Fixed-Parameter Hierarchies inside PSPACE

Guoqiang Pan
Department of Computer Science
Rice University
gqpan@cs.rice.edu

Moshe Y. Vardi
Department of Computer Science
Rice University and Microsoft Research
vardi@cs.rice.edu

Abstract

Treewidth measures the "tree-likeness" of structures. Many NP-complete problems, e.g., propositional satisfiability, are tractable on bounded-treewidth structures. In this work, we study the impact of treewidth bounds on QBF, a canonical PSPACE-complete problem. This problem is known to be fixed-parameter tractable if both the treewidth and alternation depth are taken as parameters. We show here that the function bounding the complexity in the parameters is provably nonelementary (assuming P is different than NP). This yields a strict hierarchy of fixed-parameter tractability inside PSPACE. As a tool for proving this result, we first prove a similar hierarchy for model checking QPTL, quantified propositional temporal logic. Finally, we show that QBF, restricted to instances with a slowly increasing ($\log^*$*) treewidth, is still PSPACE-complete.*

1. Introduction

Considerable attention has been dedicated recently, cf. [5], to the complexity-theoretic analysis of *constraint satisfaction*, a problem of great significance in computer science, see [6]. In particular, a focus on of this attention has been to identify tractable classes of this problem, which is NP-complete in general. A particular way to study constraint satisfaction problems is to focus on their *parametrized complexity* [8], where we analyze the complexity in terms of both the problem-instance size n and and a parameter k that relates to some property of the problem. Typically, this approach is used to capture the intuition that problems hard in the classical sense might have tractable classes, here represented as problems with a small k. The class of *fixed-parameter tractable (FPT)* problems are those that can be solved in time $f(k)\mathsf{poly}(n)$, where f is any computable function. We typically use parameterized complexity to study problems that are intractable in the classical setting, so under the usual assumptions, $f(k)$ is at least exponential. A popular parameter for structural problems is that of *treewidth*, which measures how close a structure is to being a tree (trees have treewidth 1) [15]. Many NP-complete problems, for example, Boolean satisfiability, are FPT with respect to the treewidth w, specifically $2^w n$ [7, 10].

A very recent focus of inquiry has been on *quantified* constraint satisfaction [2, 12], which is PSPACE-complete in general. It is natural to ask whether such problems exhibit fixed-parameter tractability under structural restrictions such as bounded treewidth. On one hand, it is shown in [12] that quantified constraint satisfaction remains PSPACE-hard under restrictions less restrictive than treewidth (for example, acyclic QCSP is still PSPACE-complete). On the other hand, [2] showed that quantified constraint satisfaction is FPT when treewidth, alternation depth, and domain size are all taken as parameters.

Our focus here is on investigating the complexity of quantified constraint satisfaction, specifically, *quantified Boolean formulas* (QBF), with respect to the treewidth parameter[1]. (Quantified CSP with constant domain size can be reduced to QBF with only a constant factor – logarithmic in the domain size – increase in width.) A careful analysis of the results in [2] shows that the parametric bound $f(d, w)$, where d is the alternation depth and w is the treewidth, grows nonelementarily in d; that is, the function consists of a tower of exponential that grows linearly with d. We show here that this nonelementary growth is unavoidable. Specifically, we show that, under the assumption that P is different than NP, the parametric function $f(d, n)$ must be a tower of exponentials of height almost d. This yields a strict hierarchy of parametrized complexity inside PSPACE.

The nonelementary growth of the parametrized bound suggests that QBF is not FPT with respect to treewidth alone. We prove a result that gets quite close to this; we show that QBF, restricted to instances with a slowly increasing ($\log^*$) treewidth, is still PSPACE-complete. This leaves open a small gap, which is the hardness of QBF under the assumption of bounded treewidth.

We prove our results via a technical detour of QPTL model checking. QPTL is a quantified propositional temporal logic, introduced in [16]. It is a monadic second-order logic, so its model-checking problem is closely related to the model-checking problem for the monadic second-order

[1] In [3], a modified notion of treewidth is introduced for quantified constraint satisfaction, while our focus here is on the standard notion.

logic of words, studied in [11]. It is known that while satisfiability for monadic second-order logic is nonelementarily decidable [14], its model checking problem is PSPACE-complete. Classical automata-theoretic techniques, cf. [1], entail that the model-checking problem for monadic second-order logic is FPT, where formula size is taken as the parameter. It is shown in [11] that the nonelementary lower-bound technique of [14] can be "telescoped down" to prove a nonelementary lower bound for the parametrized bound (assuming that P is different than NP). Analogously to [11], we show that the nonelementary satisfiability lower-bound for QPTL [16] can be telescoped down to yield a nonelementary lower bound for the parametrized bound (assuming that P is different than NP). The advantage of using QPTL rather than monadic second-order logic is that the latter contains both first- and second-order quantifiers, while QPTL has only second-order quantifiers. This enables us to get more refined results than those in [11], getting an FPT hierarchy in terms of alternation depth, rather than formula size.

A second advantage of using QPTL is that its model-checking problem can be reduced to QBF satisfiability, while maintaining a bounded *pathwidth*. (Pathwidth measures how close a structure is to being a path; it is a stricter measure than treewidth). Thus, the FPT complexity hierarchy result for QPTL model checking can be transferred to a similar hierarchy for QBF satisfiability.

The approach in [11] relies on the assumption that P is different than NP, so we can use it only to show that QBF for small-treewidth formulas is NP-hard. To show that QBF of small-treewidth is PSPACE-hard, we have to go beyond QPTL model checking to QPTL model-checking games, which embed quantifier alternation in the word being model checked. This built-in alternation enables us to go from NP-hardness to PSPACE-hardness, by showing that quantified satisfiability for the set of formulas with treewidth in $O(\log^*(n))$ is PSPACE-complete.

The paper is organized as follows. In Section 2, we provide basic definitions for QPTL, the tiling problem, and QBF. In Section 3, we establish a parametrized complexity hierarchy for QPTL model checking, via a reduction from tiling. In Section 4, we present a reduction from QPTL model checking to QBF, and show similar hierarchy for QBF. In Section 5, we present model-checking games and use them to prove PSPACE-completeness of small-width QBF.

2. Background

2.1. QPTL

Definition 2.1. Given a set P of propositions, the set of formulas of QPTL is the smallest set satisfying the following: Every proposition in P is a formula in QPTL; if φ and ψ are formulas in QPTL, then $(\varphi \wedge \psi)$, $(\neg\varphi)$, $X\varphi$, $F\varphi$, and $(\exists p)\varphi$ are formulas in QPTL; we use X^n as shorthand for a sequence of n X operators, $G\varphi$ as shorthand for $\neg F \neg\varphi$, and $(\forall p)\varphi$ as shorthand for $\neg(\exists p)\neg\varphi$.

We interpret QPTL formulas on finite words. A finite word model π for QPTL is a word of length n over 2^P, that is, a mapping $\pi : [0 \dots n-1] \to 2^P$, where n is the size of the model, and P is a set of propositions (which needs to contain the free propositions in the formula we are interpreting). All propositional connectives have their standard meaning. Since we are considering semantics for finite-word models, the temporal operators X and F need to consider the finiteness of the model, and so do the quantifiers. We use $\pi, i \models \varphi$ to mean the word model π at position i satisfies the formula φ. (The standard notation of $\pi \models \varphi$ now becomes a shorthand for $\pi, 0 \models \varphi$.) Of course, $\models$ is only defined when $0 \leq i < |\pi|$. The formal semantics of QPTL on a finite word model follows:

- $\pi, i \models p$ iff $p \in \pi(i)$,
- $\pi, i \models \neg\varphi$ iff $\pi, i \not\models \varphi$,
- $\pi, i \models \varphi \wedge \psi$ iff $\pi, i \models \varphi$ and $\pi, i \models \psi$,
- $\pi, i \models X\varphi$ iff $i < |\pi| - 1$ and $\pi, i+1 \models \varphi$,
- $\pi, i \models F\varphi$ iff there exists some $i \leq j < |\pi|$, where $\pi, j \models \varphi$,
- $\pi, i \models (\exists q)\varphi$ iff there is a set $Q \subseteq [0 \dots |\pi| - 1]$ of positions, such that $\pi[q \mapsto Q], i \models \varphi$, where $p \in \pi[q \mapsto Q](i)$ iff either $p \neq q$ and $p \in \pi(i)$, or $p = q$ and $i \in Q$.

Each QPTL formula φ defines a language which is the set of all πs where $\pi, 0 \models \varphi$.

Definition 2.2. The finite model-checking problem for QPTL is defined as: given a model π and a formula φ, where π is defined on a vocabulary that includes the free variables of φ, does $\pi \models \varphi$ hold?

Here, we only consider QPTL formulas in prenex normal form, where a formula can be written as $(Q_1 p_1)(Q_2 p_2) \dots (Q_k p_k)\varphi$, each Q_i is $\exists$ or $\forall$, and φ is quantifier free. The logic can be stratified into bounded-alternation layers as follows:

1. The set $\Sigma_0^{QPTL} = \Pi_0^{QPTL}$ includes the quantifier-free formulas in QPTL.
2. If φ is in Σ_k^{QPTL} or Π_{k-1}^{QPTL}, then $(\exists p)\varphi$ is in Σ_k^{QPTL}.
3. If φ is in Π_k^{QPTL} or Σ_{k-1}^{QPTL}, then $(\forall p)\varphi$ is in Π_k^{QPTL}.

The sets Σ_k^{QPTL} and Π_k^{QPTL} split QPTL into fragments where Σ_k^{QPTL} (Π_k^{QPTL}) represents the existential (universal) formulas with *alternation depth* k.

While all the QPTL formulas constructed in this paper should be in prenex normal form, we do, for simplicity of presentation, apply quantifiers to sub-formulas and later compose them with other sub-formulas. Still, we never put quantified sub-formulas under temporal operators, which allows all formulas we built to be converted to prenex normal form with a simple renaming and lifting of quantifiers without increase in either size or alternation depth. In Section 4 and 5, we also assume the QPTL formula is in negation normal form, which means that negation is applied only to atomic propositions.

2.2. Tiling Systems

Definition 2.3. A *tiling system* is a tuple $T = (D, H, V)$, where D is a finite set of *tiles*, $H \subseteq D \times D$ and $V \subseteq D \times D$ are, respectively, the *horizontal* and *vertical* adjacency relations. A *tiling problem* is the tuple (T, w, h, I), where w and h are natural numbers, and I is an *initial constraint* (see below). The tiling problem (T, w, h, I) has a solution iff there exists a mapping $F : [1 \dots w] \times [1 \dots h] \to D$, where the following requirements are satisfied:

- For all $1 \leq i < w$ and $1 \leq j \leq h$, $(F(i,j), F(i+1,j)) \in H$, and for all $1 \leq i \leq w$ and $1 \leq j < h$, $(F(i,j), F(i,j+1)) \in V$.
- F satisfies the initial constraint I.

