



QLTL Model-Checking

François Laroussinie  

IRIF, Université Paris Cité, France

Loriane Leclercq  

Ecole Centrale de Nantes, CNRS, LS2N, Nantes, France

Arnaud Sangnier 

DIBRIS, Università di Genova, Italy

Abstract

Quantified LTL (QLTL) extends the temporal logic LTL with quantifications over atomic propositions. Several semantics exist to handle these quantifications, depending on the definition of executions over which formulas are interpreted: either infinite sequences of subsets of atomic propositions (aka the “tree semantics”) or infinite sequences of control states combined with a labelling function that associates atomic propositions to the control states (aka the “structure semantics”). The main difference being that in the latter different occurrences of a control state should be labelled similarly. The tree semantics has been intensively studied from the complexity and expressivity point of view (especially in the work of Sistla [21, 22]) for which the satisfiability and model-checking problems are known to be TOWER-complete. For the structure semantics, French has shown that the satisfiability problem is undecidable [8]. We study here the model-checking problem for QLTL under this semantics and prove that it is EXPSPACE-complete. We also show that the complexity drops down to PSPACE-complete for two specific cases of structures, namely path and flat ones.

2012 ACM Subject Classification Theory of Computation → Logic

Keywords and phrases Quantified Linear-time temporal logic, Model-checking, Verification, Automata theory

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.35

Funding Loriane Leclercq: ANR project ProMiS ANR-19-CE25-0015

1 Introduction

Temporal logics (TLs) have been introduced in computer science in the late 1970’s by Pnueli [19]; they provide a powerful formalism for specifying correctness properties of evolving systems. Various kinds of temporal logics have been defined, with different expressive power and algorithmic properties. For instance, the *Computation Tree Logic* (CTL) expresses properties of the computation tree of the system under study (time is branching: a state may have several successors), and the *Linear-time Temporal Logic* (LTL) expresses properties of one execution at a time (a system is viewed as a set of executions).

In verification, we are mainly interested in two decision problems: the satisfiability problem (given a formula, decide whether there exists a model for it) and the model-checking problem (given a potential model and a formula, decide whether the formula holds true or not over the model). For the temporal logic LTL, it is well known that both these problems are PSPACE-complete. The key argument is the construction of an automaton that recognises the models (a set of infinite words) of a fixed formula. For CTL, the model-checking can be done in polynomial time while satisfiability is EXPTIME-complete (these decision procedures can also be based on automata techniques, but this time we need to use tree automata).

In terms of expressiveness, classical temporal logics like CTL or LTL still have some limitations: in particular, they lack the ability of *counting*. For instance, they cannot express that an event occurs (at least) at every even position along a path, or that a state has



© François Laroussinie, Loriane Leclercq, and Arnaud Sangnier;
licensed under Creative Commons License CC-BY 4.0

32nd EACSL Annual Conference on Computer Science Logic (CSL 2024).

Editors: Aniello Murano and Alexandra Silva; Article No. 35; pp. 35:1–35:18

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

exactly two successors. In order to cope with this, temporal logics have been extended with *propositional quantifiers* [21, 22]: in this framework, a formula $\exists p.\varphi$ is verified when there is a way of labelling the current execution by p in such a way that φ holds true.

Different semantics for quantified TLs have been studied in the literature depending on the definition of the quantifiers. For example, consider a formula Φ of the linear-time temporal logic QLTL (i.e., LTL extended with propositions quantifiers). We can interpret Φ either over an infinite word over the alphabet 2^{AP} (where AP denotes the set of atomic propositions used in Φ), or over a *labelled execution* composed by an infinite sequence of control states (some of them may have several occurrences along the execution) and a labelling function which maps each control state to a set of atomic propositions. These two points of view coincide for LTL formulas. But as soon as quantifiers are added, the two semantics are quite different: in the first one (called the *tree semantics*), each position along the execution corresponds to a new (control) state whose labelling is independent of the others. In the second one (called the *structure semantics*), two occurrences of the same control state are always labelled in the same manner.

The tree semantics has been studied extensively, especially in [22] where QLTL is proven to be as expressive as S1S (the monadic second-order logic with one successor), and the satisfiability and model-checking problems for the k -alternation fragment are shown k -EXPSpace-complete.

For QLTL with the structure semantics, the main result concerns the satisfiability problem. In this setting, the satisfiability problem is stated as follows: given a QLTL formula Φ , does there exist an infinite word $\rho \in Q^\omega$ where Q is an alphabet describing the control states, and a labelling $\ell : Q \mapsto 2^{\text{AP}}$ (where AP is the set of atomic propositions occurring in Φ) such that the labelled execution (ρ, ℓ) satisfies Φ ? French has shown that this problem is undecidable even when Q is assumed to be finite [8].

In this paper, we focus on the model-checking problem for QLTL under the structure semantics. We first explain the types of properties we can express with this logic, and then we show that the problem is EXPSpace-complete. We also consider two other special model-checking problems: when the structure is a path of the form $\rho_1 \cdot \rho_2^\omega$ and when the structure is *flat*. In both cases, we prove that the problem is then PSPACE-complete.

Related results. Adding quantifications over atomic propositions in LTL has been first considered in [21, 22] for the tree semantics: both the expressiveness and the complexity have been studied. The stutter-invariant fragment of QLTL, with a restricted notion of propositional quantification, was developed in [6]. Proof systems for QLTL were developed (still with the tree semantics), both with and without past-time modalities [11, 9].

For QLTL under the structure semantics, the main contribution is [8] where French shows that satisfiability is undecidable.

Propositional quantifications have also been studied for branching-time temporal logics. The extension of CTL* with external existential quantification was compared with tree automata over binary trees [5]. In [13], restricted quantifications are added to CTL. In [7], the satisfiability of QCTL* was proven undecidable in the structure semantics, and decidable in the tree semantics. In [14], QCTL and QCTL* are considered (both for the structure semantics and the tree semantics): the expressive power (relationship with the monadic second-order logic MSO) and complexity (for model-checking and satisfiability) are studied. In [1], the satisfiability problem of several fragments of QCTL (e.g. with only the **EX** modality) under the tree semantics is shown to be TOWER-complete, i.e. exactly as it is for the full QCTL logic. In [10], a model-checking algorithm is proposed for QCTL with the structure semantics based on a reduction to QBF and experimental results are discussed (with several QBF solvers).

The model-checking problem restricted to a path has been studied in [16, 18]. The (existential) model-checking of LTL over flat structures has been first studied in [12] where it is shown to be in NP and later on, in [4], the problem has been shown to be NP-complete even when adding past operators.

2 Model and Logic

2.1 Kripke Structures and Labelled Executions

In formal verification, systems are usually modelled with *labelled transition systems* (or *Kripke structures*): we have a finite set of control states, transitions to move from a state to another one and a labelling function which associates a set of atomic propositions with every control state. In this paper, we consider a countable set of atomic propositions denoted by AP. Kripke structures are then formally defined as follows.

► **Definition 1.** A Kripke structure (KS) is a tuple $\mathcal{K} = \langle Q, R, q_{in}, \ell \rangle$ where Q is a finite set of states, $R \subseteq Q \times Q$ is a set of transitions (we assume that for any $q \in Q$, there exists $q' \in Q$ s.t. $(q, q') \in R$), $q_{in} \in Q$ is the initial control state and $\ell: Q \rightarrow 2^{AP}$ is a labelling function.

The semantics of such a transition system is either the structure itself or its unfolding (an infinite labelled tree), and an execution can be seen either as an infinite sequence of labelled control states or an infinite word over the alphabet 2^{AP} (the underlying control states are forgotten). These two points of view coincide when considering fragments of CTL* logic: the truth value of a formula does not depend on this choice of semantics. There is no way to distinguish two different control states if their behaviours defined in terms of atomic proposition and transitions are “equivalent” (i.e. bisimilar) w.r.t. the considered logic. Classically the infinite 2^{AP} -labelled tree unfolding is used to base decision procedures over automata theory (word automata for linear-time temporal logic, or tree automata for branching-time temporal logic). And the “structure” view is used for example to develop classical algorithms for CTL-like logics (corresponding to basic graph analysis).

Adding quantifications over atomic propositions makes a big difference (see Section 3) between these two approaches. Indeed labelling control states implies that every occurrence of a state will carry the same labels. It can be seen as a second order quantification over the control states of the structure. For example it becomes possible to specify that there exists a self-loop from a given state (we can mark a single control state in order to distinguish it from other ones). With the “labelled tree” semantics, every position can be labelled independently. As in previous works on this topic, we will refer to the former semantics as the *structure semantics* and the latter as the *tree semantics*. For the linear-time logic QLTL, the most popular approach is the tree semantics: in [22], Sistla *et al.* showed important properties about its complexity and its relation with Büchi automata. Here we consider the structure semantics where the model for QLTL formulas are labelled executions.

An *execution* ρ of a Kripke structure $\mathcal{K} = \langle Q, R, q_{in}, \ell \rangle$ is an infinite sequence of states $q_0 q_1 q_2 \dots$ such that $(q_i, q_{i+1}) \in R$ for all $i \in \mathbb{N}$. Given $i \in \mathbb{N}$, we use $\rho(i)$ to denote q_i the i -th (control) state of ρ , and $\rho_{\geq i}$ to denote $q_i q_{i+1} q_{i+2} \dots$ its i -th suffix. Finally a *labelled execution* is a pair (ρ, λ) where ρ is an execution and λ is a labelling function from Q to 2^{AP} . We use $\text{Exec}_{\mathcal{K}}(q)$ [resp. $\text{Exec}^{\text{lab}}_{\mathcal{K}}(q)$] to denote the set of executions ρ [resp. labelled executions (ρ, λ)] in \mathcal{K} starting from q , i.e. such that $\rho(0) = q$.

2.2 Syntax and (Structure) Semantics of QLTL

We present now the definition of the logic QLTL, which extends the classical linear-time temporal logic LTL with quantifications over atomic propositions. The syntax of QLTL is given by the following grammar:

$$\varphi ::= q \mid \neg\varphi \mid \varphi \vee \psi \mid \mathbf{X}\varphi \mid \varphi \mathbf{U}\psi \mid \exists p. \varphi$$

where q and p range over AP.

