Quantifier Shifting for Quantified Boolean Formulas Revisited*

Simone Heisinger ✉[1] [0009−0000−7630−2791], Maximilian Heisinger[1] [0000−0001−7297−6000], Adrian Rebola-Pardo[1,2] [0000−0001−9234−4377], and Martina Seidl[1] [0000−0002−3267−4494]

[1] Institute for Symbolic Artificial Intelligence, JKU Linz, Austria
{firstname.lastname}@jku.at
[2] Institute for Logic and Computation, TU Vienna, Austria

Abstract. Modern solvers for quantified Boolean formulas (QBFs) process formulas in prenex form, which divides each QBF into two parts: the quantifier prefix and the propositional matrix. While this representation does not cover the full language of QBF, every non-prenex formula can be transformed to an equivalent formula in prenex form. This transformation offers several degrees of freedom and blurs structural information that might be useful for the solvers. In a case study conducted 20 years back, it has been shown that the applied transformation strategy heavily impacts solving time. We revisit this work and investigate how sensitive recent QBF solvers perform w.r.t. various prenexing strategies.

Keywords: Quantified Boolean Formulas · Prenexing · Normal Form Transformation

1 Introduction

Quantified Boolean formulas (QBFs), the extension of propositional formulas with quantifiers over the Boolean variables, have many applications in formal verification, synthesis, and artificial intelligence [28]. Over the last 25 years, many efficient QBF solvers have been developed [2], with clear tendency towards QBFs in prenex conjunctive normal form (PCNF). A QBF in PCNF has the form $Q_1x_1 \dots Q_nx_n.\phi$ where $Q_i \in \{\forall, \exists\}$ and ϕ is a propositional formula in conjunctive normal form. In general, encodings do not result in formulas of this structure, because of recursive definitions in the encoding or from optimizations that try to minimize the scope of variables. Origins for a non-CNF structure can be for example the use of equivalences or xors in the encoding. Therefore two transformations are required: (1) *prenexing* which shifts the quantifiers outside of the formula, and (2) transformation of the quantifier-free formula to CNF. The latter is efficiently achieved by applying the QBF-variant of the well known Tseitin transformation [31] or the optimized Plaisted-Greenbaum transformation [24]. In this work, we focus on the prenexing.

* This work was supported by the LIT AI Lab funded by the state of Upper Austria and by the Vienna Science and Technology Fund (WWTF) [10.47379/VRG11005].

Without loss of generality, formulas can be assumed to be in negation normal form (i.e. negation symbols only occur in front of variables) and cleansed (i.e. no variable occurs both bound and free, and every variable is quantified at most once). Under these conditions, prenexing is achieved by the following two rules:

$$(\mathsf{Q}x.\varphi) \circ \varphi' \Leftrightarrow \mathsf{Q}x.(\varphi \circ \varphi') \qquad \varphi \circ (\mathsf{Q}x.\varphi') \Leftrightarrow \mathsf{Q}x.(\varphi \circ \varphi')$$

with $\mathsf{Q} \in \{\forall, \exists\}$ and $\circ \in \{\wedge, \vee\}$. The formula structure imposes an ordering of the quantifiers based on the subformula relation. Quantifiers from independent parts of a formula can be freely ordered. For example, the formula $\forall x.(\exists y.(y \vee x)) \wedge (\forall z.(z \vee \neg x))$ has prenex forms $\forall x.\exists y.\forall z.\phi$ or $\forall x.\forall z.\exists y.\phi$ where $\phi = (y \vee x) \wedge (z \vee \neg x)$. Hence, a prenex form is not uniquely determined.

Egly et al. [6] suggested four different prenexing strategies that minimize the number of quantifier alternations in the prefix. Empirically, they showed that the selected prenexing strategy impacts solving performance. In this work, we revisit those prenexing strategies and give a concise formalization, which the original work lacked. We show that the original four strategies disambiguate into six unique prenexing strategies when enforcing a minimal number of quantifier alternations, and we present a tool that implements those strategies. To evaluate the impact of prenexing on modern solvers we reimplemented the generator for encoding nested counterfactuals and performed extensive experiments with these formulas and formulas from the QBFEval'08 in which a non-prenex track was organized.

2 Preliminaries

The set of *quantified Boolean formulas* $QF(X)$ over variables X is defined as follows: (1) $\top, \bot, x, \neg x, \neg\top, \neg\bot \in QF(X)$ if $x \in X$, (2) $\varphi \vee \varphi', \varphi \wedge \varphi' \in QF(X)$ if $\varphi, \varphi' \in QF(X)$, (3) if $\varphi \in QF(X)$, then $\forall x.\varphi, \exists x.\varphi \in QF(X)$ with $x \in X$.[3] In QBF $\mathsf{Q}x.\varphi$ with $\mathsf{Q} \in \{\forall, \exists\}$, the subformula φ is called the scope of variable $x \in X$ and x is said to be bound by quantifier Q. If variable x occurs in QBF φ, but φ does neither contain $\exists x$ nor $\forall x$ then x is free in φ. The set of all free variables of a QBF φ is denoted by $\mathit{free}(\varphi)$. A QBF without free variables is called closed. In the following, we assume that each variable is in the scope of at most one quantifier and that each variable occurs either free or bound, but not both in a formula. We call such formulas *cleansed*. The semantics of a QBF φ is defined by the interpretation function $[.]_\sigma \colon QF(X) \to \mathbb{B}$ where $\mathbb{B} = \{\mathbf{1}, \mathbf{0}\}$ and $\sigma \colon \mathit{free}(\varphi) \to \mathbb{B}$ is an assignment to the free variables of φ. Then $[\top]_\sigma = \mathbf{1}$, $[\bot]_\sigma = \mathbf{0}$, for any $x \in X$, $[x]_\sigma = \sigma(x)$ and $[\neg v]_\sigma = 1 - [v]_\sigma$ for $v \in X \cup \{\top, \bot\}$. Furthermore, $[\varphi_1 \wedge \varphi_2]_\sigma = \min\{[\varphi_1]_\sigma, [\varphi_2]_\sigma\}$, and $[\varphi_1 \vee \varphi_2]_\sigma = \max\{[\varphi_1]_\sigma, [\varphi_2]_\sigma\}$. Finally, $[\exists x.\varphi]_\sigma = \max\{\varphi[x|\top], \varphi[x|\bot]\}$ and $[\forall x.\varphi]_\sigma = \min\{\varphi[x|\top], \varphi[x|\bot]\}$ where $\varphi[x|t]$ denotes the QBF obtained by substituting a variable x by a truth constant $t \in \{\top, \bot\}$ in φ. Two QBFs $\varphi, \varphi' \in QF(X)$ are equivalent if $[\varphi]_\sigma = [\varphi']_\sigma$ for any assignment σ.

[3] For simplicity, we assume negations only in front of variables and truth constants.

Definition 1. *The propositional skeleton* φ_{psk} *of QBF* $\varphi \in QF(X)$ *is defined as follows:*

$$\varphi_{\text{psk}} = \begin{cases} x & \text{if } \varphi = x, \quad x \in X \cup \{\top, \bot\} \\ \neg x & \text{if } \varphi = \neg x, \quad x \in X \cup \{\top, \bot\} \\ {\varphi'}_{\text{psk}} \circ {\varphi''}_{\text{psk}} & \text{if } \varphi = \varphi' \circ \varphi'', \ \circ \in \{\wedge, \vee\} \\ {\varphi'}_{\text{psk}} & \text{if } \varphi = \mathsf{Q}V.\varphi', \quad \mathsf{Q} \in \{\forall, \exists\} \end{cases}$$

We say that a QBF φ is of the form $\mathsf{Q}V.\varphi'$ for a set of variables V whenever $\varphi = \mathsf{Q}x_1.\ldots.\mathsf{Q}x_n.\varphi'$ for some enumeration $x_1, \ldots, x_n$ of V. A QBF $\varphi \in QF(X)$ is in prenex form if it is has the structure $\Pi.\phi$ where $\Pi = \mathsf{Q}_1 x_1 \ldots \mathsf{Q}_n x_n$ is a quantifier prefix with $Q_i \in \{\forall, \exists\}$, $x_i \in X$ and $x_i \neq x_j$ for $i \neq j$ and ϕ is a propositional formula. If ϕ is also in conjunctive normal form (CNF), then φ is in *prenex conjunctive normal form* (PCNF). A propositional formula is in CNF if it is a conjunction of clauses. A clause is a disjunction of literals and a literal is a variable or a negated variable. Obviously, $(\Pi.\phi)_{\text{psk}} = \phi$ for a PCNF formula $\Pi.\phi$.

Proposition 1. *Consider QBFs* φ, φ'*, a quantifier* $\mathsf{Q} \in \{\forall, \exists\}$*, a connective* $\circ$*, and variables* x, y*.*

1. *If* $\varphi \circ (\mathsf{Q}x.\varphi')$ *is cleansed, then* $\mathsf{Q}x.(\varphi \circ \varphi')$ *is cleansed, and both formulas are equivalent.*
2. *If* $\mathsf{Q}x.\mathsf{Q}y.\varphi$ *is cleansed, then* $\mathsf{Q}y.\mathsf{Q}x.\varphi$ *is cleansed, and both formulas are equivalent.*

Forests, trees and partial orders. We will oftentimes regard trees and forests as partially ordered sets. In particular, we define a *forest* as set T equipped with a partial ordering $\leq$ such that for all elements $x \in T$, the subset $\{y \in T \mid y \leq x\}$ is totally ordered. When T is finite, this definition appropriately models the recursive concept of a forest: the elements of T are nodes, and $x \leq y$ if y is a descendant of x. We say that x is *covered* by y whenever $x \leq y$ and there is no $z \in T$ with $x < y < z$. When regarding T as a forest, this means that x is the parent of y. A forest T is a tree if, additionally, for any two elements $x, y \in T$ there is another element $z \in T$ with $z \leq x$ and $z \leq y$. For a finite T, this implies that T has a least element, which corresponds to its root.

Given a forest T, we call a list $x_1, \ldots, x_n$ a *path* in T if for all $1 \leq i < j \leq n$ we have $x_i, x_j \in T$ and $x_i < x_j$. The *height* of a forest T is defined as

$$\text{ht}(T) = \max\{n \geq 0 \mid \text{there is a path } x_1, \ldots, x_n \text{ in } T\}.$$

For a node $x \in T$, we define its *lower bounds* as $T^x = \{y \in T \mid y \leq x\}$ and its *upper bounds* as $T_x = \{y \in T \mid y \geq x\}$.

Parity-based functions. We will use a parity-based version of the floor and ceiling functions. Intuitively, $\lfloor n \rfloor_k$ (resp. $\lceil n \rceil_k$) rounds n down (resp. up) to the closest integer with the same parity as k. Formally, for integers $n, k \in \mathbb{Z}$ we define:

$$\lfloor n \rfloor_k = \max\{m \in \mathbb{Z} \mid m \leq n \text{ and } m - k \text{ is even}\}$$
$$\lceil n \rceil_k = \min\{z \in \mathbb{Z} \mid z \geq y \text{ and } m - k \text{ is even}\}$$

Direction-parametric operators. At several points our ordering-based definitions will depend on a *direction* parameter $\dagger \in \{\uparrow, \downarrow\}$. Intuitively, $\uparrow$-labeled operators use a reverse ordering, while $\downarrow$-labeled operators use an unmodified ordering. In particular, we define the following operators:

$\leq^{\uparrow}$	is	$\geq$	$\leq^{\downarrow}$	is	$\leq$	(and similarly for $\geq^{\dagger}$, $<^{\dagger}$, $>^{\dagger}$)
$\min^{\uparrow}$	is	max	$\min^{\downarrow}$	is	min	(and similarly for $\max^{\dagger}$)
$\lfloor \dots \rfloor_k^{\uparrow}$	is	$\lceil \dots \rceil_k$	$\lfloor \dots \rfloor_k^{\downarrow}$	is	$\lfloor \dots \rfloor_k$	(and similarly for $\lceil \dots \rceil_k^{\dagger}$)
$T_x^{\uparrow}$	is	T^x	$T_x^{\downarrow}$	is	T_x	(and similarly for $T_{\dagger}^x$)

3 Related Work

Already 20 years back it has empirically been shown that quantifier shifting has a severe impact on the solving performance [6] of QBF solvers. In this work, four different prenexing strategies were introduced that intuitively result in the smallest number of possible quantifier alternations. The authors noted that the presented strategies "leave room for different [prenexing] variants". In this work, we close this gap by providing a concise formalization of quantifier shifting.

The observation that the prenexing strategy impacts solving performance motivated development of several non-prenex non-CNF solvers [7,15,8,29]. With the rise of efficient preprocessing for PCNF formulas and a focus on applications with few quantifier alternations, however, solver development focused on formulas in prenex form. To deal with the information loss induced by quantifier shifting, solvers were introduced that employ dependency schemes [27] to (re-)discover and exploit variable independencies [16,22], i.e., those solvers recover information on quantifier dependencies that is hidden in the prefix. Reeves et al. presented an approach to move Tseitin variables from the inner-most quantifier block to the outer-most possible position in the quantifier prefix [26]. The exact position is determined by the variables occuring in the formula defined by the Tseitin variable. With this reordering, they observe a considerable speed-up in solver performance. Lonsing and Egly evaluated the impact of the number of quantifier alternations on recent QBF solvers [18]. In their experiments, they established a correlation between different solving paradigms like expansion or QCDCL (see [2] for a detailed discussion of such proof systems) and the number of quantifier alternations. Also, proof-theoretical investigations [1] identify the number of quantifier alternations as source of hardness for practical solving. However, to the best of our knowledge, there is no recent study that investigates the impact of quantifier shifting on the solving behavior of state-of-the-art solvers for formulas in prenex normal form.

Nowadays there is also much interest in dependency quantified Boolean formulas (DQBF) which allow for an explicit specification of quantifier dependencies. The decision problem of these formulas is NEXPTIME-complete [23], in contrast to the PSPACE-completeness of QBFs.

4 Quantifier Shifting

In this work we aim to transform arbitrary QBF formulas φ (which are not in general prenex or in CNF) into equivalent prenex QBF formulas of the form $\mathsf{Q}_1 x_1 . \ldots \mathsf{Q}_n x_n . \varphi_{\text{psk}}$. The formula φ_{psk} is not necessarily in CNF, although this can be easily achieved through the well-known Tseitin procedure.

The method we propose can be roughly summarized as follows. First, a *quantifier tree* reflecting the scope hierarchy of quantifiers in φ is constructed. Each node in this quantifier tree will then be assigned a rank with some restrictions to guarantee soundness; we call this assignment a *linearization*. Finally, the formula $\mathsf{Q}_1 x_1 . \ldots \mathsf{Q}_n x_n . \varphi_{\text{psk}}$ is constructed by enumerating the bound variables $x_1, \ldots, x_n$ by rank; thanks to the restrictions on linearizations, this formula will be equivalent to φ.

