Bounded-width QBF is PSPACE-complete*

Albert Atserias†
Universitat Politècnica de Catalunya
Barcelona, Spain
atserias@lsi.upc.edu

Sergi Oliva‡
Universitat Politècnica de Catalunya
Barcelona, Spain
oliva@lsi.upc.edu

March 15, 2013

Abstract

Tree-width is a well-studied parameter of structures that measures their similarity to a tree. Many important NP-complete problems, such as Boolean satisfiability (SAT), are tractable on bounded tree-width instances. In this paper we focus on the canonical PSPACE-complete problem QBF, the fully-quantified version of SAT. It was shown by Pan and Vardi [LICS 2006] that this problem is PSPACE-complete even for formulas whose tree-width grows extremely slowly. Vardi also posed the question of whether the problem is tractable when restricted to instances of bounded tree-width. We answer this question by showing that QBF on instances with constant tree-width is PSPACE-complete. Additionally, we introduce a family of formulas with bounded tree-width that do have short refutations in a specific proof system.

1 Introduction

Tree-width is a well-known parameter that measures how close a structure is to being a tree. Many NP-complete problems have polynomial-time algorithms on inputs of bounded tree-width. In particular, the Boolean satisfiability problem can be solved in polynomial time when the constraint graph of the input cnf-formula has bounded tree-width (cf. [1], [2]).

A natural question suggested by this result is whether QBF, the problem of determining if a fully-quantified cnf-formula is true or false, can also be solved in polynomial time when restricted to formulas whose cnf-formula has bounded tree-width. In [3], Chen concludes that the problem stays tractable if the number of quantifier alternations, as well as the tree-width, is bounded. On the negative side, Gottlob, Greco and Scarcello [4] proved that the problem stays PSPACE-complete when the number of alternations is unbounded even if the constraint graph of the cnf-formula has logarithmic tree-width (and indeed, its *incidence* graph is even a tree). By different methods, and improving upon [4], Pan and Vardi [5] show that, unless P = NP, the dependence of the running time of Chen's algorithm on the number of alternations must be non-elementary, and that the QBF problem restricted to instances of tree-width $\log^*$ in the size of the input is PSPACE-complete. All

*A preliminary version of this paper appeared in the Proceedings of the 30th International Symposium on Theoretical Aspects of Computer Science (STACS 2013). The current version is based on Chapter 3 of the second author's PhD thesis.

†Research partially supported by CICYT TIN2010-20967-C04-04 (TASSAT).

‡Research supported by CICYT TIN2007-66523 (FORMALISM).

these negative results hold also for path-width, which is a parameter that measures the similarity to a path and is in general smaller than tree-width. However, they leave open whether QBF is tractable for instances whose constraint graph has constant path-width, or even constant tree-width.

Main result and comparison to previous results In this paper, we resolve this question by showing that, even for inputs of constant path-width, QBF is PSPACE-complete. Our construction builds on the techniques from [5] with two essential differences. The first difference is that instead of reducing from the so-called *tiling-game* and producing a quantified Boolean formula of $\log^*$-smaller path-width, our reduction starts at QBF itself and produces a quantified Boolean formula whose path-width is *only* logarithmically smaller. Although this looks like backward progress, it leaves us in a position where iterating the reduction makes sense. However, in order to do so, we need to analyze which properties of the output of the reduction can be exploited by the next iteration. Here comes the second main difference: we observe that the output of the reduction has not only smaller path-width, but also smaller *window-size*, which means that any two occurences of the same variable appear close to each other in some ordering of the clauses. We call such formulas *n-leveled*, where n is a bound related to the window-size. Our main lemma exploits this structural restriction in a technical way to show that the QBF problem for n-leveled formulas reduces to the QBF problem for $O(\log n)$-leveled formulas. Iterating this reduction until we reach $O(1)$-leveled formulas yields the result.

A few more words on the differences between our methods and those in [5] and [4] are in order. The technical tool from [5] that is used to achieve n-variable formulas of $O(\log^* n)$ path-width builds on the tools from [6] and [7] that were used for showing non-elementary lower-bounds for some problems related to second-order logic. These tools are based on an encoding of natural numbers that allows the comparison of two n-bit numbers by means of an extremely smaller formula; one of size $O(\log^* n)$. It is interesting that, by explicitely avoiding this technique, our iteration-based methods take us further: beyond $O(\log^* n)$ path-width down to constant path-width. For the same reason our proof can stay purely at the level of propositional logic without the need to resort to second-order logic. Along the same lines, our method also shows that the QBF problem for n-variable formulas of constant path-width and $O(\log^* n)$ quantifier alternations is NP-hard (and Σ_iP-hard for any $i \geq 1$), while the methods from [5] could only show this for $O(\log^* n)$ path-width and $O(\log^* n)$ alternations. It is worth noting that, in view of the results in [3], these hardness results are tight up to the hidden constants in the asymptotic notation.

Structural restrictions on the generalization of QBF to unbounded domains, sometimes called QCSP, have also been studied. Gottlob et al. [4] proved that QCSP restricted to trees is already PSPACE-complete. Their hardness result for qbfs of logarithmic tree-width follows from this by *booleanization*. They also identify some new tractable fragments, and some other hardness conditions. Finally, Chen and Dalmau [8] introduced a general framework for studying structural restrictions on QCSP, and characterized the restrictions that make the problem tractable under complexity-theoretic assumptions.

Respectful tree-width and Q-resolution One of the restrictions of QCSP that Chen and Dalmau showed tractable is that the constraint graph of the instance has bounded *respectful tree-width*. Note that the tree-width of the constraint graph is independent of the quantification of the instance. Respectful tree-width is precisely a quantifier-aware parameter, that considers only tree-decompositions that are respectful with the quantification, in the sense that bottom-up algorithms

can be run on these tree-decompositions without violating precedence of quantifiers.

In this paper we observe that qbfs of bounded respectful tree-width are not only tractable but also have short Q-resolution proofs. We start by presenting different forms of quantifier-aware resolution introduced by Büning, Flögel and Karpinski [9] and Pan and Vardi [10] and show how they relate to each other. Next, we show that respectful tree-width is equivalent to *respectful induced width*. Here *induced width* refers to a measure equivalent to tree-width introduced in [11]. Finally, we show that false qbfs with bounded respectful induced width have short Q-resolution refutations, which yields our result.

As an application of this result, we show that a family of formulas inspired by one introduced by Dalmau, Kolaitis and Vardi [12], has bounded respectful tree-width. We give practical examples of how these formulas are useful.

Paper organization The paper is organized as follows. In section 2, we introduce the basic definitions. In section 3, we formalize the concept of leveled-qbf and state and prove the main lemma. In section 4, we present the main theorem of the paper, which shows how to iterate the lemma to obtain the desired result. Finally, in section 5, we introduce the Q-resolution proof system and the concept of respectful tree-width and present our results on those.

2 Preliminaries

We write $[n] := \{1, \ldots, n\}$ and $|n| := \lceil \log(n+1) \rceil$. All logarithms are base 2. Note that $|n|$ is the length of the binary encoding of n. We define $\log^{(0)} n := n$ and $\log^{(i)} n := \log(\log^{(i-1)} n)$ for $i > 0$. Also, we use $\log^* n$ as the least integer i such that $\log^{(i)} n \leq 1$.

The negation of a propositional variable x is denoted by $\overline{x}$. We also use the notation $x^{(1)}$ and $x^{(0)}$ to denote x and $\overline{x}$, respectively. Note that the notation is chosen so that $x^{(a)}$ is made *true* by the assignment $x = a$. The *underlying variable* of $x^{(a)}$ is x, and its *sign* is a. A *literal* is a variable or the negation of a variable. A *clause* is a sequence of literals. A *cnf-formula* is a sequence of clauses. The *size* of a clause is its length as a sequence, and the *size* of a cnf-formula is the sum of the sizes of its clauses. For example,

$$\phi = ((x_1, \overline{x_2}), (x_2, \overline{x_3}, x_4), (\overline{x_4})) \tag{1}$$

is a cnf-formula of size 6 made of three clauses of sizes 2, 3, and 1, respectively. If ϕ is a cnf-formula of size s, we write $\ell_1(\phi), \ldots, \ell_s(\phi)$ for the s literals of ϕ in the left-to-right order in which they appear in ϕ. For example, in (1) we have $\ell_4(\phi) = \overline{x_3}$. When ϕ is clear from the context we write ℓ_i instead of $\ell_i(\phi)$. We use $\mathrm{var}(\phi)$ to denote the set of variables occurring in a formula ϕ.

Tree-width and path-width Let ϕ be a cnf-formula with variables $X_1, \ldots, X_n$ and clauses $C_1, \ldots, C_m$. The *constraint graph* of ϕ has one vertex for every variable of ϕ and two variables are connected by an edge if and only if there is a clause which contains them both. We identify the variables of a formula with the vertices of its constraint graph.

For a given a graph $G = (V, E)$, a *tree decomposition* of G is a pair (T, L), where T is a tree and L is a function $L : V(T) \to \mathcal{P}(V)$, that satisfies the following properties:

1. $\bigcup_{t \in V(T)} L(t) = V$,
2. for every $(u, v) \in E$, there is a $t \in V(T)$ such that $u, v \in L(t)$,

3. for every $v \in V$, the subgraph of T induced by $\{t \in V(T) \mid v \in L(t)\}$ is a connected subtree.

For later convenience we assume that T is a rooted tree. Note that a graph has multiple tree-decompositions.

Given a tree-decomposition, its *width* is defined as

$$\max_{t \in V(T)} L(t) - 1.$$

The *tree-width* of a graph is the minimum among the widths of its tree-decompositions.

The tree-width of a formula is defined as the tree-width of its constraint graph.

Claim 1. *Let G be a graph and let (T, L) be a tree-decomposition of G. Then, for every $S \subseteq V(G)$ that induces a clique, there is a $t \in V(T)$ such that $S \subseteq L(t)$.*

A *path decomposition* of a graph G is a tree-decomposition (T, L) such that T is a path. The *path-width* of a graph is the minimum among the widths of its path decompositions.

