

QUANTIFIED CTL: EXPRESSIVENESS AND COMPLEXITY *

FRANÇOIS LAROUSSINIE ^a AND NICOLAS MARKEY ^b

^a LIAFA – Université Paris Diderot & CNRS
e-mail address: francoisl@liafa.univ-paris-diderot.fr

^b LSV – ENS Cachan & CNRS
e-mail address: markey@lsv.ens-cachan.fr

ABSTRACT. While it was defined long ago, the extension of CTL with quantification over atomic propositions has never been studied extensively. Considering two different semantics (depending whether propositional quantification refers to the Kripke structure or to its unwinding tree), we study its expressiveness (showing in particular that QCTL coincides with Monadic Second-Order Logic for both semantics) and characterise the complexity of its model-checking and satisfiability problems, depending on the number of nested propositional quantifiers (showing that the structure semantics populates the polynomial hierarchy while the tree semantics populates the exponential hierarchy).

1. INTRODUCTION

Temporal logics. Temporal logics extend propositional logics with modalities for specifying constraints on the order of events in time. Since [Pnu77, CE82, QS82], they have received much attention from the computer-aided-verification community, since they fit particularly well for expressing and automatically verifying (*model checking*) properties of reactive systems.

Two important families of temporal logics have been considered: linear-time temporal logics (*e.g.* LTL [Pnu77]) can be used to express properties of one single execution of the system under study, while branching-time temporal logics (*e.g.* CTL [CE82, QS82] and CTL* [EH86]) consider the execution tree. Since the 90s, many extensions of these logics have been introduced, of which alternating-time temporal logics (such as ATL, ATL* [AHK97]) extend CTL towards the study of open systems (involving several agents).

2012 ACM CCS: [**Theory of computation**]: Formal languages and automata theory; Logic—Modal and temporal logics; [**Software and its engineering**]: Software organization and properties—Software functional properties—Formal methods—Software verification; Model checking.

Key words and phrases: Temporal logics; model checking; expressiveness; tree automata.

* This is a long version of paper [DLM12], which appeared in CONCUR'12.

^b This work benefited from the support of the ERC Starting Grant EQualIS and of the EU-FP7 project Cassting.

In this landscape of temporal logics, both CTL and ATL enjoy the nice property of having polynomial-time model-checking algorithms. In return for this, both logics have quite limited expressiveness. Several extensions have been defined in order to increase this limited expressive power.

Our contributions. We are interested in the present paper in the extension of CTL (and CTL^{*}) with *propositional quantification* [Sis83, ES84]. In that setting, propositional quantification can take different meaning, depending whether the extra propositions label the Kripke structure under study (*structure semantics*) or its execution tree (*tree semantics*). While these extensions of CTL with propositional quantification have been in the air for thirty years, they have not been extensively studied yet: some complexity results have been published for existential quantification [Kup95], for the two-alternation fragment [KMTV00] and for the full extension [Fre01]; but expressiveness issues, as well as a complete study of model checking and satisfiability for the whole hierarchy, have been mostly overlooked.

We answer these questions in the present paper: in terms of expressiveness, we prove that QCTL and QCTL^{*} are equally expressive, and coincide with Monadic Second-Order Logic¹. As regards satisfiability and model-checking, we characterise the complexity of these problems depending on the quantifier alternation: under the structure semantics, the model-checking problem populates the polynomial-time hierarchy (and satisfiability is undecidable); for the tree semantics, the model-checking problem populates the exponential-time hierarchy (and so does the satisfiability problem). Finally, we also characterise the model- and formula-complexities of our problems, when one of the inputs to the model-checking problem is fixed. All these results are summarized in Tables 1 and 2, which are displayed in the conclusion of this paper.

Applications to alternating-time temporal logics. Our initial motivation for this work comes from alternating-time temporal logics. Indeed ATL also has several flaws in terms of expressiveness: namely, it can only focus on (some) zero-sum properties, *i.e.*, on purely antagonist games, in which two coalitions fight with opposite objectives. In many situations, games are not purely antagonist, but involve several independent systems, each having its own objective. Recently, several extensions of ATL have been defined to express properties of such non-zero-sum games. Among those, our logic ATL_{sc} [DLM10] extends ATL with *strategy contexts*, which provides a way of expressing interactions between strategies. Other similar approaches include Strategy Logics (SL) [CHP07, MMV10], (Basic) Strategy-Interaction Logic ((B)SIL) [WHY11], or Temporal Cooperation Logic (TCL) [HSW13].

Designing decision procedures for these extensions is much more difficult than for the standard ATL fragment. Interestingly, QCTL appears to be a convenient, uniform intermediary logic in order to obtain algorithms for ATL_{sc}, SL and related formalisms. Indeed, strategies of the players can be represented² by a finite set of atomic propositions labelling

¹This claim assumes a special notion of equivalence between formulas, since MSO is evaluated *globally* on a structure while QCTL formulas are evaluated at the initial state. This will be made clear in the paper.

²Notice that the link between strategy quantification and propositional quantification already emerges in QD μ [Pin07], which extends the *decision μ -calculus* with some flavour of propositional quantification. Also, the main motivation of [KMTV00] for studying the two-alternation fragment of QCTL is a hardness result for the control and synthesis of open systems.

the execution tree of the game structure under study. Strategy quantification is then naturally expressed as propositional quantification; since the resulting labelling is *persistent*, it can encode interactions between strategies. Notice that while the *tree semantics* of QCTL encodes plain strategies, the *structure semantics* also finds a meaning in that translation, as it may correspond to *memoryless strategies* [DLM12].

Using such a translation, any instance of the model-checking problem for ATL_{sc} (or SL) can be translated into an instance of the model-checking problem for QCTL. The algorithms proposed in this paper then yield algorithms for the former problems, which can be proved to have optimal complexity. Unfortunately, the satisfiability problem cannot follow the same reduction scheme: indeed, when translating an ATL_{sc} formula into a QCTL one, we need to know the set of agents and their allowed moves. It turns out that satisfiability is undecidable for ATL_{sc} and SL (while we prove it decidable for QCTL in the tree semantics). Interestingly, when restricting satisfiability checking to *turn-based* game structures, an alternative translation into QCTL can be used to obtain decidability of the problem.

Because they involve a lot of new definitions and technical proofs, we do not develop these questions here, and refer the interested reader to [LM14] for full details.

Related works. Extending modal logics with quantification dates back to early works of Kripke [Kri59] and Fine [Fin70]. We refer to [FM98, AP06, tC06] for more details.

(Propositional) quantification was first used in temporal logics by Sistla and others, both for linear-time [Sis83, SVW87] and branching-time temporal logics [ES84], mainly with the aim of augmenting the expressiveness of the classical logics. In the linear-time setting, the model-checking problem for the k -alternation fragment was shown k -EXPSPACE-complete [Sis83, SVW87]. The stutter-invariant fragment of QLTL, with a restricted notion of propositional quantification, was developed in [Ete99]. Proof systems for QLTL were developed in the linear-time setting, both with and without past-time modalities [KP02, FR03].

As regards branching time, the extension of CTL^* with external existential quantification (hereafter called EQ^1CTL^*) was proved as expressive as parity tree automata over binary tree [ES84]. The existential logics EQ^1CTL and EQ^1CTL^* were further studied in [Kup95], both in the structure- and in the tree semantics; model checking EQ^1CTL and EQ^1CTL^* are shown NP- and PSPACE-complete respectively (for the structure semantics) and EXPTIME- and 2-EXPTIME-complete respectively (for the tree semantics). The extensions of those logics with past-time modalities were studied in [KP95]. The extensions with arbitrary quantification were studied in [Kai97, Fre01] (in slightly different settings): satisfiability of QCTL^* was proven undecidable in the structure semantics, and decidable in the tree semantics [Fre01].

Several alternative semantics were proposed for quantification: the *amorphous* semantics defined in [Fre01] allows to take a bisimilar structure before labelling it. In [RP03], quantification is expressed as taking a synchronized product with a labelling automaton. Finally, quantification over states (rather than over atomic propositions) is studied in [PBD⁺02, CDC04], where model checking is proved PSPACE-complete (both for branching-time and for linear-time).

Finally, quantified temporal logics have found applications in model checking and control: AQ^1LTL and AQ^1CTL^* have been used to reason about *vacuity detection* (checking whether a formula is satisfied “*too easily*”) [AFF⁺03, GC04, GC12]. The one-alternation

fragments (which we call EQ^2CTL and EQ^2CTL^* hereafter) have been used in [KMTV00] to prove hardness results for the control problem with CTL and CTL^* objectives. The linear-time logic EQLTL was used as the specification language for supervisory control of Petri nets in [Mar10]. To conclude, propositional quantification was considered in the setting of *timed* temporal logics in [HRS98].

2. PRELIMINARIES

2.1. Kripke structures and trees. We fix once and for all a set AP of atomic propositions.

Definition 2.1. A Kripke structure \mathcal{S} is a 3-tuple $\langle Q, R, \ell \rangle$ where Q is a countable set of states, $R \subseteq Q^2$ is a total³ relation and $\ell: Q \rightarrow 2^{\text{AP}}$ is a labelling function. The size of \mathcal{S} , denoted with $|\mathcal{S}|$, is the size of Q (which can be infinite).

Let \mathcal{S} be a Kripke structure $\langle Q, R, \ell \rangle$. In the following, we always assume that the set of states \mathcal{S} is equipped with a total linear order \preceq . We use $\text{Succ}_{\mathcal{S}}(q)$ to denote the ordered list $\langle q'_0, \dots, q'_k \rangle$ of successors of q in \mathcal{S} (i.e., such that $(q, q'_i) \in R$ for any $0 \leq i \leq k$, and such that $q'_i \preceq q'_j$ if, and only if, $i \leq j$). We write $\text{d}_{\mathcal{S}}(q)$ for the degree of $q \in Q$, i.e., the size of $\text{Succ}_{\mathcal{S}}(q)$. Finally $\text{Succ}_{\mathcal{S}}(q, i)$ denotes the i -th successor of q in \mathcal{S} for $0 \leq i < \text{d}_{\mathcal{S}}(q)$, and this notation is extended to finite words over \mathbb{N}^* as follows: $\text{Succ}_{\mathcal{S}}(q, \varepsilon) = q$ and $\text{Succ}_{\mathcal{S}}(q, w \cdot i) = \text{Succ}_{\mathcal{S}}(\text{Succ}_{\mathcal{S}}(q, w), i)$ when $q' = \text{Succ}_{\mathcal{S}}(q, w)$ is well defined and $0 \leq i < \text{d}_{\mathcal{S}}(q')$.

An execution (or path) in \mathcal{S} is an infinite sequence $\rho = (q_i)_{i \in \mathbb{N}}$ s.t. $(q_i, q_{i+1}) \in R$ for all $i \in \mathbb{N}$. We use $\text{Path}(q)$ to denote the set of executions issued from q and $\text{Path}^f(q)$ for the set of all *finite* prefixes of executions of $\text{Path}(q)$. Given $\rho \in \text{Path}(q)$ and $i \in \mathbb{N}$, we write ρ^i for the path $(q_{i+k})_{k \in \mathbb{N}}$ of $\text{Path}(q_i)$ (the i -th suffix of ρ), ρ_i for the finite prefix $(q_k)_{k \leq i}$ (the i -th prefix), and $\rho(i)$ for the i -th state q_i . Given a path $\rho = (q_i)_{i \in \mathbb{N}}$, we write $\ell(\rho)$ for the sequence $(\ell(q_i))_{i \in \mathbb{N}}$, and $\text{Inf}(\ell(\rho))$ for the set of letters in Σ that appear infinitely many times along $\ell(\rho)$.

Definition 2.2. Let Σ be a finite set. A Σ -labelled tree is a pair $\mathcal{T} = \langle T, l \rangle$, where

- $T \subseteq \mathbb{N}^*$ is a non-empty set of finite words on \mathbb{N} satisfying the following constraints: for any non-empty word $x = y \cdot c$ in T with $y \in \mathbb{N}^*$ and $c \in \mathbb{N}$, the word y is in T and every word $y \cdot c'$ with $0 \leq c' < c$ is also in T ;
- $l: T \rightarrow \Sigma$ is a labelling function.

Let $\mathcal{T} = \langle T, l \rangle$ be a Σ -labelled tree. The elements of T are the *nodes* of \mathcal{T} and the empty word ε is the root of \mathcal{T} . Such a tree can be seen as a Kripke structure, with T as set of states, and transitions from any node $x \in T$ to any node of the form $x \cdot c \in T$, for $c \in \mathbb{N}$. The size of \mathcal{T} , and the notions of *successors* of a node x (written $\text{Succ}_{\mathcal{T}}(x)$), of *degree* of a node x (written $\text{d}_{\mathcal{T}}(x)$), of *path* issued from the root (whose set is written $\text{Path}_{\mathcal{T}}$), follow from this correspondence.

A tree has *bounded branching* if the degree of all its nodes is bounded. Given a finite set of integers $\mathcal{D} \subseteq \mathbb{N}$, a (Σ, \mathcal{D}) -tree is a Σ -labelled tree $\langle T, l \rangle$ whose nodes have their degrees in \mathcal{D} (i.e., for any $x \in T$, it holds $\text{d}_{\mathcal{T}}(x) \in \mathcal{D}$). Given a node $x \in T$, we denote with \mathcal{T}_x the (sub)tree $\langle T_x, l_x \rangle$ rooted at x , defined by $T_x = \{y \in T \mid \exists z \in T \text{ s.t. } z = x \cdot y\}$.

³I.e., for all $q \in Q$, there exists $q' \in Q$ s.t. $(q, q') \in R$.

Definition 2.3. Given a finite-state Kripke structure $\mathcal{S} = \langle Q, R, \ell \rangle$ and a state $q \in Q$, the unwinding of \mathcal{S} from q is the (bounded-degree) 2^{AP} -labelled tree $\mathcal{T}_{\mathcal{S}}(q) = \langle T_{\mathcal{S},q}, \ell_{\mathcal{T}} \rangle$ defined as follows:

- (1) $T_{\mathcal{S},q}$ contains exactly all nodes $x \in \mathbb{N}^*$ such that $\text{Succ}_{\mathcal{S}}(q, x)$ is well-defined
- (2) $\ell_{\mathcal{T}}(x) = \ell(\text{Succ}_{\mathcal{S}}(q, x))$.

If $\mathcal{D} = \bigcup_{q \in Q} \{\text{d}_{\mathcal{S}}(q)\}$, then $\mathcal{T}_{\mathcal{S}}(q)$ clearly is a $\langle 2^{\text{AP}}, \mathcal{D} \rangle$ -tree. Note also that any 2^{AP} -labelled tree can be seen as an infinite-state Kripke structure.

For a function $\ell: Q \rightarrow 2^{\text{AP}}$ and $P \subseteq \text{AP}$, we write $\ell \cap P$ for the function defined as $(\ell \cap P)(q) = \ell(q) \cap P$ for all $q \in Q$.

Definition 2.4. For $P \subseteq \text{AP}$, two (possibly infinite-state) Kripke structures $\mathcal{S} = \langle Q, R, \ell \rangle$ and $\mathcal{S}' = \langle Q', R', \ell' \rangle$ are P -equivalent (denoted by $\mathcal{S} \equiv_P \mathcal{S}'$) if $Q = Q'$, $R = R'$ and $\ell \cap P = \ell' \cap P$.

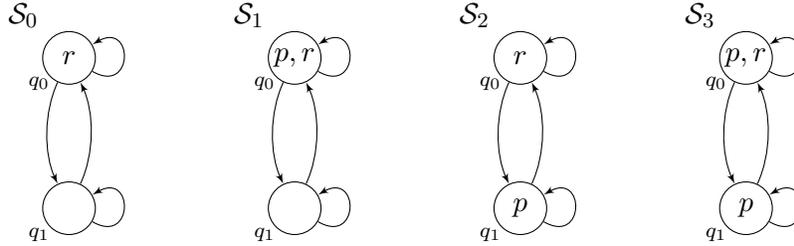


Fig. 1: Four $\{r\}$ -equivalent Kripke structures

2.2. CTL and quantified extensions.

Definition 2.5. The syntax of QCTL^* is defined by the following grammar:

$$\begin{aligned} \varphi_{\text{state}}, \psi_{\text{state}} &::= p \mid \neg \varphi_{\text{state}} \mid \varphi_{\text{state}} \vee \psi_{\text{state}} \mid \mathbf{E} \varphi_{\text{path}} \mid \exists p. \varphi_{\text{state}} \\ \varphi_{\text{path}}, \psi_{\text{path}} &::= \varphi_{\text{state}} \mid \neg \varphi_{\text{path}} \mid \varphi_{\text{path}} \vee \psi_{\text{path}} \mid \mathbf{X} \varphi_{\text{path}} \mid \varphi_{\text{path}} \mathbf{U} \psi_{\text{path}} \end{aligned}$$

where p ranges over AP . Formulas defined as φ_{state} are called state-formulas, while φ_{path} defines path-formulas. Only state formulas are QCTL^* formulas.

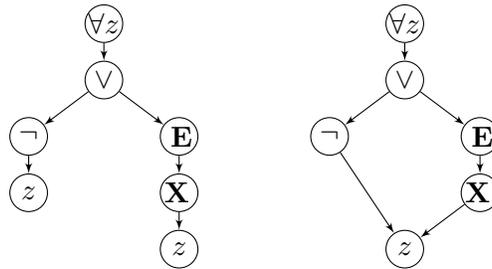


Fig. 2: Two representations of $\forall z. (\neg z \vee \mathbf{E} \mathbf{X} z)$

Naturally, any QCTL^* formula φ can be represented as a finite tree \mathcal{T}_{φ} , in which each node represents a subformula (see Fig. 2). Alternatively, formula φ can be seen as a finite

acyclic Kripke structure \mathcal{S}_φ whose unwinding is \mathcal{T}_φ . The *size* of φ is the size of \mathcal{T}_φ , and its *DAG-size* (for *directed-acyclic-graph size*) is the size of the smallest Kripke structure \mathcal{S}_φ whose unwinding is \mathcal{T}_φ . Obviously, when sharing large subformulas, the size of a formula can be significantly larger than its DAG-size.

We use standard abbreviations as: $\top = p \vee \neg p$, $\perp = \neg \top$, $\mathbf{F}\varphi = \top \mathbf{U}\varphi$, $\mathbf{G}\varphi = \neg \mathbf{F}\neg\varphi$, $\mathbf{A}\varphi = \neg \mathbf{E}\neg\varphi$, and $\forall p. \varphi = \neg \exists p. \neg\varphi$. The logic QCTL is a fragment of QCTL* where temporal modalities are under the immediate scope of path quantifiers:

Definition 2.6. *The syntax of QCTL is defined by the following grammar:*

$$\begin{aligned} \varphi_{\text{state}}, \psi_{\text{state}} ::= & p \mid \neg \varphi_{\text{state}} \mid \varphi_{\text{state}} \vee \psi_{\text{state}} \mid \exists p. \varphi_{\text{state}} \mid \\ & \mathbf{E}\varphi_{\text{state}} \mathbf{U}\psi_{\text{state}} \mid \mathbf{A}\varphi_{\text{state}} \mathbf{U}\psi_{\text{state}} \mid \mathbf{E}\mathbf{X}\varphi_{\text{state}} \mid \mathbf{A}\mathbf{X}\varphi_{\text{state}}. \end{aligned}$$

Standard definition of CTL* and CTL are obtained by removing the use of quantification over atomic proposition ($\exists p.\varphi$) in the formulas. In the following, \exists and \forall are called (*proposition*) *quantifiers*, while \mathbf{E} and \mathbf{A} are *path quantifiers*.

Given QCTL* (state) formulas φ and $(\psi_i)_i$ and atomic propositions $(p_i)_i$ appearing free in φ (*i.e.*, not appearing as quantified propositions), we write $\varphi[(p_i \rightarrow \psi_i)_i]$ (or $\varphi[(\psi_i)_i]$ when $(p_i)_i$ are understood from the context) for the formula obtained from φ by replacing each occurrence of p_i with ψ_i . Given two sublogics L_1 and L_2 of QCTL*, we write $L_1[L_2] = \{\varphi[(\psi_i)_i] \mid \varphi \in L_1, (\psi_i)_i \in L_2\}$.

2.3. Structure- and tree semantics. Formulas of the form $\exists p.\varphi$ can be interpreted in different manners (see [Kup95, Fre01, RP03]). Here we consider two semantics: the *structure semantics* and the *tree semantics*.

2.3.1. Structure semantics. Given a QCTL* state formula φ , a (possibly infinite-state) Kripke structure $\mathcal{S} = \langle Q, R, \ell \rangle$ and a state $q \in Q$, we write $\mathcal{S}, q \models_s \varphi$ to denote that φ holds at q under the structure semantics. We extend the notation to $\mathcal{S}, \rho \models_s \varphi$ when φ is a path formula and ρ is a path in \mathcal{S} . This is defined as follows:

$$\begin{aligned} \mathcal{S}, q \models_s p & \text{ iff } p \in \ell(q) \\ \mathcal{S}, q \models_s \neg \varphi_{\text{state}} & \text{ iff } \mathcal{S}, q \not\models_s \varphi_{\text{state}} \\ \mathcal{S}, q \models_s \varphi_{\text{state}} \vee \psi_{\text{state}} & \text{ iff } \mathcal{S}, q \models_s \varphi_{\text{state}} \text{ or } \mathcal{S}, q \models_s \psi_{\text{state}} \\ \mathcal{S}, q \models_s \mathbf{E}\varphi_{\text{path}} & \text{ iff } \exists \rho \in \text{Path}(q) \text{ s.t. } \mathcal{S}, \rho \models_s \varphi_{\text{path}} \\ \mathcal{S}, q \models_s \exists p.\varphi_{\text{state}} & \text{ iff } \exists \mathcal{S}' \equiv_{\text{AP} \setminus \{p\}} \mathcal{S} \text{ s.t. } \mathcal{S}', q \models_s \varphi_{\text{state}} \\ \mathcal{S}, \rho \models_s \varphi_{\text{state}} & \text{ iff } \mathcal{S}, \rho(0) \models_s \varphi_{\text{state}} \\ \mathcal{S}, \rho \models_s \neg \varphi_{\text{path}} & \text{ iff } \mathcal{S}, \rho \not\models_s \varphi_{\text{path}} \\ \mathcal{S}, \rho \models_s \varphi_{\text{path}} \vee \psi_{\text{path}} & \text{ iff } \mathcal{S}, \rho \models_s \varphi_{\text{path}} \text{ or } \mathcal{S}, \rho \models_s \psi_{\text{path}} \\ \mathcal{S}, \rho \models_s \mathbf{X}\varphi_{\text{path}} & \text{ iff } \mathcal{S}, \rho^1 \models_s \varphi_{\text{path}} \\ \mathcal{S}, \rho \models_s \varphi_{\text{path}} \mathbf{U}\psi_{\text{path}} & \text{ iff } \exists i \geq 0. \mathcal{S}, \rho^i \models_s \psi_{\text{path}} \text{ and } \forall 0 \leq j < i. \mathcal{S}, \rho^j \models_s \varphi_{\text{path}} \end{aligned}$$

Example 2.7. *As an example, consider the formula $\text{selfloop} = \forall z.(z \Rightarrow \mathbf{E}\mathbf{X}z)$. If a state q in \mathcal{S} satisfies this formula, then the particular labelling in which only q is labelled with z*

implies that q has to carry a self-loop. Conversely, any state that carries a self-loop satisfies this formula (for the structure semantics).