Example 1. Throughout our work we will use the following QBF η as a running example:

$$\exists x_1 . x_1 \wedge \Bigg(\Big(\forall y_1 . \ \big(\exists z_1 . (\neg y_1 \vee z_1) \wedge \forall u_1 . \exists v_1 . (y_1 \vee \neg u_1 \vee v_1) \wedge (\neg y_1 \vee u_1 \vee \neg v_1) \big) \wedge \big(\forall z_2 . (\exists u_2 . \neg z_2 \vee u_2) \wedge (\forall u_3 . x_1 \vee z_2 \vee u_3) \big) \Big) \vee \Big(\exists y_4 . (y_4 \wedge \forall z_4 . \exists u_4 . z_4 \wedge u_4) \vee (\neg y_4 \wedge \exists z_5 . \forall u_5 . z_5 \wedge u_5) \Big) \Big)$$

Its propositional skeleton η_{psk} is then given by:

$$x_1 \wedge \Big(\big((\neg y_1 \vee z_1) \wedge (y_1 \vee \neg u_1 \vee v_1) \wedge (y_1 \vee u_1 \vee \neg v_1) \wedge (\neg z_2 \vee u_2) \wedge (x_1 \vee z_2 \vee u_3) \big) \vee (y_4 \wedge z_4 \wedge u_4) \vee (\neg y_4 \wedge z_5 \wedge u_5) \Big)$$

Consider the following two quantifier shifts for η:

$$\eta' = \exists x_1 . \exists y_4 . \exists z_5 . \forall y_1 . \forall z_2 . \forall u_3 . \forall z_4 . \forall u_5 . \exists z_1 . \exists u_2 . \exists u_4 . \forall u_1 . \exists v_1 . \eta_{\text{psk}}$$
$$\eta'' = \exists x_1 . \exists y_4 . \exists z_5 . \exists v_1 . \forall y_1 . \forall z_2 . \forall u_3 . \forall z_4 . \forall u_5 . \exists z_1 . \exists u_2 . \exists u_4 . \forall u_1 . \eta_{\text{psk}}$$

While η' is equivalent to η, the QBF formula η'' is not. The intuitive reason is that in η'' the quantifier $\exists v_1$ has been pushed across quantifier alternation boundaries. This is exactly the situation our formalization will prevent.

Our formalization associates to each QBF a forest obtained by removing from its syntax tree all non-quantifier nodes. The remaining nodes are thus uniquely determined by a bound variable and a quantifier, and this forest contains all the information needed for quantifer shifting. Hence, we first define the abstract

concept of quantifier forests, and then we will show how to construct a quantifier forest from a QBF as above.

A *quantifier forest* is a triple $(T, \leq, q)$ where $(T, \leq)$ is a finite forest regarded as a partially ordered set (see Section 2) and $q : T \to \{\forall, \exists\}$. We call it a *quantifier tree* or *quantree* whenever $(T, \leq)$ is a tree. If $(T, \leq, q)$ is a nonempty quantree, we also define $q^\star(x) = 1$ if $q(x) = q(\min(T))$ and $q^\star(x) = 0$ otherwise. Given a QBF formula φ, its *associated quantifier forest* is a triple $(T_\varphi, \leq_\varphi, q_\varphi)$, where T_φ is the set of bound variables in φ, and $\leq_\varphi$ and q_φ are defined recursively:

- If φ is either $\top$, $\bot$ or $x \in X$, then $\leq_\varphi = \emptyset$ and $q_\varphi = \emptyset$.
- If $\varphi = \neg\varphi_0$ where $\varphi_0 \in X \cup \{\top, \bot\}$, then $\leq_\varphi = \emptyset$ and $q_\varphi = \emptyset$.
- If $\varphi = \varphi_1 \circ \varphi_2$ with $\circ \in \{\wedge, \vee\}$, then $\leq_\varphi = \leq_{\varphi_1} \cup \leq_{\varphi_2}$ and $q_\varphi = q_{\varphi_1} \cup q_{\varphi_2}$.
- If $\varphi = \mathsf{Q}x.\varphi_0$ for a quantifier Q, then $\leq_\varphi = \leq_{\varphi_0} \cup \{(x, y) \mid y \in T_\varphi\}$ and $q_\varphi = q_{\varphi_0} \cup \{(x, \mathsf{Q})\}$.

Proposition 2. *Let $(T_\varphi, \leq_\varphi, q_\varphi)$ be a quantifier forest of QBF φ. If $\varphi = \mathsf{Q}x.\varphi_0$ for a quantifier Q, then $(T_\varphi, \leq_\varphi, q_\varphi)$ is a quantree.*

Example 2. Figure 1 shows the quantree associated to the QBF η from Example 1. In general, we can only guarantee that the quantifier forest associated to a QBF is a tree when the QBF is of the form $\mathsf{Q}x.\varphi$. For example, the quantifier forest associated with $(\forall x.x) \wedge (\exists y.y)$ is a forest with two incomparable elements x and y.

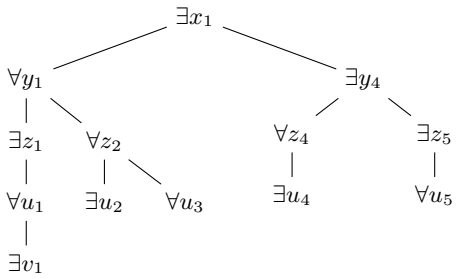
4.1 Linearizations over Quantrees

We now formalize the main object of this paper, namely the different ways the quantifiers in a formula can be rearranged into a quantifier prefix to an equivalent prenex formula. Given a QBF of the form $\mathsf{Q}x.\varphi$ for a quantifier Q, we consider its associated quantree T. We aim to construct an equivalent prenex QBF $\mathsf{Q}^1 V_i \ldots \mathsf{Q}^N V_N.\varphi_{\text{psk}}$ where $\mathsf{Q}^i \neq \mathsf{Q}^{i+1}$ for $1 \leq i < N$. To do so, each node in T (i.e. each bound variable in $\mathsf{Q}x.\varphi$) must be mapped to a single quantifier block $\mathsf{Q}^i V_i$. We call this i its *rank*. However, as shown in Example 1, assigning arbitrary ranks is unsound (i.e. the obtained prenex QBF is not equivalent to $\mathsf{Q}x.\varphi$). We show how bound variables can be ranked while preserving soundness.

Let us consider an arbitrary quantree T. A map $f : T \to \{1, \ldots, N\}$ for some $N \geq 0$ is called a *linearization* if:

1. $f(x) \leq f(y)$ for all quantree nodes $x, y \in T$ with $x \leq y$.
2. For all quantree nodes $x \in T$, $f(x)$ is odd if and only if $q^\star(x) = 1$.

Consider now a QBF of the form $\mathsf{Q}y.\varphi$ where Q is a quantifier and y is a variable, and its associated quantree $(T, \leq, q)$. In this case, since T is the set of bound variables in $\mathsf{Q}y.\varphi$, a linearization $f : T \to \{1, \ldots, N\}$ maps each bound variable $x \in T$ to an integer $f(x)$ we call its *rank*. A QBF ψ is called a *prenexation* of $\mathsf{Q}y.\varphi$ via f if ψ is of the form $\mathsf{Q}^1 V_1 \ldots \mathsf{Q}^N V_N.\varphi_{\text{psk}}$ where $V_i = \{x \in T \mid f(x) = i\}$ and $\mathsf{Q}^i = \mathsf{Q}$ (resp. $\overline{\mathsf{Q}}$) if i is odd (resp. even) for $1 \leq i \leq N$.



	$\exists$	$\forall$	$\exists$	$\forall$	$\exists$	
f	$f(\dots)=1$	$f(\dots)=2$	$f(\dots)=3$	$f(\dots)=4$	$f(\dots)=5$	strategy
f_1	$x_1y_4z_5$	$y_1z_2u_3z_4u_5$	$z_1u_2u_4$	u_1	v_1	$\forall\uparrow\uparrow/\exists\uparrow\uparrow$
f_2	$x_1y_4z_5$	$y_1z_2u_3z_4u_5$	z_1	u_1	$v_1u_2u_4$	$\forall\uparrow\downarrow$
f_3	x_1	$y_1z_2u_3$	$z_1y_4z_5$	$u_1z_4u_5$	$v_1u_2u_4$	$\exists\downarrow\uparrow$
f_4	$x_1y_4z_5$	y_1	z_1	$u_1z_2u_3z_4u_5$	$v_1u_2u_4$	$\forall\downarrow\uparrow$
f_5	$x_1y_4z_5$	$y_1z_2z_4$	$z_1u_2u_4$	$u_1u_3u_5$	v_1	$\exists\uparrow\downarrow$
f_6	x_1	y_1	$z_1y_4z_5$	$u_1z_2u_3z_4u_5$	$v_1u_2u_4$	$\forall\downarrow\downarrow/\exists\downarrow\downarrow$

Fig. 1. Above, the quantree associated to the formula η from Example 1. Below, optimal linearizations for this quantree for each strategy. In each column, the variables mapped to each rank are shown; the quantifier of each block appears in the header. Note that the optimal linearizations for strategies $\mathsf{Q}\uparrow\uparrow$ and $\mathsf{Q}\uparrow\downarrow$ assign the same rank to Q-quantified variables; this is a consequence of Lemma 2.

Theorem 1. *Let T be the quantree associated to a QBF of the form $\mathsf{Q}y.\varphi$. Consider a prenexation ψ of $\mathsf{Q}y.\varphi$ via some linearization $f : T \to \{1,\dots,N\}$. Then $\mathsf{Q}y.\varphi$ is equivalent to ψ.*

To guarantee this form $\mathsf{Q}y.\varphi$ for an arbitrary QBF φ, we can simply introduce a fresh variable y that does not occur in φ. Obviously, φ is equivalent to $\mathsf{Q}y.\varphi$.

Example 3. Figure 1 shows six linearizations for the quantree associated to the QBF η from Example 1 and Example 2. In that example, the quantifier shift η' is the prenexation of η via the linearization f_1. Note that the mapping f that would produce η'' is not a linearization, since that would violate Theorem 1. In particular, $u_1 \leq v_1$ but $f(v_1) < f(u_1)$.

4.2 Alternation Height of Quantrees

So far we have not shown that linearizations even exist. Given the theoretical and empirical impact of the number of quantifier alternations on QBF solving, we are not just interested in their existence, but rather on linearizations that minimize the maximum rank N. We will now show how to compute the minimal value of N for which linearizations exist; in fact, this value will be extremely useful to extend the ideas from [6] to arbitrary QBFs in Section 5.2.

Consider an arbitrary quantree T, and a path $x_1, \ldots, x_n$ in T. We call this path *alternating* whenever $q(x_i) \neq q(x_{i+1})$ for $1 \leq i < n$. Then we can define the *alternation height* of T as

$$\mathrm{aht}(T) = \max\{n \geq 0 \mid \text{there is an alternating path } x_1, \ldots, x_n \text{ in } T\}.$$

Intuitively, the alternation height of T is the height of the tree that results from "clumping" together all adjacent nodes with the same quantifier. It then becomes apparent that any linearization f over T must have $N \geq \mathrm{aht}(T)$, since for any alternating path $x_1, \ldots, x_n$ we have $f(x_1) < \cdots < f(x_n)$. The following result shows that this lower bound can indeed be realized:

Theorem 2. *Let T be a quantree. Then, a linearization $f : T \to \{1, \ldots, \mathrm{aht}(T)\}$ exists. Furthermore, there exists no linearization $g : T \to \{1, \ldots, N\}$ such that $0 \leq N < \mathrm{aht}(T)$.*

Observe that the number of quantifier alternations in a prenexation via a linearization grows with the value N. In the following sections, we will restrict our scope to linearizations that minimize this value, i.e. linearizations in the set

$$\mathrm{Lin}(T) = \{f : T \to \{1, \ldots, N\} \mid f \text{ is a linearization and } N = \mathrm{aht}(T)\}.$$

5 Linearization Strategies

We now follow up on the ideas from [6] and formalize them. In particular, we aim to obtain a formal definition of when does a linearization follow a given strategy, to ascertain whether strategies determine a unique linearization for each quantree, and to find simple algorithms to compute this linearization.

5.1 Strategies as Preferences over Linearizations

Here we take a non-constructive approach. For each prenexing strategy, we define a preference relation between the linearizations in $\mathrm{Lin}(T)$; linearizations that follow the strategy "better" than others are preferred. As we will show, this induces a strategy-based partial order between linearizations. The desired output of a strategy must then be a maximal element w.r.t. this partial order.

The strategies from [6] are based on the idea of pushing quantifiers of a given polarity as high or as low as possible in the quantifier hierarchy. This lends itself to a natural definition of preference.

Consider an arbitrary quantree T. Given a direction $\dagger \in \{\uparrow, \downarrow\}$ and a quantifier Q, we define the *semi-preference* relation $\precsim^{\mathsf{Q}\dagger}$ over $\mathrm{Lin}(T)$ given by $f \precsim^{\mathsf{Q}\dagger} g$ iff $f(x) \leq^{\dagger} g(x)$ for all $x \in T$ with $q(x) = \mathsf{Q}$. In other words, g is preferred over f whenever g assigns ranks to Q-nodes further in the direction $\dagger$ than f does.

Example 4. Consider the linearizations f_1, f_2 and f_6 from Figure 1. All universal variables are assigned lower ranks by both f_1 and f_2 than by f_6, so $f_6 \precsim^{\forall\downarrow} f_1$ and $f_6 \precsim^{\forall\downarrow} f_2$ hold. Note also that f_1 and f_2 assign the same ranks to universal variables, so both $f_1 \precsim^{\forall\downarrow} f_2$ and $f_2 \precsim^{\forall\downarrow} f_1$ hold. Note that this does not imply $f_1 = f_2$.

Example 4 shows that the antisymmetric property of partial orders does not hold for $\precsim^{\mathsf{Q}\dagger}$. This is intuitive: two linearizations might be just as good as each other regarding Q-nodes, while wildly differing for other nodes. Hence, for strategies to uniquely determine linearizations we need to provide preferences for both quantifiers. While this was proposed by [6], it was also noted there that uniqueness is not attained.

Example 5. The linearizations f_2 and f_3 from Figure 1 are both good candidates for linearizations for the strategy $\exists^{\downarrow}\forall^{\uparrow}$ from [6]: both assign high ranks to existential variables and low ranks to universal variables. However, there is no apparent criterion why f_2 should or should not be preferred to f_3 under that strategy.

We solve this problem by giving one quantifier priority over the other. Our *strategies* are of the form $\mathsf{Q}\dagger\ddagger$, where Q is a quantifier and $\dagger, \ddagger$ are directions. A good linearization under this strategy pushes quantifiers Q in the $\dagger$ direction, and quantifiers $\overline{\mathsf{Q}}$ in the $\ddagger$ direction; when in conflict, the former should prevail.