Quantified boolean formulas A *qbf* is a quantified Boolean formula of the form

$$\phi = Q_1 x_1 \cdots Q_q x_q(\phi'), \tag{2}$$

where $x_1, \ldots, x_q$ are propositional variables, the *matrix* ϕ' is a cnf-formula, and Q_i is either $\forall$ or $\exists$ for every $i \in \{1, \ldots, q\}$. The *size* of a qbf as in (2) is defined as the size of its matrix ϕ'. The tree-width (path-width) of a qbf is the tree-width (path-width) of its matrix. We say that $Q_1 x_1 \ldots Q_q x_q$ is the *prefix* of ϕ.

3 Leveled Formulas

In this section we state and prove the main lemma. This lemma is a reduction from n-leveled qbfs to $O(\log n)$-leveled qbfs, which is progress in our iterative argument. Before stating the lemma, we formalize the concept of leveled-qbf.

3.1 Definition of leveled qbf

Let n be a positive integer. An *n-leveled cnf-formula* is a cnf-formula ϕ in which its sequence of clauses is partitioned into *blocks* $B_1, \ldots, B_\ell$, where each block is a consecutive subsequence of clauses of ϕ, and its set of variables is partitioned into the same number of *groups* $G_1, \ldots, G_\ell$, each containing at most n variables, and such that for every $j \in \{1, \ldots, \ell - 1\}$ we have that every clause C in B_j has all its variables in $G_j \cup G_{j+1}$, and every clause C in B_ℓ has all its variables in G_ℓ. An *n-leveled qbf* is a quantified Boolean formula whose matrix is an n-leveled cnf-formula.

Observe that every qbf with n variables is an n-leveled qbf: put all clauses in a single block and all variables in a single group. However, when the sizes of the groups are limited, we get a nice structure:

Lemma 1. *Let n be a positive integer. Every n-leveled qbf has path-width at most $2n - 1$.*

Proof. Let ϕ be an n-leveled QBF with groups $G_1, \ldots, G_\ell$. Define (T, L) as the path decomposition of the matrix of ϕ where T is a path on vertices $t_1, \ldots, t_\ell$, and $L(t_i) := G_i \cup G_{i+1}$ for $i \in \{1, \ldots, \ell - 1\}$ and $L(t_\ell) := G_\ell$. Since each G_i has cardinality at most n, the claim follows. □

Now, we can formalize the statement of the main lemma.

Lemma 2. *There exist $c, d \geq 1$ and a polynomial-time algorithm that, for every $n, s \geq 1$, given an n-leveled qbf ϕ of size s, computes a $c \cdot |n|$-leveled qbf ψ of size $d \cdot s \cdot |n|$ such that $\phi \leftrightarrow \psi$.*

We devote the rest of the section to the proof of this lemma. In order to improve the readability of Boolean formulas, we use + for disjunction and $\cdot$ for conjunction.

3.2 Definition of θ

Let ϕ be a n-leveled qbf as in (2) whose matrix ϕ' is an n-leveled cnf-formula of size s with groups $G_1, \ldots, G_\ell$ and blocks $B_1, \ldots, B_\ell$. As a first step towards building ψ we define an intermediate formula θ. The formula θ contains variables $\tau_1, \ldots, \tau_s$, one for each literal in ϕ', and is defined as

$$\theta := Q_1 \boldsymbol{\tau}_1 \cdots Q_q \boldsymbol{\tau}_q (\text{NCONS}_\forall + (\text{CONS}_\exists \cdot \text{SAT}))$$

where

1. each $\boldsymbol{\tau}_j$, for $j \in [q]$, is the tuple of τ-variables corresponding to all the occurrences of the variable x_j in ϕ',
2. CONS_Q, for $Q \in \{\forall, \exists\}$, is a qbf to be defined later that is satisfied by an assignment to $\tau_1, \ldots, \tau_s$ if and only if all the variables from the same $\boldsymbol{\tau}_j$ with $Q_j = Q$ are given the same truth value,
3. NCONS_Q for $Q \in \{\forall, \exists\}$ is a qbf that is equivalent to the negation of CONS_Q,
4. SAT is a qbf to be defined later that is satisfied by an assignment to $\tau_1, \ldots, \tau_s$ if and only if every clause of ϕ' contains at least one literal $\ell_k = x^{(a)}$ such that τ_k is given value a.

This information about the constituents of θ is enough to prove the following claim.

Claim 2. $\phi \leftrightarrow \theta$

Proof. We need to prove both implications. In both cases we use a game in which two players, the existential player and the universal player, take rounds following the order of quantification of the formula to choose values for the variables quantified their way. The aim of the existential player is to show that the matrix of the formula can be made true while the aim of the universal player is to show him wrong.

In the following, for $j \in [q]$, we say that an assignment to the variables of $\boldsymbol{\tau}_j$ is *consistent* if they are given the same truth value, say $a \in \{0, 1\}$. In case the assignment is consistent, we say that a is the *corresponding assignment* for the variable x_j. Conversely, if a is an assignment to the variable x_j, the *corresponding consistent assignment* for the tuple $\boldsymbol{\tau}_j$ is the assignment that sets each variable in $\boldsymbol{\tau}_j$ to a. If an assignment to $\boldsymbol{\tau}_j$ is not consistent we call it inconsistent.

($\rightarrow$): Assume ϕ is true and let α be a winning strategy for the existential player in ϕ. We build another strategy β that guarantees him a win in θ. The construction of β will be based on the observation that, in the course of the game on θ, if the assignment given by the universal player to some $\boldsymbol{\tau}_j$ with $Q_j = \forall$ is inconsistent, then $\text{NCONS}_\forall$ is true irrespective of all other variables, and hence the matrix of θ is true. With this observation in hand, the strategy β is defined as follows: at round j with $Q_j = \exists$, if all $\boldsymbol{\tau}_1, \ldots, \boldsymbol{\tau}_{j-1}$ have been given consistent assignments up to this point

and $a_1, \ldots, a_{j-1} \in \{0,1\}$ are the corresponding assignments to the variables $x_1, \ldots, x_{j-1}$, let a_j be the assignment given to x_j by the strategy α in this position of the game on ϕ, and let the existential player assign value a_j to every variable in $\boldsymbol{\tau}_j$. If on the other hand some $\boldsymbol{\tau}_k$ with $k < j$ has been given an inconsistent assignment, let the existential player assign an arbitrary value (say 0) to every variable in $\boldsymbol{\tau}_j$. Using the observation above and the assumption that α is a winning strategy, it is not hard to see that β is a winning strategy.

($\leftarrow$): Assume θ is true and let β be a winning strategy for the existential player in θ. We build a strategy α for the existential player in ϕ. In this case the construction of α will be based on the observation that, in the course of the game on θ, as long as the universal player assigns consistent values to every $\boldsymbol{\tau}_j$ with $Q_j = \forall$, the assignment given by β to each new $\boldsymbol{\tau}_j$ with $Q_j = \exists$ must be consistent. To see this note that, if not, the universal player would have the option of staying consistent all the way until the end of the game in which case both $\text{NCONS}_\forall$ and $\text{CONS}_\exists$ would become false, thus making the matrix of θ false. With this observation in hand, the strategy α is defined as follows: at round j with $Q_j = \exists$, let $a_1, \ldots, a_{j-1} \in \{0,1\}$ be the assignment given to $x_1, \ldots, x_{j-1}$ up to this point, let $\mathbf{a}_1, \ldots, \mathbf{a}_{j-1}$ be the corresponding consistent assignments for $\boldsymbol{\tau}_1, \ldots, \boldsymbol{\tau}_{j-1}$, and let $\mathbf{a}_j$ be the assignment given by β to $\boldsymbol{\tau}_j$ in this position of the game on θ. By the observation above, since each $\mathbf{a}_k$ with $k < j$ and $Q_k = \forall$ is consistent by definition and each $\mathbf{a}_k$ with $k < j$ and $Q_j = \exists$ has been assigned according to the strategy β, the assignment $\mathbf{a}_j$ must also be consistent. Thus the existential player can set x_j to its corresponding value a_j and continue with the game.

We need to show that α is a winning strategy for the existential player on ϕ. First, if the existential player plays according to α, then the final assignment $a_1, \ldots, a_q$ that is reached in the game on ϕ is such that the corresponding assignment $\mathbf{a}_1, \ldots, \mathbf{a}_q$ in the game on ψ satisfies the matrix of θ. Since each $\mathbf{a}_j$ is consistent this means that SAT must be made true by $\mathbf{a}_1, \ldots, \mathbf{a}_q$, thus the matrix of ϕ is made true by $a_1, \ldots, a_q$. This shows that the existential player wins. □

Now, we show how to construct the qbf-formulas SAT, $\text{CONS}_\exists$ and $\text{NCONS}_\forall$. These formulas have the τ-variables as free variables and a new set of quantified variables for each literal in ϕ'. Recall that the τ-variables assign a truth value to each variable-ocurrence in ϕ'. The formula SAT will verify that these assignments satisfy all clauses of ϕ', the formula $\text{CONS}_\exists$ will verify that each existentially quantified variable is assigned consistently, and the formula $\text{NCONS}_\forall$ will verify that at least one universally quantified variable is assigned inconsistently.

3.3 Definition of sat

For every $i \in [s+1]$, we have variables μ_i and ν_i. By scanning its literals left-to-right, the formula checks that every clause of ϕ' contains at least one literal $\ell_k = x^{(a)}$ such that τ_k is given value a. To do so, μ_i and ν_i indicate the status of this process when exactly $i-1$ literals have been scanned. The intended meaning of the variables is the following:

- μ_i = "just before scanning ℓ_i, the clauses already completely scanned are satisfied, and the current clause is not satisfied yet".
- ν_i = "just before scanning ℓ_i, the clauses already completely scanned are satisfied, and the current clause is satisfied as well".

Note that ℓ_{s+1} is not a literal. Therefore, "just before scanning ℓ_{s+1}" means "just after scanning the last literal" in this case. Also, variables μ_1 and ν_1 are initialized to true and false, respectively.