Each position (i, j), where $i \in [1..w]$, $j \in [1..h]$, is called a *cell* with row i and column j.

The tiling problem is useful in representing a number of natural complexity classes. The most well known of which is the case where $h = w$ (*square tiling*), w is written in unary, and the initial condition I is empty. This version have been shown to be NP-complete by Lewis [13]. Here, we consider a constant tiling system T. This is possible because tiling systems encode runs of Turing machines, so a T that corresponds to a universal Turing machine can be used. Still, to maintain NP-completeness where we use a constant T, we need to use a different initial condition, namely, the first row needs to be fixed to a particular sequence of tiles I, where $F(1, j) = I(j)$ for $1 \leq j \leq w$ (which encodes the initial tape of the Turing machine). This allows polynomial-sized computations of Turing automaton to be encoded with the tiling problem.

Theorem 2.4. [13] There is a tiling system T (corresponding to universal Turing machines) such that The tiling-problem (T, w, h, I), where $h = w$, and I defines the first row of the tiling, is NP-complete.

In the following, we concern ourselves only with this NP-complete variant, which we denote as (T, w, I). Given a tiling F, we name the corresponding solution checking problem, in absence of the initial condition, as the *tiling check* problem, where we check whether F satisfies the horizontal and vertical relations. We also assume we can separate the tile set into two parts D' and D'', where D' does not appear on the first row, and D'' only appears on the first row. Only tiles in D'' can appear in I. This allows us to use quantifiers to guess the non-initial part of the tiling. Since $w = |I|$, and T is fixed, the size of the tiling problem is defined as $|I|$, which is the same as the size of w in unary.

2.3. QBF

Quantified Boolean formula (QBF) extends propositional logic by introducing quantifiers over the Boolean domain $\{0, 1\}$. We write formulas in QBF in prenex normal form $\psi = (Q_1 p_1)(Q_2 p_2) \dots (Q_k p_k)\psi'$, where the Qs are quantifiers, the ps are propositions, and ψ' is a propositional formula, which we call the *matrix*. By bounding the number of alternations, QBF can be stratified into classes Σ_k^{QBF} and Π_k^{QBF}, where k is the number of alternations, and a formula is in Σ_k^{QBF} if its outermost quantifier is $\exists$ and contains k alternations and in Π_k^{QBF} if its outermost quantifier is $\forall$ and contains k alternations. The complexity hierarchy of deciding these formula classes corresponds the *polynomial hierarchy* [18]. Here, we consider the case where the matrix ψ' is restricted to CNF, so the innermost universal quantifier block introduces no additional complexity. In other words, we only consider the classes Σ_k^{QBF} where k is odd and Π_k^{QBF} where k is even.

The *width* of a QBF formula is defined as the width of the *interaction graph* of its CNF matrix. The interaction graph is defined with the set of propositions as vertices, and the co-occurrence (in the same clause) relation between propositions as edges.

Definition 2.5. (Robertson and Seymour [15]) Let $G = (V, E)$ be a graph. A *tree (path)-decomposition* of G is a pair $(T, \mathcal{X})$ where $T = (I, F)$ is a tree (path) with node set I and edge set F, and $\mathcal{X} = \{X_i \mid i \in I\}$ is a family of subsets of V, one for each node of T, such that

- $\bigcup_{i \in I} X_i = V$,
- for every edge $(v, w) \in E$, there is an $i \in I$ with $\{v, w\} \subseteq X_i$, and
- for all $i, j, k \in I$, if j is on the path from i to k in T, then $X_i \cap X_k \subseteq X_j$.

The *width* of a tree (path)-decomposition is $\mathsf{max}_{i \in I} |X_i| - 1$. The *treewidth (pathwidth)* of a graph G, denoted by $tw(G)$ ($pw(G)$), is the minimum width over all possible tree (path) decompositions of G.

Bounded pathwidth is clearly a more restrictive concept than bounded treewidth, since a path-decomposition can be used without change as a tree-decomposition. In other words, for all graphs G, $tw(G) \leq pw(G)$. Thus, any complexity lower-bound result proved for bounded pathwidth also applies to bounded treewidth. In this paper, all constructions have bounded pathwidth.

3. Complexity of finite model-checking problem for QPTL

Our goal is to study the complexity of the finite model-checking problem for QPTL. First, we define the non-elementary tower function g: $g(0, n) = n$, $g(k+1, n) =$

$2^{g(k,n)}$. The inverse of g would be the $\log^k$ function: $\log^1(n) = \log n$, $\log^{k+1}(n) = \log(\log^k(n))$. Finally, the $\log^*$ function is defined by: $k = \log^*(n)$ iff k is the least natural number such that $g(k, 1) \geq n$.

3.1. Upper Bound

In [16], the satisfiability problem of QPTL is studied, i.e., for a given QPTL formula φ, whether there exists a model π such that $\pi \models \varphi$. While their analysis of QPTL is over infinite words, the automata-theoretic approach can be used to analyze QPTL also over finite words (in fact, that was the origin this approach, cf. [1]). The following lemma characterizes QPTL over finite words.

Lemma 3.1. Given a QPTL formula φ in Σ_k^{QPTL}, there exists a non-deterministic finite automaton of size $g(k, |\varphi|)$ that accepts the same language as φ.

Theorem 3.2. (Analogous to Theorem 4.1, [16]) The satisfiability problem for Σ_k^{QPTL}, where $k \geq 1$, is complete for $NSPACE(g(k-1, |\varphi|))$.

In contrast to the satisfiability problem, under a finite semantics, the model-checking problem has elementary complexity.

Theorem 3.3. The finite model-checking problem for QPTL is PSPACE-complete.

A usual parametrization of model-checking problems is to take the formula size and alternation depth as the parameter and the size of the model as the size of the problem.

Theorem 3.4. The finite model-checking problem for a formula φ in Σ_k^{QPTL} on a finite word model of size n has time complexity $O(g(k, |\varphi|)n)$.

3.2. Lower Bound

For the lower bound, we proceed by connecting the complexity of QPTL model checking on finite word models to the NP-completeness of the tiling problem. In the following, we write $\mathsf{poly}(\cdot)$ as the class of polynomially-bounded functions.

Theorem 3.5. Assuming $P \neq NP$, there exists a constant $c > 0$ such that for every $h(\cdot)$ in $\mathsf{poly}(\cdot)$, model checking for Σ_k^{QPTL} on models of size $|\pi|$, where $k > 2$, cannot be done in time $g(k-1, c|\varphi|)h(|\pi|)^2$.

The rest of this section is dedicated to the proof of Theorem 3.5. We start with an overview of the proof.

For a tiling problem (T, w, I), if we are given a tiling F satisfying I, we can reduce the tiling-check problem to QPTL model checking as follows: The tiling F is converted to a word F_w, and the tiling problem (T, w, I) is converted to a QPTL formula $\varphi'_{T,w}$. Note that the initial condition I is not checked by the formula, but instead, encoded in the model. In the following, when our concern is the formula, we use (T, w) to denote the relevent part of the tiling problem. We want $F_w \models \varphi'_{T,w}$ iff F is a solution for (T, w). Writing F in word form requires concatenating the rows of F into a word of length w^2. We order the rows from w to 1 for technical reasons. To convert (T, w) to $\varphi'_{T,w}$, we need to state row and column constraints using small QPTL formulas. Once we reduced the tiling-check problem to the finite model-checking problem for QPTL, going to the tiling (decision) problem is just a small step, using existential quantifiers to state the existence of a tiling. We add an additional existential quantifier block on the outside of $\varphi'_{T,w}$ for all the propositions that actually represent non-initial tiles, and call the resulting formula $\varphi_{T,w}$. Since now the actual tiling is quantified out in the formula, the interpretation of the propositions that correspond to non-initial tiles in the word model no longer affect the decision procedure, so the word model can contain an arbitrary mapping from cells after the first row to non-initial tiles, and the mapping from cells in the first row to initial tiles is based on I. We call such a model $\pi_{w,I}$. Thus, $\pi_{w,I} \models \varphi_{T,w}$ iff (T, w, I) has a solution. Assuming the reduction is polynomial, this particular model-checking problem is NP-hard, while fixed-parameter tractable. Assuming $P \neq NP$, a lower-bound can be shown for the function in the parameter.

The idea of using short formulas to describe very long "yardsticks" go back to the early 1970s, cf. [17, 14, 16]. The idea of "telescoping" such constructions down, using extremely short formulas to describe short yarsticks, enabling one to prove nonelementary bounds inside PSPACE, is due to [11].

We give here the proof for Theorem 3.5, under the following assumptions on the reduction. Later, we describe in detail each step in the construction of $\pi_{w,I}$ and $\varphi_{T,w}$ and show that the assumptions hold.

1. $\pi_{w,I} \models \varphi_{T,w}$ iff (T, w, I) have a solution.
2. The word $\pi_{w,I}$ has size polynomial in w.
3. The formulas $\varphi_{T,w}$ are in Σ_k^{QPTL} and are small enough so that there exists a constant $c > 0$ where $g(k-1, c|\varphi_{T,w}|)$ is in $\mathsf{poly}(w)$.
4. The construction of $\pi_{w,I}$ and $\varphi_{T,w}$ from (T, w, I) can be performed in time polynomial in w.

Proof. Choose $h(\cdot)$ in $\mathsf{poly}(\cdot)$. Assume that we have a model-checking algorithm with time complexity $g(k-1, c|\varphi|)h(|\pi|)$ for Σ_k^{QPTL}, where c is small enough to apply Assumption 3. We use this algorithm to give a polynomial algorithm for tiling, which by Theorem 2.4, is NP-complete. We are given inputs w and I (where $w = |I|$). Based on Assumption 2 and 4, we can construct $\pi_{w,I}$ in time and space polynomial in w. Based on Assumption 3, we can construct a formula $\varphi_{T,w} \in \Sigma_k^{QPTL}$, where $g(k-1, c|\varphi_{T,w}|)$ is in $\mathsf{poly}(w)$, and by Assumption 4, the time taken for the construction is in $\mathsf{poly}(w)$. Then, by Assumption 1, the

[2]In the proof of Theorem 4.2, we need to refer to the actual value of this constant, and call it c_{QPTL}

model-checking algorithm can be applied, which takes time $g(k-1, c|\varphi_{T,w}|)h(|\pi_{w,I}|)$, which by Assumptions 2 and 3, is in $\mathsf{poly}(w)$. So, if we have an algorithm for the finite model-checking problem that does better than the lower-bound, then there exists a polytime algorithm for an NP-complete problem. □

Here, readers would note that there is an exponential gap between the lower bound in Theorem 3.5 and the upper bound in Theorem 3.4. The gap can be closed by strengthening the $P \neq NP$ assumption as follows.