To formalize this idea, we liberally borrow from the somewhat similar notion of lexicographic orderings. Let us define $f \approx^{\mathsf{Q}} g$ whenever $f(x) = g(x)$ for all $x \in T$ with $q(x) = \mathsf{Q}$. In other words, $f \approx^{\mathsf{Q}} g$ holds whenever both $f \precsim^{\mathsf{Q}\dagger} g$ and $g \precsim^{\mathsf{Q}\dagger} f$ hold, regardless of the choice of direction $\dagger$. We define the *preference* relation $\preccurlyeq^{\mathsf{Q}\dagger\ddagger}$ over $\mathrm{Lin}(T)$ given by $f \preccurlyeq^{\mathsf{Q}\dagger\ddagger} g$ iff $f \precsim^{\mathsf{Q}\dagger} g$ holds, and whenever $f \approx^{\mathsf{Q}} g$ holds then $f \precsim^{\overline{\mathsf{Q}}\ddagger} g$ holds as well.

Proposition 3. *$\preccurlyeq^{\mathsf{Q}\dagger\ddagger}$ defines a partial order on $\mathrm{Lin}(T)$.*

It is easy to check that $\preccurlyeq^{\forall\dagger\dagger}$ and $\preccurlyeq^{\exists\dagger\dagger}$ are the same preference relation for $\dagger \in \{\uparrow, \downarrow\}$. Hence, our approach defines 6 unique strategies, while [6] only proposed 4 strategies. On the one hand, the $\exists^{\dagger}\forall^{\dagger}$ strategy from [6] corresponds to our $\exists\dagger\dagger$ (or, equivalently, $\forall\dagger\dagger$) strategy. On the other hand, our strategies $\exists\dagger\ddagger$ and $\forall\ddagger\dagger$ with $\dagger \neq \ddagger$ are both covered by the $\exists^{\dagger}\forall^{\ddagger}$ strategy from [6], which is not uniquely determined.

Example 6. Although we cannot yet convince the reader of this, the linearizations given in Figure 1 are the maximum element of $\mathrm{Lin}(T)$ for each of the six (unique) preference orderings $\preccurlyeq^{\mathsf{Q}\dagger\ddagger}$; the corresponding strategy is shown in the rightmost column. For now, we can foreshadow that strategies $\mathsf{Q}\dagger\uparrow$ and $\mathsf{Q}\dagger\downarrow$ assign the same ranks to Q-nodes. As shown later in Lemma 1, this holds in general.

5.2 Optimal Linearizations over a Strategy

Proposition 3 suggests this is a good direction to formalize the idea of strategies: since $\mathrm{Lin}(T)$ is finite, there exist optimal linearizations w.r.t. the preference ordering $\preccurlyeq^{\mathsf{Q}\dagger\ddagger}$. We call these linearizations $\mathsf{Q}\dagger\ddagger$-*optimal*; linearizing a quantree T through the strategy $\mathsf{Q}\dagger\ddagger$ then means computing a $\mathsf{Q}\dagger\ddagger$-optimal linearization in $\mathrm{Lin}(T)$.

Some hurdles remain unsolved, though. For one, we have not determined if $\mathsf{Q}\dagger\ddagger$-optimal linearizations are unique (i.e. if maximal elements w.r.t. $\preccurlyeq^{\mathsf{Q}\dagger\ddagger}$ are maxima as well). This is of interest because to empirically test the performance effect of quantifier shifting strategies, the outcome of applying a strategy must be reproducible at worst, and uniquely determined by definition at best. A second issue is a consequence of our non-constructive approach: we are yet to provide a procedure that computes a $\mathsf{Q}\dagger\ddagger$-optimal linearization for a given quantree.

The rest of this section is devoted to computing a closed-form expression for $\mathsf{Q}\dagger\ddagger$-optimal linearizations. Since this expression is deterministic and computable, this solves both aforementioned issues.

Overview. As we mentioned above, $\preccurlyeq^{\mathsf{Q}\dagger\ddagger}$ is somewhat similar to a lexicographic ordering in two components where the first component is ordered by $\precsim^{\mathsf{Q}\dagger}$ and the second component is ordered by $\precsim^{\overline{\mathsf{Q}}\ddagger}$. We exploit this intuition to construct $\mathsf{Q}\dagger\ddagger$-optimal linearizations: we will first optimize the first component (in our case, the ranks of Q-nodes), and then optimize the second component (the ranks of $\overline{\mathsf{Q}}$-nodes) while keeping the first component fixed.

To optimize the first component, we find a linearization $\Gamma_\dagger$, defined below in (1), that is optimal for $\precsim^{\mathsf{Q}\dagger}$. This is more precisely expressed in Lemma 1: $\Gamma_\dagger$ pushes Q-nodes further in direction $\dagger$ than any other linearization.

Interestingly, $\Gamma_\dagger$ does not depend on Q: $\Gamma_\dagger$ actually optimizes *all* nodes in the $\dagger$ direction. The second part of our method optimizes the ranks assigned to $\overline{\mathsf{Q}}$-nodes in the $\ddagger$ direction while keeping Q-nodes constant. For a general linearization f, this procedure results in a new linearization $[f]^{\mathsf{Q}\ddagger}$ defined below in (2). Lemma 2 shows that $[f]^{\mathsf{Q}\ddagger}$ is optimal for $\preccurlyeq^{\mathsf{Q}\dagger\ddagger}$ among the linearizations that assign the same ranks as f to Q-nodes. These two results are combined in Theorem 3: the unique $\mathsf{Q}\dagger\ddagger$-maximal linearization is $[\Gamma_\dagger]^{\mathsf{Q}\ddagger}$.

Theoretical results. Let us consider a quantree $(T, \leq, q)$ and a strategy $\mathsf{Q}\dagger\ddagger$. We define the mapping $\Gamma_\dagger : T \to \{1, \ldots, \mathrm{aht}(T)\}$ given by

$$\Gamma_\dagger(x) = \left\lfloor |\max^\dagger\{1, \ldots, \mathrm{aht}(T)\} - \mathrm{aht}(T^\dagger_x)| + 1 \right\rfloor^\dagger_{q^\star(x)}. \tag{1}$$

Furthermore, we define the mapping $[f]^{\mathsf{Q}\ddagger} : T \to \{1, \ldots, \mathrm{aht}(T)\}$ for $f \in \mathrm{Lin}(T)$ given by

$$[f]^{\mathsf{Q}\ddagger}(x) = \left\lfloor \min^\ddagger\{f(y) \mid y \in T^\ddagger_x \text{ and } q(y) = \mathsf{Q}\} \right\rfloor^\ddagger_{q^\star(x)}. \tag{2}$$

In (2), $\min^\ddagger$ is taken over a subset of $\{1, \ldots, \mathrm{aht}(T)\}$; we follow the convention that $\min^\ddagger(\emptyset) = \max^\ddagger\{1, \ldots, \mathrm{aht}(T)\}$.

Lemma 1. *$\Gamma_\dagger \in \mathrm{Lin}(T)$. Furthermore, for any $g \in \mathrm{Lin}(T)$, we have $g \precsim^{\mathsf{Q}\dagger} \Gamma_\dagger$.*

Lemma 2. *$[f]^{\mathsf{Q}\ddagger} \in \mathrm{Lin}(T)$ for all $f \in \mathrm{Lin}(T)$. Furthermore, $[f]^{\mathsf{Q}\ddagger} \approx^{\mathsf{Q}} f$, and for any $g \in \mathrm{Lin}(T)$ with $g \approx^{\mathsf{Q}} f$, we have $g \preccurlyeq^{\mathsf{Q}\dagger\ddagger} f$.*

Theorem 3. *Let $f \in \mathrm{Lin}(T)$ be a $\mathsf{Q}\dagger\ddagger$-optimal linearization. Then, $f = [\Gamma_\dagger]^{\mathsf{Q}\ddagger}$. In particular, $[\Gamma_\dagger]^{\mathsf{Q}\ddagger}$ is the maximum element in $(\mathrm{Lin}(T), \preccurlyeq^{\mathsf{Q}\dagger\ddagger})$.*

Example 7. Let us check that $[\Gamma_\downarrow]^{\exists\uparrow}$ is indeed f_3 for the quantree in Figure 1 for a few values. First note that $[\Gamma_\downarrow]^{\exists\uparrow}$ only depends on the values of $\Gamma_\downarrow$ for existential nodes, so we only need to compute these. In this case, $\max^\dagger\{1,\dots,\mathrm{aht}(T)\} = \mathrm{aht}(T) = 5$, $q^\star(x) = 1$, and $\mathrm{aht}(T_x^\downarrow) = \mathrm{aht}(T_x)$ is simply the maximum number of quantifier alternations below x. $\Gamma_\downarrow$ respects the tree ordering, so we obtain

$$[\Gamma_\downarrow]^{\exists\uparrow}(z_1) = \Gamma_\downarrow(z_1) = \lfloor |5 - \mathrm{aht}(T_{z_1})| + 1 \rfloor_1 = \lfloor 4 \rfloor_1 = 3 = f_3(z_1).$$

Furthermore, we can compute $[\Gamma_\downarrow]^{\exists\uparrow}(u_1)$ by checking only $\Gamma_\downarrow(z_1)$, since z_1 realizes the $\min^\uparrow$ operator in (2). Then,

$$[\Gamma_\downarrow]^{\exists\uparrow}(u_1) = \lceil \Gamma_\downarrow(z_1) \rceil_{q^\star(u_1)} = \lceil 3 \rceil_0 = 4 = f_3(u_1).$$

Example 7 suggests that $[f]^{\mathsf{Q}\ddagger}$ can be computed recursively. Indeed, the rank of a node can be computed based on the ranks of its children or parent.

Corollary 1. *Let $x \in T$ such that $q(x) \neq \mathsf{Q}$. Then,*

$$[f]^{\mathsf{Q}\ddagger}(x) = \left\lfloor \min^\ddagger\{[f]^{\mathsf{Q}\ddagger}(y) \mid x \text{ is covered by } y \in T \text{ w.r.t. } \leq^\ddagger\}. \right\rfloor^\ddagger_{q^\star(x)}$$

6 Implementation and Evaluation

We implemented the optimal linearization $[\Gamma_\dagger]^{\mathsf{Q}\ddagger}$ for each strategy $\mathsf{Q}\dagger\ddagger$ described in Section 5. Our implementation uses the BOOLEGURU framework [10], designed for efficiently working with propositional formulas and QBFs. BOOLEGURU provides a convenient parsing and serialization infrastructure for widely used formats, as well as helper functions to write formula transformations. Our extension is licensed under the MIT license and publicly available[4].

Our implementation computes a quantifier shift on an input QBF φ based on a strategy $\mathsf{Q}\dagger\ddagger$ by traversing twice the abstract syntax tree of the parsed QBF φ in a depth-first fashion. In the first pass, the propositional skeleton φ_{psk} and the quantree T are extracted. Furthermore, the values $\mathrm{aht}(T^x)$ and $\mathrm{aht}(T_x)$, which we call *height* and *depth* of x, are computed for each node $x \in T$.

The second pass is applied only to the quantree. For each node $x \in T$, we compute its rank $[\Gamma_\dagger]^{\mathsf{Q}\ddagger}(x)$. For Q-nodes, this rank is given by $\Gamma_\dagger(x)$, which is trivial to compute from the height and depth of x; for $\overline{\mathsf{Q}}$-nodes, Corollary 1 allows a recursive computation. Based on their rank, quantifier nodes in the quantree are collected in a quantifier prefix which is appended to φ_{psk}.

To apply a linearization strategy to an arbitrary formula, BOOLEGURU needs to be called with the options `:linearize-quants-{E,A}{up,down}-{up,down}` using the quantifier `E` ($\exists$) or `A` ($\forall$) and the two directions `up` ($\uparrow$) and `down` ($\downarrow$). Overall, there are eight different combinations that we evaluate in the following. However, from the discussion above, it becomes obvious that only six of those

[4] https://github.com/maximaximal/booleguru

eight strategies are different. In the implementation, all quantifiers are first extracted from an expression and processed in a separate tree. Each node contains the quantifier type, the quantified variables, and dependent quantifier nodes.

After computing the linearization, the extracted quantifiers are inserted piecewise as new expressions that wrap the originally transformed expression. This ensures the ordering of variables within the quantifier blocks stays the same. The fully quantified expression is then returned from the transformer and can either be printed using one of BOOLEGURU's serializers, or processed further.

6.1 Benchmarks

As most solvers only process formulas in prenex (conjunctive) normal form, hardly any non-prenex benchmarks are currently available. To test our implementation, we considered the benchmark set from the QBFEval 2008 and we reimplemented a generator for nested counterfactuals as described below. All used formulas and corresponding experimental logs are available at [11].

Nested Counterfactuals. We developed a novel generator for nested counterfactuals (NCFs) based on a Lua script, which is integrated into BOOLEGURU. The full encoding is described in [6]. To generate NCFs, five arguments must be provided: numbers of formulas in the background theory, numbers of variables, clauses per formula, variables per clause, nesting depth. Optionally, a sixth argument to fix the seed value for random choices. A counterfactual $\phi > \psi$ is true over a background theory T iff the minimal change of T to incorporate ϕ entails ψ. In a nested counterfactual, also ϕ or ψ are allowed to be (nested) counterfactuals. For details see [6]. We chose the range of arguments based on the description mentioned in Egly et al. [6]. We assume that the background theory T always consists of 5 randomly generated formulas. Each of these formulas consists of 2 to 10 clauses where each clause is a disjunction of 3 variables. The clauses contain randomly chosen atoms from a set of 5 variables. These atoms have a 50 percent chance of being negated. No clause may contain the same literal more than once and the clauses are non-tautological. The nesting depth of the counterfactuals ranges from 2 to 6. All possible combinations of these selected parameters result in 45 different classes. For each of these classes, 100 instances were generated to ensure that both, satisfiable and unsatisfiable results are represented. With the 8 strategies we obtain 36 000 prenexed formulas either in the non-CNF QCIR format or in QDIMACS.

Non-Prenex-Non-CNF Benchmarks from QBFEval 2008. In the QBFEval 2008, a non-prenex, non-CNF track was organized [19]. The benchmarks are available at the QBFLib.[5]. This set consists of 492 formulas in the outdated Boole format. To transform these formulas into prenex form, we first rewrote them into the related Limboole[6] format that is processable by BOOLEGURU. Again, we considered all eight options resulting in 4936 prenexed formulas.

[5] http://www.qbflib.org
[6] http://fmv.jku.at/limboole/

Table 1. Number of solved formulas per strategy and solver of QBFEval'08 set (QCIR). Diff indicates the difference between the best and the worst strategy. Each strategy has 492 formulas.