In the structure semantics, QLTL formulas are evaluated over *labelled executions* of a Kripke structure $\mathcal{K} = \langle Q, R, q_{in}, \ell \rangle$. Before providing the formal semantics, we need to introduce another notion. Given a set $P \subseteq \text{AP}$, two labellings λ and λ' from Q to 2^{AP} are said to be P -equivalent (denoted by $\lambda \equiv_P \lambda'$) iff $\lambda(q) \cap P = \lambda'(q) \cap P$ for every $q \in Q$. The semantics of QLTL formulas is then provided by the satisfaction relation \models between a labelled execution (ρ, λ) and a formula φ which is defined inductively as follows:

$$\begin{aligned} (\rho, \lambda) \models p &\text{ iff } p \in \lambda(\rho(0)) \\ (\rho, \lambda) \models \neg\varphi &\text{ iff } (\rho, \lambda) \not\models \varphi \\ (\rho, \lambda) \models \varphi \vee \psi &\text{ iff } (\rho, \lambda) \models \varphi \text{ or } (\rho, \lambda) \models \psi \\ (\rho, \lambda) \models \mathbf{X}\varphi &\text{ iff } (\rho_{\geq 1}, \lambda) \models \varphi \\ (\rho, \lambda) \models \varphi \mathbf{U}\psi &\text{ iff there exists } i \geq 0 \text{ s.t. } (\rho_{\geq i}, \lambda) \models \psi \text{ and } (\rho_{\geq j}, \lambda) \models \varphi \text{ for all } i > j \geq 0 \\ (\rho, \lambda) \models \exists p. \varphi &\text{ iff there exists a labelling } \lambda' \text{ s.t. } \lambda' \equiv_{\text{AP} \setminus \{p\}} \lambda \text{ and } (\rho, \lambda') \models \varphi \end{aligned}$$

In the sequel, we use standard abbreviations such as \top , \perp , \wedge , \Rightarrow and \Leftrightarrow . We also use the additional (classical) temporal modalities of LTL: $\mathbf{F}\varphi = \top \mathbf{U}\varphi$, $\mathbf{G}\varphi = \neg \mathbf{F}\neg\varphi$ and $\varphi \mathbf{R}\psi = \neg((\neg\varphi) \mathbf{U}(\neg\psi))$. Moreover, we use the following abbreviations related to quantifiers over atomic propositions: $\forall p. \varphi = \neg \exists p. \neg\varphi$, and for a set $P = \{p_1, \dots, p_k\} \subseteq \text{AP}$, we write $\exists P. \varphi$ for $\exists p_1 \dots \exists p_k. \varphi$ and $\forall P. \varphi$ for $\forall p_1 \dots \forall p_k. \varphi$.

Given a formula φ , we denote by $\text{SubF}(\varphi)$ the set of its subformulas and $\text{Prop}(\varphi)$ the set of the atomic propositions. A proposition p is said to be free in φ if it appears out of scope of some operator $\exists p. \dots$ or $\forall p. \dots$.

The size of a formula $\varphi \in \text{QLTL}$, denoted $|\varphi|$, is defined inductively by: $|q| = 1$, $|\neg\varphi| = |\exists p. \varphi| = |\forall p. \varphi| = |\mathbf{X}\varphi| = 1 + |\varphi|$, $|\varphi \vee \psi| = |\varphi \mathbf{U}\psi| = |\varphi \mathbf{R}\psi| = 1 + |\varphi| + |\psi|$. Moreover, we use $\text{th}(\varphi)$ to denote the *temporal height* of φ , i.e. the maximal number of nested temporal modalities in φ .

A formula is said to be in negated normal form (NNF) if negations apply only to atomic propositions. Any QLTL formula φ can be transformed into an equivalent NNF formula ψ s.t. $|\psi| = O(|\varphi|)$. Note that ψ is built from boolean operators \wedge and \vee , Temporal modalities \mathbf{X} , \mathbf{U} and \mathbf{R} , atomic propositions and their negations, and quantifiers \exists and \forall .

Two QLTL formulas φ and ψ are said to be *equivalent* (written $\varphi \equiv \psi$) iff for any labelled execution (ρ, λ) , we have $(\rho, \lambda) \models \varphi$ iff $(\rho, \lambda) \models \psi$. This equivalence is substitutive.

We write $\mathcal{K} \models_{\exists} \varphi$ when φ is satisfied by a labelled execution (ρ, ℓ) in \mathcal{K} rooted at the initial state q_{in} , and $\mathcal{K} \models_{\forall} \varphi$ when every such labelled execution in \mathcal{K} satisfy φ .

3 What can we express with QLTL?

To illustrate the kind of properties that can be expressed with QLTL with the structure semantics, we introduce the following abbreviation:

$$\exists^1 p. \varphi = \exists p. \left(\mathbf{F}p \wedge (\forall p'. (\mathbf{F}(p \wedge p') \Rightarrow \mathbf{G}(p \Rightarrow p'))) \wedge \varphi \right)$$

and its dual $\forall^1 p.\varphi = \neg\exists^1 p.\neg\varphi$. Informally $\exists^1 p.\varphi$ is satisfied if one can label exactly one control state (reachable from the current position) by p in order to make φ to be satisfied by the execution.

Now we can express the fact that a labelled execution (ρ, λ) is built from a *finite* number of control states (which is of course always the case when considering finite KS). The following formula expresses this property:

$$\Phi_{\text{finite}} = \exists^1 q.\forall p.\left((p\mathbf{U}(q \wedge p)) \Rightarrow (\mathbf{G}p)\right)$$

Assume that the property holds true for (ρ, λ) . Then there is a *last occurring* control state: the control state whose first occurrence is located after the first occurrence of every other control state. Clearly if we label this state by q , any p -labelling of all states occurring before q -state labels all states of ρ . Conversely if the formula is satisfied by some execution (ρ, λ) , it implies that the p -labelling limited to control states occurring over a finite prefix allows us to label all ρ -states.

One can also ensure that at least k distinct control states occur infinitely often along an execution with the following formula:

$$\Phi_{\geq k} = \exists^1 q_1 \dots q_k. \bigwedge_{1 \leq i \neq j \leq k} \mathbf{G}(\neg q_i \vee \neg q_j) \wedge \bigwedge_{1 \leq i \leq k} (\mathbf{GF}(q_i))$$

We can also specify that any *position* satisfying the proposition a is always followed by the same control state with:

$$\Phi_{\text{succ}} = \exists^1 q.(\mathbf{G}(a \Rightarrow \mathbf{X}q))$$

An execution (ρ, λ) is deterministic if every control state occurring along ρ is always followed by the same control state. This property can easily be expressed with QLTL:

$$\Phi_{\text{deter}} = \forall^1 q.\forall^1 q'.(\mathbf{F}(q \wedge \mathbf{X}q') \Rightarrow \mathbf{G}(q \Rightarrow \mathbf{X}q'))$$

Indeed, assume we label two control states by q and q' respectively, and the formula $(q \wedge \mathbf{X}q')$ holds true at a position along ρ , then we require that every occurrence of q is followed by q' . This is precisely the definition of deterministic execution.

When the set of control states Q is fixed and finite, any quantification $\exists p.\varphi$ is equivalent to some disjunction of all possible subsets associated with the proposition p . And then any QLTL formula is equivalent *for executions built with exactly $|Q|$ -control states* to some QLTL formula with a unique existential quantification:

► **Proposition 2.** *Let $k \geq 1$. Any QLTL formula Φ is equivalent for labelled executions containing exactly k distinct control states to some QLTL formula $\tilde{\Phi} = \exists p_0 \dots p_{2^k-1}.\hat{\Phi}$ where $\hat{\Phi}$ belongs to LTL.*

Proof. Consider a set of control states Q s.t. $|Q| = k$. We use 2^k new atomic propositions in order to label every possible subset over Q . The formula $\hat{\Phi}$ combines two parts: the first one ensures this labelling of subsets and the second one is just Φ where every existential (resp. universal) quantification is replaced by a disjunction (resp. conjunction). Formally given a QLTL formula φ , we define $\tilde{\varphi}^k$ inductively as follows:

$$\begin{aligned} \tilde{p}^k &= p \quad \widetilde{\varphi_1 \wedge \varphi_2}^k = \widetilde{\varphi_1}^k \wedge \widetilde{\varphi_2}^k & \widetilde{\neg \varphi_1}^k &= \neg \widetilde{\varphi_1}^k & \widetilde{\varphi_1 \mathbf{U} \varphi_2}^k &= \widetilde{\varphi_1}^k \mathbf{U} \widetilde{\varphi_2}^k \\ \widetilde{\mathbf{X} \varphi_1}^k &= \mathbf{X} \widetilde{\varphi_1}^k & \widetilde{\exists p.\alpha}^k &= \bigvee_{0 \leq i < 2^k} \alpha[\widetilde{p \mapsto p_i}]^k & \widetilde{\forall p.\alpha}^k &= \bigwedge_{0 \leq i < 2^k} \alpha[\widetilde{p \mapsto p_i}]^k \end{aligned}$$

And then it remains to show that for any execution (ρ, ℓ) and for any QTL formula Φ , if the number of distinct control states occurring along ρ equals k , we have:

$$(\rho, \ell) \models \Phi \quad \text{iff} \quad (\rho, \ell) \models \exists p_0 \dots p_{2^k-1}. \left(\bigwedge_{0 \leq i < j < 2^k} \mathbf{F}(p_i \leftrightarrow \neg p_j) \right) \wedge \tilde{\Phi}^k$$

The proof is based on the fact that the labelling of every p_i labels a specific subset of control states. This is ensured by the first part of the formula: there are 2^k labellings and all are different. Property 2 is then proved. \blacktriangleleft

We can observe several important differences with the tree semantics¹. For example, under the tree semantics, the formula Φ_{deter} is valid (as every position corresponds to a new control state). It is also well known that the property $\text{Even}^c(p)$ defined by “a control state is labelled by p iff at least one of its occurrences is located at an even position along the execution” can easily be expressed with the tree semantics, but this is not true anymore with the structure semantics:

► **Proposition 3.** *With the structure semantics, there is no QTL formula equivalent to the property $\text{Even}^c(p)$.*

Proof. We reuse a construction of [24] for proving that LTL cannot express that a proposition holds for every even state. Consider the set of control states $Q = \{q, q'\}$ and the execution $\rho_k = q^k \cdot q' \cdot q^\omega$ for $k > 0$. We can easily observe that the labelled execution (ρ_k, ℓ) satisfies $\text{Even}^c(p)$ if and only if (1) k is even and both q and q' are labelled by p , or (2) k is odd and only q is labelled by p .