We want to make sure that at position $i = s + 1$, i.e. after scanning the last literal, μ_{s+1} is true. Later, we will axiomatize the transition between positions i and $i + 1$. That will define μ_{i+1} and ν_{i+1} depending on μ_i, ν_i and ℓ_i according to its intended meaning. We will axiomatize this into the formula $\text{SAT}(i)$. Then, SAT is defined as

$$\text{SAT} := \exists \boldsymbol{\mu} \exists \boldsymbol{\nu} \left(\mu_1 \cdot \overline{\nu_1} \cdot \prod_{i=1}^{s} \text{SAT}(i) \cdot \mu_{s+1} \right)$$

where $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_{s+1})$ and $\boldsymbol{\nu} = (\nu_1, \ldots, \nu_{s+1})$.

Next, we formalize $\text{SAT}(i)$. For every $i \in [s]$, let $a_i \in \{0, 1\}$ denote the sign of ℓ_i, the i-th literal of ϕ', and let $k_i \in \{0, 1\}$ be the predicate that indicates whether ℓ_i is the last in literal its clause. Then, $\text{SAT}(i)$ is the conjunction of the following formulas:

$$\begin{aligned}
\mu_{i+1} \;&\leftrightarrow\; \overline{k_i}\,\mu_i\,a_i\,\overline{\tau_i} \;+\; \overline{k_i}\,\mu_i\,\overline{a_i}\,\tau_i \;+\; k_i\,\mu_i\,a_i\,\tau_i \;+\; k_i\,\mu_i\,\overline{a_i}\,\overline{\tau_i} \;+\; k_i\,\nu_i, \\
\nu_{i+1} \;&\leftrightarrow\; \overline{k_i}\,\mu_i\,a_i\,\tau_i \;+\; \overline{k_i}\,\mu_i\,\overline{a_i}\,\overline{\tau_i} \;+\; \overline{k_i}\,\nu_i.
\end{aligned}$$

In words, the axiomatization states that μ_{i+1} holds in one of three cases: 1) if ℓ_i is the last literal in its clause and the clause has been satisfied by a previous literal ($k_i\nu_i$), or 2) if ℓ_i is the last literal in its clause, this clause is not yet satisfied by a previous literal, but the truth assignment satisfies the current one ($k_i\mu_i a_i\tau_i + k_i\mu_i\overline{a_i\tau_i}$), or 3) if ℓ_i is not the last literal in its clause, this clause is not yet satisfied by a previous literal, and the truth assignment does not satisfy the current one either ($\overline{k_i}\mu_i a_i\overline{\tau_i} + \overline{k_i}\mu_i\overline{a_i}\tau_i$). The axiomatization of ν_{i+1} is similar.

Note that these two formulas can be written in cnf by writing $\leftrightarrow$ in terms of conjunctions and disjunctions and by distributing disjunctions over conjunctions. We call *i-link* a clause that contains variables only with indices i and $i+1$. Observe for later use that all clauses in the resulting cnf-formulas for $\text{SAT}(i)$ are i-links. Also, the size of SAT written in cnf is $c \cdot s$ for some constant $c \geq 1$.

3.4 Definition of cons$_\exists$

The construction of $\text{CONS}_\exists$ is a bit more complicated. It uses universally quantified variables $\{\pi_1, \ldots, \pi_s\}$ as pointers to the literals of ϕ', in one-to-one correspondance with $\{\tau_1, \ldots, \tau_s\}$. We say that pointer π_i points to literal ℓ_i. If x is the underlying variable of ℓ_i, we say that π_i points to x. Pointers that are set to true are called *activated*. We say that a pointer has been scanned if its pointed literal has been scanned. The formula checks the following: whenever exactly two pointers are activated and they point to occurrences of the same existentially quantified variable, then the truth values assigned to the pointed literals are consistent. To refer to a variable, we do not encode its identifier directly. Instead, we encode the parity of its group and its index inside this group. This is enough information to distinguish between different variables in the same or neighbouring blocks. This fact is key to our argument and will be proved later in Claim 3. The point is that this compact encoding uses only $|n| + 1$ bits per occurrence, where n is the number of variables per group, which may be much smaller than the total number of variables.

The formula uses the following variables for $i \in [s + 1]$:

- ξ_i = "just before scanning ℓ_i, all the activated pointers already scanned point to an existentially quantified variable".

- $\sigma_{i,k}$ = "just before scanning ℓ_i, exactly k activated pointers have been scanned".
- $\chi_{i,k}$ = "just before scanning ℓ_i, exactly one activated pointer has been scanned and there have been k changes of block between the pointed literal and position i, or exactly two have been scanned and there have been exactly k changes of block between the pointed literals".
- ω_i = "just before scanning ℓ_i, exactly one activated pointer has been scanned and the parity of the group of the pointed variable is equal to the parity of the block of the clause of the pointed literal, or exactly two have been scanned and the groups of the pointed variables are the same".
- κ_i = "just before scanning ℓ_i, exactly one activated pointer has been scanned and the τ-variable at the pointed position is true, or exactly two have been scanned and the truth values of the τ-variables at the pointed positions are the same".
- $\lambda_{i,b}$ = "just before scanning ℓ_i, exactly one activated pointer has been scanned and the b-th bit of the index of the pointed variable in its group is 1, or exactly two have been scanned and the b-th bit of the indices of the pointed variables in their respective groups are the same".

The variables at step $i+1$ will be axiomatized in terms of the variables at step i and ℓ_i in the formula $\textsc{cons}_\exists(i)$. The formula $\textsc{cons}_\exists$ also requires a consistency condition for all possible combinations of activated pointers. For a given combination of these pointers, the consistency condition holds if: either there is a problem with the pointers (there are not exactly two pointers activated or one is not pointing to an existentially quantified variable), or the pointed variables are not comparable (are not of the same group or do not have the same index in the group) or, they are comparable and both receive the same truth value. This consistency condition will be encoded in the formula $\textsc{cons}_\exists^{\text{acc}}$. Also, the value of the variables at position $i=1$ will be encoded in the formula $\textsc{cons}_\exists^{\text{ini}}$. Now,

$$\textsc{cons}_\exists := \forall\boldsymbol{\pi}\exists\boldsymbol{\xi}\exists\boldsymbol{\sigma}\exists\boldsymbol{\chi}\exists\boldsymbol{\omega}\exists\boldsymbol{\kappa}\exists\boldsymbol{\lambda}\left(\textsc{cons}_\exists^{\text{ini}}\cdot\prod_{i=1}^{s}\textsc{cons}_\exists(i)\cdot\textsc{cons}_\exists^{\text{acc}}\right)$$

where $\boldsymbol{\pi}=(\pi_i\,|\,1\le i\le s)$, $\boldsymbol{\xi}=(\xi_i\,|\,1\le i\le s+1)$, $\boldsymbol{\sigma}=(\sigma_{i,k}\,|\,1\le i\le s+1, 0\le k\le 2)$, $\boldsymbol{\chi}=(\chi_{i,k}\,|\,1\le i\le s+1, 0\le k\le 1)$, $\boldsymbol{\omega}=(\omega_i\,|\,1\le i\le s+1)$, $\boldsymbol{\kappa}=(\kappa_i\,|\,1\le i\le s+1)$ and $\boldsymbol{\lambda}=(\lambda_{i,b}\,|\,1\le i\le s+1, 1\le b\le |n|)$.

Next we axiomatize the introduced variables, but before that we need to introduce some notation.

Let $g_i\in[\ell]$ be the group-number of the variable underlying literal ℓ_i, let $n_i\in[|G_{g_i}|]$ be the index of this variable within G_{g_i}, and recall $a_i\in\{0,1\}$ denotes the sign of ℓ_i. For every $i\in[s]$, let $h_i\in\{0,1\}$ be the predicate that indicates whether the i-th literal ℓ_i is the last in its block or not (recall that the blocks are subsequences of consecutive clauses that partition the sequence of clauses), and recall that $k_i\in\{0,1\}$ is the predicate that indicates whether the i-th literal ℓ_i is the last in its clause or not. Next we encode the quantification of ϕ in a way that the type of quantification of each variable can be recovered from each of its occurrences: for every $i\in[s]$, let $q_i\in\{0,1\}$ be the predicate that indicates whether the variable that underlies the i-th literal ℓ_i is universally or existentially quantified in ϕ.

Finally, observe that the definition of leveled formula implies that if $b_i\in[\ell]$ is the number of the block that contains the clause to which the i-th literal belongs, then the group-number g_i is

either b_i or $b_i + 1$ whenever $1 \leq b_i \leq \ell - 1$, and is equal to ℓ if $b_i = \ell$. Accordingly, let $e_i \in \{0, 1\}$ be such that $g_i = b_i - e_i + 1$ for every $i \in [s]$. In other words, e_i indicates whether the parities of g_i and b_i agree or not.

The following claim shows that, although the number ℓ of groups is in general unbounded, a constant number of bits of information are enough to tell if the underlying variables of two literals belong to the same group:

Claim 3. *Let i, j be such that $1 \leq i < j \leq s$. Then, the underlying variables of ℓ_i and ℓ_j belong to the same group if and only if one of the following conditions holds:*

1. *$e_i = e_j$ and $b_i = b_j$, or*
2. *$e_i = 0$, $e_j = 1$, and $b_i = b_j - 1$.*

Proof. For the only if side, we have $g_i = g_j$. Then, $b_i - e_i = b_j - e_j$ and also b_i is either b_j or $b_j - 1$. If $b_i = b_j$, then $e_i = e_j$. If $b_i = b_j - 1$, then necessarily $e_i = 0$ and $e_j = 1$.