Solver	∃↑↑	∃↓↓	∃↑↓	∃↓↑	∀↑↑	∀↓↓	∀↑↓	∀↓↑	Diff.	Rel. diff. (%)
QuAbS	380	398	405	366	380	400	376	**406**	40	8.13
QFUN	340	**409**	360	339	340	407	341	361	70	14.23
CQESTO	436	**451**	430	442	436	450	447	429	22	4.47
Qute	455	464	**465**	452	455	464	454	**465**	13	2.64

6.2 Experimental Setup

All experiments were run with a timeout of 15 minutes on a cluster of dual-socket AMD EPYC 7313 @ 3.7 GHz machines running Ubuntu 22.04 with a 8 GB memory limit per task. We split the experiments into two parts: on the one hand, we consider solvers that process formulas in prenex conjunctive normal form (PCNF) and on the other hand, we consider solvers that process formulas in prenex non-CNF. For the first group of solvers that accept formulas in the QDIMACS format, we consider the following solvers: The solver DepQBF (version 6.03) is a conflict/solution driven clause/cube learning (QCDCL) solver that integrates several advanced inprocessing techniques and reasoning under the standard dependency scheme [17]. Also Qute [21] is a QCDCL solver that employs dynamic dependency learning. This solver is also able to process QCIR formulas, i.e., it is also included in the second group. The solver CAQE [25] (version 4.0.2) implements clausal selection. The solver RAReQS [14] (version 1.1) implements variable expansion in CEGAR style. Finally, dynQBF [4] [5] (version 1.1.1) is a BDD-based solver. For pre-processing, we used Bloqqer [3] and HQSpre [32]. For testing the encodings in the non-CNF QCIR format, we include the solvers QuAbS [9,30] and CQESTO [13] (version v00.0) (*sic*) that lift clausal selection to circuits, QFUN [12] (version v00.0) (*sic*), a solver that employs machine learning techniques, and Qute which was already mentioned above.

6.3 Experimental Results

In the following, we first discuss the results of the solvers that process formulas in QCIR, (i.e. formulas in prenex form but not in CNF). Second, we report on our experiments with QDIMACS formulas for the PCNF solvers.

Prenexed formulas in QCIR. The nested counterfactual benchmarks were easily solved by QCIR solvers, i.e., they could exploit the formula structure to quickly solve these formulas (all were solved in less than a second). Therefore, we focus on the formulas of the QBFEval'08 benchmark set in the following. Table 6.3 shows the results for the QCIR solvers and table 6.3 shows the number of different prefixes that were generated with all strategy combinations. For

Table 2. Number of different prefixes generated from of the 2008 non-CNF benchmark set with all strategy combinations. Each strategy has 492 formulas.

∀ \ ∃	↓↓	↓↑	↑↓	↑↑
↓↓	0	400	380	400
↓↑	380	400	0	400
↑↓	400	110	400	270
↑↑	400	380	400	0

Table 3. Number of solved formulas per strategy and solver of NCFs. Diff indicates the difference between the best and the worst strategy. Each strategy has 4500 formulas.

Solver	∃↑↑	∃↓↓	∃↑↓	∃↓↑	∀↑↑	∀↓↓	∀↑↓	∀↓↑	Diff.	Rel. diff. (%)
DepQBF	**4500**	4495	4497	**4500**	**4500**	4495	**4500**	4497	5	0.11
CAQE	37	86	**88**	37	37	86	37	**88**	51	1.13
RAReQS	**21**	12	19	16	**21**	12	16	20	9	0.2
Qute	**1012**	731	724	1010	**1012**	731	1010	724	288	6.4
dynQBF	4274	4456	4318	4467	4279	4469	**4474**	4316	200	4.44

QuAbS, QFUN, and CQESTO we see a clear difference between the best and worst shifting strategy. In contrast, Qute seems to be less sensitive regarding the prenexing, which might be related to its dynamic dependency learning. The detailed solving behavior of QFUN and CQESTO is shown in Figure 6.3. For QFUN we observe that ∃↑↑ and ∀↑↑ clearly perform best, while ∀↓↑ and ∃↑↓ seem to be less beneficial.

Prenexed formulas in QDIMACS. Table 3 shows the results of the QDIMACS solvers on the encodings of the nested counterfactuals and table 6.3 shows the number of different prefixes that were generated between all strategy combinations. DepQBF solves all formulas from 4 of the 8 strategies and most of the others, dynQBF is able to solve most of the formulas and Qute solves about

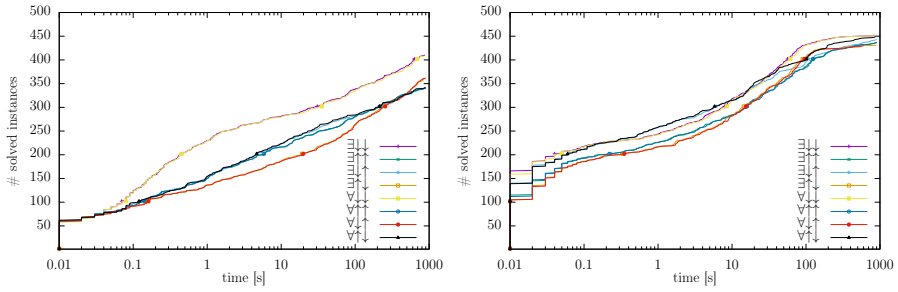


Fig. 2. Solving time of the QBFEVAL'08 set with QFUN (left) and CQESTO (right).

Table 4. Number of different prefixes generated from of the NCF benchmark set with all strategy combinations. Each strategy has 4500 formulas.

∀ \ ∃	↓↓	↓↑	↑↓	↑↑
↓↓	0	4500	4500	4500
↓↑	4500	4500	0	4500
↑↓	4500	0	4500	4500
↑↑	4500	4500	4500	0

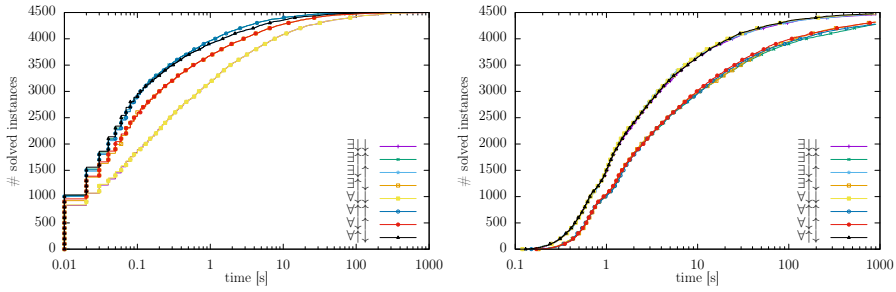


Fig. 3. Solving time of nested counterfactuals with DepQBF (left) and Qute (right).

one quarter of the formulas. Meanwhile RAReQS and CAQE hardly solve any of those. These could be connected with the observation that those solvers perform better on formulas with few quantifier alternations. For all solvers we observe that the chosen shifting strategy impacts the number of the solved formulas. Details of the runs of DepQBF and dynQBF are shown in Figure 3. For DepQBF, we observe that strategies ∃↓↓ and ∀↓↓ are clearly less preferable than strategies ∃↑↑ and ∀↑↑, while dynQBF prefers to have existential quantifiers shifted down. The QBFEval'08 benchmarks are very challenging for recent QDIMACS solvers with our encoding. Out of the 492 formulas, DepQBF solves up to 128 formulas with the best strategy (∀↓↑). dynQBF solves around 60 formulas. The other tools solve less than 30 formulas. Enabling preprocessing is beneficial for all solvers. When preprocessors Bloqqer or HQSpre simplify the formulas, then almost all formulas can be solved. With and without preprocessing, the shifting strategies have only little impact on this benchmark set. Note that more than two third of these formulas have five or less quantifier alternations.

7 Conclusion & Future Work

This paper analyzes and extends previous work from 2003 on quantifier shifting for quantified Boolean formulas. Since then, much progress has been made in the development of QBF solvers by introducing novel solving paradigms, applying efficient preprocessing techniques, and exploiting quantifier (in-)dependence.

However, most of those approaches assume formulas in prenex normal form. As a consequence, most encodings are provided in this form, which unnecessarily restricts solvers with a certain design choice. In this work, we not only formalized prenexing in a concise manner, but we also provide an efficient, publicly available tool that implements the discussed prenexing strategies and Tseitin transformation. In extensive experiments with state-of-the-art prenex CNF and non-CNF solvers, we showed that in many instances prenexing strategy selection impacts solving runtime. We showed that different solvers perform differently on different strategies, hence it was not possible to uniquely identify the best strategy. Therefore, we think it is important that solver developers and also the developers of QBF encodings exploit information available in the problem structure and do not introduce artificial restrictions.

In future work, we plan to design and evaluate further prenexing strategies and we will also revisit more non-prenex QBF encodings to obtain larger benchmark sets. At the moment, hardly any formulas in non-prenex form are available which we changed by providing the generator for encodings of nested counterfactuals. But this is a first step only. Many of the considered formulas are either too hard or too easy for recent solvers, hence more effort is necessary to obtain a larger variety of interesting benchmarks (also in the light of next QBF evaluations). Finally, we want to explore how prenexing strategies affect the generation of certificates and solutions in terms of Herbrand and Skolem functions. From first-order logic, it is well known that it is beneficial to move quantifiers as far inwards as possible to minimize the arity of the first-order Skolem functions [20].

References

1. Beyersdorff, O., Hinde, L., Pich, J.: Reasons for hardness in QBF proof systems. ACM Trans. Comput. Theory **12**(2), 10:1–10:27 (2020). https://doi.org/10.1145/3378665
2. Beyersdorff, O., Janota, M., Lonsing, F., Seidl, M.: Quantified boolean formulas. In: Handbook of Satisfiability - Second Edition, Frontiers in Artificial Intelligence and Applications, vol. 336, pp. 1177–1221. IOS Press (2021). https://doi.org/10.3233/FAIA201015
3. Biere, A., Lonsing, F., Seidl, M.: Blocked clause elimination for QBF. In: Automated Deduction - CADE-23 - 23rd International Conference on Automated Deduction, Wroclaw, Poland, July 31 - August 5, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6803, pp. 101–115. Springer (2011). https://doi.org/10.1007/978-3-642-22438-6_10
4. Charwat, G., Woltran, S.: Bdd-based dynamic programming on tree decompositions. Tech. rep., Technical report, Technische Universität Wien, Institut für Informationssysteme (2016)
5. Charwat, G., Woltran, S.: Dynamic programming-based QBF solving. In: Proceedings of the 4th International Workshop on Quantified Boolean Formulas (QBF 2016) co-located with 19th International Conference on Theory and Applications of Satisfiability Testing (SAT 2016), Bordeaux, France, July 4, 2016. CEUR Workshop Proceedings, vol. 1719, pp. 27–40. CEUR-WS.org (2016)

6. Egly, U., Seidl, M., Tompits, H., Woltran, S., Zolda, M.: Comparing different prenexing strategies for quantified boolean formulas. In: Theory and Applications of Satisfiability Testing, 6th International Conference, SAT 2003. Santa Margherita Ligure, Italy, May 5-8, 2003 Selected Revised Papers. Lecture Notes in Computer Science, vol. 2919, pp. 214–228. Springer (2003). https://doi.org/10.1007/978-3-540-24605-3_17
7. Egly, U., Seidl, M., Woltran, S.: A solver for qbfs in nonprenex form. In: ECAI 2006, 17th European Conference on Artificial Intelligence, August 29 - September 1, 2006, Riva del Garda, Italy, Including Prestigious Applications of Intelligent Systems (PAIS 2006), Proceedings. Frontiers in Artificial Intelligence and Applications, vol. 141, pp. 477–481. IOS Press (2006)
8. Goultiaeva, A., Bacchus, F.: Exploiting circuit representations in QBF solving. In: Theory and Applications of Satisfiability Testing - SAT 2010, 13th International Conference, SAT 2010, Edinburgh, UK, July 11-14, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6175, pp. 333–339. Springer (2010). https://doi.org/10.1007/978-3-642-14186-7_29
9. Hecking-Harbusch, J., Tentrup, L.: Solving QBF by abstraction. In: Proceedings Ninth International Symposium on Games, Automata, Logics, and Formal Verification, GandALF 2018, Saarbrücken, Germany, 26-28th September 2018. EPTCS, vol. 277, pp. 88–102 (2018). https://doi.org/10.4204/EPTCS.277.7
10. Heisinger, M., Heisinger, S., Seidl, M.: Booleguru, the propositional polyglot. In: Benzmüller, C., Heule, M., Schmidt, R. (eds.) Automated Reasoning - 12th International Joint Conference, IJCAR 2024, Nancy, France, July 3-6, 2024, Proceedings. Lecture Notes in Computer Science, vol. TODO @EDITOR, p. TODO The authors thank you very much for this and we are sorry for the inconvenience. Springer (2024). https://doi.org/TODO, TODOPleaseinsertthecorrectreferencetoIJCAR2024forBooleguru!
11. Heisinger, S., Heisinger, M., Rebola-Pardo, A., Seidl, M.: Artifact for "quantifier shifting for quantified boolean formulas revisited" (2024). https://doi.org/10.5281/zenodo.10634925
12. Janota, M.: QFUN: towards machine learning in QBF. CoRR **abs/1710.02198** (2017)
13. Janota, M.: Circuit-based search space pruning in QBF. In: Theory and Applications of Satisfiability Testing - SAT 2018 - 21st International Conference, SAT 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 9-12, 2018, Proceedings. Lecture Notes in Computer Science, vol. 10929, pp. 187–198. Springer (2018). https://doi.org/10.1007/978-3-319-94144-8_12
14. Janota, M., Klieber, W., Marques-Silva, J., Clarke, E.M.: Solving QBF with counterexample guided refinement. In: Theory and Applications of Satisfiability Testing - SAT 2012 - 15th International Conference, Trento, Italy, June 17-20, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7317, pp. 114–128. Springer (2012). https://doi.org/10.1007/978-3-642-31612-8_10
15. Klieber, W., Sapra, S., Gao, S., Clarke, E.M.: A non-prenex, non-clausal QBF solver with game-state learning. In: Theory and Applications of Satisfiability Testing - SAT 2010, 13th International Conference, SAT 2010, Edinburgh, UK, July 11-14, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6175, pp. 128–142. Springer (2010). https://doi.org/10.1007/978-3-642-14186-7_12
16. Lonsing, F., Biere, A.: Integrating dependency schemes in search-based QBF solvers. In: Theory and Applications of Satisfiability Testing - SAT 2010, 13th International Conference, SAT 2010, Edinburgh, UK, July 11-14, 2010. Proceed-