Now we can show that given $k, k' \geq 1$ and any LTL formula ψ s.t. $|\psi| \leq \min(k, k')$, we have: $(\rho_k, \ell) \models \psi \Leftrightarrow (\rho_{k'}, \ell) \models \psi$. The proof of this result is done by induction over $|\psi|$:

- $|\psi| = 1$: ψ is an atomic proposition, and $\ell(\rho_k(0)) = \ell(\rho_{k'}(0))$ as $k, k' \geq 1$.
- $|\psi| = n + 1$: We distinguish several cases (and omit the case of Boolean combinators):
 - $\psi = \mathbf{X}\psi_1$: consider $k, k' \geq n + 1$ and assume $(\rho_k, \ell) \models \mathbf{X}\psi_1$. Then $(\rho_{k-1}, \ell) \models \psi_1$. As both $k - 1$ and $k' - 1$ are greater or equal to $n = |\psi_1|$, we have by i.h. $(\rho_{k'-1}, \ell) \models \psi_1$ and then $(\rho_{k'}, \ell) \models \mathbf{X}\psi_1$.
 - $\psi = \psi_1 \mathbf{U} \psi_2$: consider $k, k' \geq n + 1$ and assume $(\rho_k, \ell) \models \psi_1 \mathbf{U} \psi_2$. Then there exists $i \geq 0$ s.t. $((\rho_k)_{\geq i}, \ell) \models \psi_2$ and for any $0 \leq j < \infty$ we have $((\rho_k)_{\geq j}, \ell) \models \psi_1$. We distinguish several cases:
 - * $k > k'$ and $k - i \leq k'$: then ψ_2 holds true for $((\rho_{k'})_{\geq i - (k - k')}, \ell)$ and ψ_1 is verified for any $((\rho_{k'})_{\geq j}, \ell)$ for $0 \leq j < i - (k - k')$, and this provides the result $((\rho_{k'}), \ell) \models \psi$.
 - * $k > k'$ and $k - i > k'$: In this case, we know that $((\rho_k)_{\geq i}, \ell) \models \psi_2$ is equivalent to $((\rho_{k-i}), \ell) \models \psi_2$ and $k - i$ and k' are both greater than $|\psi_2|$, which allows us to use the i.h. and deduce $((\rho_{k'}), \ell) \models \psi_2$, and then $((\rho_{k'}), \ell) \models \psi$.
 - * $k < k'$: if $(\rho_k, \ell) \models \psi_1$, then we can use the i.h. as in the previous case to deduce that for all j s.t. $(\rho_{k'-j}, \ell) \models \psi_1$ for any $0 \leq j \leq k' - k$. This ensures the result. And if $(\rho_k, \ell) \models \psi_2$, we deduce directly $(\rho_{k'}, \ell) \models \psi_2$ by the i.h. In both case, we obtain $(\rho_{k'}, \ell) \models \psi$.

Now assume that there exists some QTL formula Ψ equivalent to $\text{Even}^c(p)$. From Proposition 2, there exists a formula of the form $\Psi' = \exists P.\psi$ with $\psi \in \text{LTL}$ such that Ψ and Ψ' are equivalent over all the executions ρ_k (and more generally over any execution containing two control states). Let K be the size of ψ .

¹ The tree semantics has not been formally defined but it just consists in interpreting formulas over words in $(2^{\text{AP}})^\omega$ or equivalently over labelled executions where every control state occurs exactly once.

Consider the odd integer $\lambda = 2 \cdot K + 1$ and the labelling ℓ defined by $\ell(q) = \{p\}$ and $\ell(q') = \emptyset$. We know that $(\rho_\lambda, \ell) \models \text{Even}^c(p)$ and $(\rho_{\lambda+1}, \ell) \not\models \text{Even}^c(p)$, and therefore $(\rho_\lambda, \ell) \models \Psi'$ and $(\rho_{\lambda+1}, \ell) \not\models \Psi'$. Then there exists an extended labelling ℓ' of ℓ such that $(\rho_\lambda, \ell') \models \psi$. And from the previous result, we can deduce $(\rho_{\lambda+1}, \ell') \models \psi$ as λ and $\lambda + 1$ are greater than $|\psi|$, and this contradicts the fact $(\rho_{\lambda+1}, \ell) \not\models \Psi'$. \blacktriangleleft

Note that $\text{Even}^c(p)$ is expressible when every control state always occurs at positions of the same parity (and this explains why this property is easily expressed with the tree semantics where every position corresponds to a unique control state).

Finally we can also notice that the **U** modality cannot be expressed with **F**, **X** and quantifications (contrary to what happens in tree semantics). Let $\text{L}_Q(\mathbf{F}, \mathbf{X})$ (resp. $\text{L}(\mathbf{F}, \mathbf{X})$) be the fragment of QTLTL (resp. LTL) where the Until modality is replaced by its weak form **F**. We have:

► **Proposition 4.** *With the structure semantics, there is no $\text{L}_Q(\mathbf{F}, \mathbf{X})$ formula equivalent to the formula $a\mathbf{U}b$.*

Proof. Consider the set $Q = \{q_0, q_1, q_2\}$ and an integer $k > 0$. Let ρ_k be the finite sequence $(q_0)^k \cdot q_1$ and ρ'_k be the finite sequence $(q_0)^k \cdot q_2$. Consider the labelling function ℓ over Q defined as follows: $\ell(q_0) = \{a\}$, $\ell(q_1) = \{b\}$ and $\ell(q_2) = \emptyset$. Let π_k (resp. π'_k) be the infinite execution $(\rho_k \cdot \rho'_k)^\omega$ (resp. $(\rho'_k \cdot \rho_k)^\omega$). We clearly have $(\pi_k, \ell) \models a\mathbf{U}b$ and $(\pi'_k, \ell) \not\models a\mathbf{U}b$. Now assume that there exists some $\text{L}_Q(\mathbf{F}, \mathbf{X})$ formula Ψ equivalent to $a\mathbf{U}b$. Over the executions π_k and π'_k , such a formula would be equivalent to some formula of the form $\exists P.\psi$ with ψ an $\text{L}(\mathbf{F}, \mathbf{X})$ formula (indeed Proposition 2 does not introduce any **U** modality). And then we would have $(\pi_k, \ell) \models \exists P.\psi$ and $(\pi'_k, \ell) \not\models \exists P.\psi$. Now let ℓ' be the extended labelling ℓ' with 8 atomic propositions p_0, \dots, p_7 that label each subset of Q such that $(\pi_k, \ell') \models \psi$. Note that the $\text{L}(\mathbf{F}, \mathbf{X})$ formula ψ does not depend on k : its construction is only based on the number of control states in the structures.

It remains to show that if k is large enough (i.e. larger than the number of nested **X** on top of the formula), then we have $(\pi'_k, \ell') \models \psi$, which contradicts the hypothesis. Let $h_{\mathbf{X}}(\varphi)$ be the height of Next modalities on the top of φ . We can prove the following result: for any $\psi \in \text{L}(\mathbf{F}, \mathbf{X})$ such that $h_{\mathbf{X}}(\psi) < k - i$, we have $((\pi_k)_{\geq i}, \ell') \models \psi \Leftrightarrow ((\pi'_k)_{\geq i}, \ell') \models \psi$. The proof is done by induction over $h_{\mathbf{X}}(\psi)$:

- $h_{\mathbf{X}}(\psi) = 0$: If ψ is an atomic proposition, it is true because $\ell'((\pi_k)(i)) = \ell'((\pi'_k)(i))$ if $i < k$. If $\psi = \mathbf{F}\psi_1$ and $((\pi_k)_{\geq i}, \ell') \models \psi$, then there exists $i' \geq i$ such that $((\pi_k)_{\geq i'}, \ell') \models \psi_1$ and given the definition of executions π_k and π'_k , there exists $i'' \geq i$ such that $((\pi_k)_{\geq i'}, \ell') = ((\pi'_k)_{\geq i''}, \ell')$. The same holds when $((\pi'_k)_{\geq i}, \ell') \models \psi$.
- $h_{\mathbf{X}}(\psi) > 0$: in that case, ψ is a Boolean combination of atomic propositions, **F**-formulas or **X**-formulas. The Boolean part is easily handled by a induction over the size of the formula, and the last case is when $\psi = \mathbf{X}\psi_1$. Assume $((\pi_k)_{\geq i}, \ell') \models \mathbf{X}\psi_1$, then $((\pi_k)_{\geq i+1}, \ell') \models \psi_1$, then by i.h. we get $((\pi'_k)_{\geq i+1}, \ell') \models \psi_1$ since $h_{\mathbf{X}}(\psi_1) < k - i - 1$, and then $((\pi'_k)_{\geq i}, \ell') \models \mathbf{X}\psi_1$. The other direction is done in a similar way.