For the if side, in the first case, $g_i = b_i - e_i + 1 = b_j - e_j + 1 = g_j$. In the second case, $g_i = b_i - e_i + 1 = b_j - 1 + 1 = b_j - e_j + 1 = g_j$. Therefore, $g_i = g_j$. □

Using this claim, we axiomatize $\textsc{cons}_\exists(i)$ as the conjunction of the following formulas:

$$\begin{aligned}
\xi_{i+1} &\leftrightarrow \overline{\pi_i}\,\xi_i \;+\; \pi_i\,\xi_i\,q_i \\
\sigma_{i+1,0} &\leftrightarrow \sigma_{i,0}\,\overline{\pi_i} \\
\sigma_{i+1,1} &\leftrightarrow \sigma_{i,0}\,\pi_i \;+\; \sigma_{i,1}\,\overline{\pi_i} \\
\sigma_{i+1,2} &\leftrightarrow \sigma_{i,1}\,\pi_i \;+\; \sigma_{i,2}\,\overline{\pi_i} \\
\chi_{i+1,0} &\leftrightarrow \sigma_{i,0}\,\pi_i\,\overline{h_i} \;+\; \sigma_{i,1}\,\overline{\pi_i}\,\chi_{i,0}\,\overline{h_i} \;+\; \sigma_{i,1}\,\pi_i\,\chi_{i,0} \;+\; \sigma_{i,2}\,\chi_{i,0} \\
\chi_{i+1,1} &\leftrightarrow \sigma_{i,0}\,\pi_i\,h_i \;+\; \sigma_{i,1}\,\overline{\pi_i}\,\chi_{i,0}\,h_i \;+\; \sigma_{i,1}\,\overline{\pi_i}\,\chi_{i,1}\,\overline{h_i} \;+\; \sigma_{i,1}\,\pi_i\,\chi_{i,1} \;+\; \sigma_{i,2}\,\chi_{i,1} \\
\omega_{i+1} &\leftrightarrow \sigma_{i,0}\,\pi_i\,e_i \;+\; \sigma_{i,1}\,\overline{\pi_i}\,\omega_i \;+\; \sigma_{i,1}\,\pi_i\,(\,\chi_{i,0}\,\omega_i\,e_i \;+\; \chi_{i,0}\,\overline{\omega_i}\,\overline{e_i} \;+\; \chi_{i,1}\,\overline{\omega_i}\,e_i) \;+\; \sigma_{i,2}\,\omega_i \\
\kappa_{i+1} &\leftrightarrow \sigma_{i,0}\,\pi_i\,\tau_i \;+\; \sigma_{i,1}\,\overline{\pi_i}\,\kappa_i \;+\; \sigma_{i,1}\,\pi_i\,\kappa_i\,\tau_i \;+\; \sigma_{i,1}\,\pi_i\,\overline{\kappa_i}\,\overline{\tau_i} \;+\; \sigma_{i,2}\,\kappa_i
\end{aligned}$$

and, for all $b \in [|n|]$,

$$\lambda_{i+1,b} \;\leftrightarrow\; \sigma_{i,0}\,\pi_i\,n_{i,b} \;+\; \sigma_{i,1}\,\overline{\pi_i}\,\lambda_{i,b} \;+\; \sigma_{i,1}\,\pi_i\,\lambda_{i,b}\,n_{i,b} \;+\; \sigma_{i,1}\,\pi_i\,\overline{\lambda_{i,b}}\,\overline{n_{i,b}} \;+\; \sigma_{i,2}\,\lambda_{i,b}$$

where $n_{i,b}$ is the b-th bit of the binary encoding of n_i.

Also, we define $\textsc{cons}_\exists^{\text{ini}}$ as the conjunction of the following unit clauses:

$$\xi_1, \sigma_{1,0}, \overline{\sigma_{1,1}}, \overline{\sigma_{1,2}}, \overline{\chi_{1,0}}, \overline{\chi_{1,1}}, \overline{\omega_1}, \overline{\kappa_1}, \overline{\lambda_{1,1}}, \ldots, \overline{\lambda_{1,|n|}}.$$

Furthermore, we define $\textsc{cons}_\exists^{\text{acc}}$ as the following clause:

$$\overline{\xi_{s+1}} + \overline{\sigma_{s+1,2}} + \overline{\omega_{s+1}} + \sum_{b=1}^{|n|} \overline{\lambda_{s+1,b}} + \kappa_{s+1}.$$

Again, note that each of these formulas can be written in cnf just by writing $\leftrightarrow$ in terms of conjunctions and disjunctions and by distributing disjunctions over conjunctions, and that the clauses in the resulting cnf-formulas for $\textsc{cons}_\exists(i)$ are i-links: the (first) index of the variables they contain is either i or $i+1$. Also, the size of $\textsc{cons}_\exists$ written in cnf is $c \cdot s \cdot |n|$ for some constant $c \geq 1$.

3.5 Definition of ncons$_\forall$

The formula NCONS$_\forall$ is very similar to CONS$_\exists$, since it verifies for universally quantified variables exactly the opposite of what CONS$_\exists$ verifies for existentially quantified variables. For this reason, we proceed to its axiomatization directly.

The formula NCONS$_\forall$ is defined as

$$\text{NCONS}_\forall := \exists\boldsymbol{\pi}\exists\boldsymbol{\xi}\exists\boldsymbol{\sigma}\exists\boldsymbol{\chi}\exists\boldsymbol{\omega}\exists\boldsymbol{\kappa}\exists\boldsymbol{\lambda}\left(\text{NCONS}_\forall^{\text{ini}} \cdot \prod_{i=1}^{s} \text{NCONS}_\forall(i) \cdot \text{NCONS}_\forall^{\text{acc}}\right)$$

where $\boldsymbol{\pi}$, $\boldsymbol{\xi}$, $\boldsymbol{\sigma}$, $\boldsymbol{\chi}$, $\boldsymbol{\omega}$, $\boldsymbol{\kappa}$, $\boldsymbol{\lambda}$ are defined as before, $\text{NCONS}_\forall^{\text{ini}} := \text{CONS}_\exists^{\text{ini}}$, the formula $\text{NCONS}_\forall(i)$ is axiomatized identically to $\text{CONS}_\exists(i)$ except by replacing every occurrence of q_i by $\overline{q_i}$ for every $i \in [s]$, and the formula $\text{NCONS}_\forall^{\text{acc}}$ is the negation of $\text{CONS}_\exists^{\text{acc}}$, i.e. the following set of unit clauses:

$$\xi_{s+1}, \sigma_{s+1,2}, \omega_{s+1}, \lambda_{s+1,1}, \ldots, \lambda_{s+1,|n|}, \overline{\kappa_{s+1}}.$$

In cnf, the formula $\text{NCONS}_\forall(i)$ is again a set of i-links, and its size is $c \cdot s \cdot |n|$ for some $c \geq 1$.

3.6 Converting θ to leveled-qbf

Recall that θ was defined as $Q_1\boldsymbol{\tau}_1 \cdots Q_q\boldsymbol{\tau}_q(\text{NCONS}_\forall + (\text{CONS}_\exists \cdot \text{SAT}))$. By writing this formula in prenex form, we obtain the equivalent formula

$$\mathbf{Qz}\ (\text{NCONS}'_\forall + (\text{CONS}'_\exists \cdot \text{SAT}'))$$

where $\mathbf{Qz}$ is the appropriate prefix of quantified variables and the primed formulas are the matrices of the corresponding non-primed qbfs. We would like to write it as a leveled-qbf.

Let a and b be two new variables and let ϑ be the conjunction of the following formulas:

$$\begin{array}{l} a + \text{NCONS}'_\forall \\ b + \text{NCONS}'_\forall \\ \bar{a} + \text{CONS}'_\exists \\ \bar{b} + \text{SAT}' \end{array}$$

It is easy to see that

$$\exists a\exists b(\vartheta) \leftrightarrow \text{NCONS}'_\forall + (\text{CONS}'_\exists \cdot \text{SAT}').$$

We write ϑ in cnf. For the first disjunction $a + \text{NCONS}'_\forall$, it is enough to add a to every clause of $\text{NCONS}'_\forall$, and similarly for the others. Note that, except for the variables a and b, the result is a conjunction of i-links.

In order to make them proper i-links, we introduce new variables $\{a_1, \ldots, a_{s+1}\}$ and $\{b_1, \ldots, b_{s+1}\}$, and clauses $a_i \leftrightarrow a_{i+1}$ and $b_i \leftrightarrow b_{i+1}$ for every $i \in [s]$ to mantain consistency between the introduced variables. Now, we replace each occurrence of a and b in an improper i-link by a_i and b_i respectively. Let ψ' be the resulting formula.

Finally, define

$$\psi := \mathbf{Qz}\exists\mathbf{a}\exists\mathbf{b}(\psi')$$

where $\mathbf{a} = (a_1, \ldots, a_{s+1})$ and $\mathbf{b} = (b_1, \ldots, b_{s+1})$. Note that the construction guarantees $\psi \leftrightarrow \theta$, and by Claim 2, $\psi \leftrightarrow \phi$.

We partition the variables of ψ in groups $H_1, \ldots, H_{s+1}$ where group H_i is the set of variables with (first) index i. We also partition the clauses of ψ in blocks $C_1, \ldots, C_{s+1}$ where block C_i is the set of i-links of ψ. Note that, by the definition of i-link, all variables in C_i are contained in $H_i \cup H_{i+1}$. Therefore, ψ is a leveled-qbf with groups $H_1, \ldots, H_{s+1}$ and blocks $C_1, \ldots, C_{s+1}$.

Now, for every $i \in [s+1]$, the size of H_i is the number of variables with index i in ψ, namely $c \cdot |n|$ for some constant $c \geq 1$. Also, the size of ψ is $d \cdot s \cdot |n|$ for some constant $d \geq 1$. Therefore, ψ is a $c \cdot |n|$-leveled qbf of size $d \cdot s \cdot |n|$ such that $\phi \leftrightarrow \psi$.

Finally, it is clear that all the steps to produce ψ from ϕ can be performed in time polynomial in s, thus finishing the proof.

4 Main Theorem

In this section we prove the main result of the paper.

Theorem 1. *There exists an integer $w \geq 1$ such that QBF on inputs of path-width at most w is PSPACE-complete.*

Proof. We show that there exists a constant $n_0 \geq 1$ and a polynomial-time reduction from the canonical PSPACE-complete problem QBF to the restriction of QBF itself to n_0-leveled qbfs. Then the result will follow by setting the path-width to $w = 2n_0 - 1$ and applying Lemma 1.