ings. Lecture Notes in Computer Science, vol. 6175, pp. 158–171. Springer (2010). https://doi.org/10.1007/978-3-642-14186-7_14
17. Lonsing, F., Egly, U.: Depqbf 6.0: A search-based QBF solver beyond traditional QCDCL. In: Automated Deduction - CADE 26 - 26th International Conference on Automated Deduction, Gothenburg, Sweden, August 6-11, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10395, pp. 371–384. Springer (2017). https://doi.org/10.1007/978-3-319-63046-5_23
18. Lonsing, F., Egly, U.: Evaluating QBF solvers: Quantifier alternations matter. In: Principles and Practice of Constraint Programming - 24th International Conference, CP 2018, Lille, France, August 27-31, 2018, Proceedings. Lecture Notes in Computer Science, vol. 11008, pp. 276–294. Springer (2018). https://doi.org/10.1007/978-3-319-98334-9_19
19. Marin, P., Narizzano, M., Pulina, L., Tacchella, A., Giunchiglia, E.: Twelve years of QBF evaluations: QSAT is PSPACE-hard and it shows. Fundam. Informaticae **149**(1-2), 133–158 (2016). https://doi.org/10.3233/FI-2016-1445
20. Nonnengart, A., Weidenbach, C.: Computing small clause normal forms. In: Handbook of Automated Reasoning (in 2 volumes), pp. 335–367. Elsevier and MIT Press (2001). https://doi.org/10.1016/B978-044450813-3/50008-4
21. Peitl, T., Slivovsky, F., Szeider, S.: Dependency learning for QBF. In: Theory and Applications of Satisfiability Testing - SAT 2017 - 20th International Conference, Melbourne, VIC, Australia, August 28 - September 1, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10491, pp. 298–313. Springer (2017). https://doi.org/10.1007/978-3-319-66263-3_19
22. Peitl, T., Slivovsky, F., Szeider, S.: Dependency learning for QBF. J. Artif. Intell. Res. **65**, 180–208 (2019). https://doi.org/10.1613/jair.1.11529
23. Peterson, G., Reif, J., Azhar, S.: Lower bounds for multiplayer noncooperative games of incomplete information. Computers & Mathematics with Applications **41**(7-8), 957–992 (2001)
24. Plaisted, D.A., Greenbaum, S.: A structure-preserving clause form translation. J. Symb. Comput. **2**(3), 293–304 (1986). https://doi.org/10.1016/S0747-7171(86)80028-1
25. Rabe, M.N., Tentrup, L.: CAQE: A certifying QBF solver. In: Formal Methods in Computer-Aided Design, FMCAD 2015, Austin, Texas, USA, September 27-30, 2015. pp. 136–143. IEEE (2015). https://doi.org/10.1109/FMCAD.2015.7542263
26. Reeves, J.E., Heule, M.J.H., Bryant, R.E.: Moving definition variables in quantified boolean formulas. In: Tools and Algorithms for the Construction and Analysis of Systems - 28th International Conference, TACAS 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany, April 2-7, 2022, Proceedings, Part I. Lecture Notes in Computer Science, vol. 13243, pp. 462–479. Springer (2022). https://doi.org/10.1007/978-3-030-99524-9_26
27. Samer, M., Szeider, S.: Backdoor sets of quantified boolean formulas. J. Autom. Reason. **42**(1), 77–97 (2009). https://doi.org/10.1007/s10817-008-9114-5
28. Shukla, A., Biere, A., Pulina, L., Seidl, M.: A survey on applications of quantified boolean formulas. In: 31st IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2019, Portland, OR, USA, November 4-6, 2019. pp. 78–84. IEEE (2019). https://doi.org/10.1109/ICTAI.2019.00020
29. Tentrup, L.: Non-prenex QBF solving using abstraction. In: Theory and Applications of Satisfiability Testing - SAT 2016 - 19th International Conference, Bordeaux, France, July 5-8, 2016, Proceedings. Lecture Notes in Com-

puter Science, vol. 9710, pp. 393–401. Springer (2016). https://doi.org/10.1007/978-3-319-40970-2_24
30. Tentrup, L.: Non-prenex QBF solving using abstraction. In: Theory and Applications of Satisfiability Testing - SAT 2016 - 19th International Conference, Bordeaux, France, July 5-8, 2016, Proceedings. Lecture Notes in Computer Science, vol. 9710, pp. 393–401. Springer (2016). https://doi.org/10.1007/978-3-319-40970-2_24
31. Tseitin, G.S.: On the complexity of derivation in propositional calculus. In: Automation of Reasoning: 2: Classical Papers on Computational Logic 1967–1970, pp. 466–483. Springer Berlin Heidelberg (1983). https://doi.org/10.1007/978-3-642-81955-1_28
32. Wimmer, R., Reimer, S., Marin, P., Becker, B.: Hqspre - an effective preprocessor for QBF and DQBF. In: Tools and Algorithms for the Construction and Analysis of Systems - 23rd International Conference, TACAS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10205, pp. 373–390 (2017). https://doi.org/10.1007/978-3-662-54577-5_21

In this appendix we provide formal claims for all the results in our work. We also provide proofs for the results that are not straightforward.

In order to be as precise as possible, we first formalize the assumptions on QBFs given in Section 2. We assume an infinite set of variables. Given sets of variables X and B, we recursively define the set of *quantified Boolean formulas (QBF)* with free variables from X and bound variables from B, denoted by $\mathrm{QF}(X, B)$ as follows:

- $\top, \bot \in \mathrm{QF}(\emptyset, \emptyset)$.
- For all variables x we have $x, \neg x \in \mathrm{QF}(\{x\}, \emptyset)$.
- For all $\varphi_1 \in \mathrm{QF}(X_1, B_1)$ and $\varphi_2 \in \mathrm{QF}(X_2, B_2)$ such that $X_1 \cap B_2 = B_1 \cap X_2 \cap B_1 \cap B_2 = \emptyset$, and for all connectives $\circ \in \{\wedge, \vee\}$, we have $\varphi_1 \circ \varphi_2 \in \mathrm{QF}(X_1 \cup X_2, B_1 \cup B_2)$.
- For all $\varphi_0 \in \mathrm{QF}(X_0, B_0)$, for all variables $x \notin B_0$, and for all quantifiers $\mathsf{Q} \in \{\forall, \exists\}$, we have $\mathsf{Q}x.\varphi_0 \in \mathrm{QF}(X_0 \setminus \{x\}, B_0 \cup \{x\})$.

We say that φ is a QBF whenever $\varphi \in \mathrm{QF}(X, B)$ for some sets of variables X and B.

Proposition 1. *Let X, B be two sets of variables. Consider QBFs φ, φ', a quantifier Q, a connective $\circ$, and variables x, y.*

1. *If $\varphi \in \mathrm{QF}(X, B)$, then $X \cap B = \emptyset$.*
2. *For all sets of variables X', B', if $\varphi \in \mathrm{QF}(X, B)$ and $\varphi \in \mathrm{QF}(X', B')$, then $X = X'$ and $B = B'$.*
3. *If $\varphi \circ \varphi' \in \mathrm{QF}(X, B)$, then $\varphi' \circ \varphi \in \mathrm{QF}(X, B)$, and $\varphi \circ \varphi'$ is equivalent to $\varphi' \circ \varphi$.*
4. *If $\varphi \circ (\mathsf{Q}x.\varphi') \in \mathrm{QF}(X, B)$, then $\mathsf{Q}x.\varphi \circ \varphi' \in \mathrm{QF}(X, B)$, and $\varphi \circ (\mathsf{Q}x.\varphi')$ is equivalent to $\mathsf{Q}x.\varphi \circ \varphi'$.*
5. *If $\mathsf{Q}x.\mathsf{Q}y.\varphi \in \mathrm{QF}(X, B)$, then $\mathsf{Q}y.\mathsf{Q}x.\varphi \in \mathrm{QF}(X, B)$, and $\mathsf{Q}x.\mathsf{Q}y.\varphi$ is equivalent to $\mathsf{Q}y.\mathsf{Q}x.\varphi$.*

6. *If* $\varphi \in \mathrm{QF}(X, B)$ *and* $x \notin X$*, then* $\mathsf{Q}x.\varphi$ *is equivalent to* φ*.*
7. *If* $\varphi \in \mathrm{QF}(X, B)$*, then* $\varphi_{\mathrm{psk}} \in \mathrm{QF}(X', \emptyset)$ *for some set of variables* X' *with* $X \subseteq X' \subseteq X \cup B$*.*
8. *If* $\varphi \in \mathrm{QF}(X, \emptyset)$*, then* φ_{psk} *is equivalent to* φ*.*

Proposition 2. *Consider sets of variables* X, B *and a QBF* $\varphi \in \mathrm{QF}(X, B)$*. Then,* $(T_\varphi, \leq_\varphi, q_\varphi)$ *is a quantifier forest. Furthermore, if* $\varphi = \mathsf{Q}x.\varphi_0$ *for a quantifier* Q*, then* $(T_\varphi, \leq_\varphi, q_\varphi)$ *is a quantree.*

Theorem 1. *Let* $\varphi \in \mathrm{QF}(X, B)$ *for sets of variables* X *and* B*, and let* m *be a variable with* $m \notin B$*. Consider the quantree* $(T, \leq, q)$ *associated to* $\mathsf{Q}m.\varphi$*. If a QBF* ψ *is a prenexation of* $\mathsf{Q}m.\varphi$ *via some linearization* $f : T \to \{1, \ldots, N\}$*, then* $\mathsf{Q}m.\varphi$ *is equivalent to* ψ*.*

Proof. First we prove the case when $B = \emptyset$. Then, $\varphi_{\mathrm{psk}} = \varphi$ by Proposition 1. Furthermore, we know that $T = \{m\}$ because $\mathsf{Q}m.\varphi \in \mathrm{QF}(X \setminus \{m\}, \{m\})$. Let us call $k = f(m)$, which is odd because $m = \min(T)$. Then observe that $\{x \in T \mid f(x) = i\}$ is empty for each $1 \leq i \leq N$ with $i \neq k$; and $\{x \in T \mid f(x) = k\} = \{m\}$. Because k is odd, we then conclude $\psi = \mathsf{Q}m.\varphi_{\mathrm{psk}} = \mathsf{Q}m.\varphi$, so in particular ψ is equivalent to $\mathsf{Q}m.\varphi$.

We now show the general case by structural induction over φ.

- If $\varphi = \top$ or $\varphi = \bot$, then $\varphi \in \mathrm{QF}(\emptyset, \emptyset)$; this falls within the special case $B = \emptyset$.
- If $\varphi = x$ or $\varphi = \neg x$ for a variable x, then $\varphi \in \mathrm{QF}(\{x\}, \emptyset)$; this is covered by the special case $B = \emptyset$, too.
- If $\varphi = \varphi_1 \circ \varphi_2$ for a connective $\circ$, then we consider sets of variables X, B, X_1, B_1, X_2, B_2 where $\varphi \in \mathrm{QF}(X, B)$, $\varphi_1 \in \mathrm{QF}(X_1, B_1)$ and $\varphi_2 \in \mathrm{QF}(X_2, B_2)$ and the following hold:

  $$X = X_1 \cup X_2 \qquad B = B_1 \cup B_2 \qquad X_1 \cap B_2 = X_2 \cap B_1 = B_1 \cap B_2 = \emptyset$$

  Choose distinct variables m_1, m_2 with m_1, $m_2 \notin X \cup B \cup \{m\}$. Then, the following hold:

  $$\begin{aligned} \mathsf{Q}m.\varphi &\in \mathrm{QF}(X \setminus \{m\}, B \cup \{m\}) \\ \mathsf{Q}m_1.\varphi_1 &\in \mathrm{QF}(X_1, B_1 \cup \{m_1\}) \\ \mathsf{Q}m_2.\varphi_2 &\in \mathrm{QF}(X_2, B_2 \cup \{m_2\}) \end{aligned}$$

  Let $(T_s, \leq_s, q_s)$ be the quantree associated to $\mathsf{Q}m_s.\varphi_s$ for $1 \leq s \leq 2$. Then, $T = B \cup \{m\}$ and $T_s = B_s \cup \{m_s\}$. Let us define $k = f(m)$ and $B_i = \{x \in B \mid f(x) = i\}$ for each $1 \leq i \leq N$. Since $m = \min(T)$ and f is a linearization, k is odd and $f(x) \geq k$ for all $x \in T$. This implies that for all $1 \leq i \leq N$:

  $$\{x \in T \mid f(x) = i\} = \begin{cases} \emptyset & \text{if } 1 \leq i < k \\ B_k \cup \{m\} & \text{if } i = k \\ B_k & \text{if } k < i \leq N \end{cases}$$

Then, by Proposition 1, ψ is equivalent to a QBF of the form $\mathsf{Q}m.\mathsf{Q}^k B_k.\ldots.\mathsf{Q}^N V_N.\varphi_{\text{psk}}$.
Now choose distinct variables m_1, m_2 with m_1, $m_2 \notin X \cup B \cup \{m\}$. Then, $\mathsf{Q}m_s.\varphi_s \in \text{QF}(X_s, B_s \cup \{m_s\})$ for $1 \leq s \leq 2$, so we can consider the quantree $(T_s, \leq_s, q_s)$ associated to $\mathsf{Q}m_s.\varphi_s$. We now construct a linearization $f_s : T_s \to \{1, \ldots, N\}$ and a QBF ψ_s such that ψ_s is a prenexation of $\mathsf{Q}m_s.\varphi_s$ via f_s.
First note that $T_s = B_s \cup \{m_s\}$, and $B_s \subseteq B$. We can then define the map f_s with $f_s(x) = f(x)$ when $x \in B_s$ and $f_s(m_s) = k$. Observe that, on the one hand, $q_s(m_s) = q(m)$ and $m_s = \min(T_s)$; and on the other hand, for all $x, y \in B_s$ we have $q_s(x) = q(x)$ and $x \leq_s y$ iff $x \leq y$. From these equalities and f being a linearization over T, it easily follows that f_s is a linearization over T_s.
Let us call $B_s^i = B_i \cap B_s$ for $1 \leq i \leq N$. Then we have:

$$\begin{aligned} \{x \in T_s \mid f_s(x) = i\} &= B_i \cap B_s = \emptyset, \quad \text{for } 1 \leq i < k \\ \{x \in T_s \mid f_s(x) = k\} &= (B_k \cap B_s) \cup \{m_s\} \\ V_i &= B_i \cap B_s, \quad \text{for } k < i \leq N \end{aligned}$$

We can then build a QBF ψ_s of the form $\mathsf{Q}m_s.\mathsf{Q}^k B_s^k.\ldots.\mathsf{Q}^N B_s^N.\varphi_{s_{\text{psk}}}$. Then, $\psi_s \in \text{QF}(X_s, B_s \cup \{m_s\})$, and ψ_s is a prenexation of $\mathsf{Q}m_s.\varphi_s$ via f_s. By induction hypothesis, ψ_s is equivalent to $\mathsf{Q}m_s.\varphi_s$.
By repeatedly applying Proposition 1, $\psi_1 \circ \psi_2$ is then equivalent to a QBF of the form:

$$\mathsf{Q}m_1.\mathsf{Q}m_2.\mathsf{Q}^k(B_1^k \cup B_2^k).\ldots.\mathsf{Q}^N(B_1^N \cup B_2^N).\varphi_{1_{\text{psk}}} \circ \varphi_{2_{\text{psk}}}$$

Now observe that $\varphi_{1_{\text{psk}}} \circ \varphi_{2_{\text{psk}}} = \varphi_{\text{psk}}$, that $B_1^i \cup B_2^i = B_i$ for all $1 \leq i \leq N$, and that $m_1, m_2 \notin X$. Thus, $\psi_1 \circ \psi_2$ is equivalent to a QBF of the form:

$$\mathsf{Q}^k B_k.\ldots.\mathsf{Q}^N B_N.\varphi_{\text{psk}}$$

In other words, we have shown that ψ is equivalent to $\mathsf{Q}m.(\psi_1 \circ \psi_2)$, which is itself equivalent to $\mathsf{Q}m.(\varphi_1 \circ \varphi_2)$.