In conclusion, the formula $\exists P.\psi$ obtained from the hypothetical Ψ is fixed, as is $h_{\mathbf{X}}(\psi)$. For any $k > h_{\mathbf{X}}(\psi)$, we know that $((\pi_k), \ell') \models \psi$ if and only if $((\pi'_k), \ell') \models \psi$ and this contradicts the fact that Ψ is equivalent to $a\mathbf{U}b$. \blacktriangleleft

4 Model checking

4.1 The model-checking problem

The model-checking problem consists in verifying that a given formula is satisfied by a given model. In our framework, we can consider several variants of this problem. First, we distinguish between the existential and the universal model-checking problem:

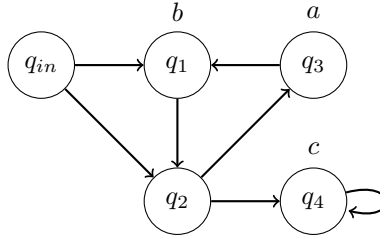
- $\text{MC}_{\exists}(\text{QLTL})$: Given a KS $\mathcal{K} = \langle Q, R, q_{in}, \ell \rangle$ and a formula $\varphi \in \text{QLTL}$, does there exist a labelled execution $(\rho, \ell) \in \text{Exec}_{\mathcal{K}}^{\text{lab}}(q_{in})$ satisfying φ , i.e. $\mathcal{K} \models_{\exists} \varphi$?
- $\text{MC}_{\forall}(\text{QLTL})$: Given a KS $\mathcal{K} = \langle Q, R, q_{in}, \ell \rangle$ and a formula $\varphi \in \text{QLTL}$, do all labelled executions $(\rho, \ell) \in \text{Exec}_{\mathcal{K}}^{\text{lab}}(q_{in})$ satisfy φ , i.e. $\mathcal{K} \models_{\forall} \varphi$?

Note that in the statement of these two problems, we assume that the initial labelling associated with the executions is the labelling ℓ of the Kripke structure \mathcal{K} . We point out the fact that these two problems are strongly related, indeed we can use for instance an algorithm for $\text{MC}_{\exists}(\text{QLTL})$ to solve $\text{MC}_{\forall}(\text{QLTL})$ since all labelled executions $(\rho, \ell) \in \text{Exec}_{\mathcal{K}}^{\text{lab}}(q_{in})$ satisfy φ iff there does not exist a labelled execution $(\rho, \ell) \in \text{Exec}_{\mathcal{K}}^{\text{lab}}(q_{in})$ satisfying $\neg\varphi$.

► **Example 5.** Let \mathcal{K} be the Kripke structure of Figure 1 rooted at q_{in} where atomic propositions are defined next to nodes. Now, we can consider the two following formulas:

- $\Psi_0 = \Phi_{\text{deter}} \Rightarrow (\mathbf{F} a \Rightarrow \mathbf{G} \neg c)$, and
- $\Psi_1 = (\mathbf{F} \mathbf{G} c) \Rightarrow \Phi_{\text{deter}}$.

where Φ_{deter} is the formula we defined previously. We can observe that Ψ_0 holds true for every execution starting from q_{in} (if the execution is deterministic, q_2 will be followed by q_4 or always by q_3), but this is not true for Ψ_1 (an execution ending with a loop in q_4 may contain both the edges $q_2 \rightarrow q_3$ and $q_2 \rightarrow q_4$). Therefore we clearly have $\mathcal{K} \models_{\forall} \Psi_0$ and $\mathcal{K} \not\models_{\forall} \Psi_1$.



■ **Figure 1** Example of model-checking problem.

4.2 Upper bound for the QLTL model-checking problems

This section is devoted to show the EXPSPACE-membership of $\text{MC}_{\exists}(\text{QLTL})$ and $\text{MC}_{\forall}(\text{QLTL})$. For this matter, we rely on alternating Büchi automata. Note that for what concerns the LTL (existential) model-checking, one technique consists in translating an LTL formula into an alternating Büchi automaton, which recognises all the models of the LTL formula, and in performing a cross-product with the Kripke structure to check whether an execution of the structure is accepted by the automaton (see, e.g. [23]). Such a method can as well be used to decide the satisfiability of LTL formulas by checking the emptiness of the associated automata. As the satisfiability problem is undecidable for QLTL [8], such a translation to a class of automata with a decidable emptiness problem will not exist. However we shall see that dealing with the model-checking problem allows us to rely on the Kripke structure to build an alternating Büchi automaton which will recognise the executions satisfying the QLTL formula. We now pursue with the formal explanation.

► **Definition 6.** An Alternating Büchi Automaton (ABA) on infinite words is a tuple $\mathcal{A} = (S, \Sigma, s_{in}, \delta, F)$ where S is a finite set of states, Σ is a finite alphabet, s_{in} is the initial state, $\delta : S \times \Sigma \rightarrow \mathcal{B}^+(S)$ is the transition function assigning a positive Boolean formula over S including false (\perp) and true (\top), to every pair (s, σ) , and $F \subseteq S$ is the accepting set of states.

In order to define how ABAs recognise infinite words over the alphabet Σ , we first need to introduce labelled trees. Given a set Γ , a Γ -labelled tree is a pair (T, σ) where:

- $T \subseteq \mathbb{N}^*$ is a tree, that is, a set of finite words in \mathbb{N}^* , called the nodes of T , such that for any $t \in T$ and $n \in \mathbb{N}$, if $t \cdot n \in T$, then $t \in T$ and $t \cdot k \in T$ for any $0 \leq k < n$. The empty word ε represents the root of T . A node $t \in T$ is said to be at depth $i \geq 0$ if its length is equal to i (the root node is hence at depth 0).
- $\sigma : T \mapsto \Gamma$ assigns an element in Γ to every node of T .

Given an automaton \mathcal{A} , a run of \mathcal{A} over an infinite word $w = a_0 a_1 \dots \in \Sigma^\omega$ is a S -labelled tree $\mathcal{T} = (T, \sigma)$ such that: $\sigma(\varepsilon) = s_{in}$ and every node t at depth i (written $|t| = i$) has k ($k \geq 0$) children t_1, \dots, t_k such that the formula $\delta(\sigma(t), a_i)$ is interpreted to true when one assigns \top to every state in $\{\sigma(t_1), \dots, \sigma(t_k)\}$ and \perp to other states. Such a run is accepting when every infinite branch of \mathcal{T} contains infinitely often nodes labelled by states in F and every finite branch ends in a node t such that $\delta(\sigma(t), a_{|t|}) = \top$. We use $\mathcal{L}(\mathcal{A})$ to denote the set of words accepted by \mathcal{A} . Deciding whether $\mathcal{L}(\mathcal{A}) = \emptyset$ is a PSPACE-complete problem [2].

We will now show how to build an ABA accepting the executions of a Kripke structure satisfying a QLTL formula. Let $\mathcal{K} = \langle Q, R, q_{in}, \ell \rangle$ be a Kripke structure and φ a QLTL formula in negated normal form. We define the ABA $\mathcal{A}_{\varphi, \mathcal{K}} = (S, Q, s_{in}, \delta, F)$ over the alphabet Q with:

- $S = \{s_{in}\} \cup Q \cup \{(\psi, \lambda) \mid \psi \in \text{SubF}(\varphi) \text{ and } \lambda : Q \mapsto 2^{\text{Prop}(\varphi)}\}$;
- F contains the elements (ψ, λ) such that ψ is not of the form $\psi_1 \mathbf{U} \psi_2$;
- and the transition function δ is defined in Figure 2.

Note that the size of S (i.e. $|S|$) is in $O(|Q| + |\varphi| \cdot 2^{|\text{Prop}(\varphi)| \cdot |Q|})$. The intuition behind this construction is that we distinguish two parts in the automaton. First, the states in Q are used to ensure that the accepting word in Q^ω corresponds to some execution in \mathcal{K} (see the definition of $\delta(q, q)$, in equations 2 to 5). The other part (the states (ψ, λ) from 6 to 14) ensures that the subformula ψ holds for true along the forthcoming execution labelled with λ . The key to the correct translation to solve the model-checking problem is that, in this ABA, the same word over Q is recognised along every branch of the execution tree. The equation 1 ensures that the two parts of the automaton are visited to check whether the word corresponds to a correct execution from the structure and that it satisfies the formula.

To state the correctness of the construction, we will use $\mathcal{A}_{\varphi, \mathcal{K}}[s]$ to denote the automaton where s is used as the initial state. We have then:

► **Lemma 7.** Let ψ be a subformula of φ , q be a control state, ρ be an infinite sequence in Q^ω and $\lambda : Q \mapsto 2^{\text{Prop}(\varphi)}$ be a labelling function (defined on the free variables in ψ). The two following properties hold:

1. $\rho \in \mathcal{L}(\mathcal{A}_{\varphi, \mathcal{K}}[q]) \Leftrightarrow \rho \in \text{Exec}_{\mathcal{K}}(q)$,
2. $\rho \in \mathcal{L}(\mathcal{A}_{\varphi, \mathcal{K}}[(\psi, \lambda)]) \Leftrightarrow (\rho, \lambda) \models \psi$.

Proof. The first result comes directly from the definition of $\delta(q, -)$ when $q \in Q$.

The second point is proved by induction over ψ . We only focus on the main cases and note that we omit argument when the reverse direction is similar to the described one:

- $\psi = p$: If $\rho \in \mathcal{L}(\mathcal{A}_{\varphi, \mathcal{K}}[(p, \lambda)])$, then by def. $\delta((p, \lambda), \rho(0)) = \top$ and we have $p \in \lambda(\rho(0))$ and then $(\rho, \lambda) \models p$. The reverse direction is similar.

$$\delta(s_{in}, q) = \delta((\varphi, \ell), q) \wedge \delta(q_{in}, q) \quad (1)$$

$$\delta(q, q) = \bigvee_{(q, q') \in R} q' \quad (2)$$

$$\delta(q, q') = \perp \quad \text{if } q \neq q' \quad (3)$$

$$\delta((\top, \lambda), q) = \top \quad (4)$$

$$\delta((\perp, \lambda), q) = \perp \quad (5)$$

$$\delta((p, \lambda), q) = \begin{cases} \top & \text{if } p \in \lambda(q) \\ \perp & \text{otherwise} \end{cases} \quad (6)$$

$$\delta((\neg p, \lambda), q) = \begin{cases} \perp & \text{if } p \in \lambda(q) \\ \top & \text{otherwise} \end{cases} \quad (7)$$

$$\delta((\psi_1 \vee \psi_2, \lambda), q) = \delta((\psi_1, \lambda), q) \vee \delta((\psi_2, \lambda), q) \quad (8)$$

$$\delta((\psi_1 \wedge \psi_2, \lambda), q) = \delta((\psi_1, \lambda), q) \wedge \delta((\psi_2, \lambda), q) \quad (9)$$

$$\delta((\mathbf{X}\psi, \lambda), q) = (\psi, \lambda) \quad (10)$$

$$\delta((\psi_1 \mathbf{U}\psi_2, \lambda), q) = \delta((\psi_2, \lambda), q) \vee (\delta((\psi_1, \lambda), q) \wedge (\psi_1 \mathbf{U}\psi_2, \lambda)) \quad (11)$$

$$\delta((\psi_1 \mathbf{R}\psi_2, \lambda), q) = \delta((\psi_2, \lambda), q) \wedge (\delta((\psi_1, \lambda), q) \vee (\psi_1 \mathbf{R}\psi_2, \lambda)) \quad (12)$$

$$\delta((\exists p.\psi, \lambda), q) = \bigvee_{P \subseteq Q} \delta((\psi, \lambda[p \rightsquigarrow P]), q) \quad (13)$$

$$\delta((\forall p.\psi, \lambda), q) = \bigwedge_{P \subseteq Q} \delta((\psi, \lambda[p \rightsquigarrow P]), q) \quad (14)$$

where $\lambda[p \rightsquigarrow P]$ denotes the labelling function λ' defined by: $\lambda'(q) = \lambda(q) \cup \{p\}$ if $q \in P$ and $\lambda'(q) = \lambda(q)$ otherwise.