Corollary 3.6. Assuming NP cannot be solved in sub-exponential time, then there exists a constant $c > 0$ such that model checking for Σ_k^{QPTL} on models of size $|\pi|$, where $k > 2$, cannot be done in time $g(k, c|\varphi|)h(|\pi|)$ where $h(\cdot)$ is in $\mathsf{poly}(\cdot)$.

For space reasons, we will only point out when we use Assumption 3, i.e., when $g(k-1, c|\varphi|)$ is polynomially bounded in w, we can always choose a smaller c' such that $g(k-1, c'|\varphi|)$ is sub-polynomial in w. In turn, $g(k, c'|\varphi|)$ is sub-exponential in w, leading to the tiling problem being solved in time sub-exponential in w.

3.2.1 Constructing the Tagged Model $\pi_{w,I}$

To facilitate easier check of row and column constraints on the word model $\pi_{w,I}$, we add annotation in the form of tags. The tags mark the column number of each cell, since we want to write the tiling in row major order. Our technique is based on a combination of techniques from [16, 11]. While both approaches allow checking of very large models by small formulas, they use different logics[3] and different constructions to assert relations between two positions that are very distant in the word model. The approach in [16] used counters that are implicitly encoded using alternation; in contrast, the approach in [11] used explicit tags[4]. Using explicit tags allows us to simplify construction of the QPTL formula, since we can now check arbitrary-sized models (unlike the counter-based approach, where the model needs to align on exact tower-of-exponentials boundaries), while at the same time maintain the alternation depth low.

Definition 3.7. A *tag of level h* is defined with the alphabet $\Sigma = \{0, 1, \langle 1\rangle, \langle /1\rangle, \ldots, \langle h\rangle, \langle /h\rangle\}$. We denote the set of tags of level h with parameter n by $\mathsf{tags}_h(n)$. Each tag represents a mapping that can be inductively defined as follows: A tag $t \in \mathsf{tags}_1(n)$ is a mapping $\{0 \ldots n-1\} \rightarrow \{0, 1\}$. A tag $t \in \mathsf{tags}_{h+1}(n)$ is a mapping $\mathsf{tags}_h(n) \rightarrow \{0, 1\}$. Tags are written as words on Σ. For $t \in \mathsf{tags}_1(n)$, t is a bracketed n-bit sequence, i.e., a word of form $\langle 1\rangle\{0|1\}^n\langle /1\rangle$, where $t(i)$ is the $(i+1)$-th bit after $\langle 1\rangle$. For $t \in \mathsf{tags}_{h+1}(n)$, t is a bracketed sequence of tags in $\mathsf{tags}_h(n)$ and a corresponding bit for each tag, i.e., $\langle h+1\rangle(t'\{0|1\})^*\langle /h+1\rangle$, where each t' is a tag in $\mathsf{tags}_h(n)$. Since t is a total function, every possible tag of level h appear exactly once in a tag of level $h+1$. Every tag $t' \in \mathsf{tags}_h(n)$ that appears inside t is called a *sub-tag* of t. For every sub-tag $t' \in \mathsf{tags}_h(n)$ in a tag $t \in \mathsf{tags}_{h+1}(n)$, the bit that appears directly after the t' is the mapped value of the t'. Equality on tags is defined as the (unordered) equality on the underlying mappings.

For example, a tag in $\mathsf{tags}_2(1)$ is $\langle 2\rangle\langle 1\rangle 0\langle /1\rangle 0$ $\langle 1\rangle 1\langle /1\rangle 1\langle /2\rangle$, which maps the sub-tag $\langle 1\rangle 0\langle /1\rangle \in \mathsf{tags}_1(1)$ to 0 and the sub-tag $\langle 1\rangle 1\langle /1\rangle \in \mathsf{tags}_1(1)$ to 1.

We now consider the size of tags.

Lemma 3.8. There are $g(h, n)$ different tags in $\mathsf{tags}_h(n)$.

The construction we use for tags is similar to that used in [11]. One important difference is that our construction requires each tag in $\mathsf{tags}_h(n)$ to be a complete mapping, where the one in [11] allowed partial maps. Our approach trades a larger tag size for better alternation efficiency.

First, we consider the size blowup in tags for our construction. We only need to provide a very relaxed bound for our final result:

Lemma 3.9. The size $s_h(n)$ of a tag in $\mathsf{tags}_h(n)$, for $n > 2$, is at most $g(h-1, 2n)$.

Our goal is to use tags to address the column for each tile. This introduces an blowup in the size of $\pi_{w,I}$ and we want to maintain polynomial size in w. Thus, the size of each tag need to be polynomial in w. We use the following lemma:

Lemma 3.10. For a given w and $h \leq \log^*(w) - 2$, there exists $n > 2$ such that:

- There are at least w distinct tags in $\mathsf{tags}_h(n)$.
- The size of each individual tag in $\mathsf{tags}_h(n)$ is at most w.

Definition 3.11. We define a mapping $\mathsf{tag}_{h,n} : N \rightarrow \mathsf{tag}_h(n)$ by induction. For the base case, $\mathsf{tag}_{1,n}(x) := \langle 1\rangle bit_{n-1}(x) \ldots bit_0(x)\langle /1\rangle$, where $bit_i(x)$ extracts the value of the ith bit of x. For the inductive case, $\mathsf{tag}_{h,n}(x) := \langle h\rangle \mathsf{tag}_{h-1,n}(g(h-1, n) - 1) bit_{g(h-1,n)-1}(x) \ldots \mathsf{tag}_{h-1,n}(0) bit_0(x)\langle /h\rangle$. In other words, $\mathsf{tag}_{h,n}(x)$ is a tagged binary number, which maps every $\mathsf{tag}_{h-1,n}(y)$ to the yth bit of x.

Next, we describe the tagged word model $\pi_{w,I}$. The construction is with respect to a parameter $k > 2$ (which is the k of Theorem 3.5). We build the word using tags of level $h = k-1$. The vocabulary of the word model is $\Sigma = \Sigma_h \cup D \cup \{\mathsf{r}\}$, where D is the set of tiles of T. We use r as a row marker. Given a tiling problem of size w, choose n where $g(h, n-1) < w \leq g(h, n)$. In other words, $n = \lceil \log^h(w) \rceil$. Now $\pi_{w,I}$ is the concatenation of

[3] In [16], QPTL is used, but on infinite models. In [11], monadic second-order logic is used, although second-order quantifiers are only used to "quantify-out the model", i.e., to check the existence of solutions. The base constructions only use first-order logic on linearly ordered structures.

[4] The tags can be seen as a form of counters, except that they only have to be checked for equality instead of "counting" anything.

$w-1$ blank rows and one initial row. The blank rows use an arbitrary non-initial tile $d_0 \in D$ (In the formula, we guess a tiling via quantification, so the use of d_0 here is purely as a placeholder.):

$$row_{blank} := \mathsf{tag}_{h,n}(0)d_0\mathsf{tag}_{h,n}(1)d_0 \dots \mathsf{tag}_{h,n}(w-1)d_0\mathsf{r}$$

The initial row is the same as defined by I:

$$row_{init} := \mathsf{tag}_{h,n}(0)I(1)\mathsf{tag}_{h,n}(1)I(2) \dots \mathsf{tag}_{h,n}(w-1)I(w)\mathsf{r}$$

Finally $\pi_{w,I} := (row_{blank})^{w-1}row_{init}$.

For ease of the construction we use in Section 5, we concatenate the rows in inverted order, where the first row is at the end of the word.

Lemma 3.12. For every k, the length of the word $\pi_{w,I}$ is polynomial in w, and $\pi_{w,I}$ can be generated in time polynomial in w. In addition, the polynomial functions do not depend on k.

Lemma 3.12 shows that Assumption 2 and 4 for Theorem 3.5 above hold.

3.2.2 Tag Comparison Using QPTL

Next we describe how to use QPTL formulas to check equality of tags, which we use to check the vertical constraints of tilings. The formula we build uses a set P of propositions. For any letter $d \in \Sigma$, we have a corresponding proposition p_d, where in the word model π, we have $\pi, i \models \mathsf{p}_d$ iff d is the ith letter in π. Other propositions in P are used to mark sets of locations in the word model; we call them *markers*. Since we often want to mark a single position, most occurrences of these propositions are restricted to be singletons. A pair of such singleton markers outline a subword by pointing to the beginning and ending positions of the subword. Given two markers p and q, the subword is denoted by $\pi_{[p,q]}$.

In the following construction, we use formula templates as follows: $\psi(p) := \psi'$ defines a template ψ with parameter p based on ψ', which is a QPTL formula. An instantiation $\psi(q)$ is a copy of ψ', where all free occurrences of p are replaced with q.

Lemma 3.13. Given $h \geq 1$ and $n \geq 1$, one can construct a formula $\varphi_{h,n}(p,p',q,q') \in \Sigma_h^{QPTL}$ of length linear in $h+n$ such that if $\pi \models \varphi_{h,n}(p,p',q,q')$, then

1. p, p', q, and q' are each true at exactly one position in the model;
2. The subwords $\pi_{[p,p']}$ and $\pi_{[q,q']}$ are equal tags in $\mathsf{tags}_h(n)$;
3. The subword $\pi_{[p,p']}$ appears before the subword $\pi_{[q,q']}$.

Proof. In the following, we first construct the formula $\varphi_{h,n}$, and then show that it satisfies the requirements in the lemma. We proceed with an inductive construction:

We present some basic templates to state some commonly used relationships for propositions. First, define $\mathsf{sing}(p)$ to assert that the proposition p is a singleton where $\mathsf{sing}(p) := Fp \wedge G(p \rightarrow XG\neg p)$. We also need to state that a proposition r is never true strictly inside a subword $\pi_{[p,q]}$ where $\mathsf{notbetween}(p,q,r) := G(p \rightarrow XG(XFq \rightarrow \neg r))$. We also need to state that two positions, both i letters away from singleton markers p and q, assert a proposition r where $\mathsf{same}^i(p,q,r) := G(p \rightarrow X^i r) \wedge G(q \rightarrow X^i r)$.

Now we construct the base case, where $\pi_{[p,p']}$ and $\pi_{[q,q']}$ mark tags in $\mathsf{tags}_1(n)$. In order to check equality, we need to state the following:

- The markers only appear once: $\mathsf{sing}(p) \wedge \mathsf{sing}(p') \wedge \mathsf{sing}(q) \wedge \mathsf{sing}(q')$.
- The range defined by the markers does not contain multiple tags at level 1: $\mathsf{notbetween}(p,p',\mathsf{p}_{\langle 1\rangle}) \wedge \mathsf{notbetween}(q,q',\mathsf{p}_{\langle 1\rangle})$.
- The markers point to the correct symbols: $G(p \rightarrow \mathsf{p}_{\langle 1\rangle}) \wedge G(p' \rightarrow \mathsf{p}_{\langle /1\rangle}) \wedge G(q \rightarrow \mathsf{p}_{\langle 1\rangle}) \wedge G(q \rightarrow \mathsf{p}_{\langle /1\rangle})$.
- The markers appear in the correct order: $G(p \rightarrow XFp') \wedge G(p' \rightarrow XFq) \wedge G(q \rightarrow XFq')$.
- The bit strings are the same: $\bigwedge_{1 \leq i \leq n}(\mathsf{same}^i(p,q,\mathsf{p}_1) \vee \mathsf{same}^i(p,q,\mathsf{p}_0))$.
- $\varphi_{1,n}(p,p',q,q')$ is the conjunction of all the formula above.