– If $\varphi = \mathsf{Q}_0 x \varphi_0$ for a quantifier Q_0 and a variable x, then $x \in B = T \setminus \{m\}$. Furthermore, $\varphi_0 \in \text{QF}(X_0, B_0)$ where $X \subseteq X_0 \subseteq X \cup \{x\}$ and $B_0 = B \setminus \{x\}$. Let $k = f(m)$ and $h = f(x)$. Since f is a linearization and $m = \min(T)$, we conclude that $k \leq h$. Furthermore, for all $y \in B_0 = T \setminus \{m, x\}$ we have $x \leq y$, and since f is a linearization, $h \leq f(y)$. These findings show that $\{y \in T \mid f(y) = i\}$ is empty for $1 \leq i < k$ and $k < i < h$, and so ψ is equivalent to a QBF of the form

$$\mathsf{Q}m.\mathsf{Q}^h x.\mathsf{Q}^h V_h.\ldots.\mathsf{Q}^N V_N.\varphi_{\text{psk}}$$

where $V_i = \{y \in B_0 \mid f(y) = i\}$ for $1 \leq i \leq N$.
Now consider the quantree $(T_0, \leq_0, q_0)$ associated to $\mathsf{Q}_0 x.\varphi_0$. Then, $T_0 = T \setminus \{m\}$, and for all $y, z \in T_0$ we have $q_0(y) = q(y)$, and $y \leq_0 z$ iff $y \leq z$. In particular, $x = \min(T_0)$, and $1 \leq f(y) \leq N - h + 1$. The map $f_0 : T_0 \to 1, \ldots, N - h + 1$ defined by $f_0(y) = f(y) - h + 1$ is then well-defined.

Let us check that f_0 is a linearization over T_0. First, for all $y, z \in T_0$ with $y \leq z$ we have $f(y) \leq f(z)$, so $f_0(y) = f(y)+(1-h) \leq f(z)+(1-h) = f_0(z)$ holds. Furthermore, for all $y \in T_0$, $f_0(y)$ is odd iff $f(y) - h + 1$ is odd iff $f(y) - f(x)$ is even iff $q(y) = q(x)$, where we have used $x = \min(T_0)$. Hence, f_0 is a linearization over T_0. Furthermore, the following equalities are easy to check:

$$\{y \in T_0 \mid f_0(y) = 1\} = V_h \cup \{x\}$$
$$\{y \in T_0 \mid f_0(y) = i\} = V_{h+i-1} \text{ , for } 1 < i \leq N - h + 1$$

We can then build a QBF ψ_0 of the form $\mathsf{Q}_0 x.{\mathsf{Q}_0}^1 V_h. \ldots .{\mathsf{Q}_0}^{N-h+1} V_N.{\varphi_0}_{\text{psk}}$. Then, $\psi_0 \in \text{QF}(X_0, B_0)$, and ψ_0 is a prenexation of $\mathsf{Q}x.\varphi_0$ via f_0. By induction hypothesis, φ_0 is equivalent to ψ_0.
Now let $1 \leq i \leq N - h + 1$. On the one hand, ${\mathsf{Q}_0}^i = \mathsf{Q}_0$ iff i is odd, and $\mathsf{Q}_0 = \mathsf{Q}$ iff h is odd. Hence, ${\mathsf{Q}_0}^i = \mathsf{Q}$ iff $i - h$ is odd. On the other hand, $\mathsf{Q}^{i+h-1} = \mathsf{Q}$ iff $i+h-1$ is odd. Now, $(i-h)-(i+h-1) = -2h+1$ is odd, so $i-h$ is odd iff $i+h-1$ is odd. We thus have shown ${\mathsf{Q}_0}^i = \mathsf{Q}^{i+h-1}$. Then, ψ_0 is of the form $\mathsf{Q}^h x.\mathsf{Q}^h V_h. \ldots \mathsf{Q}^N V_N.{\varphi_0}_{\text{psk}}$ Finally, observe that ${\varphi_0}_{\text{psk}} = \varphi_{\text{psk}}$. We then conclude that ψ is equivalent to $\mathsf{Q}m.\psi_0$, which in turn is equivalent to $\mathsf{Q}m.\varphi_0 = \varphi$.

Theorem 2. *Let $(T, \leq, q)$ be a quantree. Then, a linearization $f : T \to \{1, \ldots, \text{aht}(T)\}$ exists. Furthermore, there exists no linearization $g : T \to \{1, \ldots, N\}$ such that $0 \leq N < \text{aht}(T)$.*

Proof. First we show that for any linearization $g : T \to \{1, \ldots, N\}$ we have $N \geq \text{aht}(T)$. Let $x_1, \ldots, x_n$ be an alternating path in T with $n = \text{aht}(T)$. For any $1 \leq i < N$, we then have $x_i \leq x_{i+1}$, so $g(x_i) \leq g(x_{i+1})$. Furthermore, $q(x_i) \neq q(x_{i+1})$, which implies that $g(x_i)$ is odd iff $q(x_i) = q(\min(T))$ iff $q(x_{i+1}) \neq q(\min(T))$ iff $g(x_{i+1})$ is even. This shows that $g(x_i) \neq g(x_{i+1})$. We then conclude that $g(x_1) < \cdots < g(x_n)$, so in particular $\{1, \ldots, N\}$ has at least n elements. Hence, $N \geq n = \text{aht}(T)$.

Now we prove that there exists a linearization $f : T \to \{1, \ldots, \text{aht}(T)\}$ by induction on $\text{ht}(T)$. The base case for $\text{ht}(T) = 0$ is trivial, since then T is empty and $\text{aht}(T) = 0$. Now assume that such a linearization exists whenever $\text{ht}(T) \leq k$ for some $k \geq 0$; we show that it also exists whenever $\text{ht}(T) = k + 1$.

Let $m = \min(T)$, and define:

$$S = \{s \in T \mid m < s \text{ and there exists no } z \in T \text{ with } m < z < s\}$$

For each $s \in S$ we can consider the set T_s as a quantree with the induced ordering and quantifier mapping from T. Clearly, $m \notin T_s$ for each $s \in S$, and for distinct $s, s' \in S$, neither $s \leq s'$ nor $s' \leq s$ hold. In particular, since T is a tree, there is no $x \in T$ such that $s, s' \in T^x$. This implies that the T_s are pairwise disjoint for $s \in S$.

Now, let $x \in T$. Then, there is a path $x_1, \ldots, x_n$ in T^x such that $n = \text{ht}(T)$. Since T^x is totally ordered with $\min(T^x) = m$ and $\max(T^x) = x$, the list

$m, x_2, \ldots, x_{n-1}, x$ is a path in T^x as well. If $n = 1$, then $x = m$; otherwise, since $n = \mathrm{ht}(T)$, we have $x_2 \in S$, and then $x \in T_{x_2}$. What we have shown is that $\{m\}$ together with the T_s for $s \in S$ form a partition of T.

Observe that $\mathrm{ht}(T_s) \leq k$, so by induction hypothesis we can find a linearization $f_s : T_s \to \{1, \ldots, \mathrm{aht}(T_s)\}$. Following the partitioning above, the map $f : T \to \{1, \ldots, \mathrm{aht}(T)\}$ given by

$$f(x) = \begin{cases} 1 & \text{if } x = m \\ f_s(x) + q^\star(s) & \text{if } x \in T_s \text{ for some } s \in S \end{cases}$$

is then well-defined. Note that $1 \leq f(x) \leq qht(T)$ for all $x \in T$. When $x = m$ this is trivial. Otherwise, $x \in T_s$ for some $s \in S$, and then either $q(s) = q(m)$, in which case $f(x) \leq \mathrm{aht}(T_s) \leq \mathrm{aht}(T)$ and $q^\star(s) = 0$; or else $q(s) \neq q(m)$, in which case $f(x) \leq \mathrm{aht}(T_s) < \mathrm{aht}(T)$ and $q^\star(s) = 1$.

Now we show that f is indeed a linearization. First we show that f respects the ordering in T. Let $x, y \in T$ with $x \leq y$. If $f(x) = 1$ then trivially $f(x) \leq f(y)$. Otherwise, there is some $s \in S$ with $x \in T_s$, and then $y \in T_s$ as well. Then, $f(x) = f_s(x) + q^\star(s) \leq f_s(y) + q^\star(s) = f(y)$.

Finally, we show that f respects quantifier parity. Let $x \in T$. If $x = m$, then $q(x) = q(m)$ and $f(x) = 1$ is odd. Otherwise, $x \in T_s$ for some $s \in S$. Then, $f_s(x)$ is odd iff $q(s) = q(x)$, and $q^\star(s)$ is odd iff $q(s) = q(m)$. Then, $f(x) = f_s(x) + q^\star(s)$ is odd iff $q(x) = q(m)$.

Proposition 3. *Let T be a quantree, Q be a quantifier and $\dagger, \ddagger$ be directions. Then,*

1. *$(\mathrm{Lin}(T), \precsim^{\mathsf{Q}\dagger})$ is a quasi-ordered set (i.e. $\precsim^{\mathsf{Q}\dagger}$ is reflexive and transitive).*
2. *$\preccurlyeq^{\mathsf{Q}\dagger\ddagger}$ is a partially ordered set.*

Then, $\precsim^{\mathsf{Q}\dagger}$ defines a quasiorder on $\mathrm{Lin}(T)$ (i.e. it is reflexive and transitive).

Proof. First we show that $\precsim^{\mathsf{Q}\dagger}$ defines a quasi-order on $\mathrm{Lin}(T)$. To show reflexivity, let $f \in \mathrm{Lin}(T)$. Then, $f(x) \leq^\dagger f(x)$ for all $x \in T$, so $f \precsim^{\mathsf{Q}\dagger} f$.

To show transitivity, let $f, g, h \in \mathrm{Lin}(T)$ such that $f \precsim^{\mathsf{Q}\dagger} g$ and $g \precsim^{\mathsf{Q}\dagger} h$. Then, for all $x \in T$ with $q(x) = \mathsf{Q}$, we have $f(x) \leq^\dagger g(x) \leq^\dagger h(x)$. Hence, $f \precsim^{\mathsf{Q}\dagger} h$.

Now we show that $\preccurlyeq^{\mathsf{Q}\dagger\ddagger}$ defines a partial order on $\mathrm{Lin}(T)$. To show reflexivity, let $f \in \mathrm{Lin}(T)$. Then, $f \approx^{\mathsf{Q}} f$ and $f \approx^{\overline{\mathsf{Q}}} f$, so $f \preccurlyeq^{\mathsf{Q}\dagger\ddagger} f$.

To show transitivity, let $f, g, h \in \mathrm{Lin}(T)$ such that $f \preccurlyeq^{\mathsf{Q}\dagger\ddagger} g$ and $g \preccurlyeq^{\mathsf{Q}\dagger\ddagger} h$. Then, $f \precsim^{\mathsf{Q}\dagger} g \precsim^{\mathsf{Q}\dagger} h$. Furthermore, if $f \approx^{\mathsf{Q}} h$, then $g \precsim^{\mathsf{Q}\dagger} h \precsim^{\mathsf{Q}\dagger} f \precsim^{\mathsf{Q}\dagger} g$ holds, so $f \approx^{\mathsf{Q}} g$ and $g \approx^{\mathsf{Q}} h$ hold as well. This implies that $f \precsim^{\overline{\mathsf{Q}}\ddagger} g \precsim^{\overline{\mathsf{Q}}\ddagger} h$. Thus, $f \preccurlyeq^{\mathsf{Q}\dagger\ddagger} h$.

To show antisymmetry, let $f, g \in \mathrm{Lin}(T)$ such that $f \preccurlyeq^{\mathsf{Q}\dagger\ddagger} g$ and $g \preccurlyeq^{\mathsf{Q}\dagger\ddagger} f$. Then, $f \precsim^{\mathsf{Q}\dagger} g \precsim^{\mathsf{Q}\dagger} f$, so we have $f \approx^{\mathsf{Q}} g$. This in turn implies $f \precsim^{\overline{\mathsf{Q}}\dagger} g \precsim^{\overline{\mathsf{Q}}\dagger} f$, so $f \approx^{\overline{\mathsf{Q}}} g$. Finally, $f \approx^{\mathsf{Q}} g$ and $f \approx^{\overline{\mathsf{Q}}} g$ imply $f = g$.

In the following we prove some properties about the mappings $\Gamma_\dagger$ and $[f]^{\mathsf{Q}\ddagger}$ defined in (1) and (2). Throughout the proofs we will use some notation:

$$\begin{aligned} m &= \min T \\ R &= \{1, \ldots, \mathrm{aht}(T)\} \\ \delta(x) &= \max^\dagger(R) - \mathrm{aht}(T_x^\dagger) \\ \alpha(x) &= |\delta(x)| + 1 \\ H_x &= \{y \in T_x^\ddagger \mid q(y) = \mathsf{Q}\} \\ \mu(x) &= \min^\ddagger\{f(y) \mid y \in H_x\} \end{aligned}$$

With these definitions, the following equalities hold:

$$\begin{aligned} \Gamma_\dagger(x) &= \lfloor \alpha(x) \rfloor^\dagger_{q^\star(x)} \\ [f]^{\mathsf{Q}\ddagger} &= \lfloor \mu(x) \rfloor^\ddagger_{q^\star(x)} \end{aligned}$$

We will also extend our conventions about directions $\uparrow, \downarrow$ as follows. We write $\subseteq^\dagger$ for $\supseteq$ when $\dagger = \uparrow$, and for $\subseteq$ when $\dagger = \downarrow$. Furthermore, we say that $x_1, \ldots, x_n$ is a $\dagger$-path in T if $x_n, \ldots, x_1$ is a path in T when $\dagger = \uparrow$, or if $x_1, \ldots, x_n$ is a path in T when $\dagger = \downarrow$. This simply follows our convention that $\uparrow$ reverses definitions and $\downarrow$ preserves them.

