\mathrm{R}} \neg\varphi_n$ with $|\pi| = 4$:

$$\dfrac{\dfrac{\{x_1, a\} \quad \{\neg x_1, a\}}{\{a\}}\, r}{\dfrac{\{a\}}{\varnothing}\, p}$$

A GQBF $(+\neg)$ proof τ of the invalidity of φ_n has to begin (bottom up) by applying the $\neg$ rule.

$$\dfrac{\begin{array}{c}\tau' \\ \vdots \\ \vdash \forall x_1 \ldots \forall x_n \exists a \, (nnf(\neg\psi(x_1, a)) \vee \ldots \vee nnf(\neg\psi(x_n, a)))\end{array}}{\varphi_n \vdash}\, \neg$$

From Theorem 8.1.1 we know that the size of the proof τ' is at least 2^n. Hence the size of τ is at least 2^n too. That gives us that there is no GQBF proof of the invalidity of φ_n of a size polynomial in n. □

Lemma 11.2.1 (Exponential Q-resolution proofs [4])**.** For every n $(n \geq 1)$ there exists a formula φ_n of length $18n + 1$ which is false, and the refutation of the empty clause requires at least 2^n Q-resolution steps.

A proof can also be found in [4]. We will use this Lemma subsequently to prove relations between certain forms of GQBF and Q-resolution.

Theorem 11.2.2 (Q-resolution cannot p-simulate $\mathrm{GQBF}^{+}_{\diamond}$)**.** For every $n \geq 1$, there is a quantified boolean formula φ_n in prenex conjunctive normal form such that there is no Q-resolution proof of a polynomial size of the invalidity of φ_n but there exists a $\mathrm{GQBF}^{+}_{\diamond}$ $(+\neg)$ proof of polynomial size.

Proof. Let φ_n be defined as in Theorem 3.2 in [4]:

$$\begin{aligned}
\varphi_n = {} & \exists y_0\, \exists y_1\, \exists y_1'\, \forall x_1\, \exists y_2\, \exists y_2'\, \forall x_2\, \exists y_3\, \exists y_3'\, \ldots\, \forall x_{n-1}\, \exists x_n\, \exists x_n'\, \forall x_n \\
& \exists y_{n+1}\, \exists y_{n+2}\, \ldots\, \exists y_{n+n} \\
& (\neg y_0) \wedge \\
& (y_0 \vee \neg y_1 \vee \neg y_1') \wedge \\
& (y_1 \vee \neg x_1 \vee \neg y_2 \vee \neg y_2') \wedge (y_1' \vee x_1 \vee \neg y_2 \vee \neg y_2') \wedge \\
& (y_2 \vee \neg x_2 \vee \neg y_3 \vee \neg y_3') \wedge (y_2' \vee x_2 \vee \neg y_3 \vee \neg y_3') \wedge \\
& \vdots \\
& (y_{n-1} \vee \neg x_{n-1} \vee \neg y_n \vee \neg y_n') \wedge (y_{n-1}' \vee x_{n-1} \vee \neg y_n \vee \neg y_n') \wedge \\
& (y_n \vee \neg x_n \vee \neg y_{n+1} \vee \neg y_{n+2} \vee \ldots \vee \neg y_{n+n}) \wedge \\
& (y_n' \vee x_n \vee \neg y_{n+1} \vee \neg y_{n+2} \vee \ldots \vee \neg y_{n+n}) \wedge \\
& (x_1 \vee y_{n+1}) \wedge (x_2 \vee y_{n+2}) \wedge \ldots \wedge (x_n \vee y_{n+n}) \wedge \\
& (\neg x_1 \vee y_{n+1}) \wedge (\neg x_2 \vee y_{n+2}) \wedge \ldots \wedge (\neg x_n \vee y_{n+n})
\end{aligned}$$

Then we know by Lemma 11.2.1 (respectively Theorem 3.2 in [4]) that there is no Q-resolution proof of the invalidity of φ_n of a polynomial size. As in the proof of Theorem 11.2.1, a proof τ of the invalidity of φ_n starts with the $\neg$-rule.

$$\dfrac{\begin{array}{c}\tau' \\ \vdots \\ \vdash \psi_n\end{array}}{\varphi_n \vdash}\ \neg$$

Where ψ_n is the following formula:

$$\begin{aligned}
\psi_n = {} & nnf(\neg\varphi_n) = \\
& \forall y_0\, \forall y_1\, \forall y_1'\, \exists x_1\, \forall y_2\, \forall y_2'\, \exists x_2\, \forall y_3\, \forall y_3'\, \ldots\, \exists x_{n-1}\, \forall x_n\, \forall x_n'\, \exists x_n \\
& \forall y_{n+1}\, \forall y_{n+2}\, \ldots\, \forall y_{n+n} \\
& (y_0) \vee \\
& (\neg y_0 \wedge y_1 \wedge y_1') \vee \\
& (\neg y_1 \wedge x_1 \wedge y_2 \wedge y_2') \vee (\neg y_1' \wedge \neg x_1 \wedge y_2 \wedge y_2') \vee \\
& (\neg y_2 \wedge x_2 \wedge y_3 \wedge y_3') \vee (\neg y_2' \wedge \neg x_2 \wedge y_3 \wedge y_3') \vee \\
& \vdots \\
& (\neg y_{n-1} \wedge x_{n-1} \wedge y_n \wedge y_n') \wedge (\neg y_{n-1}' \wedge \neg x_{n-1} \wedge y_n \wedge y_n') \vee \\
& (\neg y_n \wedge x_n \wedge y_{n+1} \wedge y_{n+2} \wedge \ldots \wedge y_{n+n}) \vee \\
& (\neg y_n' \wedge \neg x_n \wedge y_{n+1} \wedge y_{n+2} \wedge \ldots \wedge y_{n+n}) \vee \\
& (\neg x_1 \wedge \neg y_{n+1}) \vee (\neg x_2 \wedge \neg y_{n+2}) \vee \ldots \vee (\neg x_n \wedge \neg y_{n+n}) \vee \\
& (x_1 \wedge \neg y_{n+1}) \vee (x_2 \wedge \neg y_{n+2}) \vee \ldots \vee (x_n \wedge \neg y_{n+n})
\end{aligned}$$