■ **Figure 2** Definition of δ .

- $\psi = \psi_1 \wedge \psi_2$: If $\rho \in \mathcal{L}(\mathcal{A}_{\varphi, \mathcal{K}}[(\psi_1 \wedge \psi_2, \lambda)])$, then by def. of δ we have $\rho \in \mathcal{L}(\mathcal{A}_{\varphi, \mathcal{K}}[(\psi_1, \lambda)])$ and $\rho \in \mathcal{L}(\mathcal{A}_{\varphi, \mathcal{K}}[(\psi_2, \lambda)])$. By ind. hyp. we can deduce $(\rho, \lambda) \models \psi_1$ and $(\rho, \lambda) \models \psi_2$, and therefore $(\rho, \lambda) \models \psi$. The reverse direction is similar.
- $\psi = \mathbf{X}\psi_1$: If $\rho \in \mathcal{L}(\mathcal{A}_{\varphi, \mathcal{K}}[(\mathbf{X}\psi_1, \lambda)])$, then by def. of δ we have $\rho_{\geq 1} \in \mathcal{L}(\mathcal{A}_{\varphi, \mathcal{K}}[(\psi_1, \lambda)])$ and by i.h. we have $(\rho_{\geq 1}, \lambda) \models \psi_1$, from which we get $(\rho, \lambda) \models \mathbf{X}\psi_1$ by the semantics of \mathbf{X} .
- $\psi = \psi_1 \mathbf{U}\psi_2$: If $\rho \in \mathcal{L}(\mathcal{A}_{\varphi, \mathcal{K}}[(\psi, \lambda)])$, then by def. of δ we have either (1) $\delta((\psi_2, \lambda), \rho(0))$ holds true for the run (that is $\rho \in \mathcal{L}(\mathcal{A}_{\varphi, \mathcal{K}}[(\psi_2, \lambda)])$) or equivalently $(\rho, \lambda) \models \psi_2$, or $\delta((\psi_1, \lambda), \rho(0))$ holds true and one successor node (in the execution tree) is recognised by the state $(\psi_1 \mathbf{U}\psi_2, \lambda)$, and so on. As any infinite branch cannot be labelled infinitely often by some $(\psi_1 \mathbf{U}\psi_2, -)$ -state, this stops at some level k with the satisfaction of $\delta((\psi_2, \lambda), \rho(0))$. From this we can deduce by i.h. that $(\rho_{\geq k}, \lambda) \models \psi_2$ and $(\rho_{\geq i}, \lambda) \models \psi_1$ for $0 \leq i < k$. This is precisely the definition of $\psi_1 \mathbf{U}\psi_2$.
Now assume $(\rho, \lambda) \models \psi_1 \mathbf{U}\psi_2$. By definition, there exists $k \geq 0$ s.t. $(\rho_{\geq k}, \lambda) \models \psi_2$ and $(\rho_{\geq i}, \lambda) \models \psi_1$ for $0 \leq i < k$. By i.h. we get $\rho_{\geq k} \in \mathcal{L}(\mathcal{A}_{\varphi, \mathcal{K}}[(\psi_2, \lambda)])$ and $\rho_{\geq i} \in \mathcal{L}(\mathcal{A}_{\varphi, \mathcal{K}}[(\psi_1, \lambda)])$ for $0 \leq i < k$. We can deduce that $\rho \in \mathcal{L}(\mathcal{A}_{\varphi, \mathcal{K}}[(\psi_1 \mathbf{U}\psi_2, \lambda)])$.
- $\psi = \exists p.\psi_1$. If $\rho \in \mathcal{L}(\mathcal{A}_{\varphi, \mathcal{K}}[(\psi, \lambda)])$, then by def. of δ , there exists some $P \subseteq Q$ s.t. $\delta((\psi_1, \lambda[p \rightsquigarrow P]), q)$ holds true for the run. By i.h. we get $(\rho, \lambda[p \rightsquigarrow P]) \models \psi_1$, which implies $(\rho, \lambda) \models \exists p.\psi_1$. ◀

Note that if we apply the previous lemma with the formula φ and the labelling ℓ of \mathcal{K} and since $\delta(s_{in}, q) = \delta((\varphi, \ell), q) \wedge \delta(q_{in}, q)$ for all $q \in Q$, we deduce that $\mathcal{L}(\mathcal{A}_{\varphi, \mathcal{K}}) \neq \emptyset$ if and only if there exists a labelled execution $(\rho, \ell) \in \text{Exec}_{\mathcal{K}}^{\text{lab}}(q_{in})$ satisfying φ . Furthermore since the number of states of $\mathcal{A}_{\varphi, \mathcal{K}}$ is in $O(|Q| + |\varphi| \cdot 2^{|\text{Prop}(\varphi)| \cdot |Q|})$ and the emptiness problem for ABA is PSPACE-complete [2], we deduce the following result.

► **Proposition 8.** $\text{MC}_{\exists}(\text{QLTL})$ and $\text{MC}_{\forall}(\text{QLTL})$ are in EXPSPACE .

4.3 Lower bound for the QLTL model-checking problems

To prove the complexity lower bound (EXPSPACE -hardness), we use a domino tiling problem that we shall now define. Let C be a finite set of colours. A tile's type is then a tuple $(c_{\text{down}}, c_{\text{left}}, c_{\text{up}}, c_{\text{right}})$ in C^4 . Let T be a finite set of *tile's type*. Given two integers a and b , a T -tiling function for the $a \times b$ -grid (with a rows and b columns) is a function $f : [0, a - 1] \times [0, b - 1] \rightarrow T$ such that for all $0 \leq i < a$ and $0 \leq j < b$, we have: (1) $f(i, j)_{\text{up}} = f(i + 1, j)_{\text{down}}$ if $i < a - 1$, and (2) $f(i, j)_{\text{right}} = f(i, j + 1)_{\text{left}}$ if $j < b - 1$. As a matter of fact, the tiling function ensures that adjacent tiles share the same colour.

When such a function exists, we say that the grid can be tiled. We now define the following tiling decision problem \mathcal{T} :

Input: a set of colours C , a set of tile's types T , an integer m (encoded in unary) and $t_{\text{init}}, t_{\text{final}} \in T$

Output: yes iff there exists an integer n and a T -tiling function f for the $n \times 2^m$ -grid such that $f(0, 0) = t_{\text{init}}$ and $f(n - 1, 2^m - 1) = t_{\text{final}}$.

This problem is EXPSPACE -complete (see e.g. [20]) and has already been used to prove complexity lower bound for variant of LTL as in [15]. We adapt here the reduction proposed in this latter work to our context.

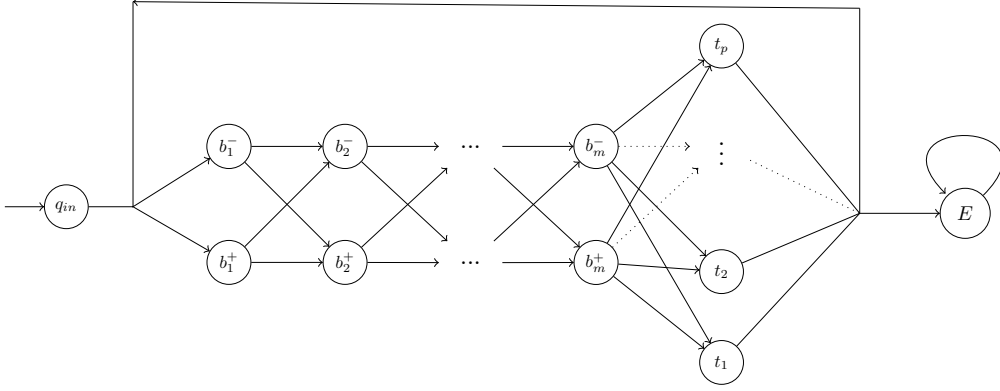
► **Proposition 9.** $\text{MC}_{\exists}(\text{QLTL})$ and $\text{MC}_{\forall}(\text{QLTL})$ are EXPSPACE -hard.

Proof. Let $\mathcal{T} = (C, T, t_{\text{init}}, t_{\text{final}}, m)$ be an instance of the tiling problem with $T = \{t_1, \dots, t_p\}$. We build a Kripke structure $\mathcal{K}_{\mathcal{T}}$ and a QLTL formula $\Phi_{\mathcal{T}}$ such that there exist a labelled execution $(\rho, \ell) \in \text{Exec}_{\mathcal{K}_{\mathcal{T}}}^{\text{lab}}(q_{in})$ satisfying φ if and only if there is some n such that the $n \times 2^m$ -grid can be tiled. The path witnessing the existence of the tiling function for the grid $n \times 2^m$ is of the form $q_{in} \cdot \left((b_1 \cdots b_m) \cdot t \right)^n \cdot E^\omega$ (with $b_j \in \{b_j^-, b_j^+\}$ for all $j \in \{1, \dots, m\}$ and $t \in \{t_1, \dots, t_p\}$). This word represents the sequence of the lines of the grid. The i -th line is listed from the cell $(i, 0)$ to cell $(i, 2^m - 1)$: each cell description starts with a sequence of m bits which encodes the cell's column number, followed by the type of tile associated with it. Such a path clearly belongs to the generic Kripke structure depicted in Figure 3 where we suppose that control states and their associated labels are the same.