Proposition 4. *Let T be a quantree, and $\dagger$ be a direction. Then, $1 \leq \Gamma_\dagger(x) \leq \mathrm{aht}(T)$ for all $x \in T$.*

Proof. Let $x \in T$. Since $\{x\} \subseteq T_x^\dagger \subseteq T$, we know that:

$$1 \leq \mathrm{aht}(T_x^\dagger) \leq \mathrm{aht}(T)$$

Since $\max^\dagger(R)$ is either 1 or $\mathrm{aht}(T)$, we can conclude:

$$0 \leq |\max^\dagger(R) - \mathrm{aht}(T_x^\dagger)| \leq \mathrm{aht}(T) - 1$$

Then, $1 \leq \alpha(x) \leq \mathrm{aht}(T)$ follows, and therefore $0 \leq \Gamma_\dagger(x) \leq \mathrm{aht}(T) + 1$ holds. There are only two ways in which the extremes might be reached:

- If $\Gamma_\dagger(x) = 0$, then $\dagger = \downarrow$, $\alpha(x) = 1$ and $\alpha(x) - q^\star(x)$ is odd. Then,

  $$\alpha(x) = |\mathrm{aht}(T) - \mathrm{aht}(T_x)| + 1 = \mathrm{aht}(T) - \mathrm{aht}(T_x) + 1 = 1$$

  implies $\mathrm{aht}(T) = \mathrm{aht}(T_x)$. Let $x_1, \ldots, x_n$ be an alternating path in T_x with $n = \mathrm{aht}(T_x)$. Because $x = \min T_x$, the list $x, x_2, \ldots, x_n$ is an alternating path too. Now, since $\alpha(x) = 1$ and $\alpha(x) - q^\star(x)$ is odd, we obtain $q^\star(x) = 0$, that is, $q(x) \neq q(m)$. The list $m, x, x_2, \ldots, x_n$ is an alternating path in T, then. But then $\mathrm{aht}(T) \geq n + 1$, which contradicts $\mathrm{aht}(T) = \mathrm{aht}(T_x) = n$.

– If $\Gamma_\dagger(x) = \mathrm{aht}(T) + 1$, then $\dagger = \uparrow$, $\alpha(x) = \mathrm{aht}(T)$ and $\alpha(x) - q^\star(x)$ is odd. Then,
$$\alpha(x) = |1 - \mathrm{aht}(T^x)| + 1 = \mathrm{aht}(T^x) - 1 + 1 = \mathrm{aht}(T^x)$$
implies $\mathrm{aht}(T^x) = \mathrm{aht}(T)$. Let $x_1, \ldots, x_n$ be an alternating path in T^x with $n = \mathrm{aht}(T^x)$. Since T^x is totally ordered, $x = \max T^x$ and $m = \min T^x$, then $m, x_2, \ldots, x_{n-1}, x$ is an alternating path too. Then, n is odd iff $q(x) = q(m)$ iff $q^\star(x) = 1$. Since $\alpha(x) - q^\star(x)$ is odd, we conclude that $n = \mathrm{aht}(T^x)$ is odd iff $\alpha(x) = \mathrm{aht}(T^x)$ is even, which is a contradiction.

We thus conclude $0 < \Gamma_\dagger(x) < \mathrm{aht}(T) + 1$, or equivalently, $0 \leq \Gamma_\dagger(x) \leq \mathrm{aht}(T)$

Proposition 5. *Let T be a quantree, and $\dagger$ be a direction. Then, $\Gamma_\dagger \in \mathrm{Lin}(T)$.*

Proof. Showing that $\Gamma_\dagger$ respects quantifier parity for any $x \in T$ is easy: $\Gamma_\dagger(x)$ is odd iff $\lfloor \alpha(x) \rfloor^\dagger_{q^\star(x)}$ is odd iff $q^\star(x)$ is odd. Since $0 \leq q^\star(x) \leq 1$, we conclude that $\Gamma_\dagger(x)$ is odd iff $q^\star(x) = 1$.

Now we show that $\Gamma_\dagger$ respects the ordering in T. Let $x, y \in T$ with $x \leq y$. Then, $T^\dagger_y \subseteq^\dagger T^\dagger_x$. Hence,$\mathrm{aht}(T^\dagger_y) \leq^\dagger \mathrm{aht}(T^\dagger_x)$ Both quantities are in the range R, so the following inequality holds:
$$\min^\dagger(R) \leq^\dagger \mathrm{aht}(T^\dagger_y) \leq^\dagger \mathrm{aht}(T^\dagger_x) \leq^\dagger \max^\dagger(R)$$
Equivalently,
$$\min^\dagger(R) - \max^\dagger(R) \leq^\dagger \delta(y) \leq^\dagger \delta(x) \leq^\dagger 0$$
Note now that for all integers a, b with $a \leq^\dagger b \leq^\dagger 0$ we have $|a| \leq |b|$. Then,
$$0 \leq |\delta(x)| \leq |\delta(y)| \leq |\min^\dagger(R) - \max^\dagger(R)| = \mathrm{aht}(T) - 1$$
and this shows $1 \leq \alpha(x) \leq \alpha(y) \leq \mathrm{aht}(T)$.

Let us choose (u, v) as (x, y) or as (y, x) so that $u \leq^\dagger v$ holds. Then, $\alpha(u) \leq^\dagger \alpha(v)$, and then we can bound:
$$\Gamma_\dagger(u) = \lfloor \alpha(u) \rfloor^\dagger_{q^\star(u)} \leq^\dagger \alpha(u) \leq^\dagger \alpha(v)$$
Let us assume that $\Gamma_\dagger(u) >^\dagger \Gamma_\dagger(v)$. Since $\Gamma_\dagger(v) = \lfloor \alpha(v) \rfloor^\dagger_{q^\star(v)}$, this can only be the case if $\alpha(u) = \alpha(v)$ holds, $\alpha(u) - q^\star(u)$ is even and $\alpha(v) - q^\star(v)$ is odd. These three combined imply that $q^\star(u) - q^\star(v)$ is odd, which in turn implies $q(u) \neq q(v)$.

Let $x_1, \ldots, x_n$ be an alternating $\dagger$-path in $T^\dagger_v$ with $n = \mathrm{aht}(T^\dagger_v)$. Since $v = \min^\dagger(T^\dagger_v)$, the list $v, x_2, \ldots, x_n$ is an alternating $\dagger$-path as well. Now, $u \leq^\dagger v$ and $q(u) \neq q(v)$, so the list $u, v, x_2, \ldots, x_n$ is an alternating $\dagger$-path in $T^\dagger_u$. Hence,
$$\mathrm{aht}(T^\dagger_u) \geq n + 1 > n = \mathrm{aht}(T^\dagger_v) \tag{3}$$

Finally, from $\alpha(u) = \alpha(v)$ we can conclude that $|\delta(u)| = |\delta(v)|$. A simple case discussion can be used to show then $\delta(u) = \delta(v)$, which in turn implies $\mathrm{aht}(T^\dagger_u) = \mathrm{aht}(T^\dagger_v)$. But this contradicts (3).

By reduction to absurd, $\Gamma_\dagger(u) \leq^\dagger \Gamma_\dagger(v)$, so $\Gamma_\dagger(x) \leq \Gamma_\dagger(y)$. This shows that $\Gamma_\dagger$ is a linearization, and together with Proposition 4 we get $\Gamma_\dagger \in \mathrm{Lin}(T)$.

Proposition 6. *Let T be a quantree, Q a quantifier and $\dagger$ be a direction. Then, $f \preccurlyeq^{\mathsf{Q}\dagger\dagger} \Gamma_\dagger$ for any $f \in \mathrm{Lin}(T)$.*

Proof. Let $f \in \mathrm{Lin}(T)$ and define $\Delta = \{x \in T \mid f(x) >^\dagger \Gamma_\dagger(x)\}$. If Δ was empty, then the result would follow: $f(x) \leq^\dagger \Gamma_\dagger(x)$ for all $x \in T$ implies both $f \precsim^{\mathsf{Q}\dagger} \Gamma_\dagger$ and $f \precsim^{\overline{\mathsf{Q}}\dagger} \Gamma_\dagger$, and from there we get $f \preccurlyeq^{\mathsf{Q}\dagger\dagger} \Gamma_\dagger$.

Let us then assume that Δ is nonempty and reach a contradiction. Choose $x \in \Delta$ maximal w.r.t. $\leq^\dagger$. Observe first that, because f is a linearization, $f(x)$ is odd iff $q^\star(x) = 1$. Then, $f(x) - q^\star x$ is even, and so

$$\lfloor q^\star(x) \rfloor^\dagger_{f(x)} = f(x) >^\dagger \Gamma_\dagger(x) = \lfloor \alpha(x) \rfloor^\dagger_{q^\star(x)}$$

This implies that $f(x) >^\dagger \alpha(x)$. Let us discuss two different cases.

- If $\mathrm{aht}(T^\dagger_x) = 1$, then $\delta(x) = \max^\dagger(R) - 1 \geq 0$ holds. Thus,

  $$\alpha(x) = |\delta(x)| + 1 = \delta(x) + 1 = \max^\dagger(R)$$

  Since $f \in \mathrm{Lin}(T)$, this leads to a contradiction:

  $$\max^\dagger(R) = \alpha(x) <^\dagger f(x) \leq^\dagger \max^\dagger(R)$$

- If $\mathrm{aht}(T^\dagger_x) > 1$, consider an alternating $\dagger$-path $x_1, \ldots, x_n$ in $T^\dagger_x$ with $n = \mathrm{aht}(T^\dagger_x)$. Since $x = \min^\dagger(\mathrm{aht}(T^\dagger_x))$, the list $x, x_2, \ldots, x_n$ is an alternating $\dagger$-path in $T^\dagger_x$, too. Then, $x \leq^\dagger x_2$ and $q(x) \neq q(x_2)$, so we have $f(x) <^\dagger f(x_2)$. Furthermore, x is maximal in Δ w.r.t. $\leq^\dagger$, so the following inequality holds:

  $$\Gamma_\dagger(x) <^\dagger f(x) <^\dagger f(x_2) \leq^\dagger \Gamma_\dagger(x)$$

  In particular, this implies $|\Gamma_\dagger(x) - \Gamma_\dagger(x_2)| \geq 2$. On the other hand,

  $$\begin{aligned} |\alpha(x_2) - \alpha(x)| &= ||\delta(x_2)| + 1 - |\delta(x)| - 1| \\ &= ||\delta(x_2)| - |\delta(x)|| \leq \\ &\leq |\delta(x_2) - \delta(x)| = \\ &= |\max^\dagger(R) - \mathrm{aht}(T^\dagger_{x_2}) - \max^\dagger(R) + \mathrm{aht}(T^\dagger_x)| \\ &= |\mathrm{aht}(T^\dagger_x) - \mathrm{aht}(T^\dagger_{x_2})| \end{aligned}$$

  Let us compute the last quantity. First, $x_2, \ldots, x_n$ is an alternating $\dagger$-path in $T^\dagger_{x_2}$. Second, for any alternating $\dagger$-path $y_1, \ldots, y_r$ in $T^\dagger_{x_2}$, the list $x, y_1, \ldots, y_r$ is also an alternating $\dagger$-path in $T^\dagger_{x_2}$. These two facts combined imply that $|\mathrm{aht}(T^\dagger_x) - \mathrm{aht}(T^\dagger_{x_2})| = 1$. Then,

  $$|\Gamma_\dagger(x_2) - \Gamma_\dagger(x)| \leq |\alpha(x_2) - \alpha(x)| + 1 \leq 2$$

  We have thus shown $|\Gamma_\dagger(x_2) - \Gamma_\dagger(x)| = 2$, so $\Gamma_\dagger(x_2) - \Gamma_\dagger(x)$ is even. This is a contradiction: from $q(x) \neq q(x_2)$ we can conclude that $\Gamma_\dagger(x_2)$ is odd iff $\Gamma_\dagger(x)$ is even, which implies that $\Gamma_\dagger(x_2) - \Gamma_\dagger(x)$ is odd.

In both cases we reach a contradiction, so Δ is empty, and so $f \preccurlyeq^{\mathsf{Q}\dagger\dagger} \Gamma_\dagger$.

Lemma 1. *Let T be a quantree, Q be a quantifier, and $\dagger$ be a direction. Then, $\Gamma_\dagger \in \mathrm{Lin}(T)$. Furthermore, for any $g \in \mathrm{Lin}(T)$, we have $g \precsim^{\mathsf{Q}\dagger} \Gamma_\dagger$.*

Proof. $\Gamma_\dagger \in \mathrm{Lin}(T)$ follows from Proposition 5. Furthermore, for all $g \in \mathrm{Lin}(T)$, Proposition 6 shows that $g \preccurlyeq^{\mathsf{Q}\dagger\dagger} \Gamma_\dagger$, so in particular $g \precsim^{\mathsf{Q}\dagger} \Gamma_\dagger$ holds.

Proposition 7. *Let T be a quantree, Q be a quantifier and $\ddagger$ be a direction. Consider a linearization $f \in \mathrm{Lin}(T)$. Then, $1 \leq [f]^{\mathsf{Q}\ddagger}(x) \leq \mathrm{aht}(T)$ for all $x \in T$.*

Proof. Let $x \in T$. Since $H_x \subseteq T$ and $f \in \mathrm{Lin}(T)$, we know that $\{f(y) \mid y \in H_x\} \subseteq R$, so we conclude $1 \leq \mu(x) \leq qhtT$. We can rewrite this inequality as:

$$\min^\ddagger(R) \leq^\ddagger \mu(x) \leq^\ddagger \max^\ddagger(R)$$

Then we can bound:

$$[f]^{\mathsf{Q}\ddagger}(x) = \lfloor \mu(x) \rfloor^\ddagger_{q^\star(x)} \leq^\ddagger \mu(x) \leq^\ddagger \max^\ddagger(R) \tag{4}$$

Now let us assume that $[f]^{\mathsf{Q}\ddagger}(x) <^\ddagger \min^\ddagger(R)$. Since $\min^\ddagger(R) \leq^\ddagger \mu(x)$, this can only be the case if $\mu(x) = \min^\ddagger(R)$ and $\mu(x) - q^\star(x)$ is odd. We show that this leads to a contradiction.

- If H_x is empty, then $\mu(x) = \min^\ddagger(\emptyset) = \max^\ddagger(R)$ holds. Then, we obtain $\min^\ddagger(R) = \max^\ddagger(R)$, which implies $\mathrm{aht}(T) = 1 = \mu(x)$. Because $\mu(x) - q^\star(x)$ is odd, we conclude $q(x) \neq q(m)$. This is however a contradiction: in this case, m, x would be a path in T, implying $1 = \mathrm{aht}(T) \geq 2$.
- If H_x is nonempty, then we can find some $y \in H_x$ with $\mu(x) = f(y)$. Since $\mu(x) - q^\star(x)$ is odd, we obtain $f(x)$ is odd iff $q^\star(x) = 1$ iff $\mu(x)$ is even iff $f(y)$ is even. Hence, $q(x) \neq q(y)$, and since $x \leq^\ddagger y$, we conclude $f(x) <^\ddagger f(y)$. This leads to a contradiction:

$$\min^\ddagger(R) \leq^\ddagger f(x) <^\ddagger f(y) = \mu(x) = \min^\ddagger(R)$$

By reduction to absurd, we conclude together with (4) that

$$\min^\ddagger(R) \leq^\ddagger [f]^{\mathsf{Q}\ddagger}(x) \leq^\ddagger \max^\ddagger(R)$$

and equivalently, $1 \leq [f]^{\mathsf{Q}\ddagger}(x) \leq \mathrm{aht}(T)$ holds.