Let c and d be the constants from the end of section 3. We choose the constant n_0 large enough so that whenever $N \geq n_0$ the following conditions are satisfied:

1. $c \cdot |N| < N$,
2. $c \cdot |c \cdot |N|| \leq \log N$,
3. $(2 \log^* N)(\log |N|) \leq \log N$,
4. $d^{2 \log^* N} \leq \log N$.

All these conditions can be met simultaneously. The idea of the reduction is to start with an arbitrary qbf formula ϕ_0 with N_0 variables and size S_0, view it as an N_0-leveled qbf, and apply Lemma 2 repeatedly until we get a n_0-leveled qbf for the large fixed constant n_0. Since the final formula will be equivalent to ϕ_0, we just need to make sure that this process terminates in a small number of iterations and that the size of the resulting formula is polynomial in S_0. We formalize this below.

Let ϕ_0 be an arbitrary qbf formula with N_0 variables and size S_0. In particular ϕ_0 is an N_0-leveled qbf of size S_0. If $N_0 \leq n_0$ then ϕ_0 is already n_0-leveled and there is nothing to do. Assume then $N_0 > n_0$. We apply Lemma 2 to get an N_1-leveled qbf of size S_1 where $N_1 = c \cdot |N_0|$ and $S_1 = d \cdot S_0 \cdot |N_0|$. By condition 1 on n_0 we get $N_1 < N_0$, which is progress. Repeating this we get a sequence of formulas $\phi_0, \phi_1, \ldots, \phi_t$, where ϕ_i is an N_i-leveled qbf of size S_i with

1. $N_i = c \cdot |N_{i-1}|$, and
2. $S_i = d^i \cdot S_0 \cdot \prod_{j=0}^{i-1} |N_j|$,

for $i \geq 1$. We stop the process at the first $i = t$ such that $N_t \leq n_0$. We claim that $t \leq 2 \log^* N_0$ and that $S_t \leq S_0 \cdot N_0 \cdot \log N_0$. This will be enough, since then the algorithm that computes ϕ_t from ϕ_0 is the required reduction as it runs in time polynomial in the size of the formula, and $\phi_0 \leftrightarrow \phi_t$.

Claim 4. *It holds that* $t \leq 2 \log^* N_0$.

Proof. First, by conditions 1 and 2 on n_0 we have

1. $N_i = c \cdot |N_{i-1}| < N_{i-1}$, and
2. $N_{i+1} = c \cdot |N_i| = c \cdot |c \cdot |N_{i-1}|| \leq \log N_{i-1}$

for every $i \geq 1$ such that $N_{i-1} > n_0$. In particular, this means that the process terminates and t exists. Unfolding the second inequality gives

$$N_{t-1} \leq \log^{(\lfloor (t-1)/2 \rfloor)} N_0.$$

However, by the choice of t we have $N_{t-1} > n_0 \geq 1$, which means that $\lfloor (t-1)/2 \rfloor < \log^* N_0$ and therefore $t \leq 2 \log^* N_0$. □

Given this bound on t, we bound S_t. We have

$$S_t = d^t \cdot S_0 \cdot \prod_{j=0}^{t-1} |N_j| \leq d^t \cdot S_0 \cdot |N_0|^t,$$

where in the inequality we used the fact that $N_i \leq N_{i-1}$ for every $i \geq 1$ such that $N_{i-1} > n_0$, by condition 1 on n_0. Now:

$$|N_0|^t \leq 2^{(2 \log^* N_0)(\log |N_0|)} \leq 2^{\log N_0} = N_0.$$

In the first inequality we used the bound on t, and in the second we used the assumption that $N_0 \geq n_0$ and condition 3 on n_0. Altogether, this gives

$$S_t \leq d^{2 \log^* N_0} \cdot S_0 \cdot N_0 \leq S_0 \cdot N_0 \cdot \log N_0,$$

which concludes the proof. Again, we used the assumption that $N_0 \geq n_0$ and condition 4 on n_0. □

5 Q-resolution and respectful tree-width

For this section, it is useful to note that a qbf can be written as

$$\phi = Q_1 X_1 \cdots Q_q X_q(\phi') \tag{3}$$

where $X_1, \ldots, X_q$ are disjoint sequences of propositional variables, and $Q_i \neq Q_{i+1}$ for $1 \leq i < q$. Of course $Q_i X_i$ means $Q_i x_1^i \ldots Q_i x_\ell^i$ for $X_i := (x_1^i, \ldots, x_\ell^i)$. Also, we say that X_i is a *quantifier block* of ϕ. Note that logical equivalence is preserved upon reordering of the variables within the same quantifier block. To establish an order between the variables in the prefix of a qbf that accounts for the quantifier blocks, we say that x *is after* y in ϕ for $x, y \in \mathrm{var}(\phi)$ if x and y belong to quantifier blocks X_i and X_j, respectively, with $i > j$. Also, for this section, all the literals in a clause have different underlying variables and, in particular, all clauses are non-tautological.

5.1 The Q-resolution proof system

In this section we define and compare some proof systems for qbfs. In [9], in an attempt to generalize resolution to qbfs, Büning et al. introduced the Q-resolution proof system, consisting of the following rules:

1. $\frac{C}{\square}$, if every $x \in \text{var}(C)$ is quantified universally.
2. $\frac{C \vee x \quad D \vee \overline{x}}{(C' \vee D')''}$, if x is quantified existentially, where
   (a) C' (resp. D') is equal to C (resp. D) except for the literals whose underlying variable is quantified universally and is after every existentially quantified variable y in $\text{var}(C)$ (resp. $\text{var}(D)$) in ϕ, and
   (b) $(C' \vee D')''$ is 1 if $(C' \vee D')$ is tautological and, otherwise, is equal to $(C' \vee D')$ except for the literals whose underlying variable is quantified universally and is after every existentially quantified variable y in $\text{var}(C') \cup \text{var}(D')$ in ϕ.

Later, Pan and Vardi [10] extended the symbolic quantifier elimination approach from cnf formulas to qbfs. They introduce a qbf solver that produces multi-resolution [13] refutations. Even though they use OBDDs to represent the clauses, the proof system that is implicit in their algorithm has the following two rules:

1. $\frac{C \vee x}{C}$, if x is quantified universally and no $y \in \text{var}(C)$ is after x in ϕ.
2. $\frac{C \vee x \quad D \vee \overline{x}}{C \vee D}$, if x is quantified existentially and no $y \in \text{var}(C) \cup \text{var}(D)$ is after x in ϕ.

In this work, we will call this proof system *weak Q-resolution*.

We introduce a simpler proof system, in the mold of weak Q-resolution, with the following rules:

1. $\frac{C \vee x}{C}$, if x is quantified universally and no $y \in \text{var}(C)$ is after x in ϕ.
2. $\frac{C \vee x \quad D \vee \overline{x}}{C \vee D}$, if x is quantified existentially.

For the moment, let us call this system *Q*-resolution*. Note that it is stronger than weak Q-resolution, since their only difference is that Q*-resolution weakens the restrictions to apply its second rule.

We say that a proof system P' *p-simulates* a proof system P if, whenever a contradiction has a P-refutation size s, it also has a P'-refutation of size polynomial in s. Also, we say that two proof systems are *p-equivalent* if they p-simulate each other. We show that Q*-resolution is, in fact, p-equivalent to Q-resolution:

Lemma 3. *The proof systems Q-resolution and Q*-resolution are p-equivalent.*

Proof. Let R_1 and R_2 be rules 1. and 2. of Q-resolution, and let R_1^* and R_2^* be rules 1. and 2. of Q*-resolution. First, we show that Q*-resolution p-simulates Q-resolution. To do so, we show that every Q-resolution step can be simulated by several Q*-resolution steps. To simulate R_1, if C is a purely universal clause, we obtain a Q*-resolution refutation of C by applying R_1^* repeatedly $|C|$

times, always on the literal whose underlying variable is the right-most in the prefix. To simulate R_2 on clauses C and D, again, repeatedly apply R_1^* on the universally quantified variables of C and D that are after every existentially quantified variable in its clause in right-to-left order, then apply R_2^* on the resulting clauses, and finally, repeteadly apply R_1^* on the universally quantified variables of the resulting clause that are after every existentially quantified variable, in right-to-left order. These are, at most, $|C| + |D|$ steps.

Second, we show that Q-resolution *p*-simulates Q*-resolution. Let $C_1^*, \ldots, C_\ell^*$ be a Q*-resolution refutation. For $i \in \{1, \ldots, \ell\}$ let

$$C_i := \begin{cases} C_i^* & \text{if } C_i^* \text{ is an initial clause,} \\ C_j & \text{if } C_i^* = R_1^*(C_j^*) \text{ for some } j < i, \\ R_2(C_j, C_k) & \text{if } C_i^* = R_2^*(C_j^*, C_k^*) \text{ with } j, k < i. \end{cases}$$

First we want to see that $C_1, \ldots, C_\ell$ is a valid Q-resolution derivation. It is clear by definition that every C_i is either an initial clause or the result of applying R_2, since in the second case C_i is already in the refutation. It remains to be seen that $\square$ can be derived from $C_1, \ldots, C_\ell$ in one more step. For that, it is enough to show that, for every $i \in \{1, \ldots, \ell\}$, we have that $C_i = C_i^*$ if C_i^* is an initial clause and that, otherwise, C_i subsumes $C_i^* \vee A_i$ for some purely universal clause A_i whose literals are after every existentially quantified variable of C_i^* in the prefix. If we suceed, just note that C_ℓ subsumes $(C_\ell^* \vee A_\ell) = (\square \vee A_\ell) = A_\ell$, and, since A_ℓ is a purely universal clause, we apply R_1 to C_ℓ to obtain $\square$. We are left to prove the claim. We will proceed by cases according to the definition of C_i^*. First, it is clear by definition that $C_i = C_i^*$ if C_i^* is an initial clause. Second, if $C_i^* = R_1^*(C_j^*)$ for some $j < i$, let l_i be the universally quantified literal that is in C_j^* and not in C_i^* and let $A_i := A_j \vee l_i$. Since C_j subsumed $C_j^* \vee A_j$, it is clear that C_i subsumes $C_i^* \vee A_i$. Third, we have that C_j subsumes $C_j^* \vee A_j$ and C_k subsumes $C_k^* \vee A_k$. By the definition of the rule and the conditions on A_j and A_k, we have that $R_2(C_j, C_k) = R_2(C_j^* \vee A_j, C_k^* \vee A_k)$. Therefore, $R_2(C_j, C_k)$ subsumes $R_2^*(C_j^* \vee A_j, C_k^* \vee A_k)$, this is, C_i subsumes C_i^* and, therefore, $C_i^* \vee A_i$ for $A_i := \emptyset$. $\square$

Since both proof systems are *p*-equivalent, to simplify notation, we will refer to both as Q-resolution for the rest of the section.