We now build a QLTL formula $\Phi_{\mathcal{T}}$ to specify what is a “tiling function” path in $\mathcal{K}_{\mathcal{T}}$. First we use b_j^\pm as a shorthand for $b_j^+ \vee b_j^-$. The formula $\Phi_{\mathcal{T}}$ is a conjunction of several properties:

- The tile t_{init} (resp. t_{final}) occurs at the first (resp. last) cell:

$$\Phi_1 = \mathbf{X} \left[\left(\bigwedge_{j=1}^m \mathbf{X}^{j-1} b_j^- \right) \wedge \mathbf{X}^m t_{\text{init}} \right] \wedge \mathbf{F} \left[\left(\bigwedge_{j=1}^m \mathbf{X}^{j-1} b_j^+ \right) \wedge \mathbf{X}^m (t_{\text{final}} \wedge \mathbf{X}E) \right]$$



■ **Figure 3** Kripke structure $\mathcal{K}_{\mathcal{T}}$ for the tiling problem.

- The ordering of the cells is correct (w.r.t. the counter encoded by the b_j s):

$$\begin{aligned} \Phi_2 = & \mathbf{G} \left[\left(b_1^\pm \wedge \neg \mathbf{X}^{m+1} E \right) \Rightarrow \left(b_1^+ \Leftrightarrow \mathbf{X}^{m+1} b_1^- \right) \right] \\ & \wedge \mathbf{G} \bigwedge_{j=1}^{m-1} \left[\left(b_j^+ \wedge \mathbf{X}^{m+1} b_j^- \right) \Leftrightarrow \mathbf{X} \left(b_{j+1}^+ \Leftrightarrow \mathbf{X}^{m+1} b_{j+1}^- \right) \right] \end{aligned}$$

The first part of Φ_2 ensures that the b_1^- and b_1^+ alternate when considering two successive cells (except at the end of the execution before looping forever at E). The second part ensures that the sign of b_{j+1} changes between two successive cells if and only if the bit b_j goes from \top (1) to \perp (0).

- The successive cells along a *row* have to agree on the colour of the shared side (that is $f(i, j)_{right} = f(i, j+1)_{left}$ for $0 \leq j < 2^m - 1$):

$$\Phi_3 = \bigwedge_{t \in T} \mathbf{G} \left[t \Rightarrow \left(\bigvee_{\substack{t' \in T \text{ s.t.} \\ t_{right} = t'_{left}}} (\mathbf{X}^{m+1} t') \vee \mathbf{X} E \vee \left(\bigwedge_{j=1}^m \mathbf{X}^j b_j^- \right) \right) \right]$$

- The successive cells along a *column* have to agree on the colour of the shared side (that is $f(i, j)_{up} = f(i+1, j)_{down}$ for $0 \leq i < n-1$). This is the difficult part because the two cells are separated by an exponential number of states. To specify this property, we need the quantification of atomic propositions: we can store the number of the cell (*i.e.* the values of the b_j s) and access the next cell with this number. For this, we define two shorthand, first we have:

$$\text{StoreNb}(p) = \left(\bigwedge_{j=1}^m \mathbf{X}^{j-1} p \right) \wedge \mathbf{G} \left(p \Rightarrow \bigvee_{j=1}^m b_j^\pm \right) \wedge \bigwedge_{j=1}^m \mathbf{F} \left((b_j^+ \wedge \neg p) \vee (b_j^- \wedge \neg p) \right)$$

StoreNb labels m consecutive states by p and ensures that only b_j s states can be labelled and that either b_j^+ or b_j^- is not labelled by p : therefore the p -labelling describes exactly the number of the column of the current cell (assuming that the formula is evaluated at the beginning of the cell, that is over b_1^+ or b_1^-). And now we can use $\text{Nb}(p) = \left(\bigwedge_{j=1}^m \mathbf{X}^j p \right)$ to locate the cells of the row “ p ”.

We can now state the property ensuring that two successive cells on the same column share the same colour on the common side:

$$\Phi_4 = \bigwedge_{t \in T} \mathbf{G} \left[\left(b_1^\pm \wedge \mathbf{X}^m t \right) \Rightarrow \left(\mathbf{XG} \left(\neg \bigwedge_{j=1}^{m-1} \mathbf{X}^j b_j^- \right) \vee \right. \right. \\ \left. \left. \exists p. [\text{StoreNb}(p) \wedge \mathbf{X} \left((\neg \text{Nb}(p)) \mathbf{U} (\text{Nb}(p) \wedge \bigvee_{\substack{t' \in T \text{ s.t.} \\ t_{up} = t'_{down}}} \mathbf{X}^{m+1} t') \right)] \right) \right]$$

The subformula $\mathbf{XG}(\neg \bigwedge_{j=1}^{m-1} \mathbf{X}^j b_j^-)$ is used to exclude the last column where there is no correspondence to ensure. The \mathbf{U} operator allows us to select exactly the next cell of the column p .

Finally we have: $\Phi_{\mathcal{T}} = \Phi_1 \wedge \Phi_2 \wedge \Phi_3 \wedge \Phi_4$. And we can easily conclude that \mathcal{T} is a positive instance of the tiling problem iff there exists an execution $(\rho, \ell) \in \text{Exec}^{\text{lab}}_{\mathcal{K}_{\mathcal{T}}}(q_{in})$ such that $(\rho, \ell) \models \Phi_{\mathcal{T}}$. \blacktriangleleft

From Propositions 8 and 9, we can deduce the main theorem about QLTL model-checking.

► **Theorem 10.** $\text{MC}_{\exists}(\text{QLTL})$ and $\text{MC}_{\forall}(\text{QLTL})$ are *EXSPACE-complete*.

4.4 Relation with QCTL*

As CTL* which extends both CTL and LTL, we can consider the logic QCTL*. Two variants of QCTL* have been defined in literature: in the first one, called FCTL* (see [7]), the formula $\exists p.\varphi$ is defined as a *path* formula (as \mathbf{U} or \mathbf{X}) and in the second variant, called QCTL* (see [14]), it is defined as a *state* formula. For the tree semantics, both logics are equally expressive, but with the structure semantics, there is a big difference in terms of expressivity (FCTL* may express the existence of an eulerian path which is not possible with QCTL*) and while the model-checking problem is PSPACE-complete for QCTL*, for FCTL* the model-checking algorithm is based on a reduction to the tree semantics and its complexity is in k -EXPTIME where k is the number of alternations of quantifications. Clearly the logic QLTL we consider here can be seen as a fragment of FCTL* but it is not included in QCTL*: the expressiveness of the two logics are different and none of the two is strictly more expressive than the other. However, it is worth noting that selecting an execution first and then looking for a labelling, as for QLTL, induces a complexity blow-up.

However we can still use the result about QCTL* model-checking to define a last verification problem for Prenex formulas $\mathcal{Q}.\psi$ with $\psi \in \text{LTL}$, where we look for labellings of the *full* structure, and *then* select paths in the structure. Such an approach corresponds to a model-checking instance for QCTL*. Formally we define these verification problems (denoted $\text{MC}_{\forall}(\text{q-LTL})$ or $\text{MC}_{\exists}(\text{q-LTL})$) as follows: given a structure $\mathcal{K} = \langle Q, q_0, R, \ell \rangle$ and a formula $\mathcal{Q}.\psi$ with $\psi \in \text{LTL}$ and \mathcal{Q} a block of quantifications, decide whether $\langle Q, q_0, R, \ell \rangle \models_{\forall} \mathcal{Q}.\psi$ with:

$$\begin{aligned} \langle Q, q_0, R, \ell \rangle \models_{\forall} \exists p. \mathcal{Q}.\psi &\Leftrightarrow \exists Q' \subseteq Q \text{ s.t. } \langle Q, q_0, R, \ell[p \mapsto Q'] \rangle \models_{\forall} \mathcal{Q}.\psi \\ \langle Q, q_0, R, \ell \rangle \models_{\forall} \forall p. \mathcal{Q}.\psi &\Leftrightarrow \forall Q' \subseteq Q, \text{ we have } \langle Q, q_0, R, \ell[p \mapsto Q'] \rangle \models_{\forall} \mathcal{Q}.\psi \end{aligned}$$

and where $\langle Q, q_0, R, \ell \rangle \models_{\forall} \psi$ with $\psi \in \text{LTL}$ is interpreted as usual. In the same way, we can define the existential variant $\text{MC}_{\exists}(\text{q-LTL})$. Clearly $\text{MC}_{\forall}(\text{q-LTL})$ and $\text{MC}_{\exists}(\text{q-LTL})$ are PSPACE-complete (PSPACE-hard due to QBF, and PSPACE-easy due to QCTL*).

This defines an interesting class of problems that are different from the ones we introduced before for QLTL and whose complexity is better. For example, if we consider a two-player turn-based game $\mathcal{G} = (Q_1, Q_2, q_0, R_{\mathcal{G}}, F_1, F_2)$ where Q_1 (resp. Q_2) are the states of Player 1 (resp. Player 2), $q_0 \in Q_1 \cup Q_2$ is the initial state and $R_{\mathcal{G}} \subseteq (Q_1 \cup Q_2) \times \{0, 1\} \times (Q_1 \cup Q_2)$ is the transition relation (NB: we assume that in every state of Player i , there are exactly two possible moves labelled by 0 and 1, and we label every transition by the number of the move it corresponds to) and F_1 (resp. F_2) is the set of winning positions of Player 1 (resp. Player 2). The existence of a memoryless strategy for Player 1 (or 2) can easily be reduced to a model-checking problem of $\text{MC}_{\forall}(\text{q-LTL})$ by considering the following Kripke structure $\mathcal{K}_{\mathcal{G}} = \langle Q, q_0, R, \ell \rangle$:

- $Q = Q_1 \cup Q_2 \cup \{(q, q', \varepsilon) \mid q \in Q_1 \text{ and } (q, \varepsilon, q') \in R_{\mathcal{G}}\}$;
- $R = \{(q, q') \mid q \in Q_2 \text{ and } (q, -, q') \in R_{\mathcal{G}}\} \cup \{(q, (q, \varepsilon, q')) \mid q \in Q_1 \text{ and } (q, \varepsilon, q') \in R_{\mathcal{G}}\} \cup \{(q, \varepsilon, q'), q'\} \mid (q, \varepsilon, q') \in R_{\mathcal{G}}\}$;
- ℓ labels the structure $\mathcal{K}_{\mathcal{G}}$ as follows: states in Q_1 (resp. Q_2) are labelled with P_1 (resp. P_2), states in F_1 (resp. F_2) are labelled with W_1 (resp. W_2). And intermediary states (q, ε, q') are labelled by C_{ε} (in order to specify which move is currently played by Player 1).