Let φ be a QCTL* formula, and consider now the formula

$$\text{uniq}(\varphi) = \mathbf{EF}(\varphi) \wedge \forall z. \left(\mathbf{EF}(\varphi \wedge z) \Rightarrow \mathbf{AG}(\varphi \Rightarrow z) \right).$$

In order to satisfy such a formula, at least one φ -state must be reachable. Assume now that two different such states q and q' are reachable: then for the particular labelling where only q is labelled with z , the second part of the formula fails to hold. Hence $\text{uniq}(\varphi)$ holds in a state (under the structure semantics) if, and only if, exactly one reachable state satisfies φ . Similarly, we can count the number of successors satisfying a given formula:

$$\begin{aligned} \mathbf{EX}_1 \varphi &= \mathbf{EX} \varphi \wedge \forall z. \left(\mathbf{EX}(\varphi \wedge z) \Rightarrow \mathbf{AX}(\varphi \Rightarrow z) \right) \\ \mathbf{EX}_{\geq k} \varphi &= \exists p_1, \dots, p_k. \left[\mathbf{AX} \left(\bigwedge_{i \neq j} \neg p_i \vee \neg p_j \right) \wedge \bigwedge_{1 \leq i \leq k} \mathbf{EX}(p_i \wedge \varphi) \right] \end{aligned}$$

As another example, let us mention that propositional quantification can be used to flatten “until” formulas:

$$\mathbf{E}\varphi_1 \mathbf{U} \varphi_2 \equiv \exists z_1, z_2. \mathbf{E}z_1 \mathbf{U} z_2 \wedge \mathbf{AG} [z_1 \Rightarrow \varphi_1 \wedge z_2 \Rightarrow \varphi_2] \quad (2.1)$$

Actually, “until” can be expressed using only “next” and “always”. This is easily achieved in QCTL*, where we would write e.g.

$$\begin{aligned} \mathbf{E}\varphi_1 \mathbf{U} \varphi_2 \equiv \exists z_1, z_2. \left[\mathbf{E} \left([z_2 \vee (z_1 \wedge \mathbf{F} z_2)] \wedge \right. \right. \\ \left. \left. \mathbf{G} [z_1 \Rightarrow \mathbf{X}(z_1 \vee z_2)] \right) \wedge \mathbf{AG} ([z_1 \Rightarrow \varphi_1 \wedge z_2 \Rightarrow \varphi_2]) \right]. \end{aligned}$$

The expression in QCTL is more involved. We rely on a more general translation through the μ -calculus [Koz83]: in this formalism, we can express the “until” modality as a fixpoint:

$$\mathbf{E}\alpha \mathbf{U} \beta \equiv \mu T. (\beta \vee \mathbf{EX}(\alpha \wedge T)).$$

Now, a least-fixpoint formula $\mu T. \varphi(T)$ (where we assume that $\varphi(T_1) \subseteq \varphi(T_2)$ whenever $T_1 \subseteq T_2$) can be expressed⁴ in QCTL as follows:

$$\mu T. \varphi(T) \equiv \exists t. \left[\mathbf{AG}(t \Leftrightarrow \varphi(t)) \wedge \forall u. [\mathbf{AG}(u \Leftrightarrow \varphi(u)) \Rightarrow \mathbf{AG}(t \Rightarrow u)] \right]$$

The first part of the formula (before quantifying over u) states that the labelling with t is a fixpoint. The second part enforces that it precisely corresponds to the least one.

⁴We have to be careful here with the exact notion of equivalence. We keep it imprecise in this example, and develop the technical details in Section 3.2, where we prove that QCTL without “until” can actually express the whole Monadic Second-Order Logic.

2.3.2. *Tree semantics.* The tree-semantics is obtained from the structure semantics by seeing the execution tree as an infinite-state Kripke structure. We write $\mathcal{S}, q \models_t \varphi$ to denote that formula φ holds at q under the tree semantics. Formally, seeing $\mathcal{T}_{\mathcal{S}}(q)$ as an infinite-state Kripke structure, we define:

$$\mathcal{S}, q \models_t \varphi \quad \text{iff} \quad \mathcal{T}_{\mathcal{S}}(q), q \models_s \varphi$$

Clearly enough, `selfloop` is always false under the tree semantics, while `uniq(φ)` holds if, and only if, φ holds at only one node of the execution tree.

Example 2.8. *Formula `acyclic` = $\mathbf{AG}(\exists z. (z \wedge \text{uniq}(z) \wedge \mathbf{AX} \mathbf{AG} \neg z))$ expresses that all infinite paths (starting from the current state) are acyclic, which for finite Kripke structures is always false under the structure semantics and always true under the tree semantics.*

2.3.3. *Equivalences between QCTL* formulas.* We consider two kinds of equivalences depending on the semantics we use. Two state formulas φ and ψ are said *s-equivalent* (resp. *t-equivalent*), written $\varphi \equiv_s \psi$ (resp. written $\varphi \equiv_t \psi$) if for any finite-state Kripke structure \mathcal{S} and any state q of \mathcal{S} , it holds $\mathcal{S}, q \models_s \varphi$ iff $\mathcal{S}, q \models_s \psi$ (resp. $\mathcal{S}, q \models_t \varphi$ iff $\mathcal{S}, q \models_t \psi$). We write $\varphi \equiv_{s,t} \psi$ when the equivalence holds for both \equiv_s and \equiv_t .

Note that both equivalences \equiv_s and \equiv_t are *substitutive*, i.e., a subformula ψ can be replaced with any equivalent formula ψ' without changing the truth value of the global formula. Formally, if $\psi \equiv_s \psi'$ (resp. $\psi \equiv_t \psi'$), we have $\Phi[\psi] \equiv_s \Phi[\psi']$ (resp. $\Phi[\psi] \equiv_t \Phi[\psi']$) for any QCTL* formula Φ .

2.4. **Fragments of QCTL*.** In the sequel, besides QCTL and QCTL*, we study several interesting fragments. The first one is the fragment of QCTL in *prenex normal form*, i.e., in which propositional quantification must be external to the CTL formula. We write EQCTL and EQCTL* for the corresponding logics⁵

We also study the fragments of these logics with limited quantification. For prenex-normal-form formulas, the fragments are defined as follows:

- for any $\varphi \in \text{CTL}$ and any $p \in \text{AP}$, $\exists p.\varphi$ is an EQ¹CTL formula, and $\forall p.\varphi$ is in AQ¹CTL;
- for any $\varphi \in \text{EQ}^k\text{CTL}$ and any $p \in \text{AP}$, $\exists p.\varphi$ is in EQ^kCTL and $\forall p.\varphi$ is in AQ^{k+1}CTL.

Symmetrically, if $\varphi \in \text{AQ}^k\text{CTL}$, then $\exists p.\varphi$ is in EQ^{k+1}CTL while $\forall p.\varphi$ remains in AQ^kCTL. Using similar ideas, we define fragments of QCTL and QCTL*. Again, the definition is inductive: Q¹CTL is the logic CTL[EQ¹CTL], and Q^{k+1}CTL = Q¹CTL[Q^kCTL]. Notice that a more refined definition of Q^kCTL could be given, where the index k would count quantifier alternation (in a way similar to EQ^kCTL) instead of the mere quantifier depth that we use here. This however requires taking care of the number of negations between two quantifiers, where “negation” here also includes hidden negations (e.g. a quantifier nested on the left-hand side of an “until” formula should be considered negated). Our results would carry on to this variant of Q^kCTL.

The corresponding extensions of CTL*, which we respectively denote with EQ^kCTL*, AQ^kCTL* and Q^kCTL*, are defined in a similar way.

⁵Notice that the logics named EQCTL and EQCTL* in [Kup95] are restrictions of our prenex-normal-form logics where only existential quantification is allowed. They correspond to our fragments EQ¹CTL and EQ¹CTL*.

Remark 2.9. Notice that EQ^kCTL and AQ^kCTL are (syntactically) included in Q^kCTL , and EQ^kCTL^* and AQ^kCTL^* are fragments of Q^kCTL^* .

3. EXPRESSIVENESS

As a preliminary remark, let us mention that propositional quantification increases the expressive power of CTL. For example, it is easy to see that the formula $\text{uniq}(P)$ defined in the previous section allows us to distinguish between two bisimilar structures; therefore such a formula cannot be expressed in CTL^* . Note also that it makes QCTL (and QCTL^*) to not be *bisimilar invariant*. This observation motivated an alternative semantics, called the *amorphous semantics*, for the propositional quantifications, in order to make QCTL (and QCTL^*) bisimilar invariant. We do not develop this semantics further, and refer the reader to [Fre01] for more details.

In this section we present several results about the expressiveness of our logics for both the structure- and the tree semantics. We show that QCTL, QCTL^* and Monadic Second-Order Logic are equally expressive. First we show that any QCTL formula is equivalent to a formula in prenex normal form (which extends to QCTL^* thanks to Proposition 3.8).

3.1. Prenex normal form. By translating path quantification into propositional quantification, we can extract propositional quantification out of purely temporal formulas: for instance, $\mathbf{EX}(Q.\varphi)$ where Q is some propositional quantification is equivalent to $\exists z.Q.(\text{uniq}(z) \wedge \mathbf{EX}(z \wedge \varphi))$. This generalizes to full QCTL under both semantics:

Proposition 3.1. *In both semantics, EQCTL and QCTL are equally expressive.*

Proof. We prove the result for structure equivalence, turning a given a QCTL formula φ into prenex normal form. The transformation being correct also for infinite-state Kripke structures, the result for tree-equivalence follows.

In the following, we assume w.l.o.g. no atomic proposition is quantified twice. We use Q to denote a sequence of quantifications, and write \bar{Q} for the dual sequence. Our translation is defined as a sequence of rewriting rules that are to be applied in a bottom-up manner, first replacing innermost subformulas with s -equivalent ones. As for CTL, we only consider the temporal modalities \mathbf{EX} , \mathbf{EU} and \mathbf{EG} (which is sufficient since $\mathbf{AX}\varphi \equiv_{s,t} \neg \mathbf{EX} \neg \varphi$ and $\mathbf{A}\varphi \mathbf{U} \psi \equiv_{s,t} \neg \mathbf{EG} \neg \psi \wedge \neg \mathbf{E} \neg \psi \mathbf{U} (\neg \varphi \wedge \neg \psi)$).

For propositional and Boolean subformulas, we have:

$$\neg Q.\varphi \equiv_s \bar{Q} \neg \varphi \qquad Q_1.\varphi_1 \vee Q_2.\varphi_2 \equiv_s Q_1.Q_2.(\varphi_1 \vee \varphi_2)$$

We now present the transformation for all three temporal modalities. Extracting a bloc of quantifiers out of an \mathbf{EX} operator can be done as follows:

$$\mathbf{EX} Q.\varphi \equiv_s \exists z.Q.(\text{uniq}(z) \wedge \mathbf{EX}(z \wedge \varphi))$$

Here variable z (which is assumed to not appear in $Q.\varphi$) is used to mark an immediate successor that satisfies $Q.\varphi$. We require z to be unique: allowing more than one successor would make the equivalence to be wrong, as can be seen on the Kripke structures \mathcal{S}_0 of Fig. 1 using formula $\mathbf{EX}(\forall p. [(\mathbf{EF} p) \Rightarrow p])$ (this formula is false, while formula $\exists z. \forall p. \mathbf{EX}(z \wedge [(\mathbf{EF} p) \Rightarrow p])$ is true by labelling both states with z).

Note that the right-hand-side formula is not yet in prenex form, because $\text{uniq}(z)$ involves a universal quantifier under a Boolean operator; applying the above rules for Boolean subformulas concludes the translation for this case.

For $\mathbf{EG}(\mathcal{Q}.\varphi)$, the idea again is to label a short (lasso-shaped) path with z , ensuring that $\mathcal{Q}.\varphi$ always holds along that path:

$$\mathbf{EG}(\mathcal{Q}.\varphi) \equiv_s \exists z. \forall z'. \mathcal{Q}.(z \wedge \mathbf{AG}(z \Rightarrow \mathbf{EX}_1 z) \wedge (\text{uniq}(z') \Rightarrow \mathbf{AG}((z \wedge z') \Rightarrow \varphi))).$$

Finally, $\mathbf{E}(\mathcal{Q}_1.\varphi_1) \mathbf{U}(\mathcal{Q}_2.\varphi_2)$ is handled by first rewriting it as

$$\exists z_1, z_2. \mathbf{E}z_1 \mathbf{U} z_2 \wedge \mathbf{AG}[z_1 \Rightarrow \mathcal{Q}_1.\varphi_1 \wedge z_2 \Rightarrow \mathcal{Q}_2.\varphi_2]$$

using Equivalence (2.1), and by noticing that $\mathbf{AG}(\mathcal{Q}.\varphi) \equiv_s \forall z. \mathcal{Q}.(\text{uniq}(z) \Rightarrow \mathbf{AG}(z \Rightarrow \varphi))$.

Before we prove correctness of the above equivalences, we introduce a useful lemma:

Lemma 3.2. *Consider a Kripke structure $\mathcal{S} = \langle Q, R, \ell \rangle$, a state q and a QCTL formula $\mathcal{Q}.\varphi$ with $\mathcal{Q} = \mathcal{Q}_1 z_1 \cdots \mathcal{Q}_k z_k$ and $\varphi \in \text{CTL}$. We have $\mathcal{S}, q \models_s \mathcal{Q}.\varphi$ if, and only if, there is a non-empty family ξ of Kripke structures such that*

- (1) *each $\mathcal{S}' \in \xi$ is of the form $\langle Q, R, \ell' \rangle$ where ℓ' and ℓ coincide over $\text{AP} \setminus \{z_1, \dots, z_k\}$;*
- (2) *for any $\mathcal{S}' = \langle Q, R, \ell' \rangle$ in ξ , any i with $\mathcal{Q}_i = \forall$, and any $\text{lab}_{z_i}: Q \rightarrow 2^{\{z_i\}}$, there exists $\langle Q, R, \ell'' \rangle \in \xi$ such that $\ell'' \cap \{z_i\} = \text{lab}_{z_i}$, and ℓ'' and ℓ' coincide over $\text{AP} \setminus \{z_i \cdots z_k\}$;*
- (3) *for all $\mathcal{S}' \in \xi$, it holds $\mathcal{S}', q \models_s \varphi$.*

A non-empty set ξ satisfying the first two properties of Lemma 3.2 is said to be $(\mathcal{Q}, \mathcal{S})$ -compatible.

Proof. The proof proceeds by induction on the number of quantifiers in \mathcal{Q} . The equivalence is trivial when there is no quantification. Now assume that the equivalence holds for some quantification \mathcal{Q} .

We first consider formula $\exists z. \mathcal{Q}.\varphi$. Assume $\mathcal{S}, q \models_s \exists z. \mathcal{Q}.\varphi$. Then there exists a structure $\mathcal{S}' = \langle Q, R, \ell' \rangle$, with ℓ' coincides with ℓ over $\text{AP} \setminus \{z\}$, such that $\mathcal{S}', q \models_s \mathcal{Q}.\varphi$. Applying the induction hypothesis to \mathcal{S}' , we obtain a family of structures satisfying conditions (1) to (3) for \mathcal{S}' and $\mathcal{Q}.\varphi$. One easily checks that the very same family also fulfills all three conditions for \mathcal{S} and $\exists z. \mathcal{Q}.\varphi$.

Conversely, if there is a family of structures satisfying all three conditions for \mathcal{S} and $(\exists z. \mathcal{Q}.\varphi)$. Pick any structure $\mathcal{S}' = \langle Q, R, \ell' \rangle$ in that family. It holds $\mathcal{S}', q \models_s \exists z. \mathcal{Q}.\varphi$, and moreover ℓ and ℓ' coincide over $\text{AP} \setminus \{z, z_1, \dots, z_k\}$, where $\{z_1, \dots, z_k\}$ are the variables appearing in \mathcal{Q} . Hence also $\mathcal{S}, q \models_s \exists z. \mathcal{Q}.\varphi$.

Now consider formula $\forall z. \mathcal{Q}.\varphi$, and assume $\mathcal{S}, q \models_s \forall z. \mathcal{Q}.\varphi$. Then for any $\mathcal{S}' = \langle Q, R, \ell' \rangle$ where ℓ and ℓ' coincide over $\text{AP} \setminus \{z\}$, we have $\mathcal{S}', q \models_s \mathcal{Q}.\varphi$. Applying the induction hypothesis, for each such \mathcal{S}' , we get a family of Kripke structures satisfying all three conditions for \mathcal{S}' and $\mathcal{Q}.\varphi$. Let ξ be the union of all those families. Then ξ clearly fulfills conditions (1) and (3). Condition (2) for universal quantifiers in \mathcal{Q} follows from the induction hypothesis. For the universal quantifier on z , pick $\mathcal{S}' = \langle Q, R, \ell' \rangle$ and lab_z . Then by construction, ξ contains a structure $\mathcal{S}'' = \langle Q, R, \ell'' \rangle$ where $\ell'' \cap \{z\} = \text{lab}_z$. By construction, ξ contains a family of structures satisfying all three conditions for \mathcal{S}'' and $\mathcal{Q}.\varphi$, which entails the result.

If conversely there is a family ξ of structures satisfying the conditions of the lemma, then for each lab_z , this family contains a structure $\mathcal{S}' = \langle Q, R, \ell' \rangle$ with $\ell' \cap \{z\} = \text{lab}_z$ and satisfying $\mathcal{Q}.\varphi$, which entails the result. \square

We now proceed to the proof of the previous equivalences. We omit the easy cases of propositional and Boolean formulas, and focus on **EX** and **EG**:

- **EX** ($\mathcal{Q}.\varphi$): Assume $\mathcal{S}, q \models_s \mathbf{EX}(\mathcal{Q}.\varphi)$ with $\mathcal{S} = \langle Q, R, \ell \rangle$. Then there exists $(q, q') \in R$ such that $\mathcal{S}, q' \models_s \mathcal{Q}.\varphi$. Therefore there exists a set ξ of Kripke structures that is $(\mathcal{Q}, \mathcal{S})$ -compatible and such that $\mathcal{S}', q' \models_s \varphi$ for every $\mathcal{S}' \in \xi$. Now consider the set ξ' defined as follows:

$$\xi' = \left\{ \mathcal{S}' = \langle Q, R, \ell' \rangle \mid \exists \langle Q, R, \ell'' \rangle \in \xi \text{ s.t. } \ell' = \ell'' \oplus \{q' \mapsto z\} \right\}$$

with:

$$(\ell \oplus \{q \mapsto x\})(r) = \begin{cases} \ell(r) \cup \{x\} & \text{if } r = q \\ \ell(r) \setminus \{x\} & \text{otherwise} \end{cases}$$

Then ξ' is $(\exists z.\mathcal{Q}, \mathcal{S})$ -compatible, and for every Kripke structure $\mathcal{S}' \in \xi'$, we have: $\mathcal{S}', q \models_s \text{uniq}(z) \wedge \mathbf{EX}(z \wedge \varphi)$. It follows $\mathcal{S}, q \models_s \exists z.\mathcal{Q}(\text{uniq}(z) \wedge \mathbf{EX}(z \wedge \varphi))$.

Now assume $\mathcal{S}, q \models_s \exists z.\mathcal{Q}(\text{uniq}(z) \wedge \mathbf{EX}(z \wedge \varphi))$. Then there exists a Kripke structure $\mathcal{S}' \equiv_{\text{AP} \setminus \{z\}} \mathcal{S}$ such that $\mathcal{S}', q \models_s \mathcal{Q}(\text{uniq}(z) \wedge \mathbf{EX}(z \wedge \varphi))$. In particular, only one state q' of \mathcal{S}' is labelled with z , and q' is a successor of q . Moreover, there exists a $(\mathcal{Q}, \mathcal{S}')$ -compatible set ξ such that for any $\mathcal{S}'' \in \xi$, it holds $\mathcal{S}'', q \models_s \mathbf{EX}(z \wedge \varphi)$. Since only q' is labelled with z , we have $\mathcal{S}'', q' \models_s \varphi$, for all $\mathcal{S}'' \in \xi$. Hence $\mathcal{S}', q' \models_s \mathcal{Q}.\varphi$, and $\mathcal{S}', q \models_s \mathbf{EX}(\mathcal{Q}.\varphi)$. Finally, the formula is independent of z , so that also $\mathcal{S}, q \models_s \mathbf{EX}(\mathcal{Q}.\varphi)$.