Now the $\mathrm{GQBF}^{+}_{\diamond}$ proof τ' of ψ_n (of polynomial length) is constructed bottom up as follows:

1. Apply the $\forall$-rule to the end-sequent $\vdash \psi_n$. In the positive branch — the branch where y_0 is replaced by $\top$ — we end up in an axiom in a short linear proof:

   The sequent to be proven is $\vdash \forall y_1 \ldots (\top \vee \ldots)$. We thus apply the rule $S3a$ and get $\vdash \forall y_1 \ldots (\top)$. $4n$ applications of the rule $S4$ suffice to remove all quantifiers and reach $\vdash \top$ which indeed is an axiom.

   In the negative branch — y_0 is replaced by $\bot$ — we get $\vdash \forall y_1 \ldots (\bot \vee (\neg\bot \wedge y_1 \wedge y_1') \vee \ldots)$. After an application of $S3b$ that removes the superfluous $\bot$ clause, an application of $S1b$ and one of $S2a$ remove the superfluous $\neg\bot$ in the first clause. We have to find a polynomial proof for the following formula:

$$
\begin{aligned}
\omega_n = {} & \forall y_1 \forall y_1' \exists x_1 \forall y_2 \forall y_2' \exists x_2 \forall y_3 \forall y_3' \ldots \exists x_{n-1} \forall x_n \forall x_n' \exists x_n \\
& \forall y_{n+1} \forall y_{n+2} \ldots \forall y_{n+n} \\
& (y_1 \wedge y_1') \vee \\
& (\neg y_1 \wedge x_1 \wedge y_2 \wedge y_2') \vee (\neg y_1' \wedge \neg x_1 \wedge y_2 \wedge y_2') \vee \\
& (\neg y_2 \wedge x_2 \wedge y_3 \wedge y_3') \vee (\neg y_2' \wedge \neg x_2 \wedge y_3 \wedge y_3') \vee \\
& \vdots \\
& (\neg y_{n-1} \wedge x_{n-1} \wedge y_n \wedge y_n') \wedge (\neg y_{n-1}' \wedge \neg x_{n-1} \wedge y_n \wedge y_n') \vee \\
& (\neg y_n \wedge x_n \wedge y_{n+1} \wedge y_{n+2} \wedge \ldots \wedge y_{n+n}) \vee \\
& (\neg y_n' \wedge \neg x_n \wedge y_{n+1} \wedge y_{n+2} \wedge \ldots \wedge y_{n+n}) \vee \\
& (\neg x_1 \wedge \neg y_{n+1}) \vee (\neg x_2 \wedge \neg y_{n+2}) \vee \ldots \vee (\neg x_n \wedge \neg y_{n+n}) \vee \\
& (x_1 \wedge \neg y_{n+1}) \vee (x_2 \wedge \neg y_{n+2}) \vee \ldots \vee (x_n \wedge \neg y_{n+n})
\end{aligned}
$$

2. Prove ω_n (by reducing it to ω_{n-1}): Apply the $\forall$-rule and split over y_1. That results in two branches, in the first one y_1 is replaced by $\top$ whereas it is replaced by $\bot$ in the second one.

   - $y_1 \longrightarrow \top$: $\vdash \forall y_1' \ldots ((\top \wedge y_1') \vee (\neg\top \wedge x_1 \wedge y_2 \wedge y_2') \vee \ldots)$. The rules $S2a$, $S1a$, $S3b$, $LU2a$, $S1b$ and $S2a$ are applied to get $\vdash \forall y_1' \ldots (y_1' \vee (\neg x_1 \wedge y_2 \wedge y_2') \vee \ldots)$. We branch over y_1'. The positive branch $\vdash \exists x_1 \ldots (\top \vee \ldots)$ can be closed with a short linear proof very similar to step (1).

     In the negative branch $\vdash \exists x_1 \ldots (\bot \vee \ldots)$ the following happens: Apply the $S3b$-rule to remove the superfluous $\bot$-clause, then apply the $\exists_\bot$-rule assigning $\bot$ to x_1. There are exactly three occurrences of x_1 in the formula and using simplification rules we derive:

     – $(\neg x_1 \wedge y_2 \wedge y_2') \longrightarrow (y_2 \wedge y_2')$

- $(\neg x_1 \wedge \neg y_{n+1}) \longrightarrow (\neg y_{n+1})$
- $(x_1 \wedge \neg y_{n+1}) \longrightarrow ()$

The formula now contains a unit clause $(\neg y_{n+1})$ that is on the top level of the matrix. We use $GU2b$ and are able to completely eliminate (including its quantifier) this variable symbol and replace it by $\top$. After simplifications we get the desired result:

$$\begin{aligned}\omega_{n-1} =& \forall y_2\, \forall y_2'\, \exists x_2\, \forall y_3\, \forall y_3'\, \ldots\, \exists x_{n-1}\, \forall x_n\, \forall x_n'\, \exists x_n\, \forall y_{n+2}\, \ldots\, \forall y_{n+n} \\ & (y_2 \wedge y_2') \vee \\ & (\neg y_2 \wedge x_2 \wedge y_3 \wedge y_3') \vee (\neg y_2' \wedge \neg x_2 \wedge y_3 \wedge y_3') \vee \\ & \vdots \\ & (\neg y_{n-1} \wedge x_{n-1} \wedge y_n \wedge y_n') \wedge (\neg y_{n-1}' \wedge \neg x_{n-1} \wedge y_n \wedge y_n') \vee \\ & (\neg y_n \wedge x_n \wedge y_{n+2} \wedge \ldots \wedge y_{n+n}) \vee \\ & (\neg y_n' \wedge \neg x_n \wedge y_{n+2} \wedge \ldots \wedge y_{n+n}) \vee \\ & (\neg x_2 \wedge \neg y_{n+2}) \vee \ldots \vee (\neg x_n \wedge \neg y_{n+n}) \vee \\ & (x_2 \wedge \neg y_{n+2}) \vee \ldots \vee (x_n \wedge \neg y_{n+n})\end{aligned}$$