Clearly a memoryless strategy for Player 1 consists in marking every Q_1 states by the move (0 or 1) corresponding to the strategy. Here with only two allowed moves, it suffices to use a single atomic proposition c . Therefore the existence of a memoryless strategy can be expressed with the following formula:

$$\Psi_{\text{strat}} = \exists c. \left[\mathbf{G} \left((P_1 \wedge c) \Rightarrow (\mathbf{X} C_1) \wedge (P_1 \wedge \neg c) \Rightarrow (\mathbf{X} C_0) \right) \Rightarrow \mathbf{F} W_1 \right]$$

5 Model Checking Paths and Flat Structures

We present here some restrictions on the considered structures which allow to obtain better complexity bounds for the model checking of QLTL formulas. We first consider ultimately periodic paths and use the facts that the model-checking problem for the branching logic QCTL (with the structure semantics) is PSPACE-complete [14] and that morally a QLTL formula over a path can be translated into a QCTL formula (in a path there is indeed no branching). We then study the model-checking problem for QLTL restricted to Kripke structures with no nested loop and show it is as well PSPACE-complete. To obtain the upper bound, we follow the same reasoning as the one presented in [4] to show that the model-checking problem for LTL with Past is in NP. It relies on two aspects: a stuttering theorem for QLTL and the fact that we can represent finitely all the executions of a flat structure with what we call iterated path schemas.

5.1 Path Model Checking

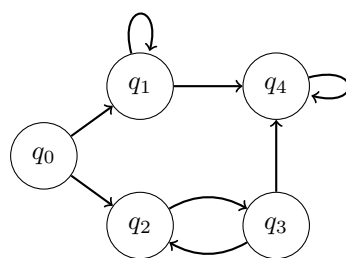
We first consider path as it is done in [17] for LTL. Given a set of states Q , a *labelled path* is an ultimately-periodic structure $(\rho_1 \cdot \rho_2^{\omega}, \ell)$ with $\rho_1 \in Q^*$, $\rho_2 \in Q^+$ and $\ell : Q \rightarrow 2^{\text{AP}}$ is a labelling function. The size of such a structure is given by the sum of the length of the sequences ρ_1 and ρ_2 . We use $\text{MC}_p(\text{QLTL})$ to denote this model-checking problem which takes as input a labelled path $(\rho_1 \cdot \rho_2^{\omega}, \ell)$ and a formula $\varphi \in \text{QLTL}$ and which asks whether $(\rho_1 \cdot \rho_2^{\omega}, \ell) \models \varphi$. Note that in a path the same control state might appear more than one time in ρ_1 and ρ_2 . We have then the following result:

► **Theorem 11.** $\text{MC}_p(\text{QLTL})$ is PSPACE-complete.

Proof. PSPACE-hardness comes from the QBF problem: a QBF instance Φ belongs to QLTL, and its validity can be directly reduced to some path model-checking problem $(q^\omega, \ell_\theta) \models \Phi$. The PSPACE-membership comes from the PSPACE-membership of the model-checking problem for QCTL in structure semantics [14]: when considering a single path, the QLTL formula φ has the same truth value as the QCTL formula $\hat{\varphi}$ with every temporal modality **X**, **U** or **R** associated with some existential or universal path quantifier. This provides the result. ◀

5.2 Flat Kripke Structures and Path Schemas

A Kripke structure is said to be *flat* (sometimes the term *weak* is used, see e.g. [12]) if every node in the underlying graph belongs to at most one simple cycle (a simple cycle is a cycle where each edge appears at most once) [3]. Figure 4 presents an example of a flat Kripke structure. Note that if we add an edge for instance from state q_4 to q_2 then the structure will not be flat anymore as the control states q_2 , q_3 and q_4 will belong to two simple cycles. It is worth noticing that path model-checking is not subsumed by flat structure model-checking (and of course, neither does the opposite) because in a path there is no restriction on the occurrences of a state, whereas in a flat structure each occurrence of a state (except the last one) is necessarily followed by the same state.



■ **Figure 4** A flat Kripke structure.

We then denote by $\text{MC}_{f,\exists}(\text{QLTL})$ (resp. $\text{MC}_{f,\forall}(\text{QLTL})$) the existential (resp. universal) model-checking problem $\text{MC}_{\exists}(\text{QLTL})$ (resp. $\text{MC}_{\forall}(\text{QLTL})$) restricted to flat Kripke structures. As for the general case, our method to solve $\text{MC}_{f,\exists}(\text{QLTL})$ can be used (with the same complexity bound) to solve $\text{MC}_{f,\forall}(\text{QLTL})$ by taking the negation of the formula.

Flat Kripke structures are easier to analyse than general structures, because we can represent their executions thanks to a finite set of path schemas (of polynomial size). Let $\mathcal{K} = \langle Q, R, q_{in}, \ell \rangle$ be a flat Kripke structures. A path p in \mathcal{K} is a finite (non-empty) sequence of control states $q_0, \dots, q_k \in Q^+$ such that $(q_i, q_{i+1}) \in R$ for all $i \in [0, k-1]$. We denote by $first(p)$ the first control state q_0 of the sequence and $last(p)$ the last one equals to q_k . A loop is then a path p such that $first(p) = last(p)$. A path schema P is an expression of the form $p_1 l_1 p_2 l_2 \dots p_k l_k$ such that :

- p_i is a path for all $i \in [1, k]$;
- l_i is a loop for all $i \in [1, k]$;
- $first(p_1) = q_{in}$ and $first(l_i) = last(p_i) = first(p_{i+1})$ for all $i \in [1, k-1]$ and $first(l_k) = last(p_k)$.

The size of a path schema $P = p_1 l_1 p_2 l_2 \dots p_k l_k$ is the sum of the lengths of each sequence composing it. We say that an execution $\rho \in Q^\omega$ respects a path schema $P = p_1 l_1 p_2 l_2 \dots p_k l_k$ iff there exists $n_1, \dots, n_{k-1} \in \mathbb{N} \setminus \{0\}$ such that $\rho = p_1 l_1^{n_1} p_2 l_2^{n_2} \dots p_{k-1} l_{k-1}^{n_{k-1}} p_k l_k^\omega$. Finally, for all $n_1, \dots, n_{k-1} \in \mathbb{N} \setminus \{0\}$, by definition of path schemas, we have that $p_1 l_1^{n_1} p_2 l_2^{n_2} \dots p_{k-1} l_{k-1}^{n_{k-1}} p_k l_k^\omega$ is an execution. From Section 3 of [4] we have the following proposition²:

► **Proposition 12** ([4]). *In a flat Kripke structure, for each execution ρ , there exists a path schema P of size smaller than $3 * |Q|$ such that ρ respects P .*

5.3 Stuttering Result for QTL

In [4], the authors have shown a general stuttering theorem for LTL with past, they have proved that if an execution of the form $\rho_1 s^M \rho_2$ (where ρ_1 and s are finite sequence of states) satisfies an LTL formula φ of temporal height at most N and if $M > 2N$, then the execution $\rho_1 s^{2N+1} \rho_2$ satisfies φ as well. In other words, to satisfy φ there is no need to repeat the infix s more than $2N + 1$ times. We shall see that we have the same result for QTL.

Let $\mathcal{K} = \langle Q, R, q_0, \ell \rangle$ be a Kripke structure (not necessarily flat) and assume that we have two executions $\rho = \rho_1 s^M \rho_2$ and $\rho' = \rho_1 s^{M'} \rho_2$ with $\rho_1, s \in Q^*$ and $\rho_2 \in Q^\omega$ and $M, M' > 2N$ for some $N \geq 2$. In Section 4 of [4], the authors present an equivalence relation (parametrised by N) between positions in ρ and ρ' which can be defined as for $i, i' \in \mathbb{N}$, we have $(\rho, i) \equiv_N (\rho', i')$ if and only if one of the following conditions holds:

1. $i, i' < |\rho_1| + N \cdot |s|$ and $i = i'$
2. $i \geq |\rho_1| + (M - N) \cdot |s|$ and $i' \geq |\rho_1| + (M' - N) \cdot |s|$ and $(i - i') = (M - M') \cdot |s|$
3. $|\rho_1| + N \cdot |s| \leq i < |\rho_1| + (M - N) \cdot |s|$ and $|\rho_1| + N \cdot |s| \leq i' < |\rho_1| + (M' - N) \cdot |s|$ and $|i - i'| = 0 \pmod{|s|}$

Intuitively, this relation states that either i and i' should be at the same position in the parts consisting of ρ_1 and the first N copies of s and in the same relative positions in the last N copies of s and in ρ_2 , otherwise i and i' should be at the same position in s . We can show following the exact same steps as for the proof of Theorem 4.1 of [4], that if φ is a QTL formula such that $\text{th}(\varphi) \leq N$ and if $(\rho, i) \equiv_N (\rho', i')$ then for all labellings λ , we have $(\rho_{\geq i}, \lambda) \models \varphi$ iff $(\rho'_{\geq i'}, \lambda) \models \varphi$. This proof is done by a double induction on the structure of φ and on N , and we should pay attention to two aspects. First, for LTL there is no need to consider labelling, however here we take them into account, but since the sequence of control states is the same (modulo the iterations of s) in ρ and ρ' , we can universally quantify on labellings (for each labelling we get two new sequences of subset of atomic propositions, and we use the fact that results for LTL hold for such sequences). Second, in order to reuse the induction reasoning, we should take care of the specific case of subformula $\exists p. \psi$. Assume hence that we have $(\rho, i) \equiv_{N'} (\rho', i')$ and that for ψ such that $\text{th}(\psi) \leq N'$, we have $(\rho_{\geq i}, \lambda') \models \psi$ iff $(\rho'_{\geq i'}, \lambda') \models \psi$ for all labelling λ' . Now for any labelling λ , we have that there exists $\lambda' \equiv_{\text{AP} \setminus \{p\}} \lambda$ s.t. $(\rho_{\geq i}, \lambda') \models \psi$ if and only if there exists $\lambda'' \equiv_{\text{AP} \setminus \{p\}} \lambda$ s.t. $(\rho'_{\geq i'}, \lambda'') \models \psi$ (simply take $\lambda' = \lambda''$). And we can hence conclude, since the temporal height of $\exists p. \psi$ is the same as ψ , that for all labellings λ , we have $(\rho_{\geq i}, \lambda) \models \exists p. \psi$ iff $(\rho'_{\geq i'}, \lambda) \models \exists p. \psi$. Finally, since $(\rho, 0) \equiv_N (\rho', 0)$ for any $N \geq 2$, we can adapt Theorem 4.1 of [4] to our case.