- **EG** ($\mathcal{Q}.\varphi$): Assume $\mathcal{S}, q \models_s \mathbf{EG}(\mathcal{Q}.\varphi)$. There must exist a lasso-shape path $\rho = q_0 q_1 q_2 \dots (q_i \dots q_j)^\omega$, with $q_0 = q$, along which $\mathcal{Q}.\varphi$ always holds. We can also assume that ρ is a *direct* path, *i.e.*, that \mathcal{S} does not contain a transition (q_k, q_l) unless $l = k + 1$ (otherwise a simpler witnessing path would exist). Thus labeling all states of ρ with z makes the formula $(z \wedge \mathbf{AG}(z \Rightarrow \mathbf{EX}_1 z))$ hold at q . Moreover, for every $k < j$, we have $\mathcal{S}, q_k \models_s \mathcal{Q}.\varphi$, so that there exists a set ξ_k of Kripke structures that are $(\mathcal{Q}, \mathcal{S})$ -compatible and such that $\mathcal{S}', q_k \models_s \varphi$ for every $\mathcal{S}' \in \xi_k$. Now, let ξ be the following set of Kripke structures:

$$\xi = \left\{ \mathcal{S}' = \langle Q, R, \ell' \rangle \mid \exists k < j. \exists \langle Q, R, \ell'' \rangle \in \xi_k \text{ s.t. } \ell' = \ell'' \oplus \{q_l \mapsto z\}_{l=0, \dots, j-1} \oplus \{q_k \mapsto z'\} \right\}.$$

For every $\mathcal{S}' \in \xi$, we have

$$\mathcal{S}', q \models_s z \wedge \mathbf{AG}(z \Rightarrow \mathbf{EX}_1 z) \wedge (\text{uniq}(z') \Rightarrow \mathbf{AG}((z \wedge z') \Rightarrow \varphi)).$$

But the set ξ is not $(\exists z.\forall z'.\mathcal{Q}, \mathcal{S})$ -compatible: it only contains Kripke structures in which z' labels a single state of ρ , while condition (2) requires that we consider all labellings. It suffices to extend ξ with arbitrary Kripke structures involving all other forms of z' -labellings to obtain a compatible set $\widehat{\xi}$. Note that the additional Kripke structures still satisfy $(\text{uniq}(z') \Rightarrow \mathbf{AG}((z \wedge z') \Rightarrow \varphi))$. Applying Lemma 3.2,

$$\mathcal{S}, q \models_s \exists z.\forall z'.\mathcal{Q}.\left(z \wedge \mathbf{AG}(z \Rightarrow \mathbf{EX}_1 z) \wedge (\text{uniq}(z') \Rightarrow \mathbf{AG}((z \wedge z') \Rightarrow \varphi)) \right).$$

Conversely, assume that this formula holds true at q in \mathcal{S} . Accordingly, let $\mathcal{S}' \equiv_{\text{AP} \setminus \{z\}} \mathcal{S}$ be the structure obtained from \mathcal{S} by extending its labelling with z in such a way that

- (1) $\mathcal{S}', q \models_s z \wedge \mathbf{AG}(z \Rightarrow \mathbf{EX}_1 z)$
- (2) $\mathcal{S}', q \models_s \forall z'.\mathcal{Q}(\text{uniq}(z') \Rightarrow \mathbf{AG}((z \wedge z') \Rightarrow \varphi))$.

The first property ensures that the z -labelling describes a lasso-shape path starting from q . The second one entails that there exists a $(\forall z' \mathcal{Q}, \mathcal{S}')$ -compatible set ξ s.t. for every $\mathcal{S}'' \in \xi$, we have $\mathcal{S}'', q \models_s \text{uniq}(z') \Rightarrow \mathbf{AG}((z \wedge z') \Rightarrow \varphi)$. This entails that for any position k along the z -path, there exists a $(\mathcal{Q}, \mathcal{S}'')$ -compatible set in which $\mathcal{S}, q_k \models_s \varphi$, which entails the result. \square

Let us briefly measure the size and alternation depth of the resulting formula: in terms of its size, the transformation never duplicates subformulas of the initial formula, so that the final size is linear in the size of the original formula. Regarding proposition quantifiers, it can be checked that the alternation depth of the resulting formula is at most two plus the sum of the number of nested quantifiers in the original formula. In the end, the number of quantifier alternations of the resulting formula is linear in the number of quantifiers in the original formula.

Remark 3.3. *The translation used in the proof above to transform any QCTL formula into an equivalent formula in prenex normal form has been defined for the structure semantics. It is still correct when considering the tree semantics but in this framework, we could define a simpler transformation (in particular, we can get rid of the $\text{dpath}(z_0, z_2)$ formula).*

3.2. QCTL and Monadic Second-Order Logic. We briefly review Monadic Second-Order Logic (MSO) over trees and over finite Kripke structures (*i.e.*, labelled finite graphs). In both cases, we use constant monadic predicates P_a for $a \in \text{AP}$ and a relation Edge either for the immediate successor relation in a 2^{AP} -labelled tree $\langle T, l \rangle$ or for the relation R in a finite KS $\langle Q, R, \ell \rangle$.

MSO is built with first-order (or individual) variables for vertices (denoted with lowercase letters x, y, \dots), monadic second-order variables for sets of vertices (denoted with uppercase letters X, Y, \dots). Atomic formulas are of the form $x = y$, $\text{Edge}(x, y)$, $x \in X$, and $P_a(x)$. Formulas are constructed from atomic formulas using the boolean connectives and the first- and second-order quantifier \exists . We write $\varphi(x_1, \dots, x_n, X_1, \dots, X_k)$ to state that x_1, \dots, x_n and X_1, \dots, X_k may appear free (*i.e.* not within the scope of a quantifier) in φ . A closed formula contains no free variable. We use the standard semantics for MSO, writing $\mathcal{M}, s_1, \dots, s_n, S_1, \dots, S_k \models \varphi(x_1, \dots, x_n, X_1, \dots, X_k)$ when φ holds on \mathcal{M} when s_i (resp. S_j) is assigned to the variable x_i (resp. X_j) for $i = 1, \dots, n$ (resp. $j = 1, \dots, k$).

In the following, we compare the expressiveness of QCTL with MSO over the finite Kripke structures (the structure semantics) and the execution trees corresponding to a finite Kripke structure (tree semantics). First note that MSO formulas may express properties on the whole trees or graphs, while our logics are interpreted over *states* of these structures. Therefore we use MSO formulas with one free variable x , which represents the state where the formula is evaluated. Moreover, we restrict the evaluation of MSO formulas to the *reachable* part of the model from the given state. This last requirement makes an important difference for the structure semantics, since MSO can express *e.g.* that a graph is not connected while QCTL can only deal with what is reachable from the initial state.

Formally, for the tree semantics, we say that $\varphi(x) \in \text{MSO}$ is t -equivalent to some QCTL* formula ψ (written $\varphi(x) \equiv_t \psi$) when for any finite Kripke structure \mathcal{S} and any state $q \in \mathcal{T}_{\mathcal{S}}$, it holds $\mathcal{T}_{\mathcal{S}}(q), q \models \varphi(x)$ iff $\mathcal{T}_{\mathcal{S}}(q), q \models_s \psi$. Similarly, for the structure semantics: $\varphi(x)$ is s -equivalent to ψ (written $\varphi(x) \equiv_s \psi$) iff for any finite Kripke structure \mathcal{S} and any state

$q \in \mathcal{S}$, it holds $\mathcal{S}_q, q \models \varphi(x)$ iff $\mathcal{S}_q, q \models_s \psi$, where \mathcal{S}_q is the reachable part of \mathcal{S} from q . For these definitions, we have:

Proposition 3.4. *Under both semantics, MSO and QCTL are equally expressive.*

Proof. The translation from QCTL to MSO is easy: translating CTL into MSO is standard and adding propositional quantifications can be managed with second-order quantifications.

Now we consider the translation from MSO to QCTL. This translation (which is valid for both semantics) is defined inductively with a set of rewriting rules. Given $\varphi(x) \in \text{MSO}$, we define $\widehat{\varphi} \in \text{QCTL}$ as follows:

$$\begin{array}{ll}
 \widehat{\neg \varphi} = \neg \widehat{\varphi} & \widehat{\varphi \wedge \psi} = \widehat{\varphi} \wedge \widehat{\psi} \\
 \widehat{P_a(x)} = a & \widehat{P_a(x_i)} = \mathbf{EF}(\mathbf{p}_{x_i} \wedge a) \\
 \widehat{x = x_i} = \mathbf{p}_{x_i} & \widehat{x_i = x_j} = \mathbf{EF}(\mathbf{p}_{x_i} \wedge \mathbf{p}_{x_j}) \\
 \widehat{x \in X_i} = \mathbf{p}_{X_i} & \widehat{x_i \in X_j} = \mathbf{EF}(\mathbf{p}_{x_i} \wedge \mathbf{p}_{X_j}) \\
 \widehat{\text{Edge}(x, x_i)} = \mathbf{EX} \mathbf{p}_{x_i} & \widehat{\text{Edge}(x_i, x_j)} = \mathbf{EF}(\mathbf{p}_{x_i} \wedge \mathbf{EX} \mathbf{p}_{x_j}) \\
 \widehat{\exists X_i. \varphi} = \exists \mathbf{p}_{X_i}. \widehat{\varphi} & \widehat{\exists x_i. \varphi} = \exists \mathbf{p}_{x_i}. \text{uniq}(\mathbf{p}_{x_i}) \wedge \widehat{\varphi}
 \end{array}$$

The last rule not listed above concerns $\widehat{\text{Edge}(x_i, x)}$, and depends on the semantics: in the tree semantics, there is no edges coming back to the root and the formula is then equivalent to false; in the structure semantics, we have to mark the root with an individual variable and use the same kind of rule as above:

$$\widehat{\text{Edge}(x_i, x)} = \begin{cases} \perp & \text{in the tree semantics.} \\ \mathbf{EF}(\mathbf{p}_{x_i} \wedge \mathbf{EX} \mathbf{p}_x) & \text{in the structure semantics.} \end{cases}$$

The correctness of the translation w.r.t. both semantics is stated in the two following Lemmas, whose inductive proofs are straightforward:

Lemma 3.5. *For any $\varphi(x, x_1, \dots, x_n, X_1, \dots, X_k) \in \text{MSO}$, any finite Kripke structure \mathcal{S} and any state q , we have:*

$$\mathcal{T}_{\mathcal{S}}(q), q, s_1, \dots, s_n, S_1, \dots, S_k \models_s \varphi(x, x_1, \dots, x_n, X_1, \dots, X_k) \quad \text{iff} \quad \mathcal{T}'_{\mathcal{S}}(q), q \models_s \widehat{\varphi}$$

where $\mathcal{T}_{\mathcal{S}}$ and $\mathcal{T}'_{\mathcal{S}}$ only differ in the labelling of propositions \mathbf{p}_{x_i} and \mathbf{p}_{X_i} : in $\mathcal{T}_{\mathcal{S}}$, no state is labelled with these propositions, while in $\mathcal{T}'_{\mathcal{S}}$, we have (1) $\mathbf{p}_{x_i} \in \ell(s)$ iff $s = s_i$ and (2) $\mathbf{p}_{X_i} \in \ell(s)$ iff $s \in S_i$.

As a special case, we get that $\mathcal{T}_{\mathcal{S}}(q), q \models \varphi(x)$ if, and only if, $\mathcal{T}_{\mathcal{S}}(q), q \models_s \widehat{\varphi}$, which entails $\varphi(x) \equiv_t \widehat{\varphi}$.

As regards the structure semantics, using similar ideas, we have:

Lemma 3.6. *For any $\varphi(x, x_1, \dots, x_n, X_1, \dots, X_k) \in \text{MSO}$, any finite Kripke structure \mathcal{S} and any state q , we have:*

$$\mathcal{S}_q, q, s_1, \dots, s_n, S_1, \dots, S_k \models \varphi(x, x_1, \dots, x_n, X_1, \dots, X_k) \quad \text{iff} \quad \mathcal{S}'_q, q \models_s \widehat{\varphi}$$

where \mathcal{S}_q and \mathcal{S}'_q only differ in the labelling of propositions \mathbf{p}_{x_i} and \mathbf{p}_{X_i} : in \mathcal{S}_q , no state is labelled with these propositions, while in \mathcal{S}'_q we have (1) $\mathbf{p}_x \in \ell(s)$ iff $s = q$, (2) $\mathbf{p}_{x_i} \in \ell(s)$ iff $s = s_i$ and (3) $\mathbf{p}_{X_i} \in \ell(s)$ iff $s \in S_i$.

In the end, after labelling state q with \mathbf{p}_x , we obtain $\mathcal{S}_q, q \models \varphi(x)$ if, and only if, $\mathcal{S}'_q, q \models_s \widehat{\varphi}$, where \mathcal{S}'_q only differs from \mathcal{S}_q by the fact that q is labelled with p_x . It follows that $\varphi(x) \equiv_s \exists \mathbf{p}_x. (\mathbf{p}_x \wedge \text{uniq}(\mathbf{p}_x) \wedge \widehat{\varphi})$. \square

Remark 3.7. *One can also notice that it is easy to express fixpoint operators with QCTL in both semantics, thus μ -calculus can be translated into QCTL. For instance, the least fixpoint equation $\mu T. [b \vee (a \wedge \mathbf{E}X T)]$ would be written as*

$$\exists T. \left[\mathbf{A}G \left(T \Leftrightarrow [b \vee (a \wedge \mathbf{E}X T)] \right) \right] \wedge \forall U. \left\{ \mathbf{A}G \left(U \Leftrightarrow [b \vee (a \wedge \mathbf{E}X U)] \right) \Rightarrow \mathbf{A}G \left(T \Rightarrow U \right) \right\}.$$

Such a formula says that there is a fixpoint T such that for any fixpoint U , T is included in U ; this precisely characterises least fixpoints. Since the μ -calculus extended with counting capabilities has the same expressiveness as MSO on trees [MR03], we get another evidence that QCTL can express all MSO properties when interpreted over trees.

3.3. QCTL and QCTL*. Finally, we show that QCTL* and QCTL are equally expressive for both semantics. The main idea of the proof is an inductive replacement of quantified subformulas with extra atomic propositions. Indeed note that for any CTL* state formula Φ and any QCTL* state formula ψ , we have $\Phi[\psi] \equiv_{s,t} \exists p_\psi. (\Phi[p_\psi] \wedge \mathbf{A}G(p_\psi \Leftrightarrow \psi))$ where p_ψ is a fresh atomic proposition. We have:

Proposition 3.8. *Under both semantics, QCTL* and QCTL are equally expressive.*

Proof. The result for the tree semantic has been shown in [Fre01]. Here we give a different translation, which is correct for both semantics. Consider a QCTL* formula Φ . The proof is by induction over the number k of subformulas of Φ that are not in QCTL. If $k = 0$, Φ already belongs to QCTL. Otherwise let ψ be one of the smallest Φ -subformulas in $\text{QCTL}^* \setminus \text{QCTL}$. Let α_i s with $i = 1, \dots, m$ be the largest ψ -subformulas belonging to QCTL (these are state formulas). Then $\psi[(\alpha_i \leftarrow p_i)_{i=1, \dots, m}]$ is a CTL* formula: every subformula of the form $\exists p. \xi$ in ψ belongs to some QCTL formula α_i , since ψ is one of the smallest $\text{QCTL}^* \setminus \text{QCTL}$ subformula. Therefore ψ is equivalent (w.r.t. both semantics) to:

$$\exists p_1 \dots \exists p_m. \left(\psi[(\alpha_i \leftarrow p_i)_{i=1, \dots, m}] \wedge \bigwedge_{i=1, \dots, m} \mathbf{A}G(p_i \Leftrightarrow \alpha_i) \right)$$

Since CTL* can be translated into the μ -calculus [Dam94], and the μ -calculus can in turn be translated into QCTL (see Remark. 3.7), we get that $\psi[(\alpha_i \leftarrow p_i)_{i=1, \dots, m}]$ is equivalent to some QCTL formula Ω . Hence

$$\psi \equiv_{s,t} \exists p_1 \dots \exists p_m. \left(\Omega \wedge \bigwedge_{i=1, \dots, m} \mathbf{A}G(p_i \Leftrightarrow \alpha_i) \right)$$

Now, consider the formula obtained from Φ by replacing ψ with the right-hand-side formula above. This formula is equivalent to Φ and has at most $k - 1$ subformulas in $\text{QCTL}^* \setminus \text{QCTL}$, so that the induction hypothesis applies. \square

From Propositions 3.1, 3.4 and 3.8, we get:

Corollary 3.9. *Under both semantics, the four logics EQCTL, QCTL and QCTL* and MSO are equally expressive.*

Remark 3.10. *In [Fre01], Tim French considers a variant of QCTL* (which we call FQCTL*), with propositional quantification within path formulas: $\exists p. \varphi_{\text{path}}$ is added in the definition of path formulas. The semantics is defined as follows:*

$$\mathcal{S}, \rho \models_s \exists p. \varphi_{\text{path}} \quad \text{iff} \quad \exists \mathcal{S}' \equiv_{\text{AP} \setminus \{p\}} \mathcal{S} \text{ s.t. } \mathcal{S}', \rho \models_s \varphi_{\text{path}}.$$

It appears that this logic is not very different from QCTL under the tree semantics: French showed that QCTL is as expressive as FQCTL*. Things are different in the structure-semantics setting, where we now show that FQCTL* is strictly more expressive than MSO. To begin with, consider the following formula:*

$$\mathbf{EG} (\exists z. \forall z'. [\text{uniq}(z) \wedge \text{uniq}(z') \wedge z \wedge \neg z'] \Rightarrow \mathbf{X} (\neg z \mathbf{U} z')).$$

This formula expresses the existence of an (infinite) path along which, between any two occurrences of the same state, all the other reachable states will be visited. This precisely corresponds to the existence of a Hamilton cycle, which is known not to be expressible in MSO [EF95, Cor. 6.3.5]. However, note that the existence of a Hamilton cycle can be expressed in Guarded Second Order Logic GSO⁶, in which quantification over sets of edges is allowed (in addition to quantification over sets of states). Still, FQCTL is strictly more expressive than GSO, as it is easy to modify the above formula to express the existence of Euler cycles:*

$$\begin{aligned} \mathbf{EG} \left(\exists x. \exists y. \forall x'. \forall y'. \left[\text{tr}(x, y) \wedge \text{tr}(x', y') \wedge \text{next_tr}(x, y) \wedge \neg \text{next_tr}(x', y') \right] \right. \\ \left. \Rightarrow \mathbf{X} (\neg \text{next_tr}(x, y) \mathbf{U} \text{next_tr}(x', y')) \right) \end{aligned}$$

where $\text{tr}(x, y) = \text{uniq}(x) \wedge \text{uniq}(y) \wedge \mathbf{EF} (x \wedge \mathbf{X} y)$ states that x and y mark the source and target of a reachable transition, and $\text{next_tr}(x, y) = x \wedge \mathbf{X} y$ states that the next transition along the current path jumps from x to y . This can be seen to express the existence of an Euler cycle, which cannot be expressed in GSO (otherwise evenness could also be expressed).

Proposition 3.11. *Under the structure semantics, FQCTL* is more expressive than QCTL* and MSO.*

Nevertheless FQCTL model checking (see next section) is decidable: for the tree semantics, it suffices to translate FQCTL* to QCTL, as proposed by French [Fre01]. The problem in the structure semantics can then be encoded in the tree semantics: for this we first need to extend the labelling of the Kripke structure \mathcal{S} with fresh propositions, one per state (e.g. assume that state q is labeled by p_q). Let \mathcal{S}' be such an extension (notice that the existence of an Euler path in such a Kripke structure can be expressed in CTL). Then any quantification $\exists P. \varphi$ in some FQCTL* formula Φ (for the structure semantics) is considered in the tree semantics. For this to be correct, we augment Φ with the extra requirement that any*

⁶This logic is called MS₂ in [CE11].

two copies of the same state receive the same labelling. We thus build a formula $\widehat{\Phi}^{S'}$, by replacing every subformula $\exists P. \psi$ in Φ with

$$\exists P. \bigwedge_{q \in Q} \left(\mathbf{EF} p_q \Rightarrow (\mathbf{EF} (p_q \wedge P) \Leftrightarrow \neg \mathbf{EF} (p_q \wedge \neg P)) \right) \wedge \widehat{\psi}^{S'}.$$

We then have: $\mathcal{S}, q \models_s \Phi$ iff $\mathcal{S}', q \models_t \widehat{\Phi}^{S'}$.

4. MODEL CHECKING

We now consider the model-checking problem for QCTL^* and its fragments under both semantics: given a finite Kripke structure \mathcal{S} , a state q and a formula φ , is φ satisfied in state q in \mathcal{S} under the structure (resp. tree) semantics? In this section, we characterise the complexity of this problem. A few results already exist, *e.g.* for EQ^1CTL and EQ^1CTL^* under both semantics [Kup95]. Hardness results for EQ^2CTL and EQ^2CTL^* under the tree semantics can be found in [KMTV00]. Here we extend these results to all the fragments of QCTL^* we have defined. We also characterize the program- and formula-complexities [Var82] of model-checking for these fragments: the formula complexity (resp. program complexity) consists in evaluating the complexity of the problem $\mathcal{S} \models \varphi$ when the model \mathcal{S} (resp. formula φ) is assumed to be fixed. Except for Theorem 4.8, our results hold true irrespective of the notion of size (classical size of DAG-size) we use for QCTL^* formulas. Appendix A proposes a short introduction to the complexity classes used in the rest of the paper (especially the polynomial-time and exponential hierarchies).

4.1. Model checking for the structure semantics.

4.1.1. *Fragments of QCTL.* First we consider the fragments of QCTL with limited quantifications: EQ^kCTL , AQ^kCTL , and Q^kCTL . Prenex-normal-form formulas are (technically) easy to handle inductively: a formula in EQ^kCTL can be checked by non-deterministically guessing a labelling and applying a model-checking procedure for $\text{AQ}^{k-1}\text{CTL}$. We prove that the model-checking problems for these fragments populate the polynomial-time hierarchy [Sto76]:

Theorem 4.1. Under the structure semantics, model checking EQ^kCTL is Σ_k^P -complete and model checking AQ^kCTL is Π_k^P -complete.

Proof. We begin with noticing that an AQ^kCTL formula is nothing but the negation of an EQ^kCTL formula. Hence it suffices to prove the result for EQ^kCTL . The case where $k = 0$ corresponds to CTL model-checking, which is PTIME-complete. For $k > 0$, hardness is easy, as EQ^kCTL model checking subsumes the following problem, which is known to be Σ_k^P -complete [Pap94]:

Problem: $\Sigma_k^P\text{SAT}$

Input: k families of variables $U_i = \{u_1^i, \dots, u_n^i\}$, and a propositional formula $\Phi(U_1, \dots, U_k)$ over $\bigcup_i U_i$;

Question: is the quantified Boolean formula $\mathcal{Q}_1 U_1 \mathcal{Q}_2 U_2 \dots \mathcal{Q}_k U_k. \Phi(U_1, \dots, U_k)$ true, where \mathcal{Q}_i is \exists (resp. \forall) when i is odd (resp. even)?