- $y_1 \longrightarrow \bot$: $\vdash \forall y_1'\, \ldots\, ((\bot \wedge y_1') \vee (\neg\bot \wedge x_1 \wedge y_2 \wedge y_2') \vee \ldots)$. We apply simplifications and derive $\vdash \forall y_1'\, \ldots\, ((x_1 \wedge y_2 \wedge y_2') \vee (\neg y_1' \wedge \neg x_1 \wedge y_2 \wedge y_2) \vee \ldots)$. Branching over y_1' yields two new branches:
  - $y_1' \longrightarrow \top$: $\vdash \exists x_1\, \ldots\, ((x_1 \wedge y_2 \wedge y_2') \vee (\neg\top \wedge \neg x_1 \wedge y_2 \wedge y_2) \vee \ldots)$ Simplify and apply $\exists_\top$, assigning $\top$ to x_1. The $\exists$-rule changes the formula in three positions:
    - $(x_1 \wedge y_2 \wedge y_2') \longrightarrow (y_2 \wedge y_2')$
    - $(\neg x_1 \wedge \neg y_{n+1}) \longrightarrow ()$
    - $(x_1 \wedge \neg y_{n+1}) \longrightarrow (\neg y_{n+1})$

    The procedure that follows is quite similar to the procedure in the branch that assigned $\top$ to y_1 — we apply $GU2b$ to eliminate the globally unit variable symbol y_{n+1}. After simplifications we again derive ω_{n-1}.
  - $y_1' \longrightarrow \top$: $\vdash \exists x_1\, \ldots\, ((x_1 \wedge y_2 \wedge y_2') \vee (\neg\bot \wedge \neg x_1 \wedge y_2 \wedge y_2) \vee \ldots)$. Simplify and apply $\exists_\top$, assigning $\top$ to x_1. The $\exists_\top$-rule changes the formula in four positions:
    - $(x_1 \wedge y_2 \wedge y_2') \longrightarrow (y_2 \wedge y_2')$
    - $(\neg x_1 \wedge y_2 \wedge y_2') \longrightarrow ()$
    - $(\neg x_1 \wedge \neg y_{n+1}) \longrightarrow ()$
    - $(x_1 \wedge \neg y_{n+1}) \longrightarrow (\neg y_{n+1})$

Use the procedure already known from the previous cases: Apply *GU2b* to eliminate the globally unit variable symbol y_{n+1}. After simplifications we get ω_{n-1} in the last branch.

Summarizing this step, the proof split up into four branches depending on the values assigned to y_1 and y'_1. The branch that assigns $\top$ to both of them can be closed with a short linear proof. The remaining three branches happen to contain the same sequent after a few steps. Thus they can be merges in the following. Figure 11.2 visualizes the proof structure described.

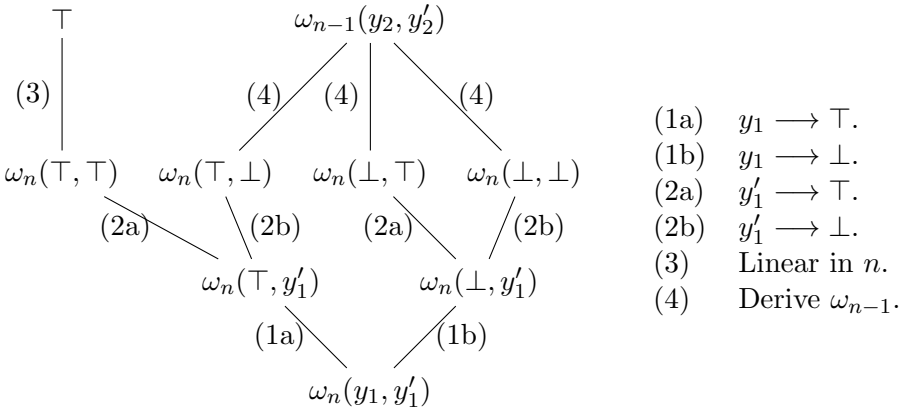


Figure 11.2.: The structure of the first reduction step

This way way we reduced ω_n to ω_{n-1} using a polynomially sized $\mathrm{GQBF}^+_\Diamond$ proof. The positive branch was closed using a polynomial sub-proof. ω_{n-1} has been reached in polynomially many steps in the remaining branches. That totals up to a polynomially proof size. The root of the full proof (of ψ_n) was polynomial too. It thus suffices to find a polynomial proof of ω_{n-1} to find a polynomial proof of ψ_n.

3. Iterate the procedure presented in step 2: Three of four branches — the remaining one is linear and short — in step 2 end in ω_{n-1} which remains to be proven. Thus, if we find a proof for this formula, we can use it to close all open branches. A proof can be found easily if we re-apply the procedure of step 2. The formula ω_{n-1} is thus reduced to ω_{n-2}.

   The newly added sub-proof is polynomial. The full proof (of ψ_n) is polynomial too as we just stack polynomial proofs in a quasi linear manner. We iterate until we reach ω_1.