By construction, $\varphi_{1,n}(p,p',q,q')$ is a formula in Σ_1^{QPTL} with size linear in n.

For the inductive case, we need to check whether two tags in $\mathsf{tags}_h(n)$ are equal. We need to check that all (internal) tags in $\mathsf{tags}_{h-1}(n)$ have matching values. We state the following:

- We still need to the same format checks as the first four entries in $\varphi_{1,n}(p,p',q,q')$, except all the references to $\langle 1\rangle$ and $\langle /1\rangle$ are changed to references to a tag of appropriate depth. We name this formula $\mathsf{format}_h(p,p',q,q')$.
- We use a proposition b to mark all starting positions of sub-tags in $\mathsf{tags}_{h-1}(n)$ that appear in $\pi_{[t,t']}$. $\mathsf{insidemarker}_h(t,t',b) := G(Ft \rightarrow \neg b) \wedge G(t' \rightarrow G\neg b) \wedge G(\neg \mathsf{p}_{\langle h-1\rangle} \rightarrow \neg b) \wedge G(t \rightarrow G((Ft' \wedge \mathsf{p}_{\langle h-1\rangle}) \rightarrow b))$
- Given (singleton) markers r, r', s, s', we can state that the pair of tags they point to map to the same value: $\mathsf{idmap}^h(r,r',s,s') := \varphi_{h-1,n}(r,r',s,s') \rightarrow (\mathsf{same}^1(r',s',\mathsf{p}_0) \vee \mathsf{same}^1(r',s',\mathsf{p}_1))$.
- We now define a formula to assert the equivalence of tags $\pi_{[p,p']}$ and $\pi_{[q,q']}$. For the meaning of the formula defined as checktags, see proof below. $\mathsf{checktags}_h(p,p',q,q') := (\exists b,b')(\forall r,r',s,s')(\mathsf{insidemarker}_h(p,p',b) \wedge \mathsf{insidemarker}_h(q,q',b') \wedge ((G(r \rightarrow b) \wedge G(s \rightarrow b')) \rightarrow \mathsf{idmap}_h(r,r',s,s')))$. Note $\mathsf{idmap}_h(r,r',s,s')$ is true whenever r' or s' does not point to the end of a tag in $\mathsf{tags}_h(n)$.

- $\varphi_{h,n}(p, p', q'q') := \mathsf{format}_h(p, p', q, q') \wedge \mathsf{checktags}_h(p, p', q, q')$.

Since in $\varphi_{h,n}$ the iterative part $\varphi_{h-1,n}$ appears under negation, and the quantifier prefix for the $\varphi_{h,n}$ outside $\varphi_{h-1,n}$ is $\exists\forall$, $\varphi_{h,n}$ in Σ_h^{QPTL} and of size linear in $h + n$.

Now we proceed to show that the construction is correct:

Claim 3.14. $\pi_{[p,p']}$ and $\pi_{[q,q']}$ are identical tags iff $\varphi_{h,n}(p, p', q, q')$. □

3.2.3 Constructing $\varphi_{T,w}$

Next, by using $\varphi_{k-1,n}(p, p', q, q')$ we construct a formula $\varphi'_{T,w}$ in Σ_k^{QPTL} to check that an annotated model is a tiling for the tiling problem (T, w) with size $w \le g(k-1, n)$.

- Every position in the word has exactly one symbol: $\varphi_D := G(\bigwedge_{s\in\Sigma}(\mathsf{p}_s \to \bigwedge_{t\neq s} \neg \mathsf{p}_t)) \wedge G(\bigvee_{s\in\Sigma} \mathsf{p}_s)$.
- Horizontal constraints are observed: Define $\varphi_H := (\forall s, s', t, t')((\mathsf{sing}(s) \wedge \mathsf{sing}(s') \wedge \mathsf{sing}(t) \wedge \mathsf{sing}(t') \wedge G(s \to \mathsf{p}_{\langle k-1\rangle}) \wedge G(s' \to \mathsf{p}_{\langle /k-1\rangle}) \wedge G(t \to \mathsf{p}_{\langle k-1\rangle}) \wedge G(t' \to \mathsf{p}_{\langle /k-1\rangle}) \wedge G(s \to Fs') \wedge G(s' \to Ft) \wedge G(t \to Ft') \wedge \mathsf{notbetween}(s, s', \mathsf{p}_{\langle k-1\rangle}) \wedge \mathsf{notbetween}(s', t, \mathsf{p}_{\langle k-1\rangle}) \wedge \mathsf{notbetween}(s', t, \mathsf{p_r}) \wedge \mathsf{notbetween}(t, t', \mathsf{p}_{\langle k-1\rangle})) \to \bigvee_{\langle d,d'\rangle\in H}(G(s' \to X\mathsf{p}_d) \wedge G(t' \to X\mathsf{p}_{d'})))$. This checks for every pair of tags $\pi_{[s,s']}$ and $\pi_{[t,t']}$, if $\pi_{[s,s']}$ appears before $\pi_{[t,t']}$, and there is no row marker or other tag markers that appear between $\pi_{[s,s']}$ and $\pi_{[t,t']}$, then the symbol that appears after $\pi_{[s,s']}$ and the symbol that appears after $\pi_{[t,t']}$ are consecutive symbols on the same row, and need to satisfy the horizontal constraint.
- Vertical constraints are observed: Define $\varphi_V := (\forall s, s', t, t')((\varphi_{k-1,n}(s, s', t, t') \wedge G(s' \to (F(\mathsf{p_r} \wedge Ft) \wedge \neg F(\mathsf{p_r} \wedge XF(\mathsf{p_r} \wedge Ft))))) \to \bigvee_{\langle d,d'\rangle\in V}(G(s' \to X\mathsf{p}_{d'}) \wedge G(t' \to X\mathsf{p}_d)))$. We know from Lemma 3.13 that $\varphi_{k-1,n}(s, s', t, t')$ checks whether $\pi_{[s,s']}$ and $\pi_{[t,t']}$ are identical tags in $\mathsf{tags}_{k-1}(n)$. In φ_V, we first check that the pair of tags are identical, i.e., they mark the same column; then we check that the pair of tags has exactly one row marker appearing between them, i.e., they are in consecutive rows; finally, we check that the tiles that appear after the pair of tags satisfy the vertical constraint.
- The formula for checking whether the model is a solution is $\varphi'_{T,w} := \varphi_D \wedge \varphi_H \wedge \varphi_V$.

For $k \ge 2$, $\varphi'_{T,w}$ is a formula in Π_{k-1}^{QPTL}, since the occurrence of $\varphi_{k-1,n}$ in $\varphi'_{T,w}$ is negative, and the quantifier block of $\varphi'_{T,w}$ outside $\varphi_{k-1,n}$ is universal.

Given a tiling problem T and a mapping F, F is a solution for (T, w) iff $F_w \models \varphi'_{T,w}$. To check whether (T, w, I) has a solution, we take $\varphi_{T,w} = (\exists \mathsf{p}_{d_0})(\exists \mathsf{p}_{d_1}) \ldots (\exists \mathsf{p}_{d_m})\varphi'_{T,w}$.

Lemma 3.15. Given a tiling instance (T, w, I), and an integer $k \ge 2$, we can generate in time $O(k + \log^{k-1}(w))$ a Σ_k^{QPTL} formula $\varphi_{T,w}$ of size $O(k + \log^{k-1}(w))$ such that $\pi_{w,I} \models \varphi_{T,w}$ iff (T, w, I) have a solution, and the coefficient in $O(k + \log^{k-1}(w))$ is independent from k.

Lemma 3.15 allows us to meet Assumption 1 needed for Theorem 3.5. We now show that we can also satisfy Assumption 3 and 4 of Theorem 3.5.

Lemma 3.16. There exists a constant $c > 0$ such that for every $k > 2$, the formula $\varphi_{T,w}$ is small enough so that $g(k-1, c|\varphi_{T,w}|)$ is in $\mathsf{poly}(w)$, and $\varphi_{T,w}$ can be constructed in time polynomial in w.

4. The complexity of low-width QBF

We now come back to the decision problem of bounded-width QBF. A variant of this problem is studied in [2] in the context of quantified constraint satisfaction and an FPT algorithm is given, where both the width and the alternation depth are taken as parameters. The function f given in the algorithm is non-elementary. We prove here that, under complexity-theoretic assumptions, the function in the parameter is indeed non-elementary.

4.1. Parameterized complexity of bounded-width QBF

Theorem 4.1. Satisfiability for Σ_k^{QBF} formulas of treewidth at most w can be solved in time $O(g(k, w)|\varphi|)$.

Proof. The result follows by a careful analysis of the algorithm in [2], counting the number of distinct trees generated by the algorithm. □

The algorithm presented in [2] is an extension of the decision procedure for bounded-width propositional formulas, which can be decided in $O(2^w|\varphi|)$ time [7, 10]. Unfortunately, it is unlikely that bounded-width QBF enjoys the same kind of tractability:

Theorem 4.2. Assuming $P \neq NP$, there exists a $c > 0$ such that for every $h(\cdot)$ in $\mathsf{poly}(\cdot)$, satisfiability for bounded-pathwidth formula ψ in Σ_k^{QBF}, where k is an odd number > 2, of pathwidth at most w cannot be solved in time $g(k-1, cw)h(|\psi|)$.