Membership in Σ_k^P is proved inductively: an EQ¹CTL instance $\exists u_1^1 \dots \exists u_k^1. \varphi$ can be solved in NP = Σ_1^P by non-deterministically picking a labelling of the Kripke structure under study with atomic propositions u_1^1 to u_k^1 , and then checking (in polynomial time) whether the CTL formula φ holds true in the resulting Kripke structure. Similarly, an EQ^kCTL formula $\exists u_1^1 \dots \exists u_k^1. \varphi$, where φ is in AQ^{k-1}CTL, can be checked by first non-deterministically labelling the Kripke structure with atomic propositions u_1^1 to u_k^1 , and checking the remaining AQ^{k-1}CTL formula φ in the resulting Kripke structure. The latter is in Π_{k-1}^P according to the induction hypothesis, so that the whole procedure is in Σ_k^P . \square

When dropping the prenex-normal-form restriction, we get

Theorem 4.2. Under the structure semantics, model checking Q^kCTL is $\Delta_{k+1}^P[O(\log n)]$ -complete.

Proof. We define the algorithm for Q^kCTL inductively: when $k = 0$, we just have a CTL model-checking problem, which is complete for PTIME = $\Delta_1^P[O(\log n)]$. Assume that we have a $\Delta_{k+1}^P[O(\log n)]$ algorithm for the Q^kCTL model-checking problem, and consider a formula $\varphi \in \mathbf{Q}^{k+1}\text{CTL}$: it can be written under the form $\varphi = \Phi[(q_i \rightarrow \exists P_i. \psi_i)_i]$ with Φ being a CTL formula involving fresh atomic propositions q_i , and $\exists P_i. \psi_i$ are subformulas⁷ of φ . The existential quantifiers in these subformulas are the outermost propositional quantifiers in φ , and ψ_i belongs to Q^kCTL, as we assume that Φ is a CTL formula. As a consequence, $\exists P_i. \psi_i$ is a state-formula, whose truth value only depends on the state in which it is evaluated. For such a formula, we can non-deterministically label the Kripke structure with propositions in P_i , and check whether ψ_i holds in the resulting Kripke structure. Computing the set of states satisfying $\exists P_i. \psi_i$ is then achieved in $\text{NP}^{\Delta_{k+1}^P[O(\log n)]}$, which is equal to Σ_{k+1}^P . Moreover, the queries for all the selected subformulas are independent and can be made in parallel. It just remains to check whether the CTL formula Φ holds, which can be achieved in polynomial time. This algorithm is thus in $\Delta_{k+2}^P[O(\log n)]$, since $\Delta_{k+2}^P[O(\log n)] = \Delta_{k+2,||}^P$ (see [Wag90]).

We prove hardness using problems PARITY(Σ_k^P), defined as follows:

Problem: PARITY(Σ_k^P)

Input: m instances of $\Sigma_k^P\text{SAT}$ $Q_1^i U_1^i \dots Q_k^i U_k^i. \Phi^i(U_1^i, \dots, U_k^i)$, where $Q_j^i = \exists$ when j is odd and $Q_j^i = \forall$ otherwise;

Question: is the number of positive instances even?

This problem is $\Delta_{k+1}^P[O(\log n)]$ -complete [Got95]. We encode it into a Q^kCTL model-checking problem. Fix $1 \leq i \leq m$; the instance $\Psi_i = Q_1^i U_1^i \dots Q_k^i U_k^i. \Phi^i(U_1^i, \dots, U_k^i)$ of $\Sigma_k^P\text{SAT}$ is encoded as in the previous reduction, using a one-state Kripke structure \mathcal{S}_i that will be labelled with atomic propositions $u_{j,l}^i$. The Q^kCTL formula to be checked is then Ψ_i itself. We label the unique state of that Kripke structure with an atomic proposition x_i , that will be used in the sequel of the reduction.

Now, consider the Kripke structure \mathcal{S} obtained as the “union” of the one-state Kripke structures above, augmented with an extra state x_{m+1} and transitions (x_i, x_{i+1}) , for each $1 \leq i \leq m$. We define $\varphi = \bigvee_{1 \leq i \leq m} (x_i \wedge \Psi_i)$. This formula holds true in those states x_i

⁷ $\exists P_i$ denotes a sequence of existential quantifications.

of \mathcal{S} whose corresponding $\Sigma_k^P\text{SAT}$ instance is positive. It remains to build a formula for “counting” these sets: we let

$$\psi_0 = \mathbf{E}(\neg\varphi \mathbf{U} x_{m+1}) \quad \text{and} \quad \psi_{i+1} = \mathbf{E}(\neg\varphi \mathbf{U} (\varphi \wedge \mathbf{E}\mathbf{X} \psi_i)).$$

It is easily seen that ψ_s holds true in state x_1 of \mathcal{S} iff exactly s of the m instances of $\Sigma_k^P\text{SAT}$ are positive. Moreover, each ψ_i has quantifier height at most k . The final formula is then the disjunction of the formulas ψ_{2i} , for $0 \leq i \leq m/2$. \square

4.1.2. *EQCTL and extensions of CTL**. When considering logics with no quantification restriction or the extensions of CTL^* , model-checking complexity becomes PSPACE-complete:

Theorem 4.3. Under the structure semantics, model checking EQCTL, QCTL, EQ^kCTL^* , AQ^kCTL^* , Q^kCTL^* , EQCTL^* and QCTL^* is PSPACE-complete.

Proof. PSPACE-hardness is straightforward because (1) any instance of QBF is a special case of a model checking problem for every logic with unbounded quantifications (EQCTL, QCTL, EQCTL^* and QCTL^*) and (2) the model-checking problem is PSPACE-hard for CTL^* [SC85], hence also for any extension thereof.

For PSPACE membership, it is sufficient to show the result for QCTL^* . Consider a formula $\Phi = \exists p_1 \dots \exists p_k. \varphi$ with $\varphi \in \text{CTL}^*$. We can easily consider the same kind of algorithm we used for EQ^kCTL in Theorem 4.1: we only replace the CTL model-checking algorithm with a CTL^* model-checking algorithm running in polynomial space [CES86]. Since $\text{NP}^{\text{PSPACE}} = \text{PSPACE}$, the resulting algorithm is in PSPACE. This clearly provides a PSPACE algorithm for any QCTL^* formula. \square

4.1.3. *Program-complexity.* Now we consider the *program complexity* (or *model complexity*) of model checking for the structure semantics. In this context, we assume that the formula is fixed, and the complexity is then expressed only in terms of the size of the model.

Theorem 4.4. Under the structure semantics, for any $k > 0$, the program-complexity of model checking is Σ_k^P -complete for EQ^kCTL and EQ^kCTL^* , and Π_k^P -complete for AQ^kCTL and AQ^kCTL^* .

Proof. Membership in Σ_k^P for EQ^kCTL and AQ^kCTL comes directly from the general algorithms (Theorem 4.1). For EQ^kCTL^* and AQ^kCTL^* , we can use the same approach: fix a formula $\Phi = \exists u_1^1 \dots \exists u_k^1. \varphi$ with $\varphi \in \text{CTL}^*$. Deciding the truth value of Φ can be done in NP by first non-deterministically guessing a labelling of the model with $\{u_1^1, \dots, u_k^1\}$ and then checking the *fixed* formula φ (model checking a fixed formula of CTL^* is NLOGSPACE-complete [Sch03]). Thus with the same argument we used for the proof of Theorem 4.1, we get a Σ_k^P algorithm for any fixed EQ^kCTL^* formula (and a Π_k^P algorithm for a AQ^kCTL^* formula).

We now prove hardness in Σ_k^P for EQ^kCTL (the results for EQ^kCTL^* , AQ^kCTL and AQ^kCTL^* are proven similarly). We begin with the case where $k = 1$ (for which the result is already given in [Kup95] with a proof derived from [HK94]): quantification is encoded in the (fixed) EQ^kCTL formula, while the model encodes the SAT formula to be checked. We begin with an NP-hardness proof for EQ^1CTL , and then explain how it can be extended to EQ^kCTL .

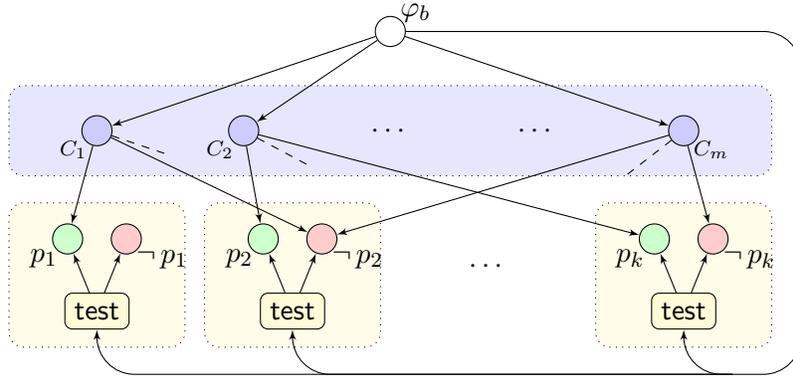


Fig. 3: The model used for proving Σ_k^P -hardness of model checking a fixed EQ^kCTL formula.

Consider an instance $\exists P. \varphi_b(P)$, where P is a set of variables. We assume w.l.o.g. that propositional formula φ_b is a conjunction of disjunctive clauses. We begin with defining the model associated to φ_b , and then build the formula, which will depend neither on φ_b , nor on P .

Write $\varphi_b = \bigwedge_{1 \leq i \leq m} \bigvee_{1 \leq j \leq n} \ell_{i,j}$, where $\ell_{i,j}$ is in $\{p_k, \neg p_k \mid p_k \in P\}$. The model is defined as follows:

- it has one initial state, named φ_b , m states named C_i for $1 \leq i \leq m$, and $3|P|$ states named p_k , $\neg p_k$ and $\text{test}(p_k)$ for each $p_k \in P$.
- there is a transition from φ_b to each C_i and to each $\text{test}(p_k)$, a transition from each $\text{test}(p_k)$ to the corresponding p_k and $\neg p_k$, and a transition from each C_i to its constitutive literals $\ell_{i,j}$. Finally, each p_k and $\neg p_k$ carries a self-loop.
- states $\text{test}(p_k)$ are labelled with an atomic proposition test , which is the only atomic proposition in the model.

Figure 3 displays an example of this construction. The intuition is as follows: one of the states p_k and $\neg p_k$ will be labelled (via the EQ^kCTL formula) with an extra proposition \oplus . That exactly one of them is labelled will be checked by the test -states. That the labelling defines a satisfying assignment will be checked by the C_i -states. The formula writes as follows:

$$\Phi = \exists \oplus . [\mathbf{AX} (\text{test} \Rightarrow (\mathbf{EX} \oplus \wedge \mathbf{EX} \neg \oplus)) \wedge \mathbf{AX} (\neg \text{test} \Rightarrow \mathbf{EX} \oplus)].$$

One is easily convinced that a labelling with \oplus defines a valuation of the propositions in P (by the first part of Φ), and that φ_b evaluates to true under that valuation (by the second part of Φ). Conversely, a satisfying assignment can be used to prove that Φ holds true in the model.

This reduction can be extended to prove Σ_k^P -hardness of model checking a fixed formula of EQ^kCTL . Consider an instance of $\Sigma_k^P\text{SAT}$ of the form $\exists P_1 \dots Q_k P_k. \varphi_b(P_1, \dots, P_k)$, assuming w.l.o.g. that the sets P_i are pairwise disjoint. The model now involves k test -propositions test_1 to test_k , and a test -state associated with a proposition in P_l is labelled with test_l . The rest of the construction is similar. Assuming that k is even (in which case Q_k is universal, and φ_b is a disjunction of conjunctive clauses—the dual case being similar),

formula Φ_k then writes as follows:

$$\Phi_k = \exists \oplus_1 \dots \forall \oplus_k . [\mathbf{AX}(\text{test}_{2i+1} \Rightarrow (\mathbf{EX} \oplus_{2i+1} \wedge \mathbf{EX} \neg \oplus_{2i+1})) \wedge \\ [\mathbf{AX}(\text{test}_{2i+2} \Rightarrow (\mathbf{EX} \oplus_{2i+2} \wedge \mathbf{EX} \neg \oplus_{2i+2}))] \Rightarrow (\mathbf{AX}(\neg \text{test} \Rightarrow \mathbf{EX} \oplus))].$$

□

For Q^kCTL and Q^kCTL^* , we have:

Theorem 4.5. Under the structure semantics, for any $k > 0$, the program-complexity of model checking is $\Delta_{k+1}^{\text{P}}[O(\log n)]$ -complete for Q^kCTL and Q^kCTL^* .

Proof. To prove membership in $\Delta_{k+1}^{\text{P}}[O(\log n)]$ for Q^kCTL^* , we reuse the same algorithm as for Theorem 4.2: we get the same complexity for Q^kCTL^* and Q^kCTL because program-complexity for CTL^* is in PTIME (as for CTL).

Now we prove hardness in $\Delta_{k+1}^{\text{P}}[O(\log n)]$ for the fixed-formula model-checking problem for Q^kCTL . Fix some k , and consider of $\text{PARITY}(\Sigma_k^{\text{P}})$, made of m instances of $\Sigma_k^{\text{P}}\text{SAT}$, which we write $\Phi^i(U_1^i, \dots, U_k^i)$ (assuming w.l.o.g. that they all begin with an existential quantifier). We begin with defining a partial view of the Kripke structure that we will use for the construction: it has an initial state init and a final state final , and, for each $1 \leq i \leq m$, four states labelled with i and either 0 or 1 (to indicate the parity of the number of positive formulas up to Φ_i) and either \oplus or \ominus (to indicate the validity of the i -th instance). Transitions are defined as follows: from init , there is a transition to $(1, 0, \oplus)$ and $(1, 0, \ominus)$; from $(i, 0, \oplus)$ and $(i, 1, \ominus)$, there are transitions to $(i+1, 1, \oplus)$ and to $(i+1, 1, \ominus)$; from $(i, 1, \oplus)$ and $(i, 0, \ominus)$, there are transitions to $(i+1, 0, \oplus)$ and $(i+1, 0, \ominus)$. Finally, there is a transition from $(m, 0, \ominus)$ and $(m, 1, \oplus)$ to final , and self-loops on final , $(m, 0, \oplus)$ and $(m, 1, \ominus)$. Consider a path in such a Kripke structure, and assume that we can enforce that the path visits a \oplus -state if, and only if, the corresponding $\Sigma_k^{\text{P}}\text{SAT}$ instance is positive. Then this path reaches final if, and only if, the total number of positive instances is even. Otherwise, the path will be stuck in $(m, 0, \oplus)$ or in $(m, 1, \ominus)$. In other words, formula $\mathbf{EF} \text{ final}$ holds true if, and only if, the number of positive instances is even.

It remains to enforce the correspondence between positive states and positive instances of $\Sigma_k^{\text{P}}\text{SAT}$. This is achieved using the reduction of the proof of Theorem 4.4: we first extend the above Kripke structure by plugging, at each state (i, j, k) , one copy of the Kripke structure built in the proof of Theorem 4.4. Now, the formula to be checked in the resulting structure has to be reinforced as follows:

$$\Psi_k = \mathbf{E}(\oplus \Leftrightarrow \tilde{\Phi}_k) \mathbf{U} \text{ final}$$

where $\tilde{\Phi}_k$ is (a slightly modified version of) the formula built in the proof of Theorem 4.4. □

When model checking a fixed formula of QCTL^* (hence with fixed alternation depth), there is no hope of being able to encode arbitrary alternation: the program complexity of QCTL^* (and QCTL) model checking thus lies in the small gap between PH and PSPACE, unless the polynomial-time hierarchy collapses:

Theorem 4.6. Under the structure semantics, the program-complexity of model checking is PH-hard but not in PH, and in PSPACE but not PSPACE-hard, for EQCTL , QCTL , EQCTL^* and QCTL^* (unless the polynomial-time hierarchy collapses).

Proof. From Theorem 4.4, model-checking a fixed formula in EQCTL or EQCTL* is PH-hard. Membership in PSPACE follows from Theorem 4.3. If these problems were in PH, they would lie in Σ_k^P for some k , and the polynomial-hierarchy would collapse. Similarly, if they were PSPACE-hard, then a fixed formula (in EQCTL or EQCTL*, hence in EQ^kCTL or EQ^kCTL* for some k) could be used to encode any instance of QSAT, again collapsing the polynomial-time hierarchy. The same method applies to QCTL and QCTL* thanks to Theorem 4.5. \square

4.1.4. *Formula-complexity.* Now we consider the formula complexity of model checking for the structure semantics. In this context we assume that the the *model* is assumed to be fixed, and the complexity is then expressed only in term of the size of the formula. We will see that every complexity result obtained for combined complexity also holds for the formula complexity: these logics are expressive enough to provide complexity lower bounds for fixed models.

In Theorem 4.1, complexity lower-bounds for model-checking EQ^kCTL and AQ^kCTL are proved with a fixed model. Therefore these results apply also to formula complexity:

Theorem 4.7. Under the structure semantics, the formula-complexity of model checking is Σ_k^P -complete for EQ^kCTL and Π_k^P -complete for AQ^kCTL.

For Q^kCTL, we have the following result:

Theorem 4.8. Under the structure semantics, the formula-complexity is $\Delta_{k+1}^P[O(\log n)]$ -complete for Q^kCTL when considering the *DAG-size* of QCTL formulas. When considering the standard size of formulas, the problem is in $\Delta_{k+1}^P[O(\log n)]$ and $\text{BH}(\Sigma_k^P)$ -hard.

Proof. Membership in $\Delta_{k+1}^P[O(\log n)]$ can be proved using the same algorithm as in the proof of Theorem 4.2, and noticing that its complexity is unchanged when considering the DAG-size of the formula.

In order to prove hardness in $\Delta_{k+1}^P[O(\log n)]$, we again reduce PARITY (Σ_k^P) to a model-checking problem for Q^kCTL over the Kripke structure S with one state and a self-loop as follows: consider an instance \mathcal{I} of PARITY (Σ_k^P) consisting in m instances Ψ_i ($i = 1, \dots, m$) of Σ_k^P SAT. We let $\alpha^1 = \neg\Psi_1$ and $\alpha^{i+1} = (\neg\Psi_{i+1} \wedge \alpha^i) \vee (\Psi_{i+1} \wedge \neg\alpha^i)$. Clearly α^i holds in S iff there is an even number of positive instances in the set $\{\Psi_1, \dots, \Psi_i\}$, so that the instance \mathcal{I} is positive iff α^m holds in S . However, since α_i is duplicated in the definition of α_{i+1} , the reduction is in logarithmic space only if we represent the formula as a DAG.

If we consider the usual notion of size of a formula, one can easily see that formula complexity of Q^kCTL model checking is Σ_k^P -hard and Π_k^P -hard. Actually, as CTL is closed under Boolean combinations, the problem is hard for any level of the Boolean hierarchy $\text{BH}(\Sigma_k^P)$ over Σ_k^P (we refer to [Hem98] for more details about Boolean hierarchies). \square

Finally formula complexity of CTL* model-checking is already PSPACE-hard [Sch03] and any QBF instance can be reduced to a model-checking problem for EQCTL over a fixed structure. This provides the complexity lower-bounds of the following result (the complexity upper-bound come from the general case, see Theorem 4.3):

Theorem 4.9. Under the structure semantics, the formula-complexity is PSPACE-complete for EQ^kCTL*, AQ^kCTL*, Q^kCTL*, EQCTL, QCTL, EQCTL*, and QCTL*.

$k = 0$, this is easy:

$$\text{yardstick}_0^n(s, t) = \mathbf{AG} \left(s \Rightarrow \left((\mathbf{AX})^n t \wedge \bigwedge_{0 \leq k < n} (\mathbf{AX})^k \neg t \right) \right). \quad (4.2)$$

For the subsequent cases, we use propositional quantification to insert a number of intermediary points (labelled with r), at distance $F(k-1, n)$ apart. We then associate with each occurrence of r a counter, encoded in binary (with least significant bit on the right) using a fresh proposition c on the $F(k-1, n)$ cells between the present occurrence of r and the next one. Our global formula then looks as follows:

$$\text{yardstick}_k^n = \exists r. \exists c. (\text{graduation}_k(r, s, t) \wedge \text{counter}_k(c, r, s, t)). \quad (4.3)$$

When $k = 1$, $\text{graduation}_1(r, s, t)$ is rather easy (notice that we allow graduations outside the $[s, t]$ -interval):

$$\text{graduation}_1(r, s, t) = \mathbf{AG} ((s \vee t) \Rightarrow r) \wedge \text{yardstick}_0^n(r, r).$$

As regards the counter, we have to enforce that, between s and t , it has value zero exactly at s and value $2^n - 1$ exactly at t , and that it increases between two consecutive r -delimited intervals:

$$\text{counter}_1(c, r, s, t) = \text{zeros}_1(c, r, s, t) \wedge \text{ones}_1(c, r, s, t) \wedge \text{increment}_1(c, r, s, t)$$

$$\text{zeros}_1(c, r, s, t) = \mathbf{AG} (s \Leftrightarrow (r \wedge \neg c \wedge \mathbf{AX} \mathbf{A}(\neg c \mathbf{U} r)))$$

$$\text{ones}_1(c, r, s, t) = \mathbf{AG} ((r \wedge \mathbf{AX} \mathbf{A}(\neg r \mathbf{U} t)) \Rightarrow \mathbf{A}(c \mathbf{U} t))$$

$$\text{increment}_1(c, r, s, t) = \mathbf{AG} (s \Rightarrow (\mathbf{AG} ((c \Leftrightarrow (\mathbf{AX})^n c) \Leftrightarrow \mathbf{AX} \mathbf{A}(\neg r \mathbf{U} (\neg c \wedge \neg r)))))$$

The first two formulas are easy: zeros_1 requires that s be the only position that can be followed by only zeros until the next occurrence of r ; ones_1 expresses that in the last r -delimited interval before t , c always equals 1. Finally, increment_1 requires that, starting from s , the value of c is changed from one interval to the next one if, and only if, c equals one in all subsequent positions of the first interval. One can check that this correctly encodes the incrementation of the counter. In the end, yardstick_1 is an EQ^1CTL formula.