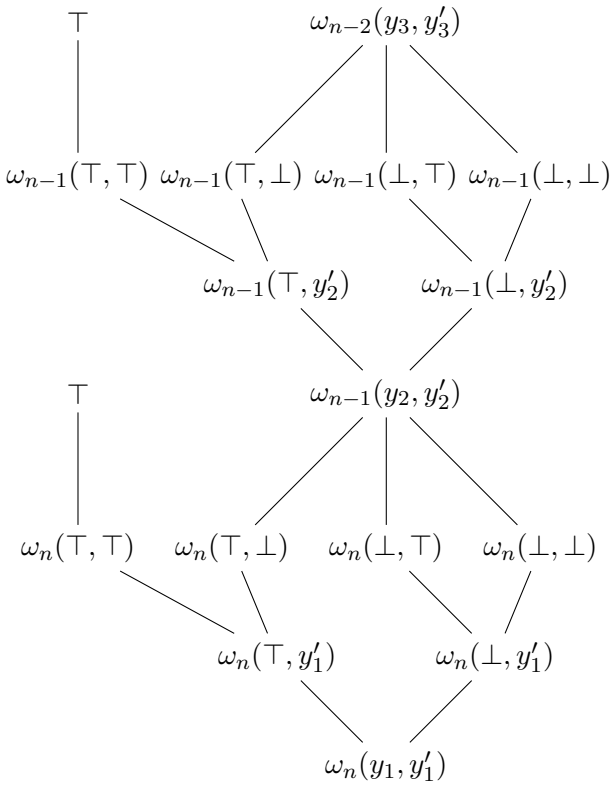


Figure 11.3.: Composing Reduction Steps

4. Close the last open branch:

$$\begin{aligned}\omega_1 =& \forall y_n \, \forall y_n' \, \exists x_n \, \forall y_{n+n} \\ & (y_n \wedge y_n') \vee \\ & (\neg y_n \wedge x_n \wedge y_{n+n}) \vee \\ & (\neg y_n' \wedge \neg x_n \wedge y_{n+n}) \vee \\ & (\neg x_n \wedge \neg y_{n+n}) \vee \\ & (x_n \wedge \neg y_{n+n})\end{aligned}$$

Clearly, if a proof for ω_1 exists, it cannot exceed a size polynomial in n. From $\neg\varphi_n$ is true and the soundness and completeness of $\mathrm{GQBF}^+_\diamond$ we can deduce that a proof indeed exists. For the sake of completeness, a proof can be constructed as follows (simplification rules are applied implicitly):

$$\dfrac{\dfrac{\vdots\,\tau_1}{\vdash \forall y_{n+n}\,(y_{n+n} \vee \neg y_{n+n})}}{\dfrac{\vdash \top \quad \dfrac{\vdash \exists x_n \forall y_{n+n}\,((\neg x_n \wedge y_{n+n}) \vee \ldots)}{\vdash \mathbf{Q}(y'_n \vee (\neg y'_n \wedge \neg x_n \wedge y_{n+n}) \vee \ldots)}\,\forall}{\vdash \omega_1}\,\exists_\bot \qquad \dfrac{\dfrac{\vdots\,\tau_2 \qquad \vdots\,\tau_3}{\vdash \mathbf{Q}(y'_n \vee (x_n \wedge y_{n+n}) \vee \ldots)}\,\forall}{\vdash \omega_1}\,\forall}$$

whereas τ_1 is:

$$\dfrac{\vdash \top \qquad \vdash \top}{\vdash \forall y_{n+n}\,(y_{n+n} \vee \neg y_{n+n})}\,\forall$$

and τ_2 is:

$$\dfrac{\dfrac{\vdots\,\tau_1}{\vdash \forall y_{n+n}\,(y_{n+n} \vee \neg y_{n+n})}}{\vdash \exists x_n \forall y_{n+n}\,(x_n \wedge y_{n+n}) \vee (\neg x_n \wedge \neg y_{n+n}) \vee (x_n \wedge \neg y_{n+n})}\,\exists_\top$$

and τ_3 is:

$$\dfrac{\dfrac{\vdots\,\tau_1}{\vdash \forall y_{n+n}\,(y_{n+n} \vee \neg y_{n+n})}}{\vdash \exists x_n \forall y_{n+n}\,(x_n \wedge y_{n+n}) \vee (\neg x_n \wedge y_{n+n}) \vee (\neg x_n \wedge \neg y_{n+n}) \vee (x_n \wedge \neg y_{n+n})}\,\exists_\top$$

Now we can merge all our steps and create a complete proof of $\neg\varphi$. Its structure can seen in Figure 11.4. Thin lines represent branches of polynomial length whereas thick lines represent multiple polynomially lengthened branches that begin in a common sequent and end in another common sequent. The size of the full proof is surely polynomial.

□

Theorem 11.2.3 (G can effectively p-simulate Q-resolution)**.** There exists a transformation $R\colon \mathcal{QBF} \times \mathbb{N} \to \mathcal{QBF}$ such that for each Q-resolution-refutable formula φ the following holds:

- $\pi\colon \vdash_{\mathrm{R}} \neg\varphi$ be the shortest Q-resolution proof of the invalidity of φ.
- $m \geq |\pi|$
- $R(\varphi, m) \equiv \varphi$
- R runs in a time polynomial in $|\varphi| + m$.
- There exists a polynomial $p \in \mathbf{P}[\mathbb{N}]$ and a G proof τ of $R(\varphi, m)$ such that $|\tau| = p(|\pi|)$.