Now, for a qbf ϕ with matrix ϕ' and for variables x, y quantified existentially and universally respectively in ϕ, we define

$$\phi'^{(\exists x)} := \{C \vee D \mid C \vee x \in \phi' \text{ and } D \vee \overline{x} \in \phi'\} \cup \{C \in \phi' \mid x \notin \text{var}(C)\}, \text{ and}$$
$$\phi'^{(\forall x)} := \{C \mid C \vee x \in \phi' \text{ or } C \vee \overline{x} \in \phi'\} \cup \{C \in \phi' \mid x \notin \text{var}(C)\}.$$

We write $\phi'^{(Q_1 x_1, Q_2 x_2)}$ instead of $(\phi'^{(Q_2 x_2)})^{(Q_1 x_1)}$. Note that $x \notin \text{var}(\phi^{(Qx)})$. We prove the following lemma:

Lemma 4. *Let ψ be a cnf formula and let $\mathbf{Q}$ be any prefix. Then,*

1. $\mathbf{Q}\psi^{(\exists x)} \models \mathbf{Q}\exists x\psi$, *and*
2. $\mathbf{Q}\psi^{(\forall x)} \models \mathbf{Q}\forall x\psi$.

Proof. For the first claim, let A be an assignment that satisfies $\psi^{(\exists x)}$. Let A_0, A_1 be extensions of A that assign $x := 0$ and $x := 1$ respectively. If neither satisfies ψ, then ψ contains at least a pair of clauses $C \vee x$ and $D \vee \overline{x}$ such that $A_0(C) = 0$ and $A_1(D) = 0$. But then, $C \vee D$ belongs to $\psi^{(\exists x)}$ and $A(C \vee D) = 0$ causing a contradiction. Therefore, since at least one of A_0 and A_1 satisfies ψ, we have that A satisfies $\exists x\psi$. For the second claim, just note that $\psi^{(\forall x)} \models \psi$. $\square$

Note that completeness of weak Q-resolution (and therefore, also Q-resolution) is proved by repeated applications of this lemma: let $\mathbf{Q}$ be the prefix of ϕ. Then, $\phi'^{(\mathbf{Q})}$ is either empty, and therefore the formula is true, or contains just $\square$, and therefore the formula is false.

Various efforts have been directed to determine families of qbfs for which the Q-resolution proof system is polynomially bounded. Aspvall et al. [14] showed that (weak) Q-resolution is polynomially bounded for bijunctive-qbfs, that is, formulas with at most two literal per clause. Later, Büning et al. [9] showed the same for Horn-qbfs. They also proved that extended-Horn qbfs, that is, qbfs in which the existentially quantified part of each clause is Horn and the universal part is arbitrary, require exponential-size Q-resolution refutations.

Observe that Theorem 1 implies that, unless NP=PSPACE, no proof-system is polynomially bounded for qbfs of bounded tree-width (and even path-width), as otherwise one could guess a polynomial-size refutation and check it in polynomial time. However, some families of qbfs with bounded tree-width have polynomial-size Q-resolution refutations. For example, if we allow only existential quantifiers, the problem becomes equivalent to boolean satisfiability of cnfs (CNF-SAT), and Alekhnovich and Razborov [15] showed that cnfs of bounded branch-width (which is equivalent to bounded tree-width) have polynomial-size resolution (and therefore, (weak) Q-resolution) refutations. We devote the rest of the section to describe a larger family of qbfs with bounded tree-width for which (weak) Q-resolution is polynomially bounded.

5.2 Respectful tree-width

As defined in section 2, the tree-width of a qbf is the tree-width of its matrix, and therefore, it is independent of its prefix. Multiple algorithms on cnfs that are tractable on instances with bounded tree-width are not applicable to qbfs, since the tree decompositions that they use do not mesh well with the quantification of the variables. To tackle this problem, Chen and Dalmau [16] introduced what we call here *respectful tree-width*, a concept analogous to tree-width, but on tree decompositions that are, in some sense, respectful with the prefix of the formula, so that the algorithms for cnfs make sense.

Let ϕ be a qbf and let (T, L) be a tree decomposition of its matrix. Let r be the root of T. Define t_x as the closest vertex to r in T such that $x \in L(t_x)$. For a pair of variables $x, y \in \mathrm{var}(\phi)$, we say that x *is under* y if $t_x \neq t_y$ and t_y is in the (unique) path from r to t_x in T. We say that (T, L) is *respectful* with the prefix of ϕ if, for every pair of variables $x, y \in \mathrm{var}(\phi)$, if x is under y, then y is not after x. A *respectful tree decomposition* of ϕ is one that is respectful with its prefix. The *respectful tree-width* of ϕ is the minimum width among its respectful tree decompositions.

The main result of this section is that Q-resolution is polynomially bounded on qbfs of bounded respectful tree-width. The proof of this lemma makes use of a construction on graphs defined by Dechter and Pearl [11] named *induced graph*.

A pair $(H, \prec)$ is an induced graph of G if $\prec$ defines a strict total order on the vertices of G, and H is the closure of G under the following operation: for every $x, y, z \in V(H)$ such that $x \prec z$ and $y \prec z$, if (x, z) and (y, z) are edges, add (x, y) as an edge. The *width* of an induced graph is $\max_{x \in V(H)} |\{(x, y) \in E(H) \mid y \prec x\}|$. The *induced width* of a graph is the minimum among the widths of its induced graphs.

Given G and $\prec$, the usual way to obtain H, as proposed by Dechter and Pearl, is through the following process: one vertex of $V(H)$ at the time and in order opposite to $\prec$, add edges (x, y) for every pair x, y of neighbors of the current vertex z such that $x \prec z$ and $y \prec z$.

Let ϕ be a qbf and let $(H, \prec)$ be an induced graph of its matrix. We say that $(H, \prec)$ is *respectful* with the prefix of ϕ if, for every pair of variables $x, y \in \text{var}(\phi)$, if $x \prec y$ then x is not after y. A *respectful induced graph* of ϕ is one that is respectful with its prefix. The *respectful induced width* of ϕ is the minimum width among its respectful induced graphs.

Observe the following claim:

Claim 5. *Let ϕ be a qbf as in (3) and let $(H, \prec)$ be a respectful induced graph of ϕ. Let S be the sequence of variables in ϕ in the order defined by $\prec$. Then, $S = Y_1, \ldots, Y_q$, where Y_i is a permutation of X_i for every $i \in \{1, \ldots, q\}$. Moreover, $Q_1 Y_1 \ldots Q_q Y_q(\phi')$ is logically equivalent to ϕ.*

In [17], Arnborg et al. show that a qbf has a tree decomposition of width w if and only if its constraint graph is a partial w-tree. Along the same lines, Freuder [2] shows that a qbf has an induced graph of width w if and only if its constraint graph is a partial w-tree. By composing these theorems, we obtain that a qbf has a tree decomposition of width w if and only if it has an induced graph of width w. In [18], Dechter gives a direct proof of the *if* side of this statement in terms of bucket elimination. Using the construction by Decther, we present a direct proof of the whole statement in graph-theoretic terms and show that our constructions preserve respectfulness.

Lemma 5. *Let ϕ be a qbf. Then ϕ has a respectful tree decomposition of width w if and only if it has a respectful induced graph of width w.*

Proof. Let G be the constraint graph of ϕ. First, let (T, L) be a respectful tree decomposition of ϕ of width w. We will construct a respectful induced graph of ϕ of the same width. Define $\prec$ as $x \prec y$ if y is under x in (T, L) and arbitrarily if neither is under the other. Let H be such that $(H, \prec)$ is an induced graph of G. We show that (T, L) is also a tree decomposition of H by induction on the number of edges of H. If $|E(H)| = |E(G)|$, then $H = G$ and we are done. If $|E(H)| > |E(G)|$, let (x, y) be an edge of $E(H) \setminus E(G)$. By definition of H, there is a $z \in V(H)$ such that $x \prec z$ and $y \prec z$ and both (x, z) and (y, z) belong to $E(H)$. By induction hypo (T, L) is a tree decomposition of $H - (x, y)$. We have to show that $x, y \in L(t)$ for some $t \in V(T)$. Let T_z be the connected subtree of T induced by the vertices $t \in V(T)$ such that $z \in L(t)$ and let t_z be the root of T_z. We will show that, in fact, both x and y belong to t_z. Let $T_z^x := \{t \in V(T_z) \mid x \in L(t)\}$. Since (T, L) is a tree decomposition of $H - (x, y)$ and $(x, z) \in E(H)$, we have that T_z^x is non-empty. Let $t_z^x \in T_z^x$ be the closest vertex to t_z among them. If $t_z^x \neq t_z$, then x is under z and, by the definition of $\prec$, we have that $z \prec x$, which is a contradiction. Therefore, $t_z^x = t_z$, which implies $x \in L(t_z)$. The same argument can be made to show that $y \in L(t_z)$, proving the claim. Define $\prec_R$ as $x \prec_R y$ if y is under x in (T, L) and $x \prec_R y$ if x occurs before y in the prefix of ϕ and neither is under the other. Let H_R be such that $(H_R, \prec_R)$ is an induced graph of G. Note that, since $\prec_R$ is a particular case of $\prec$, we have that (T, L) is also a tree decomposition of H_R. To see that $(H_R, \prec_R)$ is respectful we show that, if $x \prec_R y$, then x is not after y. We have two cases: first, if y is under x, then, since (T, L) is respectful, we have that x is not after y; and second, if x occurs before y in the prefix, of course x is not after y. Finally, for every $x \in V(H_R)$, by definition of H_R, the vertices of $V_x := \{x\} \cup \{y \mid y \prec x \text{ and } (x, y) \in E(H_R)\}$ form a clique. By Claim 1, for every $x \in V(H_R)$ there is a $t \in V(T)$ such that $V_x \subseteq L(t)$. Therefore,

$$\max_{x \in V(H_R)} |\{(x, y) \in E(H_R) \mid y \prec x\}| \leq \max_{x \in V(H_R)} |V_x| - 1 \leq \max_{t \in V(T)} |L(t)| - 1 \leq w.$$