For any $k \geq 2$, yardstick_k is obtained using similar ideas, with slightly more involved formulas.

$$\text{graduation}_k(r, s, t) = \mathbf{AG} ((s \vee t) \Rightarrow r) \wedge \forall u. \forall v. \left[(\text{delimiters}(u, v) \wedge \text{yardstick}_{k-1}^n(u, v)) \Rightarrow \right. \\ \left. (\mathbf{AG} (u \Rightarrow \mathbf{AF} (r \wedge \mathbf{AF} v)) \wedge \mathbf{AG} ((r \wedge \mathbf{AF} v \wedge \neg \mathbf{AF} u) \Rightarrow \mathbf{AX} \mathbf{A}(\neg r \mathbf{U} v))) \right].$$

Roughly, this states that the labelling with r has to satisfy the constraint that, between any two points u and v at distance $F(k-1, n)$ apart, there must be exactly one r . Notice that formula yardstick_{k-1}^n appears negated in graduation_k . Regarding the counter, formulas zeros_k and ones_k are the same as zeros_1 and ones_1 , respectively. Incrementation is handled using the same trick as for graduation_k :

$$\text{increment}_k(c, r, s, t) = \forall u. \forall v. \left[(\text{delimiters}(u, v) \wedge \text{yardstick}_{k-1}^n(u, v)) \Rightarrow \right. \\ \left. \mathbf{AG} ((s \wedge \mathbf{AF} u) \Rightarrow (\mathbf{AG} (((u \wedge c) \Leftrightarrow \mathbf{AG} (v \Rightarrow c)) \Leftrightarrow (\mathbf{AX} \mathbf{A} \neg r \mathbf{U} (\neg c \wedge \neg r))))) \right]$$

This formula is a mix between increment_1 , in that it uses the same trick of requiring that the value of c is preserved if there is a zero at a lower position, and the labelling with u and v to consider all positions that are at distance $F(k-1, n)$ apart.

Now, since yardstick_{k-1}^n is, by induction hypothesis, in $\text{EQ}^{k-1}\text{CTL}$, formula yardstick_k^n is in EQ^kCTL (notice that yardstick_{k-1}^n appears negated after the universal quantifiers on u and v).

We now explain how we encode the problem whether a word y is accepted by an alternating Turing machine equipped with a tape of size $E(k-1, |y|)$ into an EQ^kCTL model-checking problem. Assume we are given such a Turing machine $\mathcal{M} = \langle Q, q_0, \delta, F \rangle$ on a two-letter alphabet $\Sigma = \{\alpha, \beta\}$, and an input word $y \in \Sigma^n$. An execution of \mathcal{M} on y is encoded as (abstractly) depicted on Fig. 5, with one configuration being encoded as a sequence of cells, and branching occurring only between two consecutive configurations.

With \mathcal{M} , we associate a Kripke structure $\mathcal{S}_{\mathcal{M}} = \langle S, R, \ell \rangle$ where $S = (Q \cup \{\epsilon\}) \times (\Sigma \cup \{\circ\}) \cup \{\#\}$ (where \circ denotes empty cells of the tape and $\#$ will be used to delineate the successive configurations of \mathcal{M}), $R = S \times S$ is the complete transition relation, and ℓ labels each state with its name (hence the initial set of atomic propositions is S). We write s_0 for the state $\#$, where our formula will be evaluated.

The execution tree of $\mathcal{S}_{\mathcal{M}}$ from s_0 contains as branches any word in $s_0 \cdot S^\omega$. We use symbol $\#$ to divide that tree into slices of height $F(k-1, n)$: formula

$$\# \wedge \forall u. \forall v. \left[(\text{delimiters}(u, v) \wedge \text{yardstick}_{k-1}^n(u, v)) \Rightarrow \right. \\ \left. \mathbf{AG}((u \wedge \#) \Rightarrow \mathbf{AX} \mathbf{A}(\neg \#) \mathbf{U}(v \wedge \#)) \right] \quad (4.4)$$

enforces that along any branch, symbol $\#$ occurs at every level multiple of $F(k-1, n)$. Notice that this formula is in $\text{AQ}^{k-1}\text{CTL}$, since the yardstick_{k-1}^n formula is in $\text{EQ}^{k-1}\text{CTL}$. Notice that when $k = 1$, a simpler CTL formula can be used.

Now, not all branches of the execution tree of $\mathcal{S}_{\mathcal{M}}$ are needed in order to represent an accepting execution of \mathcal{M} : only states labelled with $\#$ may have several successors (see Fig. 5). In order to keep track of the relevant branches, we label them with a fresh, existentially-quantified proposition a . The fact that branching only occurs at $\#$ -nodes can be expressed as

$$a \wedge \mathbf{AG}(a \Rightarrow \mathbf{EX} a) \wedge \mathbf{AG} \left[(a \wedge \neg \#) \Rightarrow \bigwedge_{p \neq q \in S} \neg(\mathbf{EX}(a \wedge p) \wedge \mathbf{EX}(a \wedge q)) \right].$$

Enforcing the initial state of the Turing machine (namely, that the tape contains y , Turing machine is in state q_0 and the tape head is on the first letter of y) is straightforward (by expressing that a must label the corresponding sequence of states in the tree). Expressing “local” requirements on the encoding of a configuration (*e.g.* that each configuration contains exactly one state representing a position for the tape head) is straightforward, using the delimiter $\#$. The fact that an accepting state is reached along any a -branch is also easy. It only remains to express that there is a transition linking any configuration with its successor configurations. This can be achieved using similar formulas as formula (4.4), using delimiters u and v to ensure that the content of the tape is preserved and that the tape head has been moved by one position.

In the end, the global formula has an external existential quantification on a , followed by formulas in $\text{AQ}^{k-1}\text{CTL}$ similar to formula (4.4) (of CTL formulas when $k = 1$). The whole formula is then in EQ^kCTL , which concludes the proof that model checking this logic is k -EXPTIME-hard. \square

When using CTL*, the above proof can be improved to handle one more exponential: indeed, using CTL*, formula $\mathbf{yardstick}_0^n(s, t)$ can be made to enforce that the distance between s and t is 2^n . This way, using k quantifier alternations, we can encode the computation of an alternating Turing machine running in space $(k + 1)$ -exponential. In the end:

Theorem 4.11. Model checking EQ^kCTL^* under the tree semantics is $(k + 1)$ -EXPTIME-hard (for positive k).

Algorithms for the tree semantics. We use tree-automata techniques to develop model-checking algorithms for our logics. We recall the definitions and main results of this classical setting, and refer to [MS87, MS95, Tho97, KVV00] for a more detailed presentation.

We begin with defining alternating tree automata, which we will use in the proof. This requires the following definition: the set of *positive boolean formulas* over a finite set P of propositional variables, denoted with $\text{PBF}(P)$, is the set of formulas defined as

$$\text{PBF}(P) \ni \zeta ::= p \mid \zeta \wedge \zeta \mid \zeta \vee \zeta \mid \top \mid \perp$$

where p ranges over P . That a valuation $v: P \rightarrow \{\top, \perp\}$ satisfies a formula in $\text{PBF}(P)$ is defined in the natural way. We abusively say that a subset P' of P satisfies a formula $\varphi \in \text{PBF}(P)$ iff the valuation $\mathbb{1}_{P'}$ (mapping the elements of P' to \top and the elements of $P \setminus P'$ to \perp) satisfies φ . Since negation is not allowed, if $P' \models \varphi$ and $P' \subseteq P''$, then also $P'' \models \varphi$.

Definition 4.12. Let Σ be a finite alphabet. Let $\mathcal{D} \subseteq \mathbb{N}$ be a finite subsets of degrees. An alternating parity \mathcal{D} -tree automaton on Σ , or $\langle \mathcal{D}, \Sigma \rangle$ -APT, is a 4-tuple $\mathcal{A} = \langle Q, q_0, \tau, \Omega \rangle$ where

- Q is a finite set of states,
- $q_0 \in Q$ is the initial state,
- τ is a family of transition functions $(\tau_d)_{d \in \mathcal{D}}$ such that for all $d \in \mathcal{D}$, it holds $\tau_d: Q \times \Sigma \rightarrow \text{PBF}(\{0, \dots, d-1\} \times Q)$,
- $\Omega: Q \rightarrow \{0, \dots, k-1\}$ is a parity acceptance condition.

The size of \mathcal{A} , denoted by $|\mathcal{A}|$, is the number of states in Q . The range k of Ω is the index of \mathcal{A} , denoted by $\text{idx}(\mathcal{A})$.

A non-deterministic \mathcal{D} -tree automaton on Σ , or $\langle \mathcal{D}, \Sigma \rangle$ -NPT, is a $\langle \mathcal{D}, \Sigma \rangle$ -APT where for any $d \in \mathcal{D}$, $q \in Q$ and $\sigma \in \Sigma$, we have: $\tau_d(q, \sigma) = \bigvee_i \left(\bigwedge_{0 \leq c < d} (c, q_{i,c}) \right)$.

We now define the semantics of our tree automata. Notice that contrary to the classical setting, where tree automata are defined to deal with fixed-arity trees, we better use the setting of [KVV00], where the transition function depends on the arity of the node where it is applied. Let $\mathcal{A} = \langle Q, q_0, \tau, \text{Acc} \rangle$ be an $\langle \mathcal{D}, \Sigma \rangle$ -APT, and $\mathcal{T} = \langle T, l_{\mathcal{T}} \rangle$ be a $\langle \Sigma, \mathcal{D} \rangle$ -tree. An *execution tree* of \mathcal{A} on \mathcal{T} is a $T \times Q$ -labelled tree $\mathcal{E} = \langle E, p \rangle$ such that

- $p(\epsilon) = (\epsilon, q_0)$,
- for each node $e \in E$ with $p(e) = (t, q)$ and $d = d_{\mathcal{T}}(t)$, there exists a subset $\xi = \{(c_0, q'_0), \dots, (c_m, q'_m)\} \subseteq \{0, \dots, d-1\} \times Q$ such that $\xi \models \tau_d(q, l_{\mathcal{T}}(t))$, and for $i = 0, \dots, m$, we have $e \cdot i \in E$ and $p(e \cdot i) = (t \cdot c_i, q'_i)$.

We write p_Q for the labelling function restricted to the second component: when $p(e) = (t, q)$, then $p_Q(e) = q$. Given an infinite path $\pi \in \text{Path}_{\mathcal{E}}$ in an execution tree, $p_Q(\pi)$ is the set of

states of visited along π , and $\text{Inf}(p_Q(\pi))$ is the set of states visited infinitely many times. An execution tree is *accepting* if $\min\{\Omega(q) \mid q \in \text{Inf}(p_Q(\pi))\}$ is even for every infinite path $\pi \in \text{Path}_{\mathcal{E}}$. A tree \mathcal{T} is *accepted* by \mathcal{A} if there exists an accepting execution tree of \mathcal{A} on \mathcal{T} .

Deciding whether a given tree is accepted by a tree automaton is decidable. More precisely, given a tree automaton \mathcal{A} and a regular tree \mathcal{T} (*i.e.*, a tree for which there exists a finite Kripke structure \mathcal{S} and a state q such that $\mathcal{T} = \mathcal{T}_{\mathcal{S}}(q)$), the problem whether \mathcal{T} is accepted by \mathcal{A} is decidable. Moreover, given a tree automaton \mathcal{A} , the problem whether \mathcal{A} accepts some tree at all is also decidable⁸, and when the answer is positive, \mathcal{A} accepts a regular tree. We summarise these results in the following theorem:

Theorem 4.13. The problem whether an APT \mathcal{A} with d priorities accepts regular tree \mathcal{T} represented as a Kripke structure \mathcal{S} can be solved in time $O((|\mathcal{A}| \cdot |\mathcal{S}|)^d)$. Checking the emptiness of an APT \mathcal{A} is EXPTIME-complete [Löd13]. Additionally, If \mathcal{A} accepts some infinite tree, then it accepts a regular one [Rab72].

We now recall some standard properties of APT, which we will use later to define our model-checking algorithm for Q^kCTL . First note that the use of Boolean formulae in the transition function makes the treatment of operations like union, intersection and complement easy to handled with APT and there is no cost in term of the size of the resulting automata [MS87]. Now we assume we are given a Kripke structure $\mathcal{S} = \langle Q, R, \ell \rangle$ on a set AP of atomic propositions, and we write \mathcal{D} for the set of degrees in \mathcal{S} .

Lemma 4.14. [KVV00] *Given a CTL formula φ over AP and a set \mathcal{D} of degrees, we can construct a $\langle \mathcal{D}, 2^{\text{AP}} \rangle$ -APT \mathcal{A}_{φ} accepting exactly the 2^{AP} -labelled \mathcal{D} -trees satisfying φ . The automaton \mathcal{A}_{φ} has size linear in the size of φ , and uses a constant number of priorities.*

Sketch of proof. We only describe the construction, and refer to [KVV00] for a detailed proof of the result. W.l.o.g. we assume that negations in φ are followed by atomic propositions; This might require adding the two extra modalities **EW** and **AW**, which satisfy the following equivalences:

$$\begin{aligned} \neg(\mathbf{E}\varphi \mathbf{U} \psi) &\equiv \mathbf{A}(\neg\psi) \mathbf{W}(\neg\psi \wedge \neg\varphi) \\ \neg(\mathbf{A}\varphi \mathbf{U} \psi) &\equiv \mathbf{E}(\neg\psi) \mathbf{W}(\neg\psi \wedge \neg\varphi) \end{aligned}$$

The automaton $\mathcal{A}_{\varphi} = \langle Q_{\varphi}, q_0, \tau, \Omega_{\varphi} \rangle$ is defined as follows:

- Q_{φ} is the set of state subformulas (not including \top and \perp),
- the initial state q_0 is φ ,
- given a degree $d \in \mathcal{D}$, $\psi \in Q_{\varphi}$ and $\sigma \in 2^{\text{AP}}$, we define $\tau_d(\psi, \sigma)$ as follows:

$$\begin{aligned} \tau_d(P, \sigma) &= \begin{cases} \top & \text{if } P \in \sigma \\ \perp & \text{otherwise} \end{cases} & \tau_d(\neg P, \sigma) &= \begin{cases} \perp & \text{if } P \notin \sigma \\ \top & \text{otherwise} \end{cases} \\ \tau_d(\psi_1 \wedge \psi_2) &= \tau_d(\psi_1) \wedge \tau_d(\psi_2) & \tau_d(\psi_1 \vee \psi_2) &= \tau_d(\psi_1) \vee \tau_d(\psi_2) \\ \tau_d(\mathbf{E}\mathbf{X}\psi, \sigma) &= \bigvee_{0 \leq c < d} (c, \psi) & \tau_d(\mathbf{A}\mathbf{X}\psi, \sigma) &= \bigwedge_{0 \leq c < d} (c, \psi) \end{aligned}$$

$$\tau_d(\mathbf{E}\psi_1 \mathbf{U} \psi_2, \sigma) = \tau_d(\psi_2, \sigma) \vee \left(\tau_d(\psi_1, \sigma) \wedge \bigvee_{0 \leq c < d} (c, \mathbf{E}\psi_1 \mathbf{U} \psi_2) \right)$$

⁸Note that for an APT, emptiness checking and universality checking have the same complexity because building the complement automaton can be done efficiently.

$$\tau_d(\mathbf{E}\psi_1 \mathbf{W} \Psi_2, \sigma) = \tau_d(\psi_2, \sigma) \vee \left(\tau_d(\psi_1, \sigma) \wedge \bigvee_{0 \leq c < d} (c, \mathbf{E}\psi_1 \mathbf{W} \psi_2) \right)$$

$$\tau_d(\mathbf{A}\psi_1 \mathbf{U} \Psi_2, \sigma) = \tau_d(\psi_2, \sigma) \vee \left(\tau_d(\psi_1, \sigma) \wedge \bigwedge_{0 \leq c < d} (c, \mathbf{A}\psi_1 \mathbf{U} \psi_2) \right)$$

$$\tau_d(\mathbf{A}\psi_1 \mathbf{W} \Psi_2, \sigma) = \tau_d(\psi_2, \sigma) \vee \left(\tau_d(\psi_1, \sigma) \wedge \bigwedge_{0 \leq c < d} (c, \mathbf{A}\psi_1 \mathbf{W} \psi_2) \right)$$

- the acceptance condition is defined as

$$\begin{aligned} \Omega_\varphi(\mathbf{E}\psi_1 \mathbf{U} \psi_2) &= \Omega_\varphi(\mathbf{A}\psi_1 \mathbf{U} \psi_2) = 1 \\ \Omega_\varphi(\mathbf{E}\psi_1 \mathbf{W} \psi_2) &= \Omega_\varphi(\mathbf{A}\psi_1 \mathbf{W} \psi_2) = 2 \end{aligned}$$

Note that the definition of Ω_φ for other nodes (*i.e.*, boolean combinations) is not needed as these nodes cannot be visited infinitely often along a branch. \square

QCTL quantifications over atomic propositions will be handled with the projection operation on tree automata. To this aim, we will use the following lemma (notice that it requires *non-deterministic* tree automata as input):

Lemma 4.15. [MS85] *Let \mathcal{A} be a $\langle \mathcal{D}, 2^{\text{AP}} \rangle$ -NPT, with $\text{AP} = \text{AP}_1 \cup \text{AP}_2$. For all $i \in \{1, 2\}$, we can build a $\langle \mathcal{D}, 2^{\text{AP}} \rangle$ -NPT \mathcal{B}_i such that, for any 2^{AP} -labelled \mathcal{D} -tree \mathcal{T} , it holds: $\mathcal{T} \in \mathcal{L}(\mathcal{B}_i)$ iff $\exists \mathcal{T}' \in \mathcal{L}(\mathcal{A}). \mathcal{T} \equiv_{\text{AP}_i} \mathcal{T}'$. The size and index of \mathcal{B}_i are those of \mathcal{A} .*

In order to use this result, we will have to apply the *simulation theorem*, which allows for turning APTs into NPTs. Having varying degrees does not change the result (for example, one can adapt the proofs of Lemma 3.9 and Theorem 3.10 in [Löd13] in order to get the result in our extended setting):

Lemma 4.16. [MS95] *Let \mathcal{A} be a $\langle \mathcal{D}, 2^{\text{AP}} \rangle$ -APT. We can build an $\langle \mathcal{D}, 2^{\text{AP}} \rangle$ -NPT \mathcal{N} accepting the same language as \mathcal{A} , and such that $|\mathcal{N}| \in 2^{O(|\mathcal{A}| \cdot \text{idx}(\mathcal{A}) \cdot \log(|\mathcal{A}| \cdot \text{idx}(\mathcal{A})))}$ and $\text{idx}(\mathcal{N}) \in O(|\mathcal{A}| \cdot \text{idx}(\mathcal{A}))$.*

Now we are ready to describe the construction of the automaton for Q^kCTL :

Theorem 4.17. *Given a Q^kCTL formula φ over AP and a set \mathcal{D} of degrees, we can construct a $\langle \mathcal{D}, 2^{\text{AP}} \rangle$ -APT \mathcal{A}_φ accepting exactly the 2^{AP} -labelled \mathcal{D} -trees satisfying φ . The automaton \mathcal{A}_φ has size k -exponential and number of priorities $(k - 1)$ -exponential in the size of φ .*

Proof. We proceed by induction over k .

- if $\varphi \in \text{Q}^1\text{CTL}$, then φ is of the form $\Phi[(\psi_i)_{1 \leq i \leq m}]$ where Φ is a CTL formula and $(\psi_i)_{1 \leq i \leq m}$ are EQ^1CTL formulas. We handle each ψ_i separately. Assume that $\psi_i = \exists p_1 \dots \exists p_l. \psi'$ with $\psi' \in \text{CTL}$. From Lemma 4.14, one can build an APT $\mathcal{A}_{\psi'}$ recognizing the \mathcal{D} -trees satisfying ψ' ; moreover, $|\mathcal{A}_{\psi'}|$ is in $O(|\psi'|)$ and $\text{idx}(\mathcal{A}_{\psi'}) = 2$. Applying Lemma 4.16, we get an equivalent NPT $\mathcal{N}_{\psi'}$ whose size is in $2^{O(|\psi'| \cdot \log(|\psi'|))}$ and number of priorities is in $O(|\psi'|)$. Applying Lemma 4.15 (to $\mathcal{N}_{\psi'}$ and for atomic propositions p_1, \dots, p_l), we get an NPT $\mathcal{B}_{\psi_i} = \langle Q_{\psi_i}, q_0^{\psi_i}, \tau_{\psi_i}, \Omega_{\psi_i} \rangle$ recognizing the models of ψ_i . The size of \mathcal{B}_{ψ_i} is in $2^{O(|\psi_i| \cdot \log(|\psi_i|))}$, and its number of priorities is in $O(|\psi_i|)$.

Now to complete the construction it remains to construct the final automaton $\mathcal{A}_\varphi = \langle Q, q_0, \tau, \Omega \rangle$. It is based on the APT associated with the CTL context $\Phi[-]$ (w.r.t. Lemma 4.14) and the different NPTs built for the subformulas ψ_i . Indeed the transition

function τ follows the rules of Lemma 4.14 for $\Phi[-]$ and we just add the two following rules to deal with the subformulae ψ_i and their negations⁹:

– $\tau(\psi_i, \sigma) = \tau_{\psi_i}(q_0^{\psi_i}, \sigma)$, and

– $\tau(\neg\psi_i, \sigma) = \tau_{\bar{\psi}_i}(q_0^{\bar{\psi}_i}, \sigma)$ where $\tau_{\bar{\psi}_i}$ is the transition function of $\overline{\mathcal{B}_{\psi_i}}$ (the dual of \mathcal{B}_{ψ_i}).

Therefore \mathcal{A}_φ is an APT whose size is in $2^{O(|\varphi| \cdot \log(|\varphi|))}$ and its number of priorities is in $O(|\varphi|)$.

- if $\varphi \in \mathbf{Q}^k\text{CTL}$ with $k > 1$, the construction follows almost the same steps as in the base case. Here φ is of the form $\Phi[(\psi_i)_{1 \leq i \leq m}]$, where Φ is a CTL formula and each ψ_i belongs to $\text{EQ}^1\text{CTL}[\mathbf{Q}^{k-1}\text{CTL}]$, *i.e.*, is of the form $\exists p_1 \dots \exists p_l. \psi'$ with $\psi' \in \mathbf{Q}^{k-1}\text{CTL}$.