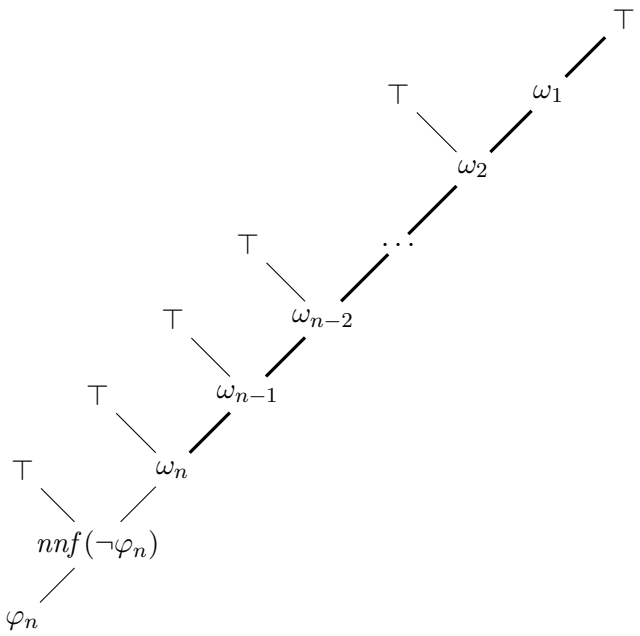


Figure 11.4.: A complete proof of the invalidity of φ_n.

Proof. By Theorem 11.1.5.

□

Equipped with this new knowledge, we can extend the hierarchy of sequent style calculi and augment it with Q-resolution. A graph displaying the relations can be seen in Figure 11.5.

11.3. Simulation between GQBF and 2-PCNF-Evaluation

For a special class of quantified boolean formulas there exists an algorithm that decides the evaluation problem for them in a time linear in the size of the input formula. This class of formulas is the set of quantified boolean formulas in prenex conjunctive normal form that contain only dual clauses. We call a clause dual if it contains exactly two literals. The algorithm is presented in [2]. It works the following way:

- Create a graph representation of the input formula $\varphi = \bigwedge_{i=1}^{n} c_i$: For each variable symbol x that appears in φ create
  - a vertex named x.
  - a vertex named $\neg x$.
  - an edge $(\neg x, x)$.

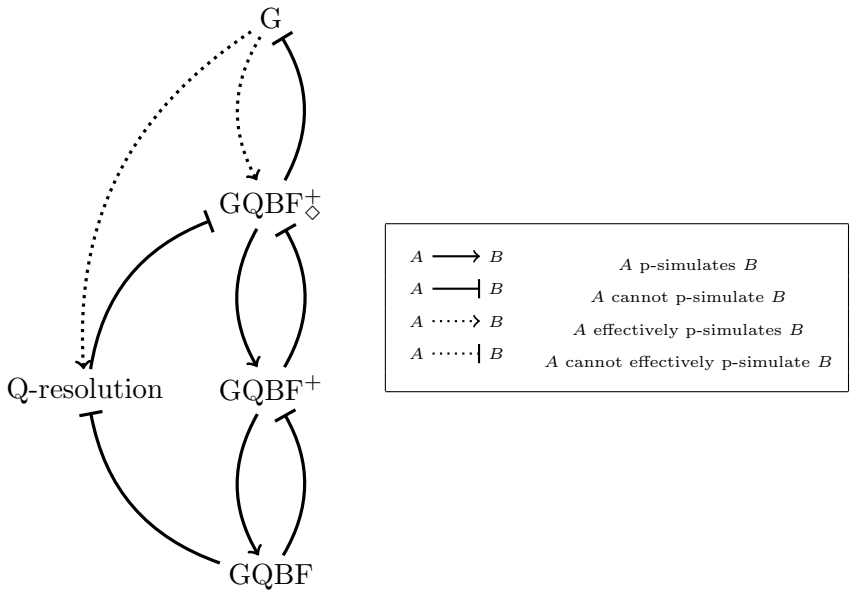
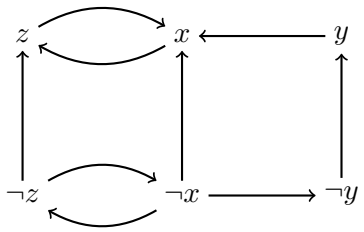


Figure 11.5.: An extended hierarchy of polynomial simulation results.

For each clause $c_i = (\ell_1 \vee \ell_2)$ of φ create

- an edge $(\neg\ell_1, \ell_2)$.
- an edge $(\neg\ell_2, \ell_2)$.

The formula $\forall x\, \exists y\, \forall z\, (x \vee \neg y) \wedge (\neg x \vee z) \wedge (x \vee \neg z)$ thus yields the following graph:



- Partition the graph into its strongly connected components. This can be done in a time liner in the size of the graph. In our example the strongly connected components are $\{x, z\}$, $\{\neg x, \neg z\}$, $\{y\}$ and $\{\neg y\}$.
- The formula φ is true if and only if none of the following conditions hold (Theorem 2 in [2]):
  - An existential vertex v is in the same strong component as its complement $\neg v$.

- A universal vertex u is in the same strong component as an existential vertex e and the variable symbol u is quantified within the scope of the quantifier of e.
- There is a path from a universal vertex u_1 to another universal vertex u_2 or to $\neg u_1$.

In our example the third condition is met. There is a connection from x to z whereas both are universally quantified variable symbols. The formula φ is false.

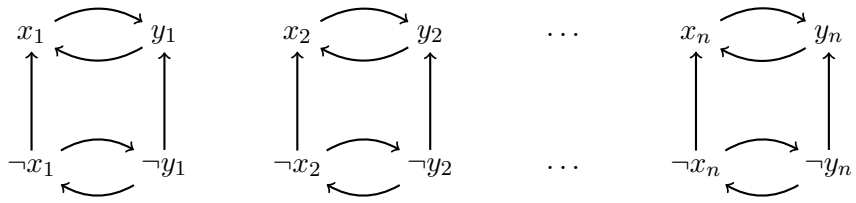
Theorem 11.3.1 (GQBF$^+$ cannot p-simulate 2-PCNF-Evaluation)**.** There is a countably infinite family of formulas such that for members of this family there exists a 2-PCNF-Evaluation that is linear in the size of the formula but there is no GQBF$^+$ proof of polynomial size.