Proposition 8. *Let T be a quantree, Q be a quantifier and $\ddagger$ be a direction. Consider a linearization $f \in \mathrm{Lin}(T)$. Then, $[f]^{\mathsf{Q}\ddagger} \in \mathrm{Lin}(T)$.*

Proof. First, we show that $[f]^{\mathsf{Q}\ddagger}$ respects quantifier parity. Given any $x \in T$, we have $[f]^{\mathsf{Q}\ddagger}(x) = \lfloor \mu(x) \rfloor^\ddagger_{q^\star(x)}$, so $[f]^{\mathsf{Q}\ddagger}(x)$ is odd iff $q^\star(x)$.

Now we show that $[f]^{\mathsf{Q}\ddagger}$ respects the ordering in T. Let $x, y \in T$ with $x \leq y$. Then we can choose (u, v) as (x, y) or as (y, x) so that $u \leq^\ddagger v$. Then, $H_u \subseteq^\ddagger H_v$, which implies $\mu(u) \leq^\ddagger \mu(v)$. Hence,

$$[f]^{\mathsf{Q}\ddagger}(u) = \lfloor \mu(u) \rfloor^\ddagger_{q^\star(u)} \leq^\ddagger \mu(u) \leq^\ddagger \mu(v)$$

Let us assume that $[f]^{\mathsf{Q}\ddagger}(u) >^{\ddagger} [f]^{\mathsf{Q}\ddagger}(v)$. This can only be the case if $\mu(u) = \mu(v)$ holds, $\mu(u) - q^\star(v)$ is even, and $\mu(v) - q^\star(v)$ is odd. These three properties together imply that $q^\star(u) - q^\star(v)$ is odd, which further implies $q(u) \neq q(v)$. Since $u \leq^{\ddagger} v$, we obtain $u <^{\ddagger} v$. We show that this leads to a contradiction:

- If $q(u) = \mathsf{Q}$, then $u \in H_u$, so
$$\mu(v) = \mu(u) \leq^{\ddagger} f(u) <^{\ddagger} f(v)$$
holds. Now, if H_v is empty, we have $f(v) \leq^{\dagger} \max^{\ddagger}(R) = \mu(v)$. Otherwise, we can find some $w \in H_v$ with $\mu(v) = f(w)$, so $f(v) \leq^{\ddagger} f(w) = \mu(v)$. Either way, we conclude that $\mu(v) <^{\ddagger} f(v) \leq^{\dagger} \mu(v)$, which is a contradiction.
- If $q(u) \neq \mathsf{Q}$, then $q(v) = \mathsf{Q}$, so $v \in H_v$. Then we can find some $w \in H_v$ such that $f(w) = \mu(v)$. Observe that $q(v) = q(w) = \mathsf{Q}$ and $\mu(v) - q^\star(v)$ is odd. Then, $q^\star(v) = 0$ iff $\mu(v)$ is odd iff $f(w)$ is odd iff $q^\star(w) = 1$ iff $q^\star(w) = 1$, which is a contradiction.

By reduction to absurd, we conclude that $g(u) \leq^{\ddagger} g(v)$, and equivalently, $g(x) \leq g(y)$.

Proposition 9. *Let T be a quantree, Q be a quantifier and $\ddagger$ be a direction. Consider a linearization $f \in \mathrm{Lin}(T)$. Then, $g \approx^{\mathsf{Q}} [f]^{\mathsf{Q}\ddagger}$ and $g \precsim^{\overline{\mathsf{Q}}\ddagger} [f]^{\mathsf{Q}\ddagger}$ for any $g \in \mathrm{Lin}(T)$ such that $f \approx^{\mathsf{Q}} g$.*

Proof. Let $g \in \mathrm{Lin}(T)$ with $[f]^{\mathsf{Q}\ddagger} \approx^{\mathsf{Q}} g$. First we show $[f]^{\mathsf{Q}\ddagger} \approx^{\mathsf{Q}} g$. Let $x \in T$ with $q(x) = \mathsf{Q}$. Then, $x \in H_x$, so $\mu(x) \leq^{\ddagger} f(x)$. Now, since H_x is nonempty, there is some $y \in H_x$ with $f(y) = \mu(x)$. Note that $x \leq^{\ddagger} y$, so:
$$f(x) \leq^{\ddagger} f(y) = \mu(x) \leq^{\ddagger} f(x)$$
This shows $f(x) = \mu(x)$. Now, $f(x)$ is odd iff $q^\star(x) = 1$. In particular, $\mu(x) - q^\star(x)$ is even. Furthermore, $f \approx^{\mathsf{Q}} g$, so $f(x) = g(x)$. Hence,
$$[f]^{\mathsf{Q}\ddagger}(x) = \lfloor \mu(x) \rfloor^{\ddagger}_{q^\star(x)} = \mu(x) = f(x) = g(x)$$
which shows $[f]^{\mathsf{Q}\ddagger}(x) \approx^{\mathsf{Q}} g$.

Now we show $g \precsim^{\overline{\mathsf{Q}}\ddagger} [f]^{\mathsf{Q}\ddagger}$ holds. Let $x \in T$ such that $q(x) \neq \mathsf{Q}$. observe that $g(x)$ is odd iff $q^\star(x) = 1$, so $g(x) = \lfloor g(x) \rfloor^{\ddagger}_{q^\star(x)}$ holds.

- If H_x is empty, then $\mu(x) = \min^{\ddagger}(\emptyset) = \max^{\ddagger}(R)$. From $g(x) \leq^{\ddagger} \max^{\ddagger}(R)$ we obtain:
$$g(x) = \lfloor g(x) \rfloor^{\ddagger}_{q^\star(x)} \leq^{\ddagger} \left\lfloor \max^{\ddagger}(R) \right\rfloor^{\ddagger}_{q^\star(x)} = \lfloor \mu(x) \rfloor^{\ddagger}_{q^\star(x)} = [f]^{\mathsf{Q}\ddagger}(x)$$
- If H_x is nonempty, then we can find some $y \in H_x$ such that $\mu(x) = f(y)$. Since $q(y) = \mathsf{Q}$ and $x \leq^{\ddagger} y$, we have:
$$g(x) <^{\ddagger} g(y) = f(y) = \mu(x)$$
Then we obtain:
$$g(x) = \lfloor g(x) \rfloor^{\ddagger}_{q^\star(x)} \leq^{\ddagger} \lfloor \mu(x) \rfloor^{\ddagger}_{q^\star(x)} = [f]^{\mathsf{Q}\ddagger}(x)$$

Either way, we have shown $g(x) \leq^{\ddagger} [f]^{\mathsf{Q}\ddagger}(x)$ for any $x \in T$ with $q(x) \neq \mathsf{Q}$. Hence, $g \precsim^{\overline{\mathsf{Q}}\ddagger} [f]^{\mathsf{Q}\ddagger}$ holds.

Lemma 2. *Let T be a quantree, Q be a quantifier and $\dagger, \ddagger$ be directions. Consider a linearization $f \in \mathrm{Lin}(T)$. Then, $[f]^{\mathsf{Q}\ddagger} \in \mathrm{Lin}(T)$. Furthermore, $[f]^{\mathsf{Q}\ddagger} \approx^{\mathsf{Q}} f$, and for any $g \in \mathrm{Lin}(T)$ with $g \approx^{\mathsf{Q}} f$, we have $g \preccurlyeq^{\mathsf{Q}\dagger\ddagger} [f]^{\mathsf{Q}\ddagger}$.*

Proof. $f \in \mathrm{Lin}(T)$ is shown by Propostion 8. $[f]^{\mathsf{Q}\ddagger} \approx^{\mathsf{Q}} f$ follows from Proposition 9 by considering $f \approx^{\mathsf{Q}} f$. Now, let $g \in \mathrm{Lin}(T)$ with $g \approx^{\mathsf{Q}} f$. Proposition 9 shows that $[f]^{\mathsf{Q}\ddagger} \approx^{\mathsf{Q}} g$ and $g \precsim^{\overline{\mathsf{Q}}\ddagger} [f]^{\mathsf{Q}\ddagger}$ hold. The former also implies $g \precsim^{\mathsf{Q}\dagger} [f]^{\mathsf{Q}\ddagger}$. Hence, $g \preccurlyeq^{\mathsf{Q}\dagger\ddagger} [f]^{\mathsf{Q}\ddagger}$.

Theorem 3. *Let T be a quantree, Q be a quantifier and $\dagger, \ddagger$ be directions. Then, $f \preccurlyeq^{\mathsf{Q}\dagger\ddagger} [\Gamma_\dagger]^{\mathsf{Q}\ddagger}$ for all $f \in \mathrm{Lin}(T)$. In particular, $[\Gamma_\dagger]^{\mathsf{Q}\ddagger}$ is the unique $\mathsf{Q}\dagger\ddagger$-optimal linearization in $\mathrm{Lin}(T)$.*

Proof. Let $f \in \mathrm{Lin}(T)$. Then, by Lemma 1, $f \precsim^{\mathsf{Q}\dagger} \Gamma_\dagger$. Since Lemma 2 shows that $\Gamma_\dagger \approx^{\mathsf{Q}} [\Gamma_\dagger]^{\mathsf{Q}\ddagger}$, we conclude that $f \precsim^{\mathsf{Q}\dagger} [\Gamma_\dagger]^{\mathsf{Q}\ddagger}$.

- If $f \approx^{\mathsf{Q}} [\Gamma_\dagger]^{\mathsf{Q}\ddagger}$ does not hold, then $f \preccurlyeq^{\mathsf{Q}\dagger\ddagger} [\Gamma_\dagger]^{\mathsf{Q}\ddagger}$ holds by definition.
- If $f \approx^{\mathsf{Q}} [\Gamma_\dagger]^{\mathsf{Q}\ddagger}$ holds, then $f \approx^{\mathsf{Q}} \Gamma_\dagger$ holds as well, and Lemma 2 then implies $f \preccurlyeq^{\mathsf{Q}\dagger\ddagger} [\Gamma_\dagger]^{\mathsf{Q}\ddagger}$.

Either way, we conclude $f \preccurlyeq^{\mathsf{Q}\dagger\ddagger} [\Gamma_\dagger]^{\mathsf{Q}\ddagger}$.

Corollary 1. *Let T be a quantree, Q be a quantifier and $\dagger$ be a direction. Let $f \in \mathrm{Lin}(T)$. Then, for all $x \in T$ such that $q(x) \neq \mathsf{Q}$, the following equality holds:*

$$[f]^{\mathsf{Q}\ddagger}(x) = \Big\lfloor \min^{\ddagger}\{[f]^{\mathsf{Q}\ddagger}(y) \mid x \text{ is covered by } y \in T \text{ w.r.t. } \leq^{\ddagger}\} \Big\rfloor^{\ddagger}_{q^\star(x)}$$

Proof. Let us define for all $x \in T$ the following:

$$\begin{aligned} S_x &= \{y \in T \mid x \text{ is covered by } y \text{ w.r.t. } \leq^{\ddagger}\} \\ \lambda(x) &= \min^{\dagger}\{G(y) \mid y \in S_x\} \end{aligned}$$

Then we need to show that $[f]^{\mathsf{Q}\ddagger}(x) = \lfloor \lambda(x) \rfloor^{\ddagger}_{q^\star(x)}$ for all $x \in T$ with $q(x) \neq \mathsf{Q}$.

- If S_x is empty, then since H_x is empty as well. Then, $\lambda(x) = \mu(x) = \min^{\ddagger}(\emptyset)$, so $[f]^{\mathsf{Q}\ddagger}(x) = \lfloor \mu(x) \rfloor^{\ddagger}_{q^\star(x)} = \lfloor \lambda(x) \rfloor^{\ddagger}_{q^\star(x)}$ holds.
- If S_x is nonempty and H_x is empty, then for all $y \in S_x$ we have $H_y \subseteq H_x =$, so $\mu(y) = \max^{\ddagger}(\emptyset) = \mu(x)$. Since $x \leq^{\ddagger} y$ and $y \notin H_x$, we conclude $q(y) \neq \mathsf{Q}$. Hence, $q^\star(x) = q^\star(y)$. Then,

$$[f]^{\mathsf{Q}\ddagger}(y) = \lfloor \mu(y) \rfloor^{\ddagger}_{q^\star(y)} = \lfloor \mu(x) \rfloor^{\ddagger}_{q^\star(x)}$$

which is independent of y. Then we obtain:

$$\begin{aligned}\lfloor \lambda(x) \rfloor^{\ddagger}_{q^\star(x)} &= \left\lfloor \min^{\ddagger}\{ [f]^{\mathsf{Q}\ddagger}(y) \mid y \in S_x \} \right\rfloor^{\ddagger}_{q^\star(x)} = \\ &= \left\lfloor \min^{\ddagger}\{ \lfloor \mu(x) \rfloor^{\ddagger}_{q^\star(x)} \} \right\rfloor^{\ddagger}_{q^\star(x)} = \\ &= \left\lfloor \lfloor \mu(x) \rfloor^{\ddagger}_{q^\star(x)} \right\rfloor^{\ddagger}_{q^\star(x)} = \lfloor \mu(x) \rfloor^{\ddagger}_{q^\star(x)} = [f]^{\mathsf{Q}\ddagger}(x)\end{aligned}$$

– If both S_x and H_x are nonempty, then we can find $u \in S_x$ and $w \in H_x$ such that $\lambda(x) = [f]^{\mathsf{Q}\ddagger}(u)$ and $\mu(x) = f(w)$. Furthermore, we can find some $v \in S_x$ such that $v \leq^{\ddagger} w$. Observe also that $q(w) = \mathsf{Q}$, so $[f]^{\mathsf{Q}\ddagger}(w) = f(w)$. Then we have:

$$\lambda(x) = [f]^{\mathsf{Q}\ddagger}(u) \leq^{\ddagger} [f]^{\mathsf{Q}\ddagger}(v) \leq^{\ddagger} [f]^{\mathsf{Q}\ddagger}(w) = f(w) = \mu(x)$$

We thus conclude $\lfloor \lambda(x) \rfloor^{\ddagger}_{q^\star(x)} \leq^{\ddagger} [f]^{\mathsf{Q}\ddagger}(x)$. On the other hand, $x \leq^{\ddagger} u$, so $[f]^{\mathsf{Q}\ddagger}(x) \leq^{\ddagger} [f]^{\mathsf{Q}\ddagger}(u) = \lambda(x)$. This implies:

$$[f]^{\mathsf{Q}\ddagger}(x) = \lfloor \mu(x) \rfloor^{\ddagger}_{q^\star(x)} = \left\lfloor \lfloor \mu(x) \rfloor^{\ddagger}_{q^\star(x)} \right\rfloor^{\ddagger}_{q^\star(x)} = \left\lfloor [f]^{\mathsf{Q}\ddagger}(x) \right\rfloor^{\ddagger}_{q^\star(x)} \leq^{\ddagger} \lfloor \lambda(x) \rfloor^{\ddagger}_{q^\star(x)}$$

Both inequalities combined yield $\lfloor \lambda(x) \rfloor^{\ddagger}_{q^\star(x)} = [f]^{\mathsf{Q}\ddagger}(x)$.