From the induction hypothesis, we can build an APT $\mathcal{A}'_{\psi'}$ recognizing the \mathcal{D} -trees satisfying ψ' , and whose size is $(k-1)$ -exponential and whose number d of priorities is $(k-2)$ -exponential in $|\psi'|$. Applying Lemma 4.16, we get an equivalent NPT $\mathcal{N}_{\psi'}$ whose size is k -exponential in $|\psi'|$ (precisely in $2^{O(|\mathcal{A}'_{\psi'}| \cdot \log(|\mathcal{A}'_{\psi'}| \cdot d))}$) and whose number of priorities is $|\mathcal{A}'_{\psi'}| \cdot d$, *i.e.*, $(k-1)$ -exponential in $|\psi'|$. From Lemma 4.15 (applied to $\mathcal{N}_{\psi'}$ and for propositions p_1, \dots, p_l), we get an NPT \mathcal{B}_{ψ_i} recognizing the models of ψ_i . Again the size and number of priorities of \mathcal{B}_{ψ_i} are identical to those of $\mathcal{N}_{\psi'}$.

Now we finish the construction as before in combining these NPTs with the APT provided by the CTL context Φ . This provides an APT \mathcal{A}_φ whose size is k -exponential in $|\varphi|$ and its number of priorities is $k-1$ -exponential. \square

Combining this with the result of Theorem 4.13, we get our final result for $\mathbf{Q}^k\text{CTL}$:

Theorem 4.18. Model checking $\mathbf{Q}^k\text{CTL}$ under the tree semantics is in $k\text{-EXPTIME}$ (for positive k).

The proof is easily adapted to the quantified extensions of CTL^* :

Theorem 4.19. Model checking $\mathbf{Q}^k\text{CTL}^*$ under the tree semantics is in $(k+1)\text{-EXPTIME}$ (for positive k).

Proof. The proof proceeds along the same lines as in the proof of Theorem 4.17. However, we have to build automata for a CTL^* formula in the base case, so that the automaton for a $\mathbf{Q}^1\text{CTL}^*$ formula has size 2-exponential and number of priorities exponential.

During the induction step, we consider automata for $\mathbf{Q}^{k-1}\text{CTL}^*$ formulas, apply the simulation theorem and projection, and combine them with an (exponential-size) automaton for a CTL^* formula [KVVW00]. One can easily see that the resulting automaton has size $k+1$ -exponential, and number of priorities k -exponential. Theorem 4.13 then entails the result. \square

From Theorems 4.10, 4.11, 4.18 and 4.19, we obtain:

Corollary 4.20. *Under the tree semantics, for any $k > 0$, model checking EQ^kCTL , AQ^kCTL and $\mathbf{Q}^k\text{CTL}$ is $k\text{-EXPTIME}$ -complete, and model checking EQ^kCTL^* , AQ^kCTL^* and $\mathbf{Q}^k\text{CTL}^*$ is $(k+1)\text{-EXPTIME}$ -complete.*

It follows:

Theorem 4.21. Model checking EQCTL , QCTL , EQCTL^* and QCTL^* under the tree semantics is TOWER -complete.

⁹Remember the construction for CTL formulae assumes that negations precede atomic propositions.

4.2.1. *Program-complexity.* When fixing the formula, the problem becomes much easier (in terms of its theoretical complexity): given φ , one can build an automaton \mathcal{A}_φ^2 such that for every Kripke structure \mathcal{S} , deciding whether $\mathcal{S}, q \models_t \varphi$ is equivalent to deciding whether the unfolding of a *variant* of \mathcal{S} is accepted by \mathcal{A}_φ^2 which can be done in polynomial time (because $|\mathcal{A}_\varphi^2|$ is assumed to have constant size).

First consider a finite Kripke structure $\mathcal{S} = \langle Q, R, \ell \rangle$. We define $\mathcal{S}_2 = \langle Q_2, R_2, \ell_2 \rangle$ with $Q_2 = Q \cup Q_{int}$, $\ell_2(q) = \ell(q)$ for $q \in Q$ and $\ell_2(q) = \{\mathbf{p}_{int}\}$ for $q \in Q_{int}$ (\mathbf{p}_{int} is a fresh atomic proposition). Finally Q_{int} and R_2 are defined in order to mimic \mathcal{S} -transitions with nodes of degree 2: if q_1, \dots, q_k (with $k > 1$) are the k successors of q in \mathcal{S} , we have a transition $(q, q_\epsilon) \in R_2$ and q_ϵ is the root of a complete binary tree with k leaves q_1, \dots, q_k . For this, we need $k - 1$ internal nodes (including q_ϵ). If q has a unique successor q' in \mathcal{S} , we just add the transition (q, q') to R_2 . Thus the size of Q_{int} is $\sum_{q \in Q} d(x) - 1$.

Now consider a QCTL formula Φ (that does not use \mathbf{p}_{int}). Let $\widehat{\Phi}$ be the QCTL formula defined as follows:

$$\begin{array}{ll} \widehat{\mathbf{E}\varphi \mathbf{U}\psi} = \mathbf{E}(\mathbf{p}_{int} \vee \widehat{\varphi}) \mathbf{U}(\neg \mathbf{p}_{int} \wedge \widehat{\psi}) & \widehat{\varphi \wedge \psi} = \widehat{\varphi} \wedge \widehat{\psi} \\ \widehat{\mathbf{A}\varphi \mathbf{U}\psi} = \mathbf{A}(\mathbf{p}_{int} \vee \widehat{\varphi}) \mathbf{U}(\neg \mathbf{p}_{int} \wedge \widehat{\psi}) & \widehat{\varphi \vee \psi} = \widehat{\varphi} \vee \widehat{\psi} \\ \widehat{\mathbf{E}\mathbf{X}\varphi} = \mathbf{E}\mathbf{X} \mathbf{E}\mathbf{p}_{int} \mathbf{U}(\neg \mathbf{p}_{int} \wedge \widehat{\varphi}) & \widehat{P} = P \\ \widehat{\exists P. \varphi} = \exists P. \widehat{\varphi} & \widehat{\neg \varphi} = \neg \widehat{\varphi} \end{array}$$

The following lemma which relates the truth value of Φ on \mathcal{S} with that of $\widehat{\Phi}$ on \mathcal{S}_2 :

Lemma 4.22. *For any Kripke structure $\mathcal{S} = \langle Q, R, \ell \rangle$, for any QCTL formula Φ , and for any $q \in Q$, we have $\mathcal{S}, q \models_t \Phi$ if, and only if, $\mathcal{S}_2, q \models_t \widehat{\Phi}$.*

Proof. The proof is done by structural induction over Φ . □

Theorem 4.23. Under the tree semantics, the program-complexity of model-checking is PTIME-complete for all our fragments between EQ¹CTL and AQ¹CTL to QCTL*.

Proof. Let Ψ be a QCTL* formula. By Proposition 3.8, we can build an equivalent QCTL formula Φ . Now consider $\widehat{\Phi}$ as above. Let $\mathcal{A}_{\widehat{\Phi}}$ be the alternating tree automaton built for proving Theorem 4.18 when $\mathcal{D} = \{1, 2\}$. It is easy to see that $\mathcal{A}_{\widehat{\Phi}}$ recognizes the 2^{AP} -labeled trees having nodes with degrees in $\{1, 2\}$ and satisfying $\widehat{\Phi}$.

Now consider a model-checking instance $\mathcal{S}, q \models_t \Phi$, it clearly reduces to $\mathcal{S}_2, q \models_t \widehat{\Phi}$ and then to $\mathcal{T}_{\mathcal{S}_2}(q) \in \mathcal{L}(\mathcal{A}_{\widehat{\Phi}})$. This last problem can be solved in polynomial time (in $|\mathcal{S}_2|$) when the size of $\mathcal{A}_{\widehat{\Phi}}$ is assumed to be fixed: following Theorems 4.13 and 4.17, the polynomial has degree $(k - 1)$ -exponential in $|\Phi|$ when $\Phi \in \text{Q}^k\text{CTL}$.

We now prove hardness in PTIME, by reducing the **CIRCUIT-VALUE** problem (actually a simplified version of it, with no negation [GJ79]) to a model-checking problem for a fixed formula in EQ¹CTL. Pick a monotone circuit with fan-out 2 (*i.e.*, an acyclic Kripke structure in which all states are labelled with \bigvee or \bigwedge and have two outgoing edges, except for two “terminal” states respectively labelled with 0 and 1, and which only have a self-loop as outgoing edges). The value of the circuit is the value of its initial node, defined in the natural way. See Fig. 6 for an instance of **CIRCUIT-VALUE** (self-loops omitted) whose initial node evaluates to 1.

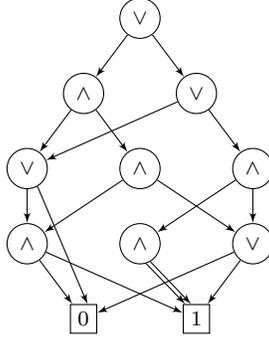


Fig. 6: A monotone boolean circuit

Consider the CTL formula

$$\varphi = \mathbf{AG} \left[(1 \Rightarrow p) \wedge (0 \Rightarrow \neg p) \wedge (\bigwedge \Rightarrow (p \Leftrightarrow \mathbf{AX} p)) \wedge (\bigvee \Rightarrow (p \Leftrightarrow \mathbf{EX} p)) \right]$$

and choose a labelling of the circuit with a new atomic proposition p satisfying φ (in the initial state). By an easy induction, we get that a state is labelled with p if, and only if, its value is 1. As a consequence, the circuit has value one if, and only if, the initial state of its associated Kripke structure satisfies the fixed EQ^1CTL formula $\exists p. (p \wedge \varphi)$. The same equivalence holds for the AQ^1CTL formula $\forall p. (\varphi \Rightarrow p)$, as it is easily seen that only one p -labelling satisfies φ . \square

4.2.2. *Formula-complexity.* The reductions used in general cases can be made to work with a fixed model. Thus we have:

Theorem 4.24. Under the tree semantics, the formula-complexity of model-checking is k -EXPTIME-complete for EQ^kCTL and Q^kCTL , and it is $(k+1)$ -EXPTIME-complete for EQ^kCTL^* and Q^kCTL^* , for positive k .

Proof. Membership in k -EXPTIME (resp. in $(k+1)$ -EXPTIME) follows from the general case. We can adapt the hardness proof for EQ^kCTL (Theorem 4.10) to work with a fixed Kripke structure \mathcal{S} . Let \mathcal{M} be a $(k-1)$ -EXPSPACE alternating Turing machine $\langle Q, q_0, \delta, F \rangle$ with $Q = \{q_1, \dots, q_l\}$ and let y be an input word over Σ . Assume (w.l.o.g.) that $\Sigma = \{\alpha, \beta\}$. In order to reuse the reduction performed for the general case, we need to encode the unfolding of $\mathcal{S}_{\mathcal{M}}$ (defined in the proof of Theorem 4.10) with an unfolding of some fixed Kripke structure \mathcal{S} (hence with a fixed arity and a fixed labelling). For this we use the Kripke structure $\mathcal{S} = \langle S, R, \ell \rangle$ with $S = \{s_0, s_1\}$, $R = \{(s_0, s_0), (s_0, s_1), (s_1, s_0), (s_1, s_1)\}$ and $\ell(s_i) = \emptyset$ for $i = 0, 1$: its unfolding is a binary tree and it contains no atomic proposition. We use intermediary states (labeled by \mathbf{p}_{int}) to encode the arity of the states of $\mathcal{S}_{\mathcal{M}}$ (i.e., every “regular” node s will be the root of some finite binary tree labeled with \mathbf{p}_{int} and whose leaves correspond to all possible successors of s in $K_{\mathcal{M}}$), and we use an additional existential propositional quantification to describe the states (content of the tape, control state, separator, ...). Let AP be the set $\{q_1, \dots, q_l, \alpha, \beta, \#\}$. The EQ^kCTL specification used to describe the existence of an accepting run of \mathcal{M} over y is the following formula Φ where

$\Phi_{\mathcal{M},y}$ is a slight variant of the EQ^kCTL formula used in the proof of Theorem 4.10 and Φ_b is a CTL formula described below:

$$\Phi = \exists \mathbf{p}_{int} \exists q_0 \dots \exists q_l. \exists \alpha. \exists \beta. \exists \# . (\Phi_b \wedge \Phi_{\mathcal{M},y})$$

Formula Φ_b describes the labelling of the binary tree with $\{\mathbf{p}_{int}, p_{q_0}, \dots, p_{q_l}, \alpha, \beta, \#\}$:

- $\neg \mathbf{p}_{int}$ is true initially,
- every state satisfying $\neg \mathbf{p}_{int}$ satisfies exactly one proposition in $\{\alpha, \beta, \#\}$ and at most one in $\{p_{q_0}, \dots, p_{q_l}\}$ can be associated with α or β ,
- every $\neg \mathbf{p}_{int}$ state can reach (along a \mathbf{p}_{int} -labeled path) every $\neg \mathbf{p}_{int}$ state with one the following labellings:

$$\{\{\alpha\}, \{\beta\}, \{\#\}, \{\alpha, q_0\}, \dots, \{\alpha, q_l\}, \{\beta, q_0\}, \dots, \{\beta, q_l\}\}$$

- and every \mathbf{p}_{int} state satisfies $\mathbf{AF} \neg \mathbf{p}_{int}$ (every intermediary state is inevitably followed by a regular state).

In $\Phi_{\mathcal{M},y}$, the only change we have to do is for $\text{yardstick}_0^n(s, t)$: we replace \mathbf{AX}^n by a sequence of n nested formulae of the form “ $\mathbf{AX}(\mathbf{Ap}_{int} \mathbf{U}(\neg \mathbf{p}_{int} \wedge \dots))$ ” in order to find the n -th cell without considering intermediary states. Clearly Φ is still in EQ^kCTL and we have $y \in \mathcal{L}(\mathcal{M})$ iff $K, s_0 \models \Phi$.

The same approach can be done for EQ^kCTL^* and Q^kCTL^* with an extra exponential level due to the ability of CTL^* to describe a run of length 2^n . \square

As a consequence of the previous result, we have:

Theorem 4.25. Under the tree semantics, the formula-complexity of model-checking is TOWER-complete for EQCTL , QCTL , EQCTL^* and QCTL^* .

5. SATISFIABILITY

We now address the satisfiability problem for our two semantics: given formula φ in QCTL^* (or a fragment thereof), does there exist a finite Kripke structure \mathcal{S} such that $\mathcal{S}, q \models \varphi$ for some state q of \mathcal{S} ?

Structure semantics. As a corollary of our Prop. 3.4 and of the undecidability of MSO over finite graphs [See76], we have

Theorem 5.1. For the structure semantics, satisfiability of QCTL and QCTL^* is undecidable.

Notice that satisfiability of QCTL and QCTL^* is clearly recursively enumerable: the finite Kripke structures can be enumerated and checked against the considered formula, using our model-checking algorithms.

Similar undecidability results were already proved by French: in [Fre01], he proved that satisfiability of QCTL (more precisely, AQ^1CTL) is undecidable over *infinite-state* Kripke structure; in [Fre03], he proved that satisfiability of QLTL over finite-state Kripke structures is undecidable (from which we get the result for AQ^1CTL^*).

To complete the picture, we prove the following results:

Theorem 5.2. Under the structure semantics, the satisfiability problem is undecidable for AQ^kCTL (when $k \geq 1$) and EQ^kCTL (when $k \geq 2$).

Notice that satisfiability for EQ^1CTL and EQ^1CTL^* under the structure semantics is nothing but satisfiability of CTL and CTL^* , respectively: if φ is in CTL^* , $\exists p.\varphi$ is satisfiable if, and only if, φ is.

The core of the reduction is an EQ^2CTL formula characterising those Kripke structures that are finite grids¹⁰. We say that a Kripke structure is a finite grid when it is *linear* (i.e., the underlying graph (Q, R) is isomorphic to the graph $L_m = ([1, m], \{(i, i + 1) \mid 1 \leq i \leq m - 1\} \cup \{(m, m)\})$ for some m) or when it is isomorphic to a product $L_m \times L_n$ for some integers m and n . The outermost existential quantifiers of our EQ^2CTL formula can then be handled together with the existential quantification on the Kripke structure (because we deal with satisfiability), so that our reduction will work for AQ^1CTL .

The main idea behind our formula is a labelling of every other *horizontal* (resp. *vertical*) *line* of the structures with atomic propositions h (resp. v). Using universal quantification, we impose alternation between lines of h -states and lines of $\neg h$ -states using the following formula of the following form, enforcing small *squares* as depicted on Fig. 7:

$$\forall \gamma. \mathbf{AG} \left[(h \wedge \mathbf{EX} (h \wedge \mathbf{EX} (\neg h \wedge \gamma))) \Rightarrow \mathbf{EX} (\neg h \wedge \mathbf{EX} (\neg h \wedge \gamma)) \right].$$

The full formula and a proof that it correctly characterises finite grids are given in Appendix B.

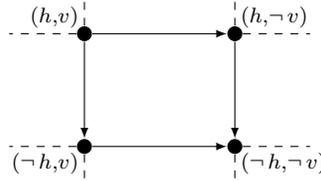


Fig. 7: The horizontal and vertical lines of a grid

Proof of Theorem 5.2. We consider the following tiling problem: given a finite set of different tiles, is it possible to tile any finite grid? This problem is easily shown undecidable, e.g. by encoding the computations of a Turing machine.

Now, we encode the dual problem, expressing the existence of a finite grid for which any tiling has a local mismatch. This can be achieved using the formula for grids and an additional (universal) quantification for placing tiles on that grid, for which a CTL formula will express the existence of a position where neighboring tiles do not match. \square

Tree semantics. We prove the following result:

Theorem 5.3. Under the tree semantics, satisfiability is $(k + 1)$ -EXPTIME-complete for AQ^kCTL and Q^kCTL , and it is k -EXPTIME-complete for EQ^kCTL (for positive k).

Proof. We first prove $(k + 1)$ -EXPTIME-hardness for AQ^kCTL satisfiability. This is a straightforward adaptation of the proof of Theorem 4.10: in that proof, from an input word y and an alternating Turing machine \mathcal{M} equipped with a tape of size $E(k, |y|)$, we built an $\text{EQ}^{k+1}\text{CTL}$

¹⁰This follows the same ideas as in [Fre01] for proving undecidability of AQ^1CTL over *infinite* Kripke structures. However, the restriction to finite grids comes with many more technicalities (not all states will have two successors, to begin with).

formula of the form $\exists a. \Phi$, where Φ is in AQ^kCTL , which is true in the Kripke structure $\mathcal{S}_{\mathcal{M}}$ if, and only if, \mathcal{M} accepts the input word y . Now we claim that it is also equivalent to the satisfiability of $\Phi \wedge \chi_{\mathcal{M}}$, where $\chi_{\mathcal{M}}$ is an AQ^1CTL -formula that encodes the Kripke structure $\mathcal{S}_{\mathcal{M}}$. Letting $\text{AP} = (Q \cup \{\epsilon\}) \times (\Sigma \cup \{\circ\}) \cup \{\#\}$, formula $\chi_{\mathcal{M}}$ expresses the fact that $\#$ is the initial state, and that any reachable state has exactly one successor labelled with p , for each $p \in \text{AP}$ (see the proof of Theorem 4.10 for the definition of $\mathcal{S}_{\mathcal{M}}$). More precisely, $\chi_{\mathcal{M}}$ can be written as

$$\# \wedge \mathbf{AG} \left[\bigvee_{p \in \text{AP}} \left(p \wedge \bigwedge_{q \in \text{AP} \setminus \{p\}} \neg q \right) \right] \wedge \forall z. \bigwedge_{p \in \text{AP}} \mathbf{AG} \left[\mathbf{EX} p \wedge (\mathbf{EX} (p \wedge z) \Rightarrow \mathbf{AX} (p \Rightarrow z)) \right].$$

We write $\Phi_{\mathcal{M}}$ for $\Phi \wedge \chi_{\mathcal{M}}$.

First assume that $\exists a. \Phi$ is true in $\mathcal{S}_{\mathcal{M}}$ (for the tree semantics). Then also $\exists a. (\Phi_{\mathcal{M}})$ is true in that Kripke structure (notice that $\chi_{\mathcal{M}}$ does not depend on a). Applying Theorem 4.17 to $\Phi_{\mathcal{M}}$ for $\mathcal{D} = \{|\text{AP}|\}$, we know that there is an alternating tree automaton $\mathcal{A}_{\Phi_{\mathcal{M}}}$ accepting exactly the \mathcal{D} -trees in which $\Phi_{\mathcal{M}}$ holds. Since $\exists a. (\Phi_{\mathcal{M}})$ is true in $\mathcal{S}_{\mathcal{M}}$, we know that \mathcal{A}_{Φ} accepts at least one tree. From Theorem 4.13, it accepts a regular tree. As a consequence, there is a finite-state Kripke structure where $\Phi_{\mathcal{M}}$ holds. Conversely, if $\Phi_{\mathcal{M}}$ is satisfiable, then $\exists a. \Phi_{\mathcal{M}}$ also is (for the tree semantics). Pick a Kripke structure \mathcal{S} satisfying $\exists a. \Phi_{\mathcal{M}}$. Then \mathcal{S} satisfies $\chi_{\mathcal{M}}$, so that the unwindings of \mathcal{S} and of $\mathcal{S}_{\mathcal{M}}$ are the same, and state $\#$ of $\mathcal{S}_{\mathcal{M}}$ satisfies $\exists a. \Phi$.

This proves the lower bound for AQ^kCTL and Q^kCTL . As EQ^kCTL contains $\text{AQ}^{k-1}\text{CTL}$, it also gives the complexity lower-bound for EQ^kCTL .