Proof. Let

$$\varphi_n = \forall x_1 \exists y_1 \ldots \forall x_n \exists y_n \left(\psi(x_1, y_1) \wedge \ldots \wedge \psi(x_n, y_n)\right)$$

with $\psi(x, y) = ((x \supset y) \wedge (y \supset x))$ be a closed quantified boolean formula in negation normal form of size $(12n - 1)$. Then, by Lemma 8.1.2, there exists no GQBF$^+$ proof of φ_n of a size polynomial in n.

As φ_n is in prenex conjunctive normal form and each conjunct is a dual clause, we can apply the algorithm presented above to create a proof of φ_n that is linear in n. The following graph represents φ_n:



Clearly, none of the three conditions if violated anywhere. □

Theorem 11.3.2 (G can effectively p-simulate 2-PCNF-Eval)**.**

Proof. By Theorem 11.1.5. □

We can further extend our hierarchy of polynomial simulation results. It can be seen in Figure 11.6.

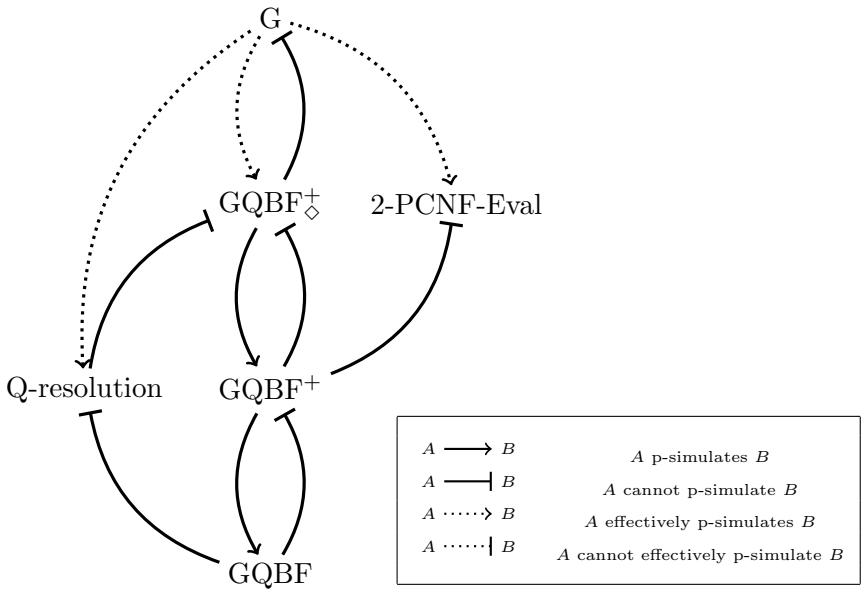


Figure 11.6.: A hierarchy of polynomial simulation results.

Part VII.

Conclusion

12. Conclusion

In previous parts, especially in Part IV and VI, we presented infinite families of formulas for which some calculi can or cannot produce proofs of polynomial size.

When we speak of polynomially provable formulas, we speak of formulas for which a proof of polynomial length with respect to the size of the actual formula exists. Furthermore, when speaking about polynomially provable formulas we always refer to a specific calulus. A formula φ may be polynomially provable with calculus A but not polynomially provable with calulus B. Also note that the complexity of proof search does not always is not always tied to size of a proof. There may exists families of formulas that have very short proofs that are very hard to find in a huge search space. On the other hand there may exist proofs that are large in size but can be found in a time that is nearly identical to their size. In this thesis emphasis is mostly put on the sizes of proofs rather than on the complexity of proof search.

Based on the families from Part IV and VI we established a complete hierarchy of the sequent-style calculi presented: GQBF, GQBF$^+$, GQBF$^+_\diamond$ and G. It was shown that the minimal GQBF calculus is exponentially weaker than the full calculus having simplification rules. The full calculus itself cannot polynomially simulate its exponentially stronger version that allows to construct proofs structured as directed acyclic graphs. It was also shown that G, a sequent calculus with *cut* and structural rules, that produces proofs in the form of directed acylic graphs is exponentially stronger than GQBF$^+_\diamond$. Figure 12.1 presents this hierarchy.

We can subdivide the set of (closed) quantified boolean formulas with respect to their provability using sequent-style calculi. While there are infinite families of formulas that can be proven with the minimal calulus in a short way (e.g., A_n), there are also infinite families that cannot be proven polynomially. Even GQBF$^+_\diamond$ might be too weak to produce short proofs for them. In between there is a whole hierarchy of formulas that require certain features of a calculus to be proven in a short manner. Table 12.1 presents examples for such families for each level of the hierarchy presented in Figure 12.1. It also provides references to the locations within this thesis where their upper and lower bounds of proof size with respect to a certain calculus are proven.