Second, let $(H, \prec)$ be a respectful induced graph of ϕ of width w. We will construct a respectful tree decomposition of the same width. Let T be a graph with one vertex t_x for every $x \in V(H)$

and one edge (t_y, t_x) where y is the biggest (with respect to $\prec$) neighbor of x in H such that $y \prec x$. Note that T is acyclic, since for every $x \in V(H)$, we have that t_x is connected to at most one vertex t_y such that $y \prec x$. As defined, T is not rooted and may not be connected, but we will fix this at the end of the proof. Let L be defined by $L(t_x) := \{x\} \cup \{y \mid (y, x) \in E(H) \text{ and } y \prec x\}$ for every $x \in V(H)$. Next, we show that (T, L) is a respectful tree decomposition of G of width w. First, we have that $\bigcup_{t \in V(T)} L(t) = V(G)$ since $V(G) = V(H)$ and, for every $x \in V(H)$, we have that $x \in L(t_x)$. Second, for every $(x, y) \in E(G)$, we have $x, y \in L(t_x)$ if $y \prec x$ and $x, y \in L(t_y)$ if $x \prec y$. Third, we have to show that for every $x \in V(G)$, the subgraph of T induced by $\{t \in V(T) \mid x \in L(t)\}$ is a connected subtree. Recall that H has the property that, for every $x, y, z \in V(H)$ such that $x \prec z$ and $y \prec z$, if (x, z) and (y, z) are in $E(H)$, also (x, y) is in $E(H)$. It is enough to see that, if $x \in L(t)$, the unique shortest path $t_1, \ldots, t_\ell$ with $t_1 = t$ and $t_\ell = t_x$ is such that $x \in L(t_i)$ for every $i \in \{1, \ldots, \ell\}$. We prove this by induction on i. If $i = 1$, by hypothesis we have $x \in L(t_1)$. Now, let $i > 1$ and, as induction hypothesis, assume x belongs to $L(t_1), \ldots, L(t_{i-1})$. We want to show that x belongs to $L(t_i)$ also. Let $y, z \in V(H)$ be such that $t_{i-1} = t_z$ and $t_i = t_y$. Since, by induction hypothesis, $x \in L(t_z)$, we have that $x \prec z$ and that (x, z) is in $E(H)$. Also, since $(t_y, t_z) \in E(T)$, we have that (y, z) is in $E(H)$. Now, we show that $x \prec y$ by cases: if $z \prec y$, then $x \prec y$, since $x \prec z$. If $y \prec z$, then also $x \prec y$, since otherwise y would not have been the biggest (with respect to $\prec$) neighbour of z such that $y \prec z$ (x would satisfy the conditions and would be bigger than y). Note that, by the construction of T, every vertex $t_u \in V(T)$ has at most one neighbour in $V_u(T) := \{t_v \in V(T) \mid v \prec u\}$. Suppose, for the sake of contradiction, that $z \prec y$. Then, (y, z) is the single edge that connects y to $V_y(T)$. But since the path does not repeat edges, it cannot lead to any other vertex in $V_y(T)$. Since t_x is in $V_y(T)$, this is a contradiction. Therefore, we have that $y \prec z$. Finally, since $x \prec z$ and $y \prec z$ and (x, z) and (y, z) both belong to $E(H)$ and $(H, \prec)$ is an induced graph, also (x, y) belongs to $E(H)$. And then, since $x \prec y$, we have that $x \in L(t_i)$. We make sure now that the graph that we built is rooted and connected. Let $T_1, \ldots, T_k$ be the connected components of T. For $i \in \{1, \ldots, k\}$, let r_i be the unique vertex of T_i such that $|L(r_i)| = 1$. Let r be a fresh vertex and let $T_C = (V_C, E_C)$ with $V_C := V(T) \cup \{r\}$ and $E_C := V(E) \cup \bigcup_{i \in [k]} \{(r, r_i)\}$ be a rooted tree with r in the root. Note that T_C is connected. Also, let L_C be the extension of L to V_C such that $L_C(r) = \emptyset$. Note that (T_C, L_C) is respectful, since if x is under y, by construction of T_C surely $y \prec x$, and, since $(H, \prec)$ is respectful, y is not after x. Finally, (T_C, L_C) has width

$$\max_{t \in V(T_C)} |L_C(t)| - 1 = \max_{t \in V(T)} |L(t)| - 1 = \max_{x \in V(H)} |\{y \mid (x, y) \in E(H) \text{ and } y \prec x\}| \leq w.$$

□

Corollary 1. *Let ϕ be a qbf. Then ϕ has respectful tree-width w if and only if it has respectful induced width w.*

In a different setting, Chen and Dalmau [16] show that quantified constraint satisfaction problems, which generalize QBFs to unbounded domains, are tractable if they have bounded respectful tree-width. We show here the corresponding result for Q-resolution: it is polynomially bounded for qbfs of bounded respectful tree-width.

Lemma 6. *Let ϕ be a false qbf sentence with n variables, m clauses and respectful tree-width w. Then, there is a weak Q-resolution refutation of ϕ of size $O(m + n \cdot 3^w)$.*

Proof. By Lemma 5, we have that ϕ has respectful induced width w. Let $(H, \prec)$ be a respectful induced graph of ϕ of width w. Let $Y := (R_1 y_1, \ldots, R_n y_n)$ be the sequence of variables of ϕ in order $\prec$ together with its quantifier in ϕ, and, for $i \in \{1, \ldots, n\}$, let $Y_i := (R_i y_i, \ldots, R_n y_n)$ be the i-th suffix of Y.

Since $(H, \prec)$ is respectful with the prefix of ϕ, by Claim 5 we have that $R_1 y_1 \ldots R_n y_n (\phi')$ is equivalent to ϕ. Moreover, since $\phi \equiv \square$, by Lemma 4 we have that $\phi'^{(Y)} \models \square$ and also that $\square \in \phi'^{(Y)}$. Then, the sequence $(\phi', \phi'^{(Y_n)}, \ldots, \phi'^{(Y_1)})$ makes a valid Q-resolution refutation of ϕ.

Finally, note that every $\phi'^{(Y_i)}$ has at most 3^w clauses not already in the sequence, since every variable is connected to at most w variables of $\phi'^{(Y_{i-1})}$, and there are a total of 3^w possible clauses that can be formed with w variables. Therefore, the size of the refutation is $O(m + n \cdot 3^w)$. $\square$

5.3 Formulas with bounded respectful tree-width

In the previous section we have shown that false qbfs with bounded respectful tree-width have short Q-resolution refutations. In this section we introduce a family of formulas with this property and show some formulas that belong to this family and may have real-world applications.

5.3.1 qbfs with bounded number of variables

Let $x_1, \ldots, x_k$ be propositional variables. A k-qbf is defined recursively as follows:

1. any clause on variables $x_1, \ldots, x_k$ is a k-qbf,
2. if ϕ and ψ are k-qbfs, then $\phi \wedge \psi$ is a k-qbf,
3. if ϕ is a k-qbf, then $\exists x_i$ is a k-qbf, where $i \in \{1, \ldots, k\}$, and
4. if ϕ is a k-qbf, then $\forall x_i$ is a k-qbf, where $i \in \{1, \ldots, k\}$.

Notice that we allow a variable to be quantified more than once. The recursive construction of a k-qbf defines a (rooted) labeled tree, whose leaves are labeled with the clauses of the formula and whose internal vertices are either labeled with a $\wedge$ and have two children, or labeled with $\exists$ or $\forall$ and have a single child. For a k-qbf ϕ, we say that (T_ϕ, K_ϕ) is its *associated tree* in the sense described above, where T_ϕ is a tree of the indicated form and $K_\phi : V(T_\phi) \to \mathcal{C} \cup \{\wedge, \exists x_1, \ldots, \exists x_s, \forall x_1, \ldots, \forall x_s\}$ where $\mathcal{C}$ is the set of all clauses on the variables $x_1, \ldots, x_k$. We say ϕ is the *associated formula* of the pair (T_ϕ, K_ϕ).

This family of formulas is the propositional version of one introduced by Dalmau et al. in [12], extended by allowing universal quantification. Their framework allows, given a QBF, to rewrite it as a logically equivalent k-QBF. Here we want to achieve exactly the opposite: given a k-qbf, rewrite it as a logically equivalent qbf. To do so, given a k-qbf ϕ, consider the following rewriting rules:

1. **A-Rule**: Associativity of conjunction is applied to subformulas of ϕ.
2. **C-Rule**: Commutativity of conjunction is applied to subformulas of ϕ.
3. **$\exists$-Rule**: a subformula of ϕ of the form $(\psi \wedge (\exists x \theta))$ is replaced by the formula $(\exists x(\psi \wedge \theta))$, provided the variable x does not occur in ψ.
4. **$\forall$-Rule**: a subformula of ϕ of the form $(\psi \wedge (\forall x \theta))$ is replaced by the formula $(\forall x(\psi \wedge \theta))$, provided the variable x does not occur in ψ.

5. **R-∃-Rule**: a subformula of ϕ of the form $(\exists x\psi)$ is replaced by the formula $(\exists y)\psi[x/y]$, where y does not occur in ψ and $\psi[x/y]$ is obtained from ψ by replacing all free occurrences of x in ψ by y.
6. **R-∀-Rule**: a subformula of ϕ of the form $(\forall x\psi)$ is replaced by the formula $(\forall y)\psi[x/y]$, where y does not occur in ψ and $\psi[x/y]$ is obtained from ψ by replacing all free occurrences of x in ψ by y.

It is clear that the application of these rules preserves logical equivalence.