Now we prove membership in $(k+1)\text{-EXPTIME}$ for Q^kCTL satisfiability. Let Φ be a Q^kCTL formula. Let $\widehat{\Phi}$ be the corresponding formula over $\{1, 2\}$ -trees, as defined at Section 4.2.1. Let Φ_2 be the CTL formula $\neg \mathbf{p}_{int} \wedge \mathbf{AG} \mathbf{AF} \neg \mathbf{p}_{int}$. The following lemma allows us to reduce the satisfiability problem for Φ to the satisfiability problem for $\widehat{\Phi} \wedge \Phi_2$ for $\{1, 2\}$ -trees:

Lemma 5.4. *There exists a Kripke structure \mathcal{S} with a state q such that $\mathcal{S}, q \models_t \Phi$ if, and only if, there exists a regular $\{1, 2\}$ -tree \mathcal{T} such that $\mathcal{T}, \varepsilon \models_s \widehat{\Phi} \wedge \Phi_2$.*

Proof. If $\mathcal{S}, q \models_t \Phi$ then $\mathcal{S}_2, q \models_t \widehat{\Phi}$ (see Section 4.2.1). Since $\mathcal{S}_2, q \models_t \Phi_2$, the result follows.

We now prove the converse direction. Assume $\mathcal{T}, \varepsilon \models_s \widehat{\Phi} \wedge \Phi_2$. Note that the root satisfies $\neg \mathbf{p}_{int}$ (enforced by Φ_2). Now consider the tree \mathcal{T}' where the intermediary state (those labeled with \mathbf{p}_{int}) has been removed, and replaced with direct transitions from their father node to their child nodes. This new tree is still regular and it can be easily proved to satisfy Φ (by construction of $\widehat{\Phi}$). \square

From the previous result, it remains to build an $\langle \{1, 2\}, 2^{\text{AP}} \rangle$ -APT for $\widehat{\Phi} \wedge \Phi_2$, as described in Theorem 4.17. The size of this automaton is k -exponential in Φ . Checking emptiness can be done in time $(k+1)$ -exponential.

Now consider the case of some EQ^kCTL formula Φ . Let Φ' be the $\text{AQ}^{k-1}\text{CTL}$ formula obtained from Φ by removing the outermost existential block of quantifiers. Clearly Φ' is satisfiable iff Φ is: satisfiability implicitly uses an existential quantification over atomic propositions, which can include the first block of existential quantifications. This provides us with a $k\text{-EXPTIME}$ algorithm for EQ^kCTL satisfiability. \square

The result is lifted to fragments of QCTL^* as follows:

Theorem 5.5. Under the tree semantics, satisfiability is $(k+2)$ -EXPTIME-complete for AQ^kCTL^* and Q^kCTL^* , and it is $(k+1)$ -EXPTIME-complete for EQ^kCTL^* .

Proof. As explained above, using CTL^* allows us to gain an exponential level in the reduction: $\text{yardstick}_0^n(s, t)$ can enforce that the distance between s and t is 2^n . \square

Finally we have:

Corollary 5.6. Under the tree semantics, satisfiability is TOWER-complete for EQCTL , QCTL , EQCTL^* and QCTL^* .

Remark 5.7. In our definition, the satisfiability problem for the tree semantics asks for the existence of a finite Kripke structure. Another satisfiability problem can be considered, asking for the existence of a labeled tree T satisfying the input formula. This version of the problem is also decidable, and is equivalent to the previous one. Indeed, assume that such a T exists, then there is a finitely branching tree T' satisfying Φ , because Φ is equivalent to some MSO formula and MSO has the finite-branching property. As $T' \models_s \Phi$, we clearly have that $\widehat{\Phi} \wedge \Phi_2$ is satisfiable by some $\{1, 2\}$ -tree; thanks to Lemma 5.4, we get that there exists a finite Kripke structure satisfying Φ for the tree semantics.

6. CONCLUSIONS

	satisfiability	model checking	formula-compl.	program-compl.
EQ^kCTL	EXPTIME-c. for EQ^1CTL 2-EXPTIME-c. for EQ^1CTL^* Undecidable otherwise (Th. 5.2)	Σ_k^P -complete (Th. 4.1, 4.4, 4.7)		
Q^kCTL		$\Delta_{k+1}^P[O(\log n)]$ -complete ¹¹ (Th. 4.2, 4.5, 4.8)		
EQ^kCTL^*		PSPACE-complete		Σ_k^P -complete (Th. 4.4)
Q^kCTL^*				$\Delta_{k+1}^P[O(\log n)]$ -c.(Th. 4.5)
(E)QCTL (E)QCTL*				PH-hard, in PSPACE (Th. 4.6)

Table 1: Results for the structure semantics

	satisfiability	model checking	formula-compl.	program-compl.
EQ^kCTL	k-EXPTIME-c.(Th. 5.3)	k-EXPTIME-complete (Cor. 4.20, Th. 4.24)		PTIME-complete (Th. 4.23)
Q^kCTL	$(k+1)$ -EXPTIME-c.(Th. 5.3)			
EQ^kCTL^*	$(k+1)$ -EXPTIME-c.(Th. 5.5)	$(k+1)$ -EXPTIME-complete (Cor. 4.20, Th. 4.24)		
Q^kCTL^*	$(k+2)$ -EXPTIME-c.(Th. 5.5)			
(E)QCTL (E)QCTL*	TOWER-complete (Th. 4.21, 4.25, Cor. 5.6)			

Table 2: Results for the tree semantics.

¹¹Hardness in $\Delta_{k+1}^P[O(\log n)]$ for the formula-complexity of Q^kCTL model checking only holds when considering the DAG-size of formulas. With the standard size, the problem is $\text{BH}(\Sigma_k^P)$ -hard, and in $\Delta_{k+1}^P[O(\log n)]$.

While it was introduced thirty years ago, the extension of CTL with propositional quantifiers had never been studied in details. Motivated by a correspondence with temporal logics for multi-player games, we have proposed an in-depth study of that logic, in terms of its expressiveness (most notably, we proved that propositional quantification fills in the gap between temporal logics and monadic second-order logic), of its model-checking problem (which we proved is decidable—see a summary of our results in Tables 1 and 2) and of its satisfiability (which is partly decidable).

Temporal logics extended with propositional quantification are simple, yet very powerful extensions of classical temporal logics. They have a natural semantics, and optimal algorithms based on standard automata constructions. Their powerful expressiveness is very convenient to encode more intricate problems over extensions of temporal logics, as was done for multi-agent systems in [LM14]. We expect that QCTL will find more applications in related areas shortly.

Acknowledgement. We thank Thomas Colcombet, Olivier Serre and Sylvain Schmitz for helpful comments during the redaction of this paper. We also thank the reviewers for their many valuable remarks.

REFERENCES

- [AFF⁺03] R. Armoni, L. Fix, A. Flaisher, O. Grumberg, N. Piterman, A. Tiemeyer, and M. Y. Vardi. Enhanced vacuity detection in linear temporal logic. In *Proceedings of the 15th International Conference on Computer Aided Verification (CAV'03)*, LNCS 2725, p. 368–380. Springer, 2003.
- [AHK97] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS'97)*, p. 100–109. IEEE Comp. Soc. Press, 1997.
- [AP06] H. Arló-Costa and E. Pacuit. First-order classical modal logic. *Studia Logica*, 84(2):171–210, 2006.
- [CDC04] K. Chatterjee, P. Dasgupta, and P. P. Chakrabarti. The power of first-order quantification over states in branching and linear time temporal logics. *Information Processing Letters*, 91(5):201–210, 2004.
- [CE82] E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Proceedings of the 3rd Workshop on Logics of Programs (LOP'81)*, LNCS 131, p. 52–71. Springer, 1982.
- [CE11] B. Courcelle and J. Engelfriet. *Graph Structure and Monadic Second-Order Logic, a Language Theoretic Approach*. Cambridge University Press, 2011.
- [CES86] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.
- [CHP07] K. Chatterjee, T. A. Henzinger, and N. Piterman. Strategy logic. In *Proceedings of the 18th International Conference on Concurrency Theory (CONCUR'07)*, LNCS 4703, p. 59–73. Springer, 2007.
- [Dam94] D. R. Dams. CTL^{*} and ECTL^{*} as fragments of the modal μ -calculus. *Theoretical Computer Science*, 126(1):77–96, 1994.
- [DLM10] A. Da Costa, F. Laroussinie, and N. Markey. ATL with strategy contexts: Expressiveness and model checking. In *Proceedings of the 30th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'10)*, LIPIcs 8, p. 120–132. Leibniz-Zentrum für Informatik, 2010.

- [DLM12] A. Da Costa, F. Laroussinie, and N. Markey. Quantified CTL: Expressiveness and model checking. In *Proceedings of the 23rd International Conference on Concurrency Theory (CONCUR'12)*, LNCS 7454, p. 177–192. Springer, 2012.
- [EF95] H.-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer, 1995.
- [EH86] E. A. Emerson and J. Y. Halpern. "Sometimes" and "not never" revisited: On branching versus linear time temporal logic. *Journal of the ACM*, 33(1):151–178, 1986.
- [ES84] E. A. Emerson and A. P. Sistla. Deciding full branching time logic. *Information and Control*, 61(3):175–201, 1984.
- [Ete99] K. Etessami. Stutter-invariant languages, ω -automata, and temporal logic. In *Proceedings of the 11th International Conference on Computer Aided Verification (CAV'99)*, LNCS 1633, p. 236–248. Springer, 1999.
- [Fin70] K. Fine. Propositional quantifiers in modal logic. *Theoria*, 36(3):336–346, 1970.
- [FM98] M. Fitting and R. L. Mendelsohn. *First-Order Modal Logic*, Synthese Library. Number 277 in Synthese Library. Springer, 1998.
- [FR03] T. French and M. Reynolds. A sound and complete proof system for QPTL. In *Proceedings of the 4th Workshop on Advances in Modal Logic (AIML'02)*, p. 127–148. King's College Publications, 2003.
- [Fre01] T. French. Decidability of quantified propositional branching time logics. In *Proceedings of the 14th Australian Joint Conference on Artificial Intelligence (AJCAI'01)*, LNCS 2256, p. 165–176. Springer, 2001.
- [Fre03] T. French. Quantified propositional temporal logic with repeating states. In *Proceedings of the 10th International Symposium on Temporal Representation and Reasoning and of the 4th International Conference on Temporal Logic (TIME-ICTL'03)*, p. 155–165. IEEE Comp. Soc. Press, 2003.
- [GC04] A. Gurfinkel and M. Chechik. Extending extended vacuity. In *Proceedings of the 5th International Conference on Formal Methods in Computer-Aided Design (FMCAD'04)*, LNCS 3312, p. 306–321. Springer, 2004.
- [GC12] A. Gurfinkel and M. Chechik. Robust vacuity for branching temporal logic. *ACM Transactions on Computational Logic*, 13(1), 2012.
- [GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979.
- [Got95] G. Gottlob. NP trees and Carnap's modal logic. *Journal of the ACM*, 42(2):421–457, 1995.
- [Hem98] H. Hempel. *Boolean Hierarchies – On Collapse Properties and Query Order*. PhD thesis, Friedrich-Schiller Universität Jena, Germany, 1998.
- [HK94] J. Y. Halpern and B. M. Kapron. Zero-one laws for modal logic. *Annals of Pure and Applied Logic*, 69(2-3):157–193, 1994.
- [HRS98] T. A. Henzinger, J.-F. Raskin, and P.-Y. Schobbens. The regular real-time languages. In *Proceedings of the 25th International Colloquium on Automata, Languages and Programming (ICALP'98)*, LNCS 1443, p. 580–591. Springer, 1998.
- [HSW13] C.-H. Huang, S. Schewe, and F. Wang. Model-checking iterated games. In *Proceedings of the 19th International Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS'13)*, LNCS 7795, p. 154–168. Springer, 2013.
- [Kai97] R. Kaivola. *Using Automata to Characterise Fixed Point Temporal Logics*. Phd thesis, School of Informatics, University of Edinburgh, UK, 1997.
- [KMTV00] O. Kupferman, P. Madhusudan, P. S. Thiagarajan, and M. Y. Vardi. Open systems in reactive environments: Control and synthesis. In *Proceedings of the 11th International Conference on Concurrency Theory (CONCUR'00)*, LNCS 1877, p. 92–107. Springer, 2000.
- [Koz83] D. C. Kozen. Results on the propositional μ -calculus. *Theoretical Computer Science*, 27:333–354, 1983.

- [KP95] O. Kupferman and A. Pnueli. *Once and for all*. In *Proceedings of the 10th Annual Symposium on Logic in Computer Science (LICS'95)*, p. 25–35. IEEE Comp. Soc. Press, 1995.
- [KP02] Y. Kesten and A. Pnueli. Complete proof system for QPTL. *Journal of Logic and Computation*, 12(5):701–745, 2002.
- [Kri59] S. A. Kripke. A completeness theorem in modal logic. *Journal of Symbolic Logic*, 24(1):1–14, 1959.
- [Kup95] O. Kupferman. Augmenting branching temporal logics with existential quantification over atomic propositions. In *Proceedings of the 7th International Conference on Computer Aided Verification (CAV'95)*, LNCS 939, p. 325–338. Springer, 1995.
- [KVW00] O. Kupferman, M. Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model-checking. *Journal of the ACM*, 47(2):312–360, 2000.
- [LM14] F. Laroussinie and N. Markey. Augmenting ATL with strategy contexts. Research Report LSV-14-05, Laboratoire Spécification et Vérification, ENS Cachan, France, 2014. 45 pages.
- [LMS01] F. Laroussinie, N. Markey, and Ph. Schnoebelen. Model checking CTL⁺ and FCTL is hard. In *Proceedings of the 4th International Conference on Foundations of Software Science and Computation Structure (FoSSaCS'01)*, LNCS 2030, p. 318–331. Springer, 2001.
- [Löd13] C. Löding. Automata on Infinite Trees (preliminary version for the handbook of the AutoMathA project), 2013.
- [Mar10] M. B. Martins. *Supervisory Control of Petri Nets using Linear Temporal Logic*. Thèse de doctorat, Instituto Superior Técnico, Universidade Técnica de Lisboa, Portugal, 2010.
- [MMV10] F. Mogavero, A. Murano, and M. Y. Vardi. Reasoning about strategies. In *Proceedings of the 30th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'10)*, LIPIcs 8, p. 133–144. Leibniz-Zentrum für Informatik, 2010.
- [MR03] F. Moller and A. Rabinovich. Counting on CTL^{*}: on the expressive power of monadic path logic. *Information and Computation*, 184(1):147–159, 2003.
- [MS85] D. E. Muller and P. E. Schupp. Alternating automata on infinite objects, determinacy and Rabin's theorem. In *Automata on Infinite Words – École de Printemps d'Informatique Théorique (EPIT'84)*, LNCS 192, p. 99–107. Springer, 1985.
- [MS87] D. E. Muller and P. E. Schupp. Alternating automata on infinite trees. *Theoretical Computer Science*, 54(2-3):267–276, 1987.
- [MS95] D. E. Muller and P. E. Schupp. Simulating alternating tree automata by nondeterministic automata: New results and new proofs of the theorems of Rabin, McNaughton and Safra. *Theoretical Computer Science*, 141(1-2):69–107, 1995.
- [Pap94] Ch. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.
- [PBD⁺02] A. C. Patthak, I. Bhattacharya, A. Dasgupta, P. Dasgupta, and P. P. Chakrabarti. Quantified computation tree logic. *Information Processing Letters*, 82(3):123–129, 2002.
- [Pin07] S. Pinchinat. A generic constructive solution for concurrent games with expressive constraints on strategies. In *Proceedings of the 5th International Symposium on Automated Technology for Verification and Analysis (ATVA'07)*, LNCS 4762, p. 253–267. Springer, 2007.
- [Pnu77] A. Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science (FOCS'77)*, p. 46–57. IEEE Comp. Soc. Press, 1977.
- [QS82] J.-P. Queille and J. Sifakis. Specification and verification of concurrent systems in CESAR. In *Proceedings of the 5th International Symposium on Programming (SOP'82)*, LNCS 137, p. 337–351. Springer, 1982.

- [Rab72] M. O. Rabin. *Automata on infinite objects and Church's thesis*, Regional Conference Series in Mathematics. Number 13 in Regional Conference Series in Mathematics. American Mathematical Society, 1972.
- [RP03] S. Riedweg and S. Pinchinat. Quantified μ -calculus for control synthesis. In *Proceedings of the 28th International Symposium on Mathematical Foundations of Computer Science (MFCS'03)*, LNCS 2747, p. 642–651. Springer, 2003.
- [SC85] A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *Journal of the ACM*, 32(3):733–749, 1985.
- [Sch03] Ph. Schnoebelen. The complexity of temporal logic model checking. In *Proceedings of the 4th Workshop on Advances in Modal Logic (AIML'02)*, p. 481–517. King's College Publications, 2003.
- [Sch13] S. Schmitz. Complexity hierarchies beyond elementary. Research Report cs.CC/1312.5686, arXiv, 2013.
- [See76] D. G. Seese. *Entscheidbarkeits- und Interpretierbarkeitsfragen monadischer Theorien zweiter Stufe gewisser Klassen von Graphen*. PhD thesis, Humboldt-Universitt zu Berlin, German Democratic Republic, 1976.
- [Sis83] A. P. Sistla. *Theoretical Issues in the Design and Verification of Distributed Systems*. PhD thesis, Harvard University, Cambridge, Massachusetts, USA, 1983.
- [Sto76] L. J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):1–22, 1976.
- [SVW87] 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.
- [tC06] B. ten Cate. Expressivity of second order propositional modal logic. *Journal of Philosophical Logic*, 35(2):209–223, 2006.
- [Tho97] W. Thomas. Languages, automata and logics. In *Handbook of Formal Languages*, p. 389–455. Springer, 1997.
- [Var82] M. Y. Vardi. The complexity of relational query languages. In *Proceedings of the 14th Annual ACM Symposium on the Theory of Computing (STOC'82)*, p. 137–146. ACM Press, 1982.
- [Wag90] K. W. Wagner. Bounded query classes. *SIAM Journal on Computing*, 19(5):833–846, 1990.
- [WHY11] F. Wang, C.-H. Huang, and F. Yu. A temporal logic for the interaction of strategies. In *Proceedings of the 22nd International Conference on Concurrency Theory (CONCUR'11)*, LNCS 6901, p. 466–481. Springer, 2011.

APPENDIX A. THE POLYNOMIAL-TIME AND EXPONENTIAL HIERARCHIES

In this section, we briefly define the complexity classes that are used in the paper. More details about those complexity classes can be found *e.g.* in [Pap94].

We write **PTIME** (resp. **NP**) for the class of decision problems that can be decided in polynomial time by a deterministic (resp. non-deterministic) Turing machine.

Given a decision problem \mathcal{L} (for instance SAT), a *Turing machine with oracle \mathcal{L}* is a Turing machine equipped with an extra tape (the *oracle tape*) and three special states q_{oracle} , q_{yes} and q_{no} . During the computation, whenever the Turing machine enters state q_{oracle} , it directly jumps to q_{yes} if the instance of \mathcal{L} written on the oracle tape is positive, and to q_{no} otherwise.

We write $\text{PTIME}^{\mathcal{L}}$ for the set of languages accepted by deterministic Turing machines with oracle \mathcal{L} and halting in polynomial time (in the size of the input). We write $\text{NP}^{\mathcal{L}}$ for the analogous class defined with non-deterministic machines. When \mathcal{L} is complete for some complexity class \mathcal{C} , we also write $\text{PTIME}^{\mathcal{C}}$ for $\text{PTIME}^{\mathcal{L}}$ (and $\text{NP}^{\mathcal{C}}$ for $\text{NP}^{\mathcal{L}}$). Notice that these definitions do not depend on the selected \mathcal{C} -complete problem. We also define $\text{PTIME}^{\mathcal{C}[\log n]}$ as the subclass of problems of $\text{PTIME}^{\mathcal{C}}$ that can be solved by a deterministic Turing machine in polynomial time but with only a logarithmic number of visits to the q_{oracle} -state. Finally, $\text{PTIME}_{\parallel}^{\mathcal{C}}$ is the subclass of problems of $\text{PTIME}^{\mathcal{C}}$ that can be solved in polynomial time by a deterministic Turing machine slightly different from the previous ones: several queries can be written on the oracle tape, but once the oracle state is visited, the oracle tape cannot be modified anymore. This can be seen as solving all queries in parallel.

The *polynomial-time hierarchy* is the following sequence of complexity classes, defined recursively with $\Sigma_0^{\text{P}} = \text{PTIME}$ and

$$\begin{aligned} \Sigma_{i+1}^{\text{P}} &= \text{NP}^{\Sigma_i^{\text{P}}} & \Pi_{i+1}^{\text{P}} &= \text{coNP}^{\Sigma_i^{\text{P}}} & \Delta_{i+1}^{\text{P}} &= \text{PTIME}^{\Sigma_i^{\text{P}}} \\ \Delta_{i+1}^{\text{P}}[O(\log n)] &= \text{PTIME}_{\parallel}^{\Sigma_i^{\text{P}}[\log n]} & \Delta_{i+1, \parallel}^{\text{P}} &= \text{PTIME}_{\parallel}^{\Sigma_i^{\text{P}}[\log n]} \end{aligned}$$

The classes in this hierarchy lie between **PTIME** and **PSPACE** (and it is an open problem whether the hierarchy collapses). Notice for instance that $\Sigma_1^{\text{P}} = \text{NP}$, since an oracle for **PTIME** problems is useless to a non-deterministic Turing machine running in polynomial time. Similarly, $\Delta_1^{\text{P}} = \text{PTIME}$. On the other hand, $\Delta_2^{\text{P}} = \text{PTIME}^{\text{NP}}$ contains those problems that can be solved in polynomial time by a deterministic Turing machine with an **NP** oracle. This includes both **NP** and **coNP**.

A natural problem in Σ_i^{P} is the problem **QSAT** $_i$ of finding the truth value of

$$\exists X_1. \forall X_2. \exists X_3 \dots Q_i X_i. \varphi(X_1, X_2, X_3, \dots, X_i)$$

where φ is a boolean formula with variables in X_1 to X_i . The dual problem (where the sequence of quantifications begins with a universal one) is Π_i^{P} -complete. The problem **SNSAT** $_i$, made of several instances of **QSAT** $_i$ where the k -th instance uses the truth value of the $k - 1$ previous ones, is an example of a Δ_{i+1}^{P} -complete problem [LMS01].

The complexity class **PH** is the union of all the classes in the polynomial-time hierarchy. Equivalently, $\text{PH} = \bigcup_{i \in \mathbb{N}} \Sigma_i^{\text{P}}$. Clearly, $\text{PH} \subseteq \text{PSPACE}$. Whether these two classes are equal is open. Notice that **PH** is not known to contain complete problems: if a problem is complete for **PH**, then it is at least as hard as any other problem in **PH**, but on the other hand is belongs to Σ_i^{P} for some i , which implies that the polynomial-time hierarchy would collapse.