Like we did within the hierarchy of sequent-style calculi, we compared GQBF, GQBF$^+_\diamond$ and G to Q-resolution and established exponential separations, effectively p-simulation results and families of formulas that can be used to separate Q-resolution from other calculi. It was shown that GQBF cannot polynomially

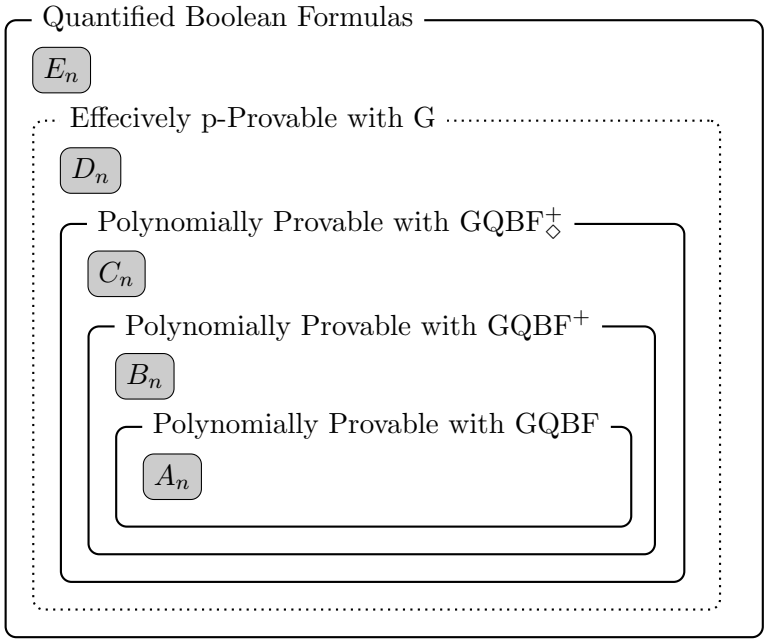


Figure 12.1.: A Hierarchy of Sequent-Style Calculi

simulate Q-resolution whereas Q-resolution itself is exponentially weaker then full GQBF when proofs in the form of directed acyclic graphs are permitted. Again, a collection of these results can be seen in the Figure 12.2. Table 12.2 provides examples for separating families and references to relevant proofs.

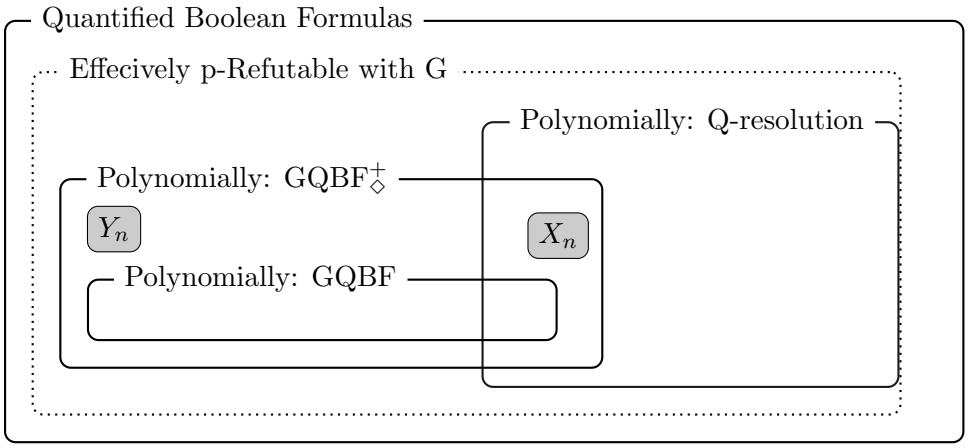


Figure 12.2.: Relations between Sequential Calculi and Q-resolution

Altough we established a lower bound and an upper bound for the features a

Family	Members	Proof of Complexity
A_n	$\exists x_1 \ldots \exists x_n (x_1 \vee \ldots \vee x_n)$	Simple
B_n	$\exists a \forall x_1 \ldots \forall x_n \exists b ((x_1 \vee a \vee b) \wedge \ldots \wedge (x_n \vee a \vee b))$	Theorem 8.1.1
C_n	$\forall x_1 \exists y_1 \ldots \forall x_n \exists y_n (\psi(x_1, y_1) \wedge \ldots \wedge \psi(x_n, y_n))$	Theorem 8.2.1
D_n	$\forall x_1 \exists y_1 \ldots \forall x_n \exists y_n \exists z_1 \exists z'_1 \ldots \exists z_n \exists z'_n$ $(\psi'(x_1, y_1, z_1, z'_1) \wedge \ldots \wedge \psi'(x_n, y_n, z_n, z'_n))$	Theorem 11.1.6
E_n	From [18] it is known that having an optimal calculus is equivalent to NP $\cap$ co-NP having complete languages. As the latter is considered rather unlikely, we can safely assume the existence of an infinite family E_n.	

$\psi(x, y) = (\neg x \vee y) \wedge (\neg y \vee x)$
$\psi'(x, y, a, b) = \psi(x, y) \wedge (x \vee a \vee b) \wedge (\neg x \vee a \vee b) \wedge (a \vee \neg a) \wedge (b \vee \neg b)$

Table 12.1.: Formulas from various levels of the hierarchy

sequent calculus has to offer to be able to polynomially simulate Q-resolution, the exact set of features remains an open question. While G, a full-fledged sequent calculus with a cut rule can indeed effectively p-simulate Q-resolution, it is not researched yet if GQBF$^+_\Diamond$ or even GQBF$^+$ suffice to do so too. It also remains an open question if, in the other direction, Q-resolution can effectively p-simulate GQBF$^+$.

When we were concerned with effective, truth-preserving transformations on quantified boolean formulas, we showed that moving quantifiers outwards, to the root of the structure tree of a formula, increases proof size in general. We showed that there are infinite families of formulas that can be proven polynomially by GQBF while their prenexed versions have only proofs that are exponentially bigger in size. For GQBF and GQBF$^+$ there exist formulas that can be proven polynomially while some of their prenexed equivlanets cannot be proven polynomially with the same calulus. Thus prenexing increases proof size and makes proof search more difficult [6]. Here remains an open questions too: Does there exist a family of formulas for which any possible normal form is at least one level higher than the original formula? We proved that for the lowest level of our hierarchy that is true. No answer can currently be given for higher levels. That implies that a good choice of prenexing strategy is essential to keep proof size low.