Proof. We combine Theorem 3.5 above with a translation from QPTL finite model checking to QBF satisfiability to show the lower bound. In Theorem 4.5 that follows, we show that for every QPTL formula $\varphi \in \Sigma_k^{QPTL}$, with odd k, and every word model π, we can construct a QBF formula $\psi = \varphi_{Q,\pi}$ of pathwidth at most $2|\varphi| - 1$ and alternation depth k such that $\models \psi$ iff $\pi \models \varphi$. Also, ψ is of size $O(|\varphi||\pi|)$, so there is a constant c' such that the construction takes time at most $c'|\varphi||\pi|$ and $|\psi| \le c'|\varphi||\pi|$. Consider

$c = c_{QPTL}/2$. For every $k > 2$ and for every polynomially bounded function $f'(\cdot)$, we have that $g(k-1, c(2|\varphi| - 1))f'(|\varphi|) < g(k-1, c_{QPTL}|\varphi|)$ for large enough $|\varphi|$. This is because $g(k-1, c(2|\varphi|-1)) = g(k-1, c_{QPTL}|\varphi| - c_{QPTL}/2)$, which is far smaller than $g(k-1, c_{QPTL}|\varphi|)$. For example, for every positive constant d, $g(2,x)/g(2,x-d)$ is not polynomially bounded on x. Now, suppose we can decide the satisfiability of ψ in time $g(k-1, cw)h(|\psi|)$, we can use it to decide the truth of $\pi \models \varphi$ through deciding ψ in time $g(k-1, c_{QPTL}|\varphi|)h'(|\pi|)$, with a polynomial $h'(\cdot)$. First, the time taken to solve the model checking of φ on π through QBF is $g(k-1, c(2|\varphi|-1))h(c'|\varphi||\pi|) + c'|\varphi||\pi| \leq g(k-1, c(2|\varphi|-1))(h(c'|\varphi||\pi|) + c'|\varphi||\pi|)$. Because $h(\cdot)$ is polynomial, there exist polynomial functions $f'(\cdot)$ and $h'(\cdot)$ such that $h(c'|\varphi||\pi|) + c'|\varphi||\pi| \leq f'(|\varphi|)h'(|\pi|)$. Also, from our assumption, $g(k-1, c(2|\varphi|-1))f'(\varphi) \leq g(k-1, c_{QPTL}(|\varphi|))$ for large enough $|\varphi|$. Thus, $g(k-1, c(2|\varphi|-1))h(c'|\varphi||\pi|) + c'|\varphi||\pi| \leq g(k-1, c_{QPTL}|\varphi|)h'(|\pi|)$ for large enough $|\varphi|$. This contradicts with Theorem 3.5 under the same $P \neq NP$ assumption. □

Note we have the similar one-exponential gap between the upper and lower bound, and it can be closed in the same way as the bounds for QPTL model checking. A direct consequence is that low-width QBF can not be solved with an FPT algorithm with an elementary blowup in the parameters k and w, where k is the alternating depth and w is the width.

Definition 4.3. The class of $\log^*$-pathwidth QBF is the set of QBF formulas ψ that have pathwidth $O(\log^*(|\psi|))$.

Theorem 4.4. The class of $\log^*$-pathwidth QBF is NP-hard.

Proof. NP-hardness is by a reduction from the NP-hard tiling problem. For a tiling instance (T, w, I), We first use the construction in Section 3 to encode it as a QPTL model-checking instance, then use the translation presented for Theorem 4.5 below to translate it to a QBF instance. For the first step of the construction, take the alternation depth k to be the least odd number $\geq \log^*(w) - 2$ (in other words, k is either $\log^*(w) - 2$ or $\log^*(w) - 1$). Because $k - 1 \geq \log^*(w) - 3$, we have that $\log^{k-1}(w) \leq 16$. By Lemma 3.15, the size of the QPTL formula $\varphi_{T,w}$ is $O(k + \log^{k-1}(w)) = O(k)$. By Lemma 3.12, the model $\pi_{w,I}$ has size $h(w)$ for some polynomial function h. Now we use Theorem 4.5 to translate the finite model-checking problem $\pi_{w,I} \models \varphi_{T,w}$ to a QBF instance $\psi = \varphi_{Q,\pi_{w,I},T,w}$ that has size $|\psi| = O(|\varphi_{T,w}||\pi_{w,I}|) = O(k \times h(w))$ and pathwidth at most $2|\varphi_{T,w}| - 1$. Since $k = O(\log^*(w))$, $O(k \times h(w))$ is polynomial in w. Similarly, the pathwidth of ψ is at most $2|\varphi_{T,w}| - 1$, which is in $O(k) = O(\log^*(w))$ and in turn, $O(\log^*(|\psi|))$ (since $|\psi|$ is polynomial in w). Thus, the family of $\log^*$-pathwidth QBF problems is $NP - hard$. □

In contrast, propositional satisfiability problems that have $O(\log(|\psi|))$ pathwidth are still in P [9].

4.2. Translating QPTL finite model checking to QBF

Theorem 4.5. Given a formula φ in Σ_k^{QPTL} (for odd k) that is in negation normal form and a finite word model π, we can generate in time $O(|\varphi||\pi|)$ a formula $\varphi_{Q,\pi}$ in Σ_k^{QBF} of size $O(|\varphi||\pi|)$ and pathwidth at most $2|\varphi| - 1$, such that $\pi \models \varphi$ iff $\models \varphi_{Q,\pi}$.

Proof. We perform the following translation from QPTL to QBF. Consider the QPTL formula $\varphi = (\exists x_{1,1}) \ldots (\forall x_{2,1}) \ldots (\exists x_{k,1}) \ldots \varphi'$. We first construct a propositional model π_Q such that $\pi_Q \models \varphi_Q$ iff $\pi \models \varphi$. We can then eliminate the propositional model by describing it as a conjunction of unit literals and conjoining with φ_Q (this does not increase the width of the formula). We establish the correspondence between $\pi \models \varphi$ and $\pi_Q \models \varphi_Q$ inductively.

The base case is when φ is quantifier free. A propositional model π_Q can capture all information encoded in the word model π: For each free proposition q of φ we create $|\pi|$ Boolean propositions $p_{q,i}$, $0 \leq i < |\pi|$, defined by $\pi_Q(p_{q,i}) = 1$ iff $q \in \pi(i)$. We refer to this set of Boolean propositions as P_Q.

We write $\mathsf{sub}(\varphi)$ for the set of sub-formulas of φ, and $\mathsf{sub}'(\varphi)$ for the set of non-atomic sub-formulas of φ, in other words, $\mathsf{sub}'(\varphi) = \mathsf{sub}(\varphi) - \{q, \neg q | q \text{ is a proposition defined in } \varphi\}$. With each ψ in $\mathsf{sub}'(\varphi)$, we associate $|\pi|$ new propositions $p_{\psi,i}$. Essentially, we use a propositional encoding of accepting runs of the automaton constructed in [19] (adapted to finite words). The formula φ_Q can be encoded in CNF form by unrolling the formula onto the structure as follows. (For ease of understanding, clause groups for closely related propositions are written using the $\leftrightarrow$ operator.)

- In the following, whenever $\psi = a$ or $\psi = \neg a$, substitute $p_{a,i}$ or $\neg p_{a,i}$ for $p_{\psi,i}$.
- $\psi = X\psi'$: $C_\psi := (\neg p_{\psi,|\pi|-1}) \wedge \bigwedge_{0 \leq i < |\pi|-1} (p_{\psi,i} \leftrightarrow p_{\psi',i+1})$
- $\psi = F\psi'$: $C_\psi := (p_{\psi,|\pi|-1} \leftrightarrow p_{\psi',|\pi|-1}) \wedge \bigwedge_{0 \leq i < |\pi|-1} (p_{\psi,i} \leftrightarrow (p_{\psi,i+1} \vee p_{\psi',i}))$ (because $\pi, i \models F\psi$ iff $\pi, i \models \psi$ or $\pi, i+1 \models F\psi$.)
- $\psi = G\psi'$: $C_\psi := (p_{\psi,|\pi|-1} \leftrightarrow p_{\psi',|\pi|-1}) \wedge \bigwedge_{0 \leq i < |\pi|-1} (p_{\psi,i} \leftrightarrow (p_{\psi,i+1} \wedge p_{\psi',i}))$ (Because $\pi, i \models G\psi$ iff $\pi, i \models \psi$ and $\pi, i+1 \models G\psi$.)
- $\psi = \psi' \wedge \psi''$: $C_\psi := \bigwedge_{0 \leq i < |\pi|} p_{\psi,i} \leftrightarrow (p_{\psi',i} \wedge p_{\psi'',i})$
- $\psi = \psi' \vee \psi''$: $C_\psi := \bigwedge_{0 \leq i < |\pi|} p_{\psi,i} \leftrightarrow (p_{\psi',i} \vee p_{\psi'',i})$

Now, we have $\varphi'_Q := p_{\varphi,0} \wedge \bigwedge_{\psi \in \mathsf{sub}'(\varphi)} C_\psi$ in CNF with $O(|\pi||\varphi|)$ clauses. We proceed to quantify out the propositions that correspond to valuations of subformulas in $\mathsf{sub}'(\varphi) = \{\psi_1, \psi_2 \ldots \psi_m\}$. Thus, π_Q define exactly the set of propositions we need to interpret $\varphi_Q := (\exists p_{\psi_1,0}) \ldots (\exists p_{\psi_m,|\pi|-1})\varphi'_Q$. φ_Q is in turn the *QBF translation* of φ.

Claim 4.6. If φ is quantifier free, then $\pi \models \varphi$ iff $\pi_Q \models \varphi_Q$.

We use QBF quantifiers to simulate QPTL quantifiers inductively. Assume we can translate ψ to ψ_Q in QBF. The inductive case is:

- $\varphi = (\exists x)\psi$: Take $\varphi_Q = (\exists p_{x,0}) \dots (\exists p_{x,|\pi|-1})\psi_Q$
- $\varphi = (\forall x)\psi$: Take $\varphi_Q = (\forall p_{x,0}) \dots (\forall p_{x,|\pi|-1})\psi_Q$

Clearly, φ_Q is in prenex normal form. The following claims are immediate.

Claim 4.7. $\pi \models \varphi$ iff $\pi_Q \models \varphi_Q$.

Claim 4.8. The matrix of $\varphi_{Q,\pi}$ have pathwidth at most $2|\varphi| - 1$.

Not only is the translation width efficient, it is also alternation efficient as well, in that

- For $\varphi \in \Sigma_i^{QPTL}$ where i is even, $\varphi_{Q,\pi} \in \Sigma_{i+1}^{QBF}$.
- For $\varphi \in \Sigma_i^{QPTL}$ where i is odd, $\varphi_{Q,\pi} \in \Sigma_i^{QBF}$.

The construction of $\varphi_{Q,\pi}$ can be performed in time linear to its size, i.e., $O(|\varphi||\pi|)$. In summary, the formula $\varphi_{Q,\pi}$ we constructed meets all the requirements posed by Theorem 4.5. $\square$

5. PSPACE-hardness for $\log^*$-pathwidth QBF

In section 3 and 4, we showed that unlike propositional satisfiability, QBF is hard even when the width is greatly limited (to $\log^*(n)$). To keep the low-width version as hard as tiling (NP), $\log^*(n)$ alternations are needed. Our goal is to increase the hardness of the low-width class beyond NP. One would expect $O(n)$ alternations to be needed to capture PSPACE. We move therefore from model checking to model-checking games in order to boost the alternation depth, while keeping the formulas small.

We would like to show the following improvement over Theorem 4.4:

Theorem 5.1. The decision problem for the class of $\log^*$-pathwidth QBF is PSPACE-complete.