² In [4], the size of path schema is bounded by $2 * |R|$, since we consider here sequence of control states, we use 3 as a constant to stay on the safe side.

► **Proposition 13.** *Let $N \geq 2$ and $M, M' > 2N$ and $\rho = \rho_1 s^M \rho_2$ and $\rho' = \rho_1 s^{M'} \rho_2$ with $\rho_1, s \in Q^*$ and $\rho_2 \in Q^\omega$. For all QLTL formulas φ such that $\text{th}(\varphi) \leq N$, we have $(\rho, \ell) \models \varphi$ iff $(\rho', \ell) \models \varphi$.*

5.4 Algorithm for Flat Kripke Structures

We now present the algorithm to solve $\text{MC}_{f,\exists}(\text{QLTL})$. Let $\mathcal{K} = \langle Q, R, q_{in}, \ell \rangle$ be a flat Kripke structure and φ a QLTL formula such that $\text{th}(\varphi) \leq N$. Assume there exists a labelled execution $(\rho, \ell) \in \text{Exec}^{\text{lab}}_{\mathcal{K}}(q_0)$ satisfying φ . Using Proposition 12, there exists a path schema $P = p_1 l_1 p_2 l_2 \dots p_k l_k$ (of size smaller than $3 * |Q|$) and $n_1, \dots, n_{k-1} \in \mathbb{N} \setminus \{0\}$ such that $\rho = p_1 l_1^{n_1} p_2 l_2^{n_2} \dots p_{k-1} l_{k-1}^{n_{k-1}} p_k l_k^\omega$. Now for all $i \in \{1, \dots, k-1\}$, we define $n'_i = \min(n_i, 2N+5)$ and let ρ' be the execution $p_1 l_1^{n'_1} p_2 l_2^{n'_2} \dots p_{k-1} l_{k-1}^{n'_{k-1}} p_k l_k^\omega$, thanks to Proposition 13, we get that $(\rho', \ell) \models \varphi$. This gives us the path for a non-deterministic PSPACE-algorithm. We seek for a path schema $P = p_1 l_1 p_2 l_2 \dots p_k l_k$ and for $(k-1)$ positive naturals n'_1, \dots, n'_{k-1} smaller than $2N+5$ such that $(p_1 l_1^{n'_1} p_2 l_2^{n'_2} \dots p_{k-1} l_{k-1}^{n'_{k-1}} p_k l_k^\omega, \ell) \models \varphi$. Note that $p_1 l_1^{n'_1} p_2 l_2^{n'_2} \dots p_{k-1} l_{k-1}^{n'_{k-1}} p_k l_k^\omega$ is of polynomial size in the size of the flat Kripke structure \mathcal{K} and the formula φ and that checking whether $(p_1 l_1^{n'_1} p_2 l_2^{n'_2} \dots p_{k-1} l_{k-1}^{n'_{k-1}} p_k l_k^\omega, \ell) \models \varphi$ can be done in polynomial space thanks to Theorem 11. We use then Savitch's theorem to obtain a PSPACE-algorithm. For the lower bound, the proof is the same as for the path model-checking.

► **Theorem 14.** *$\text{MC}_{f,\exists}(\text{QLTL})$ and $\text{MC}_{f,\forall}(\text{QLTL})$ are PSPACE-complete.*

6 Conclusion

We studied the model-checking problem for QLTL for the structure semantics. (In this semantics, executions are seen as an infinite sequence of control states, together with a labelling function that associates atomic propositions to the control states.) To begin with, we proved that the model-checking problem is EXPSPACE-complete. To obtain a better understanding of this semantics, we have considered some variants of the model-checking problem, with a restriction on the form of the structures: path model-checking and model-checking of flat structures. Both problems turn out to be PSPACE-complete. Our results are summarised in Figure 5. We also showed that the problems $\text{MC}_{\forall}(\text{q-LTL})$ and $\text{MC}_{\exists}(\text{q-LTL})$ corresponding to another way of considering quantifications are PSPACE-complete.

problem:	$\text{MC}_{\exists}(\text{QLTL})$	$\text{MC}_{\forall}(\text{QLTL})$	$\text{MC}_p(\text{QLTL})$	$\text{MC}_{f,\exists}(\text{QLTL})$	$\text{MC}_{f,\forall}(\text{QLTL})$
complexity:	EXPSPACE-complete			PSPACE-complete	

■ **Figure 5** Complexity of QLTL model-checking.

The interesting properties that this semantics can express (as the finite number of reachable control states, the determinism of an execution) leads us to continue studying this semantics by working on the satisfiability problem of fragments of QLTL and on the expressivity of prenex formulas.

References

- 1 B. Bednarczyk and S. Demri. Why Does Propositional Quantification Make Modal and Temporal Logics on Trees Robustly Hard? *Logical Methods in Computer Science*, 18(3):5:1–5:46, July 2022.
- 2 A. K. Chandra, D. Kozen, and L. J. Stockmeyer. Alternation. *J. ACM*, 28(1):114–133, 1981.

- 3 H. Comon and Y. Jurski. Multiple counter automata, safety analysis and PA. In *CAV'98*, volume 1427 of *LNCS*, pages 268–279. Springer, 1998.
- 4 S. Demri, A. K. Dhar, and A. Sangnier. Taming past LTL and flat counter systems. *Inf. Comput.*, 242:306–339, 2015.
- 5 E. A. Emerson and A. P. Sistla. Deciding full branching time logic. *Information and Control*, 61(3):175–201, June 1984.
- 6 K. Etessami. Stutter-invariant languages, ω -automata, and temporal logic. In Nicolas Halbwachs and Doron A. Peled, editors, *CAV'99*, volume 1633 of *LNCS*, pages 236–248. Springer-Verlag, July 1999.
- 7 T. French. Decidability of quantified propositional branching time logics. In *AJCAI'01*, volume 2256 of *LNCS*, pages 165–176. Springer-Verlag, December 2001.
- 8 T. French. Quantified propositional temporal logic with repeating states. In *TIME-ICTL'03*, pages 155–165. IEEE Comp. Soc. Press, July 2003.
- 9 T. French and M. Reynolds. A sound and complete proof system for QPTL. In Philippe Balbiani, Nobu-Yuki Suzuki, Frank Wolter, and Michael Zakharyashev, editors, *AIML'02*, pages 127–148. King's College Publications, 2003.
- 10 A. Hossain and F. Laroussinie. QCTL model-checking with QBF solvers. *Inf. Comput.*, 280:104642, 2021.
- 11 Y. Kesten and A. Pnueli. Complete proof system for QPTL. *Journal of Logic and Computation*, 12(5):701–745, October 2002.
- 12 L. Kuhtz and B. Finkbeiner. Weak Kripke structures and LTL. In *CONCUR'11*, volume 6901 of *LNCS*, pages 419–433. Springer, 2011.
- 13 O. Kupferman. Augmenting branching temporal logics with existential quantification over atomic propositions. In *CAV'95*, volume 939 of *LNCS*, pages 325–338. Springer-Verlag, July 1995.
- 14 F. Laroussinie and N. Markey. Quantified CTL: expressiveness and complexity. *Logical Methods in Computer Science*, 10(4), 2014.
- 15 F. Laroussinie, N. Markey, and P. Schnoebelen. Temporal logic with forgettable past. In *LICS'02*, pages 383–392. IEEE Computer Society, 2002.
- 16 N. Markey and P. Schnoebelen. Model checking a path. In *CONCUR'03*, volume 2761 of *LNCS*, pages 248–262. Springer, 2003.
- 17 N. Markey and P. Schnoebelen. Model checking a path. In *CONCUR'03*, volume 2761 of *LNCS*, pages 248–262. Springer, 2003.
- 18 Markey N and P. Schnoebelen. Mu-calculus path checking. *Inf. Process. Lett.*, 97(6):225–230, 2006.
- 19 A. Pnueli. The temporal logic of programs. In *FOCS'77*, pages 46–57. IEEE Comp. Soc. Press, October–November 1977.
- 20 F. Schwarzentruber. The complexity of tiling problems. *CoRR*, abs/1907.00102, 2019.
- 21 A. P. Sistla. *Theoretical Issues in the Design and Verification of Distributed Systems*. PhD thesis, Harvard University, Cambridge, Massachusetts, USA, 1983.
- 22 A. P. Sistla, M. Y. Vardi, and P. Wolper. The complementation problem for Büchi automata with applications to temporal logics. *Theoretical Computer Science*, 49:217–237, 1987.
- 23 M. Y. Vardi. Alternating automata and program verification. In Jan van Leeuwen, editor, *Computer Science Today: Recent Trends and Developments*, volume 1000 of *LNCS*, pages 471–485. Springer, 1995.
- 24 P. Wolper. Temporal logic can be more expressive. *Inf. Control.*, 56(1/2):72–99, 1983.