Note that every k-qbf of size s can be rewritten as a qbf as in (2) by the following steps: first, repeatedly apply the R-Rules with fresh variables $x_{k+1}, \ldots, x_s$ until no variable in the formula occurs quantified more than once. Second, repeatedly apply ∃-Rule and ∀-Rule, always on the outermost possible quantifier (and A-Rule and C-Rule, as necessary, to reorder the conjunctions in order to apply the other rules) until we obtain the form (2). It is clear that this can be done in a number of steps polynomial in s and that the resulting formula ϕ^R will be over the variables $x_1, \ldots, x_s$. Also, let K^R_ϕ be equal to K_ϕ but appropiately applying the renaming performed by the R-Rules on the clauses at the leaves.

For a tree T and $t \in V(T)$, let T^t be the subtree of T rooted at t. Let ϕ_t be the associated formula of (T^t_ϕ, K_ϕ) and let ϕ^R_t be the associated formula of (T^t_ϕ, K^R_ϕ). Now, define $L_\phi : V(T_\phi) \to \mathcal{P}(\{x_1, \ldots, x_s\})$ as

$$L_\phi(t) := \{x \mid x \text{ is free in the formula } \phi^R_t\}$$

for every $t \in V(T_\phi)$.

We prove the following claim:

Claim 6. *The pair (T_ϕ, L_ϕ) is a respectful tree decomposition of ϕ^R of width $k-1$.*

Proof. First, note that every clause of ϕ^R is precisely $K^R_\phi(t)$ for some leaf t of T_ϕ. Since $t = T^t_\phi$, the associated formula of (t, K^R_ϕ) is precisely the clause $K^R_\phi(t)$, and therefore, all of its variables are free in it. Therefore, for every clause C of ϕ^R, there is a leaf t of T_ϕ for which $L(t) = \text{var}(C)$, and also, $\bigcup_{t \in V(T_\phi)} L(t) = \text{var}(\phi'^R)$. Second, for $x \in \text{var}(\phi^R)$, let t_x be the (unique) child of the (unique) vertex of t of T_ϕ such that $K^R_\phi(t)$ is of the form Qx for $Q \in \{\exists, \forall\}$, and the root of T if there is none. Note that $x \in L(t)$ if and only if both $t \in V(T^{t_x})$ and for some leaf t' of T^t, we have $x \in L(t')$. Then, the subgraph of T_ϕ induced by $\{t \in \text{var}(\phi^R) \mid x \in L(t)\}$ is precisely the union of the (unique) paths from t_x to a leaf t' of T_ϕ such that $x \in L(t')$. Since all of these paths have their beginning at t_x, this is a connected subtree. Finally, note that for every $t \in T_\phi$, we have $|L(t)| \leq k$ since, in case $|L(t)| > k$ for some $t \in T_\phi$, that would imply that ϕ^R_t has more than k free variables, which is not possible, since, before renaming, ϕ (and therefore, ϕ_t) has only k variables in total. □

Corollary 2. *Every k-qbf is logically equivalent to a qbf with respectful tree-width $k-1$.*

Note that, together with Lemma 6, this gives that, as long as $k \leq c \cdot \log n$ for some constant c, for every false k-qbf we can obtain a logically equivalent qbf and a short Q-resolution refutation of the second. Next, we see examples of k-qbfs for which this result may be useful.

5.3.2 Bounded model checking

An alternating finite state machine is a nondeterministic state machine whose states are of two types: ∃-states or ∀-states. On a given input, the machine accepts if there is at least one transition

leaving every $\exists$-state such that for every transition leaving every $\forall$-state, the machine ends up reaching an accepting state. Consider an alternating finite state machine with n states and with m transitions leaving each state, in which every transition leaving an $\exists$-state leads to a $\forall$-state and viceversa. States and transitions leaving each state are labeled with a number encoded in binary as $\bar{x} = x_1, \ldots, x_{|n|}$ and $\bar{y} = y_1, \ldots, y_{|m|}$, respectively. Also, define the ternary relation R as $R(\bar{x}, \bar{y}, \bar{x}')$ if, from state $\bar{x}$, using transition $\bar{y}$, we can reach state $\bar{x}'$ in a single step. Let $I(\bar{x})$ indicate that $\bar{x}$ is an initial state, and let $Z(\bar{x})$ indicate that $\bar{x}$ is a Z-state.

We want to obtain a proof of the following statement, common in the context of bounded model checking: no Z-state is accessible from an I-state in at most ℓ steps. We call this statement $P_{\leq \ell}$. Note that this problem can be reduced to obtaining a proof of P_t for every $0 \leq t \leq \ell$. We focus on this last problem, which is equivalent to finding a refutation of $\neg P_t$, which is equal to $\exists \bar{x}(I(\bar{x}) \wedge \psi_t(\bar{x}))$ where

$$\begin{aligned}
&\psi_0(\bar{x}) = Z(\bar{x}), \\
&\psi_{i+1}(\bar{x}) = \exists \bar{y} \exists \bar{x}'(R(\bar{x}, \bar{y}, \bar{x}') \wedge \psi_i(\bar{x}')) && \text{for odd } i \geq 0, \\
&\psi_{i+1}(\bar{x}') = \forall \bar{y} \exists \bar{x}(R(\bar{x}', \bar{y}, \bar{x}) \wedge \psi_i(\bar{x})) && \text{for even } i \geq 0.
\end{aligned}$$

Observe that, by writing $I(\bar{x})$, $Z(\bar{x})$ and $R(\bar{x}, \bar{y}, \bar{x}')$ as cnfs, the formula $\neg P_t$ that we obtain is a $(2|n| + |m|)$-qbf. Therefore, if P_t is true, we can obtain a Q-resolution refutation of a qbf expressing $\neg P_t$ of size exponential in $2|n| + |m|$, that is, polynomial in the number of states and the size of the formula.

By defining the formulas encoding $I(\bar{x})$, $Z(\bar{x})$ and $R(\bar{x}, \bar{y}, \bar{x}')$ appropiately, we can use this to model multiple real-world situations. We present a couple of examples:

Verification of software with human interaction In this case, the alternating finite state machine models the interaction between a user and a computer interface: $\exists$-states are those waiting for a response of the system and $\forall$-states are those waiting for a response from the user. The initial state is the initial configuration of the software, the Z-states are those in which the software crashes or reaches an undesired point. Finally, the relation R is defined by the work-flow of the program. We want to make sure that, from the initial state of the program, for every input of the user into the interface, there is a possible response of the program in such a way that the user cannot crash the system before ℓ interactions.

Two-player games by turns In this case, the alternating finite state machine models the strategies of the players: the $\exists$-states model the positions in which the first player has to move and the $\forall$-states model the positions in which his adversary has to move. The initial state is the initial configuration of the game and the Z-state is a winning or losing configuration, depending on what we want to prove. The relation R defines the legal moves of the players. What we can prove here is that, starting with the initial configuration of the game, the first player cannot win the game (or lose it) before $\ell + 1$ rounds have been played.

References

[1] R. Dechter and J. Pearl, Tree clustering for constraint networks, Artificial Intelligence 38 (1989) 353-366.

[2] E. Freuder, Complexity of k-tree structured constraint satisfaction problems, Proceedings of the 8th National Conference on Artificial Intelligence (AAAI) 1 (1990) 4-9.

[3] H. Chen, Quantified constraint satisfaction and bounded treewidth, Proceedings of the 16th European Conference on Artificial Intelligence (ECAI) (2004) 161-165.

[4] G. Gottlob, G. Greco, and F. Scarcello, The complexity of quantified constraint satisfaction problems under structural restrictions, Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI) (2005) 150-155.

[5] G. Pan and M.Y. Vardi, Fixed-parameter hierarchies inside PSPACE, Proceedings of the 21th IEEE Symposium on Logic in Computer Science (LICS) (2006) 27-36.

[6] A. Meyer, Weak monadic second order theory of succesor is not elementary recursive, Logic Colloquium. Lecture Notes in Mathematics 453 (1975) 132-154.

[7] M. Frick and M. Grohe, The complexity of first-order and monadic second-order logic revisited, Proceedings of the 17th IEEE Symposium on Logic in Computer Science (LICS) (2002) 215-224.

[8] H. Chen and V. Dalmau, Decomposing Quantified Conjunctive (or Disjunctive) Formulas, Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science (LICS) (2012) 205-214.

[9] H.K. Büning, A. Flögel and M. Karpinski, Resolution for quantified Boolean formulas, Information and Computation 117 (1995) 12-18.

[10] G. Pan and M.Y. Vardi, Symbolic decision procedures for QBF, Proceedings of the 10th International Conference on Principles and Practice of Constraint Programming (CP) (2004) 453-467.

[11] R. Dechter and J. Pearl, Network-based heuristics for constraint-satisfaction problems, Artificial Intelligence 34 (1987) 1-38.

[12] V. Dalmau, P. Kolaitis and M.Y. Vardi, Constraint satisfaction, bounded treewidth and finite-variable logics, Proceedings of the 8th International Conference in Principles and Practice of Constraint Programming (CP) (2006) 223-254.

[13] P. Chatalic and L. Simon, Multi-resolution on compressed sets of clauses, Proceedings of the 12th IEEE International Conference on Tools with Artificial Intelligence (ICTAI) (2000) 2-10.

[14] B. Aspvall, M.F. Plass and R.E. Tarjan, A linear-time algorithm for testing the truth of certain quantified boolean formulas, Information Processing Letters 8 (1979) 121-123.

[15] M. Alekhnovich and A.A. Razborov, Satisfiability, branch-width and Tseitin tautologies, Proceedings of the 43rd Symposium on Foundations of Computer Science (FOCS) (2002) 593-603.

[16] H. Chen and V. Dalmau, From pebble games to tractability: An ambidextrous consistency algorithm for quantified constraint satisfaction, Proceedings of the 19th Conference on Computer Science Logic (CLS) (2005) 232-247.

[17] S. Arnborg, D.G. Corneil and A. Proskurowski, Complexity of finding embeddings in a k-tree, SIAM Journal on Algebraic Discrete Methods 8 (1987) 277-284.

[18] R. Dechter, Constraint processing, Morgan Kaufmann, 2003.