Proof. The problem is trivially in PSPACE. For PSPACE-hardness, we start from a PSPACE-complete tiling-game problem and combine two translation results to achieve a polynomial reduction. Given a tiling-game instance (T, w, I), we use Theorem 5.5 to convert it into a QPTL model-checking game $(\pi_{w,I}, \varphi_{T,w,G}, G)$. The model $\pi_{w,I}$ and the play order G have size polynomial in w, and $\varphi_{T,w,G}$ has size $O(\log^*(w))$. Then, by applying Theorem 5.3, we convert the QPTL model-checking game $(\pi_{w,I}, \varphi_{T,w,G}, G)$ to a QBF formula $\psi = \varphi_{Q,\pi_{w,I},w,G}$ that has size polynomial in $|\pi_{w,I}|$, $|G|$, and $|\varphi_{T,w,G}|$. In turn, $|\psi|$ is polynomial in w. Also, ψ have pathwidth at most $2|\varphi_{T,w,G}| - 1$, which is in $O(\log^*(w))$, and in turn, $O(\log^*(|\psi|))$. Thus, the family of $\log^*$-pathwidth QBF problems is PSPACE-complete. $\square$

5.1. Model-checking games

First, we extend finite model checking for QPTL to a two-player game. Instead of being given a model to check, two players (the Constructor and the Spoiler) play a game to update a model for a specific formula. The Constructor's goal is to satisfy the formula in the resulting model, under opposition by the Spoiler.

Definition 5.2. A *finite model-checking game* for QPTL is defined as follows: A finite-word model is a map $\pi : [0 \dots n-1] \rightarrow 2^P$. The input to the game is a formula φ (which uses both the propositions in π as well as two distinguished propositions $\mathsf{p}_\exists$ and $\mathsf{p}_\forall$), a word model π, the number m of rounds, and a *play order* G. The play order G labels each position $0 \dots n-1$ in the model π either with a *round number*, which is an integer between 0 and $m-1$, or leaves it unlabelled. Thus, for each round number i, we have a set G_i of positions in the model defined by $G_i = \{d | G(d) = i\}$. Two players, the Constructor and the Spoiler, attempt to update π under the play order G. The play order is processed in increasing round order i. If i is even, the Constructor updates π^i to π^{i+1} (with $\pi^0 = \pi$) on every position d in G_i by choosing a new assignment $x_d \in 2^P$ and updating π^i with x_d. In summary, $\pi^{i+1} = \pi^i[d_1 \mapsto x_{d_1}][d_2 \mapsto x_{d_2}] \dots [d_k \mapsto x_{d_k}]$, where $G_i = \{d_1, \dots, d_k\}$. Here, $\pi[d \mapsto x](i) = \pi(i)$ if $d \neq i$ and $\pi[d \mapsto x](d) = x$. If i is odd, the Spoiler performs the update instead of the Constructor. The model π^m is extended onto the distinguished propositions $\mathsf{p}_\exists$ and $\mathsf{p}_\forall$ to become π', where $\mathsf{p}_\exists \in \pi'(d)$ only if $G(d)$ is even, and $\mathsf{p}_\forall \in \pi'(d)$ only if $G(d)$ is odd. π' is a winning model for the Constructor iff $\pi' \models \varphi$. The Constructor wins a model-checking game iff given π, G, m, and φ, he has a strategy to make the resulting model π' winning, and we write it as $\pi \models_G \varphi$.

The *finite model-checking game* for QPTL is a conservative extension of the finite model-checking problem on QPTL, since given an empty G, it is exactly the finite model-checking problem for QPTL. We can use a translation similar to that of Section 4 to convert QPTL model-checking games to QBF.

Theorem 5.3. Given a QPTL model-checking game (π, φ, G), we can construct in polynomial time a QBF formula $\varphi_{Q,\pi,G}$ such that $\pi \models_G \varphi$ iff $\models \varphi_{Q,\pi,G}$. The size of $\varphi_{Q,\pi,G}$ is polynomial in $|\varphi|$, $|\pi|$, and $|G|$, and the pathwidth of $\varphi_{Q,\pi,G}$ is at most $2|\varphi| - 1$.

5.2. Reducing tiling games to model-checking games

In Section 3, we showed that we can reduce the tiling problem (T, w, I) to a QPTL finite model-checking problem with model size in $\mathsf{poly}(w)$ and formula size in $O(k + \log^k(w))$. This result is combined with that of Theorem 4.4 to show that QBF formulas of $\log^*(n)$ width is NP-hard.

The ideal result would be to show PSPACE-completeness, i.e., restricting QBF to $\log^*(n)$ width does not reduce its complexity. To do that, we need a tiling problem that is PSPACE-complete and has a polynomial number of cells. This requires inherent alternation in the problem in the form of a game. We use the following model of the tiling game:

Definition 5.4. A *tiling game* uses a tiling system T and is played on a board with h rows and w columns with an initial condition I. Two players, the Constructor, and the Spoiler plays by tiling alternating rows, starting with the Constructor on row 1. The resulting tiling F is a winning tiling for the Constructor if one of the following holds:

1. The tiling F is a solution for the tiling problem (T, h, w, I).
2. The smallest j such that $(F(i, j-1), F(i, j)) \notin V$ or $(F(i-1, j), F(i, j)) \notin H$ is even.

The second condition states that the Spoiler is the first to produce a failing tile. The Constructor wins the game (T, h, w, I) iff he has a strategy to force a winning tiling.

A unary, square version of the tiling game (T, w, I), where $h = w$ and I specifies the first row, like most variants of square tiling games [4], is PSPACE-complete. We use finite model-checking games for QPTL to encode tiling games. The construction of the model $\pi_{w,I}$ from w is the same as in section 3.2.1. The play order G have w rounds, where each round G_{i-1} for $1 \leq i \leq w$ contains the set of positions in $\pi_{w,I}$ that corresponds to the cells (i, x) for $1 < x \leq w$. The formula is constructed as follows. Condition 1 is encoded by the $\varphi'_{T,w}$ of section 3.2.3. To encode condition 2, we guess the first position where the Spoiler made a failing play using a new variable e (and auxiliary variables e_D, e_H, and e_V). The following checks need to be performed. First, we check that e is a singleton and on a cell that is played by the Spoiler. Second, we check the tiling against the constraints. But, now the requirement for a correct tiling is reduced; the tiling is not necessarily a solution, as errors can occur on rows after the error marker e. Since we use an inverted row-major order, the correctness checks are not applied on positions before e in the word model, i.e., whenever Fe holds. Third, we check that an actual error occurred on the error marker. There are three types of errors, guessed by the variables e_D, e_H, and e_V. 1) The player chooses an assignment that does not correspond to a tile, i.e., more than one of p_ds or none of the p_ds are assigned to true. 2) The horizontal constraint is violated. 3) The vertical constraint is violated.

Theorem 5.5. Given a tiling game (T, w, I), we can construct in polynomial time a QPTL model-checking game $(\pi_{w,I}, \varphi_{T,w,G}, G)$ such that the Constructor wins the tiling game (T, w, I) iff $\pi_{w,I} \models_G \varphi_{T,w,G}$, the model $\pi_{w,I}$ and the play order G have size polynomial in w, and the formula $\varphi_{T,w,G}$ has size $O(\log^*(w))$.

Acknowledgements

Supported in part by NSF grants CCR-9988322, CCR-0124077, CCR-0311326, and ANI-0216467, by BSF grant 9800096, and by Texas ATP grant 003604-0058-2003. Part of this work was done while the second author was visiting the Isaac Newton Institute for Mathematical Science, as part of a Special Programme on Logic and Algorithms.

References

[1] J. Büchi. Weak second-order arithmetic and finite automata. *Zeit. Math. Logik und Grundl. Math.*, 6:66–92, 1960.

[2] H. Chen. Quantified constraint satisfaction and bounded treewidth. In R. L. de Mántaras and L. Saitta, editors, *ECAI*, pages 161–165. IOS Press, 2004.

[3] H. Chen and V. Dalmau. From pebble games to tractability: An ambidextrous consistency algorithm for quantified constraint satisfaction. In *CSL 2005*, pages 232–247, 2005.

[4] B. Chlebus. Domino-tiling games. *J. Comp. Sys. Sci.*, 32:374–392, 1986.

[5] N. Creignou, S. Khanna, and M. Sudan. Complexity classifications of Boolean constraint satisfaction problems. *Monographs on Discrete Applied Mathematics*, 2001.

[6] R. Dechter. *Constraint Processing*. Morgan Kaufmman, 2003.

[7] R. Dechter and J. Pearl. Tree clustering for constraint networks. *Artificial Intelligence*, pages 353–366, 1989.

[8] R. Downey and M. Fellows. *Parametrized Complexity*. Springer-Verlag, 1999.

[9] A. Ferrara, G. Pan, and M. Vardi. Treewidth in verification: Local vs. global. In *LPAR 2005*, pages 489–503, 2005.

[10] E. Freuder. Complexity of k-tree structured constraint satisfaction problems.

[11] M. Frick and M. Grohe. The complexity of first-order and monatic second-order logic revisited. In *LICS'02*, pages 215–224, 2002.

[12] G. Gottlob, G. Greco, and F. Scarcello. The complexity of quantified constraint satisfaction problems under structural restrictions. In *IJCAI 05*, pages 150–155, 2005.

[13] H. Lewis. Complexity of solvable cases of the decision problem for the predicate calculus. In *FOCS 1978*, pages 35–47, 1978.

[14] A. Meyer. Weak monadic second order theory of successor is not elementary-recursive. In *Logic Colloquium, Lecture Notes in Mathematics 453*. Springer-Verlag, 1975.

[15] N. Robertson and P. Seymour. Graph minors. ii. algorithmic aspects of treewidth. *J. of Algorithms*, 7:309–322, 1986.

[16] A. Sistla, M. Vardi, and P. Wolper. The complementation problem for Büchi automana with applications to temporal logic. *Theo. Comp. Sci.*, 49:217–237, 1987.

[17] L. Stockmeyer. *The complexity of decision problems in automate theory and logic*. PhD thesis, Dept. of Elec. Eng., MIT, 1974.

[18] L. Stockmeyer. The polynomial hierarchy. *Theo. Comp. Sci.*, 3:1 – 22, 1976.

[19] M. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 15:1 – 37, 1994.

Appendix A: Proofs of Theorems in Section 3

Lemma 3.1. Given a QPTL formula φ in Σ_k^{QPTL}, there exists a non-deterministic finite automaton of size $g(k, |\varphi|)$ that accepts the same language as φ.

Proof. We use the standard automata-theoretic technique [1, 16, 19], applied to nondeterminsitic automata on finite words. The induction is on the structure of the formula.