Family	**Members**	**Proof of Complexity**
X_n	$\exists x_1 \ldots \exists x_n \forall a \, (\psi''(x_1, a) \wedge \ldots \wedge \psi''(x_n, a))$	Theorem 11.2.1
Y_n	$\exists y_0 \, \exists y_1 \, \exists y_1' \, \forall x_1 \, \exists y_2 \, \exists y_2' \, \forall x_2 \, \exists y_3 \, \exists y_3' \, \ldots$ $\forall x_{n-1} \exists x_n \, \exists x_n' \, \forall x_n \, \exists y_{n+1} \, \exists y_{n+2} \, \ldots \, \exists y_{n+n}$ $(\neg y_0) \wedge$ $(y_0 \vee \neg y_1 \vee \neg y_1') \wedge$ $(y_1 \vee \neg x_1 \vee \neg y_2 \vee \neg y_2') \wedge (y_1' \vee x_1 \vee \neg y_2 \vee \neg y_2') \wedge$ $(y_2 \vee \neg x_2 \vee \neg y_3 \vee \neg y_3') \wedge (y_2' \vee x_2 \vee \neg y_3 \vee \neg y_3') \wedge$ $\vdots$ $(y_{n-1} \vee \neg x_{n-1} \vee \neg y_n \vee \neg y_n') \wedge (y_{n-1}' \vee x_{n-1} \vee$ $\neg y_n \vee \neg y_n') \wedge$ $(y_n \vee \neg x_n \vee \neg y_{n+1} \vee \neg y_{n+2} \vee \ldots \vee \neg y_{n+n}) \wedge$ $(y_n' \vee x_n \vee \neg y_{n+1} \vee \neg y_{n+2} \vee \ldots \vee \neg y_{n+n}) \wedge$ $(x_1 \vee y_{n+1}) \wedge (x_2 \vee y_{n+2}) \wedge \ldots \wedge (x_n \vee y_{n+n}) \wedge$ $(\neg x_1 \vee y_{n+1}) \wedge (\neg x_2 \vee y_{n+2}) \wedge \ldots \wedge (\neg x_n \vee y_{n+n})$	Theorem 11.2.2
$\psi''(x, a) \quad = \quad (a \vee x) \wedge (a \vee \neg x) \wedge (\neg a \vee x) \wedge (\neg a \vee \neg x)$		

Table 12.2.: Separating Formulas

Bibliography

[1] Ansótegui, C., Gomes, C. P., and Selman, B. The Achilles' Heel of QBF. In *AAAI* (2005), pp. 275–281.

[2] Aspvall, B., Plass, M. F., and Tarjan, R. E. A Linear-Time Algorithm for Testing the Truth of Certain Quantified Boolean Formulas. *Inf. Process. Lett. 8*, 3 (1979), 121–123.

[3] Baaz, M., and Leitsch, A. On Skolemization and Proof Complexity. *Fundam. Inform. 20*, 4 (1994), 353–379.

[4] Büning, H. K., Karpinski, M., and Flögel, A. Resolution for Quantified Boolean Formulas. *Inf. Comput. 117*, 1 (1995), 12–18.

[5] Cook, S. A. The complexity of theorem-proving procedures. In *STOC* (1971), pp. 151–158.

[6] Egly, U., Seidl, M., and Woltran, S. A solver for QBFs in negation normal form. *Constraints 14*, 1 (2009), 38–79.

[7] Gasarch, W. I. The P=?NP Poll. *SIGACT NEWS 33*, 2 (2002), 34–47.

[8] Jussila, T., Biere, A., Sinz, C., Kröning, D., and Wintersteiger, C. M. A First Step Towards a Unified Proof Checker for QBF. In *SAT* (2007), pp. 201–214.

[9] Klieber, W., Sapra, S., Gao, S., and Clarke, E. M. A Non-prenex, Non-clausal QBF Solver with Game-State Learning. In *SAT* (2010), pp. 128–142.

[10] Krajíček, J., and Pudlák, P. Quantified propositional calculi and fragments of bounded arithmetic. *Zeitschr. f, math. Logik und Grundlagen d. Math. 36*, 1 (1990), 29–46.

[11] Leitsch, A. *The resolution calculus.* Springer, 1997.

[12] Lonsing, F., and Biere, A. Nenofex: Expanding NNF for QBF Solving. In *SAT* (2008), pp. 196–210.

[13] Narizzano, M., Pulina, L., and Tacchella, A. Report of the Third QBF Solvers Evaluation. *JSAT 2*, 1-4 (2006), 145–164.

[14] Papadimitriou, C. H. *Computational Complexity.* Addison Wesley, 1994.

[15] Peschiera, C., Pulina, L., Tacchella, A., Bubeck, U., Kullmann, O., and Lynce, I. The Seventh QBF Solvers Evaluation (QBFEVAL'10). In *SAT* (2010), pp. 237–250.

[16] Pitassi, T., and Santhanam, R. Effectively polynomial simulations. In *ICS* (2010), pp. 370–382.

[17] Plaisted, D. A., and Greenbaum, S. A Structure-Preserving Clause Form Translation. *J. Symb. Comput. 2*, 3 (1986), 293–304.

[18] Sadowski, Z. On an Optimal Quantified Propositional Proof System and a Complete Language for NP ∩ co-NP. In *FCT* (1997), pp. 423–428.

[19] Stockmeyer, L. J., and Meyer, A. R. Word Problems Requiring Exponential Time: Preliminary Report. In *STOC* (1973), pp. 1–9.

[20] Takeuti, G. *Proof Theory.* Studies in Logic and the Foundations of Mathematics. North-Holland, 1975.

[21] Tseitin, G. On the complexity of derivation in propositional calculus. *Studies in Constructive Mathematics and Mathematical Logic* (1968), 115–125.

[22] Wrathall, C. Complete Sets and the Polynomial-Time Hierarchy. *Theor. Comput. Sci. 3*, 1 (1976), 23–33.