The class k -EXPTIME (resp. k -EXPSPACE) is the class of decision problems that can be solved by a deterministic Turing machine running in time (resp. space) $O(\exp_k(p(n)))$ for some polynomial p , where \exp_k is defined inductively as follows:

$$\exp_1(n) = 2^n \qquad \exp_k(n) = 2^{\exp_{k-1}(n)}$$

For instance, $\exp_5(n) = 2^{2^{2^{2^{2^n}}}}$ (and $\exp_5(1)$ is a number with 65.536 binary digits). It is not difficult to prove that

$$k\text{-EXPTIME} \subseteq k\text{-EXPSPACE} \subseteq (k+1)\text{-EXPTIME},$$

and one of the two inclusions is strict (*i.e.*, $k\text{-EXPTIME} \subsetneq (k+1)\text{-EXPTIME}$).

In the same way as for the polynomial-time hierarchy, we let **ELEMENTARY** be the union of all the classes in the exponential hierarchy: $\mathbf{ELEMENTARY} = \bigcup_{k \in \mathbb{N}} k\text{-EXPTIME}$. Since this hierarchy is known to be strict, **ELEMENTARY** cannot have complete problems.

In order to define classes above **ELEMENTARY**, we define the function $\mathbf{tower}(n) = \exp_n(1)$. The class **TOWER** is then the class of problems that can be decided by a deterministic Turing machine in time $O(\mathbf{tower}(p(n)))$ where p is a polynomial. It is quite clear that $\mathbf{ELEMENTARY} \subseteq \mathbf{TOWER}$ (because $\exp_k(p(n)) \leq \mathbf{tower}(k + p(n))$). But now, as argued in [Sch13], **TOWER** has complete problems, and **TOWER**-hardness can be proved by showing k -EXPTIME-hardness for all k using *uniform* reductions.

APPENDIX B. CHARACTERISING GRID-LIKE STRUCTURES WITH EQ²CTL

Let $\mathcal{S} = \langle Q, R, \ell \rangle$ be an arbitrary (finite-state) Kripke structure. Then \mathcal{S} is a grid (in a sense that will be made precise at Prop. B.6) if it can be labelled with atomic propositions s, h, v, l, r, t and b in such a way that the following conditions are fulfilled:

- all states have at least one successor, and at most two:

$$\mathbf{AG} \mathbf{EX} \top \wedge \forall \alpha, \beta. \mathbf{AG} \left[\left(\mathbf{EX} (\alpha \wedge \beta) \wedge \mathbf{EX} (\alpha \wedge \neg \beta) \right) \Rightarrow \left((\mathbf{AX} \alpha) \wedge \bigwedge_{\substack{h_s \in \{h, \neg h\} \\ v_s \in \{v, \neg v\}}} \left[(h_s \wedge v_s) \Rightarrow (\mathbf{EX} (\neg h_s \wedge v_s) \wedge \mathbf{EX} (h_s \wedge \neg v_s)) \right] \right) \right] \quad (\text{B.1})$$

Notice that the formula also requires that when a state has two successors, then they are labelled differently w.r.t. both h and v .

- the Kripke structure has exactly one self-loop, which has only one outgoing transition (the loop itself). It is the role of atomic proposition s to mark that state:

$$\mathbf{uniq}(s) \wedge \mathbf{AG} (s \Rightarrow \mathbf{AG} s) \wedge \mathbf{AF} s. \quad (\text{B.2})$$

The first two conjuncts impose that the state labelled with s has only a self-loop as outgoing transition. The third conjunct requires that all paths eventually reach s , which means that the structure is acyclic (except at s).

- the atomic propositions h and v (for *horizontal* and *vertical*) are used to define the direction of the grid. This contains several formulas: first, we impose that the initial state has two successors:

$$(h \wedge v) \wedge \mathbf{EX} (h \wedge \neg v) \wedge \mathbf{EX} (\neg v \wedge h) \quad (\text{B.3})$$

We then impose that if a state has two successors, then those two successors have a common successor, as depicted on Fig. 7. This is expressed as follows:

$$\forall \gamma. \mathbf{AG} \left[\bigwedge_{d \in \{h, \neg h, v, \neg v\}} (d \wedge \mathbf{EX} d \wedge \mathbf{EX} \neg d) \Rightarrow \left(\bigwedge \begin{array}{l} \mathbf{EX} (d \wedge \mathbf{AX} \gamma) \Rightarrow \mathbf{EX} (\neg d \wedge \mathbf{EX} (\neg d \wedge \gamma)) \\ \mathbf{EX} (\neg d \wedge \mathbf{AX} \gamma) \Rightarrow \mathbf{EX} (d \wedge \mathbf{EX} (\neg d \wedge \gamma)) \end{array} \right) \right] \quad (\text{B.4})$$

In the same vein, we also impose another formula, which we will use as a sufficient condition for having two successors:

$$\forall \gamma. \mathbf{AG} \left[\bigwedge_{d \in \{h, \neg h, v, \neg v\}} d \Rightarrow \left(\begin{array}{c} \mathbf{EX} (d \wedge \mathbf{EX} (\neg d \wedge \gamma)) \\ \Leftrightarrow \\ \mathbf{EX} (\neg d \wedge \neg s \wedge \mathbf{EX} (\neg d \wedge \gamma)) \end{array} \right) \right] \quad (\text{B.5})$$

We also impose globally that each state can be reached from the initial state by two particular paths that are made of one ‘‘horizontal part’’ followed by a ‘‘vertical part’’ (and conversely). This is expressed as follows:

$$\forall \gamma. \left[\mathbf{EF} \gamma \Rightarrow \bigvee_{\substack{h_q \in \{h, \neg h\} \\ v_q \in \{v, \neg v\}}} \bigvee_{\substack{h_\gamma \in \{h, \neg h\} \\ v_\gamma \in \{v, \neg v\}}} \mathbf{E}h_q \mathbf{U} (h_q \wedge \mathbf{E}v_\gamma \mathbf{U} (v_\gamma \wedge \gamma)) \wedge \mathbf{E}v_q \mathbf{U} (v_q \wedge \mathbf{E}h_\gamma \mathbf{U} (h_\gamma \wedge \gamma)) \right] \quad (\text{B.6})$$

Symmetrically, from any state, the s -state can be reached using similar paths. Here we have to enumerate the possible values for h and v in the s -state:

$$\bigvee_{\substack{h_s \in \{h, \neg h\} \\ v_s \in \{v, \neg v\}}} \mathbf{AG} \left[\bigvee_{\substack{h_q \in \{h, \neg h\} \\ v_q \in \{v, \neg v\}}} \mathbf{E}h_q \mathbf{U} (h_q \wedge \mathbf{E}v_s \mathbf{U} (v_s \wedge s)) \wedge \mathbf{E}v_q \mathbf{U} (v_q \wedge \mathbf{E}h_s \mathbf{U} (h_s \wedge s)) \right] \quad (\text{B.7})$$

- finally, we label the left, right, top and bottom borders of the grid, which we define as follows:

$$\mathbf{A}(v \wedge l) \mathbf{U} (\neg v \wedge \neg l) \wedge \mathbf{AG} (\neg v \Rightarrow \mathbf{AG} \neg l) \wedge \mathbf{AG} (r \Leftrightarrow (\mathbf{AG} v \vee \mathbf{AG} \neg v)) \\ \wedge \mathbf{A}(h \wedge t) \mathbf{U} (\neg h \wedge \neg t) \wedge \mathbf{AG} (\neg h \Rightarrow \mathbf{AG} \neg t) \wedge \mathbf{AG} (b \Leftrightarrow (\mathbf{AG} h \vee \mathbf{AG} \neg h)) \quad (\text{B.8})$$

Propositions h and v are used to mark ‘‘horizontal’’ and ‘‘vertical’’ lines. A successor u' of a state u is a *horizontal successor* if both u and u' have the same labelling w.r.t. h . Similarly, it is a *vertical successor* when they have the same labelling w.r.t. v . Similarly, a *horizontal path* (resp. *vertical path*) is a path in \mathcal{S} along which the truth value of h (resp. of v) is constant. Finally, we let L , R , T and B be the sets of states labelled with l , r , t and b , respectively.

We now give a few intermediary results that will be convenient for proving that the conjunction of the formulas above characterises grids. These lemmas assume that the initial state q of \mathcal{S} satisfies the conjunction of Formulas (B.1) to (B.8), and that all states of \mathcal{S} are reachable from q .

Lemma B.1. *Pick two states u and u' in \mathcal{S} , such that there is a transition between u and u' (in either direction). Then*

- either u and u' are the same state (hence they are labelled with s),
- or they have the same labelling with h if, and only if, they have different labelling with v .

In other terms, a successor of a state is either a horizontal successor or a vertical successor, and not both (except for the state carrying the self-loop).

Proof. We proceed by contradiction: pick a transition (u, u') (not the self-loop) such that u and u' are both labelled with h and v (the other cases would be similar). We assume that u is (one of) the minimal such u , with “minimal” here being defined w.r.t. the sum of the length of all paths from the initial state q of \mathcal{S} to u .

First notice that it cannot be the case that $u = q$, because of Formulas (B.1) and (B.3). Hence u must have a predecessor w . By minimality of u , that state satisfies the condition in the lemma. We assume that w is labelled with h and $\neg v$ (the other case, with $\neg h$ and v , would be similar). From Formula (B.5) (with $d = \neg v$), there must be a successor w' of w labelled with $\neg v$, and having u' as successor (this is the common successor with u). According to Formula (B.1), w' is labelled with $\neg h$. Now, applying Formula (B.4) in w for proposition h , there must be another common successor u'' to u and w' , labelled with $\neg h$. We get a contradiction, since u now has two successors but does not satisfies Formula (B.1). \square

Lemma B.2. *If a state is in L and not in T (or in T and not in L) then it has exactly one predecessor. Only the initial state is both in L and T . Symmetrically, any state in R or in B has only one successor, and the only state in $R \cap B$ is labelled with s .*

Proof. According to Formula (B.8), the initial state must be labelled with l and t . Moreover, the initial state has two successors, labelled with $(h, \neg v)$ and $(\neg h, v)$, according to Formula (B.3). Formula (B.8) enforces that the former state satisfies $\neg l$, and the latter satisfies $\neg t$. Hence no other state will ever satisfy $l \wedge t$.

Now, consider a state u labelled with l and not with t , and assume it has two predecessors w and w' . From Formula (B.8), any path between the initial state q and state u can only visit v -states. So there must be some state between q and u having two v -successors, which is forbidden by Formula (B.1). The proof for t is similar.

Now, assume that some state u in R has two successors. From Formula (B.8), both successors will be labelled with v or both with $\neg v$, which again contradicts Formula (B.1).

Finally, if a state u is labelled with both r and b , then Formula (B.8) imposes that all its successors must have the same labelling as u w.r.t. h and v . From Lemma B.1, u is labelled with s . \square

Lemma B.3. *Pick a state u , different from the initial state. Then u is in L (resp. in T) if, and only if, it has no horizontal (resp. vertical) predecessor.*

Similarly, pick a state u not labelled with s . Then $u \in R$ (resp. $u \in B$) if, and only if, it has no horizontal (resp. vertical) successor.

Proof. We begin with proving the equivalence for the horizontal case. The vertical case is similar. We consider the four possible labellings of u w.r.t. h and v :

- if $u \models h \wedge v$: from Formula (B.6), there exists a path from q to u that is made of two parts: first a vertical path visiting only v -states, followed by a horizontal path visiting only h -states. In case u has no h -predecessor, it must be the case that the second part is trivial, so that u can be reached from q by a path visiting only v -states. From Formula (B.8), u is labelled with l .

Conversely, if u is in L , from Lemma B.2 it has only one predecessor. That predecessor must be labelled with v (Formula (B.8)), hence it cannot be labelled with h (Lemma B.1). So u has no h -predecessor.

- if $u \models \neg h \wedge v$: the same arguments apply, replacing h with $\neg h$.
- if $u \models h \wedge \neg v$: again from Formula (B.6), we get the existence of a path from q to u visiting only v -states first, and only h -states in a second part. Now, because $u \models \neg v$, the second part must contain at least two states, so that u has a predecessor labelled with h . Moreover, as u is labelled with $\neg v$, it cannot be in L (Formula (B.8)).
- if $u \models \neg h \wedge \neg v$, the same arguments apply.

Now, the proof for R (and B) is even simpler: Formula (B.8) precisely says that the states labelled with r (resp. b) are precisely those that have only vertical (resp. horizontal) successors. Apart for the s -state, this entails that those states do not have horizontal (resp. vertical) successors. \square

Lemma B.4. *Pick two states u and u' in L (resp. in T). Then there is a vertical (resp. horizontal) path between u and u' (in one or the other direction).*

Proof. From Formula (B.8), there exist vertical paths from the initial state q to u and to u' . Let π be the longest common prefix of these two paths: it contains at least q . Consider its last state x : if x is u or u' , then our result follows. Otherwise, there are two distinct vertical paths from x to u and u' , which means that x has two vertical successors, contradicting Formula (B.1). The proof for T is similar. \square

Following Lemma (B.4), we can define a binary relation \preceq_L on L (resp. \preceq_T on T) by letting $u \preceq_L u'$ if, and only if, there is a (vertical) path from u to u' (resp. $u \preceq_T u'$ if, and only if, there is a horizontal path from u to u'). These are easily seen to be ordering relations, because \mathcal{S} is (mostly) acyclic. Lemma B.4 entails that these orders are total, with q as minimal element.

Now, pick a state u . By Formulas (B.6) and (B.8), there exist (at least) one l -state x such that there is a horizontal path from x to u . Similarly, there is (at least) one t -state y with a vertical path from y to u . Notice that from Formula (B.1), we know that there is only one (maximal) horizontal (resp. vertical) path starting in any given state. We now prove that the states x and y above are uniquely determined from u .

Lemma B.5. *Let u be any state of \mathcal{S} . Let x and x' be l -states (resp. t -states) such that u is on the horizontal (resp. vertical) paths from x and from x' . Then $x = x'$.*

Proof. We prove the ‘‘horizontal’’ case, the other one being similar. So we assume we have two different states x and x' , and w.l.o.g. that $x' \preceq_L x$. We show that there exists a ‘‘grid’’ containing q , x and u , i.e., a sequence of states $(u_{i,j})_{0 \leq i \leq m, 0 \leq j \leq n}$, having the following properties:

- (1) for all $0 \leq i \leq m - 1$ and $0 \leq j \leq n$, $(u_{i,j}, u_{i+1,j})$ is a horizontal transition;
- (2) for all $0 \leq i \leq m$ and $0 \leq j \leq n - 1$, $(u_{i,j}, u_{i,j+1})$ is a vertical transition;
- (3) $u_{0,0} = q$, $u_{0,n} = x$ and $u_{m,n} = u$.

The grid is built by repeatedly applying Formula (B.5) as follows: first, there is a unique path from q to x , which defines the values $(u_{0,j})_{0 \leq j \leq n}$. Similarly, the unique path from x to u defines the values $(u_{i,n})_{0 \leq i \leq m}$. Notice that (3) is fulfilled with this definition.

Now we apply Formula (B.5) to $u_{0,n-1}$, which has a vertical successor $u_{0,n}$ followed by a horizontal successor $u_{1,n}$. Hence there must exist a horizontal successor $u_{1,n-1}$ of $u_{0,n-1}$ of which $u_{1,n}$ is a vertical successor. The same argument applies to all states between $u_{0,n-2}$ and $u_{0,0}$, thus forming a vertical path $(u_{1,j})_{0 \leq j \leq n}$. The same argument applies again to form the subsequent vertical paths, until building path $(u_{m,j})_{0 \leq j \leq n}$.

Since $x' \preceq_L x$, there must exist an integer k for which $u_{0,k} = x'$. By construction, we know that u appears on the horizontal path originating from x' . But it cannot be the case that there is a p such that $u_{p,k} = u$: this would give rise to a cycle on u . Hence there must be a horizontal path from $u_{m,k}$ to u .

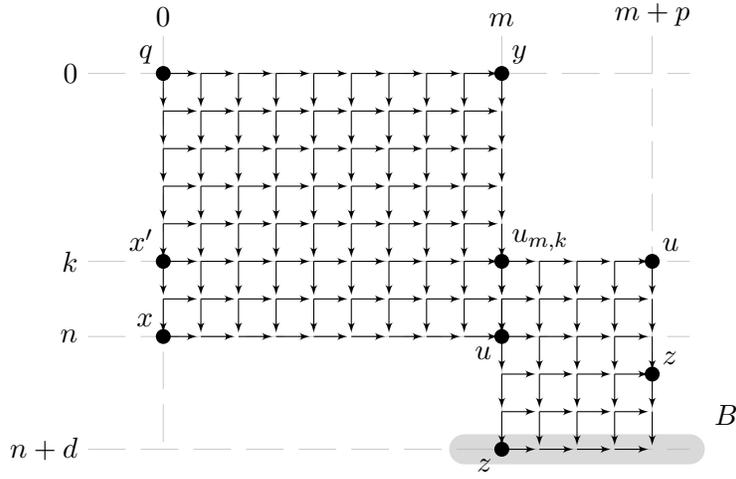


Fig. 8: Overview of the construction of the proof of Lemma B.5

Let us recap the situation: we have a state $(u_{m,k})$ from which there is a non-trivial vertical path $(u_{m,k+j})_{0 \leq j \leq n-k}$ to u , as well as a non-trivial horizontal path to u , which we write $(u_{m+i,k})_{0 \leq i \leq p}$ (see Fig. 8). Now, from Formula (B.7), from u there is a vertical path to a b -state z . That path is unique, thanks to Formula (B.1). Hence there is a unique integer d and a unique sequence of states $(u_{m,n+i})_{0 \leq i \leq d}$ that forms a vertical path from $u_{m,n} = u$ to a b -state $u_{m,n+d} = z$. Notice that z is not labelled with s , because it has vertical predecessors that have horizontal successors. Now, starting from the horizontal path $(u_{m+i,k})_{0 \leq i \leq p}$ and the vertical path $(u_{m,k+j})_{0 \leq j \leq n-k+d}$ and applying Formula (B.4), we build a grid $(u_{m+i,k+j})_{0 \leq i \leq p, 0 \leq j \leq n-k+d}$. But since there is only one maximal vertical path from u , it must be the case that $u_{m+p,k+d} = z$: hence this state is not labelled with s , but it has a vertical successor and belongs to B , which is a contradiction. \square

We are now ready for proving our result:

Proposition B.6. *Write φ for the conjunction of all formulas above. Then $\mathcal{S}, q \models \varphi$ if, and only if, the part of \mathcal{S} that is reachable from q is a (two-dimensional) grid (i.e., it can be defined as the product of two finite-state “linear” Kripke structures).*

Proof. One the one hand, it is clear that a grid can be labelled with h, v, l, r, t, b and s in such a way that φ holds.

We now prove the converse, assuming that all the states of \mathcal{S} are reachable from q . We assume that \mathcal{S} is labelled with h, v, l, r, t, b and s in a way that witnesses all the formulas constituting φ .

Using h and v , we define the following relations: two states u and u' are *h-equivalent* if there is a sequence $(u_i)_{0 \leq i \leq k}$ of states such that $u_0 = u$, $u_k = u'$, the labelling of $(u_i)_I$ is constant w.r.t. h , and for all $0 \leq i \leq k-1$, there is a transition (u_i, u_{i+1}) or (u_{i+1}, u_i) . Notice that in particular u and u' have the same h -labelling. *v-equivalence* is defined similarly. Clearly enough, these are equivalence relations, and we write \mathcal{H} and \mathcal{V} for the sets of equivalence classes of these relations. Each set forms a partition of the set of states of \mathcal{S} .

We also define the sets L, R, T and B as the sets of states labelled with the corresponding atomic propositions (l, r, t and b , respectively). First notice that L and R are sets of \mathcal{V} , and T and B are in \mathcal{H} :

- $L \in \mathcal{V}$: the initial state must be labelled with l (and v). We prove that L is the v -equivalence class of q : first, L contains precisely those states that are reachable from q via a vertical path. Since any vertical predecessor of a state in L is in L (Lemma B.2), we get the result.
- $R \in \mathcal{V}$: the s -state is in R , and we prove that R is the v -equivalence class of the s -state. For more clarity, we assume that the s -state is labelled with v . Then R contains exactly the v -states from which only v -states are reachable. This proves the result.

The proof for T and B is similar.

Now, from Formula (B.6) and Lemma B.5, any $H \in \mathcal{H}$ contains exactly one element from L , which we write $l(H)$. Similarly, any $V \in \mathcal{V}$ is the equivalence class of a unique element of T , denoted with $t(V)$. Then any $H \in \mathcal{H}$ and $V \in \mathcal{V}$ have non-empty intersection: this can be proven by building a grid containing q , $l(H)$ and $t(V)$. If some H and V were to have two states in common, we would be in a similar situation as in the proof of Lemma B.5. As a consequence, all the elements of \mathcal{H} contain the same number of states (namely, the number of elements of \mathcal{V}). Similarly for the elements of \mathcal{V} . Finally, since each state u in a set H of \mathcal{H} also belongs to \mathcal{V} , it can be associated with an element $t(u)$ of T . This gives rise to an order in each set H . One easily sees that these orders are compatible with the “vertical successor” relation, meaning that if $u \prec_H u'$, then their vertical successors x and x' satisfy $x \prec_{H'} x'$. Hence the graph of \mathcal{S} is isomorphic to the product of L and T (augmented with a self-loop on their last states). \square

Notice that we only characterised two-dimensional grids. One-dimensional grids can be characterised by

$$\exists r. \forall \gamma. \mathbf{AG} (\mathbf{EX} \top \wedge (\mathbf{EX} \gamma \Rightarrow \mathbf{AX} \gamma)) \wedge (\mathbf{EF} (r \wedge \gamma) \Rightarrow \mathbf{AG} (r \Rightarrow \gamma)) \wedge \mathbf{EF} \mathbf{AG} r.$$

The first conjunct enforces that each state has exactly one successor; the second conjunct expresses the fact at most one state is labelled with r . The last conjunct enforces that the r -state is eventually reached and never escaped, thus requiring that it has a self-loop.