For the base case, where ψ is in Σ_1^{QPTL}, we build a finite automaton with size $g(1, |\psi|)$ as follows. For the quantifier-free part of ψ, which we call ψ', we can build a (non-deterministic) finite automaton of size $2^{|\psi'|} \leq g(1, |\psi|)$ that accepts the same language as ψ' [16]. We can now project out the quantifier variables of ψ; this does not increase the size of the automaton.

For the inductive case, given a formula ψ in Σ_k^{QPTL}, it is in the form $(\exists p_1) \dots (\exists p_n) \neg \psi'$, where ψ' is a formula in Σ_{k-1}^{QPTL}. The induction hypothesis states that the language accepted by ψ' can be accepted by an automaton A' of size $g(k-1, |\psi'|)$. We build an automaton A that accepts the same language as ψ by complementing A' and projecting the result onto the free variables of ψ. Complementing A' requires an exponential blowup in the size of A', and projection can be applied with no additional size increase. Thus, the size of A is in $2^{g(k-1,|\psi'|)} \leq g(k, |\psi|)$. $\square$

Theorem 3.2. (Analogous to Theorem 4.1, [16])The satisfiability problem for Σ_k^{QPTL}, where $k \geq 1$, is complete for $NSPACE(g(k-1, |\varphi|))$.

Proof. The automaton in Lemma 3.1 can be built "on-the-fly", and emptiness can be checked in non-deterministic logspace in the size of the automaton. The lower bound can be proved as in [16], as the proof there uses only finite prefixes of infinite words to describe Turing-machine computations. $\square$

Theorem 3.3. The finite model-checking problem for QPTL is PSPACE-complete.

Proof. The PSPACE upper-bound is a direct corollary of the PSPACE-completeness of monadic second-order logic (MSO) model checking on words [11], since the temporal operators of QPTL can be syntactically interpreted in MSO. PSPACE-hardness comes from the fact that the QBF decision problem is a special case of QPTL finite model-checking problem on a model with only one state. $\square$

Theorem 3.4. The finite model-checking problem for a formula φ in Σ_k^{QPTL} on a finite word model of size n has time complexity $O(g(k, |\varphi|)n)$.

Proof. We also note that all automaton transformations used in Lemma 3.1 can be performed in the same time bound as their corresponding space bound. As a result, the automaton that accepts the same language as φ have size $g(k, |\varphi|)$ and can be build in time $O(g(k, |\varphi|))$. Checking whether an NFA of size m accepts a word of length n can be performed in time $O(mn)$. Thus, model checking of QPTL can be performed in time $O(g(k, |\varphi|)n)$. $\square$

Lemma 3.8. There are $g(h, n)$ different tags in $\mathsf{tags}_h(n)$.

Proof. We proceed by induction. For the base case, tags of level 1 are bit strings of length n, so there are $2^n = g(1, n)$ different tags in $\mathsf{tags}_1(n)$. For the inductive case, $\mathsf{tags}_{i+1}(n)$ is the power set $\mathcal{P}(\mathsf{tags}_i(n))$, so by the inductive hypothesis, the number of tags is $2^{g(i,n)} = g(i+1, n)$. $\square$

Lemma 3.9. The size $s_h(n)$ of a tag in $\mathsf{tags}_h(n)$, for $n > 2$, is at most $g(h-1, 2n)$.

Proof. We proceed by induction. For the base case, where $h = 1$, we know the size of the tag $s_1(n)$ is $n + 2 \leq 2n = g(0, 2n)$. For the inductive case, we use the properties of g where for $h \geq 1$, $g(h, n)^2 \leq g(h, 2n)$, and for each $c > 0$, $g(h, n) + c \leq g(h, n + c)$. Both can be proved by a simple induction on h. Now $s_h(n) = g(h-1, n)(s_{h-1}(n) + 1) + 2 \leq g(h-1, n)(g(h-2, 2n) + 1) + 2 \leq g(h-1, n)(g(h-2, 2n) + 2)$. Since $h \geq 2$ and $n > 2$, we have $g(h-2, 2n) + 2 \leq g(h-2, 2n+2) \leq g(h-2, 2^n) \leq g(h-1, n)$. Thus, $s_h(n) \leq g(h-1, n)^2 \leq g(h-1, 2n)$. $\square$

Lemma 3.10. For a given w and $h \leq \log^*(w) - 2$, there exists $n > 2$ such that:

- There are at least w distinct tags in $\mathsf{tags}_h(n)$.
- The size of each individual tag in $\mathsf{tags}_h(n)$ is at most w.

Proof. We choose n such that $g(h, n-1) < w \leq g(h, n)$ so, by Lemma 3.8, we have at least w distinct tags. Since $h < \log^*(w) - 1$, we have $g(h, 2) \leq g(\log^*(w) - 2, 2) = g(\log^*(w) - 1, 1) \leq w$, so $n > 2$. By Lemma 3.9, each tag in $\mathsf{tags}_h(n)$ is of size at most $g(h-1, 2n) \leq g(h-1, 2^{n-1}) = g(h, n-1) < w$. $\square$

Lemma 3.12. For every k, the length of the word $\pi_{w,I}$ is polynomial in w, and $\pi_{w,I}$ can be generated in time polynomial in w. In addition, the polynomial functions do not depend on k.

Proof. We look at instance sizes $w > g(k, 1)$. Since $h = k - 1 \leq \log^*(w) - 2$, we can apply Lemma 3.10 to show that every tag has size less than w. So, $|\pi_{w,I}| \leq (w(w+1)+1)w$, which is in $O(w^3)$, and the coefficient is independent from k. Intuitively, the tagged word $\pi_{w,I}$ is obtained by tagging each position in a tiling with its column address in binary, tagging each address bit with its address in binary, and so on, iterated up to h levels. Each level adds a logarithmic term of overhead (i.e., $\log(w)$, $\log(\log(w))$, and so on) in the time needed to construct the label. Since $\pi_{w,I}$ is of length $O(w^3)$, we can generate it in time $O(w^4)$. $\square$

Claim 3.14. $\pi_{[p,p']}$ and $\pi_{[q,q']}$ are identical tags iff $\varphi_{h,n}(p, p', q, q')$.

Proof. Again, we proceed by induction. The base case for $\varphi_{1,n}$ is quite simple. The fact that the singletons p, p', q, q' point to the correct position and are in the correct order is asserted by the formula. For well-formedness, we also need to check that the subwords $\pi_{[p,p']}$ and $\pi_{[q,q']}$ form actual tags, instead of being fragments that only mark part of a tag or fragments that cross multiple tags. Since the input word is constructed such that all tag boundary pairs must contain complete tags, $\pi_{[p,p']}$ or $\pi_{[q,q']}$ cannot be too short. And for $\pi_{[p,p']}$ or $\pi_{[q,q']}$ to be too long, they have to contain multiple tags, but we have asserted that tag markers $\langle 1 \rangle$ does not appear inside the subword. The equality of the subwords $\pi_{[p,p']}$ and $\pi_{[q,q']}$ is also directly asserted by the formula.

The inductive case is slightly more involved. Assuming that the construction is correct for level $h-1$, we show that it is correct for level h. format_h asserts that both tags are well formed exactly like the base case. In $\mathsf{checktags}_h$, we use the quantifier on b and b' to choose a set of positions that marks tags in $\mathsf{tags}_{h-1}(n)$, which we need to compare equality on. The set of positions b marks all starting positions of tags in $\mathsf{tags}_{h-1}(n)$ inside the subword $\pi_{[p,p']}$ (which is a tag in $\mathsf{tags}_h(n)$), and b' marks all starting positions of tags in $\mathsf{tags}_{h-1}(n)$ inside the subword $\pi_{[q,q']}$. The variables r, r', s, and s' are universally quantified. If any of them are not a singleton, or does not outline tags in $\mathsf{tags}_{h-1}(n)$, or are in the wrong order, $\mathsf{idmap}_h(r, r', s, s')$ returns true. If $\pi_{[r,r']}$ and $\pi_{[s,s']}$ do point to tags of the right type, but they are not tags we are actively checking (as marked by b and b'), the antecedent $(G(r \to b) \wedge G(s \to b'))$ part fails, skipping the check. Otherwise, idmap_h is used to check that for all pair of tags in $\mathsf{tags}_{h-1}(n)$ between $\pi_{[p,p']}$ and $\pi_{[q,q']}$, if they are the same, they map to the same value. Since the tags $\pi_{[p,p']}$ and $\pi_{[q,q']}$ are well formed by construction, this proves $\pi_{[p,p']}$ and $\pi_{[q,q']}$ are the same tag in $\mathsf{tags}_h(n)$ iff $\varphi_{h,n}(p, p', q, q')$. □

Lemma 3.15. Given a tiling instance (T, w, I), and an integer $k \geq 2$, we can generate in time $O(k + \log^{k-1}(w))$ a Σ_k^{QPTL} formula $\varphi_{T,w}$ of size $O(k + \log^{k-1}(w))$ such that $\pi_{w,I} \models \varphi_{T,w}$ iff (T, w, I) have a solution, and the coefficient in $O(k + \log^{k-1}(w))$ is independent from k.

Proof. Since $\varphi_{T,w}$ guesses a tiling non-deterministically and checks it through $\varphi'_{T,w}$, we have $\pi_{w,I} \models \varphi_{T,w}$ iff (T, w, I) has a solution, satisfying Assumption 1 of Theorem 3.5. Our choice of n such that $g(k-1, n-1) < w \leq g(k-1, n)$ means $n = \lceil \log^{k-1}(w) \rceil$, so $|\varphi'_{T,w}|$ and, in turn, $|\varphi_{T,w}|$ are in $O(k+n) = O(k + \log^{k-1}(w))$. The coefficient in $O(k + \log^{k-1}(w))$ is a constant based on our formula schemas. The formula can be generated in time linear in its size. □

Lemma 3.16. There exists a constant $c > 0$ such that for every $k > 2$, the formula $\varphi_{T,w}$ is small enough so that $g(k-1, c|\varphi_{T,w}|)$ is in $\mathsf{poly}(w)$, and $\varphi_{T,w}$ can be constructed in time polynomial in w.

Proof. By Lemma 3.15, $|\varphi_{T,w}|$ is in $O(k + \log^{k-1}(w))$. So, there is a constant c and w_c such that for all $w > w_c$, $c|\varphi_{T,w}| < \log^{k-1}(w)$. Thus, $g(k-1, c|\varphi_{T,w}|) < g(k-1, \log^{k-1}(w)) = w$. We can construct $\varphi_{T,w}$ in time $O(\log^{k-1}(w))$, which is much smaller than $O(w)$. □

Appendix B: Proofs of Theorems in Section 4

Claim 4.6. If φ is quantifier free, then $\pi \models \varphi$ iff $\pi_Q \models \varphi_Q$.

Proof.
- $\Rightarrow$: We have $\pi \models \varphi$. We can extend π_Q with an assignment A on the quantified variables corresponding to subformulas in $\mathsf{sub}'(\varphi)$ such that $\pi_Q, A \models \varphi'_Q$ (thus $\pi_Q \models \varphi_Q$) by annotating the model with satisfied subformulas as follows: For every $\psi \in \mathsf{sub}'(\varphi)$, $A(p_{\psi,i}) = 1$ iff $\pi, i \models \psi$. Now we look at the clauses in φ'_Q. We have $A(p_{\varphi,0}) = 1$ since $\pi, 0 \models \varphi$. Every clause in $\bigwedge_{\psi\in\mathsf{sub}'(\varphi)} C_\psi$ holds because they encode exactly the semantics of QPTL as defined in Section 2. Thus, $\pi_Q, A \models \varphi'_Q$, and $\pi_Q \models (\exists p_{\psi_1,0}) \dots (\exists p_{\psi_m,|\pi|-1})\varphi'_Q$.
- $\Leftarrow$: A similar construction can be performed as above. We have $\pi_Q \models (\exists p_{\psi_1,0}) \dots (\exists p_{\psi_m,|\pi|-1})\varphi'_Q$. Thus, we have an assignment A on the quantified variables such that $\pi_Q, A \models \varphi'_Q$. We use induction on the structure of φ to show for any sub-formula ψ of φ, if $p_{\psi,i} \in A \cup \pi_Q$, then $\pi, i \models \psi$. For the base case of atomic sub-formulas, we have from the construction of π_Q that $\pi_Q \models p_{a,i}$ iff $\pi, i \models a$. For all inductive cases, there exist clauses in φ'_Q to ensure the connection between ψ and $\psi'(\psi'')$ for $\psi = X\psi'$, $\psi = F\psi'$, $\psi = G\psi'$, $\psi = \psi' \wedge \psi''$, $\psi = \psi' \vee \psi''$. Since there is a unit clause in φ'_Q to assert $p_{\varphi,0}$, we have $A \models p_{\varphi,0}$. From the induction, we have $\pi \models \varphi$. $\square$

Claim 4.8. The matrix of $\varphi_{Q,\pi}$ have pathwidth at most $2|\varphi| - 1$.

Proof. The matrix of $\varphi_{Q,\pi}$ have clauses connecting at most two neighboring unrolling of propositions, so there is the following path decomposition $\{\{p_{\psi,0}, p_{\psi,1} \mid \psi \in \mathsf{sub}(\varphi)\}, \{p_{\psi,1}, p_{\psi,2} \mid \psi \in \mathsf{sub}(\varphi)\}, \dots, \{p_{\psi,|\pi|-2}, p_{\psi,|\pi|-1} \mid \psi \in \mathsf{sub}(\varphi)\}\}$, where the requirements of a path decomposition is met:

- The decomposition is a path.
- All constraints (clauses) occur in one path node.
- Every proposition only occurs in a sub-path (two consecutive nodes) of the decomposition.

The width of the decomposition is $2|\varphi| - 1$. $\square$

Appendix C: Proofs of Theorems in Section 5

Theorem 5.3. Given a QPTL model-checking game (π, φ, G), we can construct in polynomial time a QBF formula $\varphi_{Q,\pi,G}$ where $\pi \models_G \varphi$ iff $\models \varphi_{Q,\pi,G}$. The size of $\varphi_{Q,\pi,G}$ is polynomial in $|\varphi|$, $|\pi|$, and $|G|$, and the pathwidth of $\varphi_{Q,\pi,G}$ is at most $2|\varphi| - 1$.

Proof. We use the QBF translation of QPTL finite model checking presented in Theorem 4.5 as a tool. First, we construct φ_Q in QBF from φ and π. Next, we unroll the quantifiers implicit in G onto φ_Q. For each round in G where $G_i = \{d_1, \dots d_k\}$, the corresponding QBF prefix block Q_i is $(qp_{x_1,d_1}) \dots (qp_{x_j,d_1})(qp_{x_1,d_2}) \dots (qp_{x_j,d_k})$, where $P = \{x_1, \dots x_j\}$ is the set of propositions in φ excluding $\mathsf{p}_\exists$ and $\mathsf{p}_\forall$, and q is $\exists$ if i is even and $\forall$ if i is odd. I.e., for every round i in G, there has $|P| \times |G_i|$ additional quantifiers in the prefix to represent the choices made by the player corresponding to G_i. The formula $\varphi_{Q,G}$ is in turn $Q_1 \dots Q_m \varphi_Q$. The propositional model π_Q is generated from π in the same manner as Theorem 4.5, except that the assignment to $p_\exists$ and $p_\forall$ is defined by G instead of by π. π_Q can be written as a quantifier-free QBF formula of unit clauses (as usual, when lifting the quantifiers in $\varphi_{Q,G}$ to the overall formula, we only need to keep the unit clauses in π_Q that correspond to the free propositions of $\varphi_{Q,G}$) and conjoined with $\varphi_{Q,G}$ to make $\varphi_{Q,\pi,G}$. Now we have $\pi \models_G \varphi$ iff $\models \varphi_{Q,\pi,G}$. Since $|\varphi_Q|$ is $O(|\pi||\varphi|)$, $|\varphi_{Q,\pi,G}|$ is in $O(|\pi||\varphi| + |G|)$. And because $\varphi_{Q,\pi,G}$ have the same matrix as $\varphi_{Q,\pi}$ except for unit clauses referring to $\mathsf{p}_\exists$ and $\mathsf{p}_\forall$, the same path decomposition can be used to show a pathwidth of at most $2|\varphi| - 1$. $\square$

Theorem 5.5. Given a tiling game (T, w, I), we can construct in polynomial time a QPTL model-checking game $(\pi_{w,I}, \varphi_{T,w,G}, G)$ such that the Constructor wins the tiling game (T, w, I) iff $\pi_{w,I} \models_G \varphi_{T,w,G}$, the model $\pi_{w,I}$ and the play order G have size polynomial in w, and the formula $\varphi_{T,w,G}$ has size $O(\log^*(w))$.

Proof. We give the formula here:

- $\varphi_e := \mathsf{sing}(e) \wedge G(e \rightarrow \mathsf{p}_\forall) \wedge G(e \leftrightarrow (e_D \vee e_H \vee e_V))$. This formula checks the error is performed by the Spoiler and guesses the type of the error.
- $\varphi'_D := G((\neg e_D \wedge (\mathsf{p}_\forall \vee \mathsf{p}_\exists)) \rightarrow (Fe \vee ((\bigvee_{d\in D} \mathsf{p}_d) \wedge (\bigwedge_{d,d'\in D, d\neq d'} \neg(\mathsf{p}_d \wedge \mathsf{p}_{d'}))))) \wedge (e_D \rightarrow ((\bigwedge_{d\in D} \neg \mathsf{p}_d) \vee (\bigvee_{d,d'\in D, d\neq d'} \mathsf{p}_d \wedge \mathsf{p}_{d'}))))$
- $\varphi'_H := (\forall s, s', t, t')((\mathsf{sing}(s) \wedge \mathsf{sing}(s') \wedge \mathsf{sing}(t) \wedge \mathsf{sing}(t') \wedge G(s \rightarrow \mathsf{p}_{\langle k-1\rangle}) \wedge G(s' \rightarrow \mathsf{p}_{\langle /k-1\rangle}) \wedge G(t \rightarrow \mathsf{p}_{\langle k-1\rangle}) \wedge G(t' \rightarrow \mathsf{p}_{\langle /k-1\rangle}) \wedge G(s \rightarrow Fs') \wedge G(s' \rightarrow Ft) \wedge G(t \rightarrow Ft') \wedge \mathsf{notbetween}(s, s', \mathsf{p}_{\langle k-1\rangle}) \wedge \mathsf{notbetween}(s', t, \mathsf{p}_{\langle k-1\rangle}) \wedge \mathsf{notbetween}(s', t, \mathsf{p}_\mathsf{r}) \wedge \mathsf{notbetween}(t, t', \mathsf{p}_{\langle k-1\rangle})) \rightarrow (((\bigvee_{\langle d,d'\rangle\in H}(G(s' \rightarrow X\mathsf{p}_d) \wedge G(t' \rightarrow X\mathsf{p}_{d'}))) \vee G(s' \rightarrow Fe)) \wedge (G(s' \wedge Xe_H) \rightarrow \neg(\bigvee_{\langle d,d'\rangle\in H}(G(s' \rightarrow X\mathsf{p}_d) \wedge G(t' \rightarrow X\mathsf{p}_{d'}))))))$

- $\varphi'_V := (\forall s, s', t, t')((\varphi_{k-1,n}(s, s', t, t') \wedge G(s' \rightarrow (F(\mathsf{p_r} \wedge Ft) \wedge \neg F(\mathsf{p_r} \wedge XF(\mathsf{p_r} \wedge Ft))))) \rightarrow ((\bigvee_{\langle d,d' \rangle \in V}(G(s' \rightarrow X\mathsf{p}_{d'}) \wedge G(t' \rightarrow X\mathsf{p}_d)) \vee (s' \rightarrow Fe)) \wedge (G(s' \wedge Xe_V) \rightarrow \neg(\bigvee_{\langle d,d' \rangle \in V}(G(s' \rightarrow X\mathsf{p}_d) \wedge G(t' \rightarrow X\mathsf{p}_{d'})))))$
- Condition 2 is encoded by $\varphi_\forall := (\exists e, e_D, e_H, e_V)(\varphi_e \wedge \varphi'_H \wedge \varphi'_V \wedge \varphi'_D)$.
- The formula for the model-checking game is $\varphi_{T,w,G} := \varphi'_{T,w} \vee \varphi_\forall$.

In the model-checking game, G encodes the same play order for the Constructor and the Spoiler as the tiling game. φ is a disjunction on the two winning conditions. The $\varphi'_{T,w}$ portion of $\varphi_{T,w,G}$ checks that the Constructor wins the tiling game through producing an tiling that is a solution to (T, w, I). The $\varphi_\forall$ portion of $\varphi_{T,w,G}$ checks that the Constructor wins the tiling game by guessing the position of the first failing play and checking that it is actually an error belonging to the Spoiler, as well as checking that all plays before the error are correct. So $\pi_{w,I} \models \varphi_{T,w,G}$ iff the Constructor wins (T, w, I). For the size bound, we use the same formula alternation depth k as Theorem 4.4, where k is the least odd number $\geq \log^*(w) - 2$. $\pi_{w,I}$ has the same polynomial $O(w^4)$ size as in Lemma 3.12, and G has size at most $|\pi_{w,I}|$. By construction, $\varphi_{T,w,G}$ has size $O(|\varphi_{k-1,n}|)$, which by Theorem 3.13, is $O(k - 1 + \log^{k-1}(w))$. Since $\log^{k-1}(w)$ is bounded by a constant from our choice of k, $|\varphi_{T,w,G}|$ is in $O(k)$, which is $O(\log^*(w))$. □