Quantified CTL: expressiveness and model checking

Arnaud Da Costa, François Laroussinie, Nicolas Markey

June 2012

Research report LSV-12-02 (Version 1.1 – 2020/04/24)

Laboratoire Spécification & Vérification

École Normale Supérieure de Cachan
61, avenue du Président Wilson
94235 Cachan Cedex France

Quantified CTL: expressiveness and model checking

Arnaud Da Costa[1], François Laroussinie[2], and Nicolas Markey[1]

[1] LSV – CNRS & ENS Cachan
[2] LIAFA – Univ. Paris Diderot & CNRS

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 characterize the complexity of its model-checking problem, 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). We also show how these results apply to model checking ATL-like temporal logics for games.

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 (*e.g.* via *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* [AHK02]) extend CTL towards the study of open systems (involving several agents).

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 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 Logic. Regarding model checking, we consider both prenex-normal-form formulas (EQCTL) and general formulas (QCTL), and our results are summarized in the table below (where k in EQ^{k}CTL and Q^{k}CTL refers to some measure of quantification height of formulas, see Section 2.4). Finally, we also characterize the model- and formula-complexities of our problems, when one of the inputs to the model-checking problem is fixed.

	structure semantics	tree semantics
EQ^{k}CTL	$\Sigma^{\mathsf{P}}_{\mathsf{k}}$-c.	k-EXPTIME-c.
Q^{k}CTL	$\Delta^{\mathsf{P}}_{\mathsf{k}+1}[O(\log n)]$-c.	
EQ^{k}CTL*, Q^{k}CTL*	PSPACE-c.	k+1-EXPTIME-c.
EQCTL, QCTL, EQCTL*,QCTL*		non-elementary

Applications to alternating-time temporal logics. 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 [CHP07,MMV10] or (B)SIL [WHY11].

Interestingly, the model-checking problem for these extensions of ATL$_{sc}$ (and also for Strategy Logics) can be seen as a QCTL model-checking problem[3]: strategy quantification in ATL is naturally encoded using propositional quantification of QCTL; since this labelling is *persistent*, it can encode interactions between strategies. We give the full encoding in Section 5. 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*.

Related works. Propositional quantification was also defined and studied on LTL [Sis83,SVW87,KP95], where the model-checking problem for the k-alternation fragment was settled complete for k-EXPSPACE. In the branching-time setting,

[3] Notice that the link between games and propositional quantification already emerges in Q$_{\mathrm{D}}\mu$ [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 results are more sparse: CTL and CTL* with only existential quantification was studied in [Kup95], where model checking is shown NP- and PSPACE-complete resp. (for the structure semantics) and EXPTIME- and 2-EXPTIME-complete resp. (for the tree semantics). The two-alternation fragment was then studied in [KMTV00] (only for the tree semantics): model checking is 2-EXPTIME- and 3-EXPTIME-complete, respectively for CTL or CTL*. Finally, satisfiability of the full extension (with arbitrary quantification) was studied in [Fre01].

Several other semantics have also been defined in the literature: the amorphous semantics is somewhat intermediary between structure- and tree semantics, and considers bisimilar structures before labelling with extra atomic propositions [Fre01]. Alternative semantics are proposed and studied in [RP03,PBD$^+$02].

Besides the above-mentioned applications of QCTL to open systems, let us mention that QCTL has also been used in the setting of three-valued model checking, where *partial* Kripke structures are considered (*i.e.*, Kripke structures where the truth value of some atomic propositions may be unknown) [BG99].

2 Preliminaries

2.1 Kripke structures and trees

We fix once and for all a set AP of atomic propositions.

Definition 1. *A* Kripke structure $\mathcal{S}$ *is a* 3*-tuple* $\langle Q, R, \ell \rangle$ *where* Q *is a countable*[4] *set of states,* $R \subseteq Q^2$ *is a total relation (*i.e.*, for all* $q \in Q$*, there exists* $q' \in Q$ *s.t.* $(q, q') \in R$*), and* $\ell \colon Q \to 2^{\mathsf{AP}}$ *is a labelling function.*

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. We use $\mathsf{Exec}(q)$ to denote the set of executions issued from q and $\mathsf{Exec}^{\mathsf{f}}(q)$ for the set of all finite prefixes of executions of $\mathsf{Exec}(q)$. Given $\rho \in \mathsf{Exec}(q)$ and $i \in \mathbb{N}$, we write ρ^i for the path $(q_{i+k})_{k \in \mathbb{N}}$ of $\mathsf{Exec}(q_i)$ (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.

Definition 2. *Let* Σ *and* S *be two finite sets. A* Σ-labelled S-tree *is a pair* $\mathcal{T} = \langle T, l \rangle$*, where*

- $T \subseteq S^*$ *is a non-empty set of finite words on* S *satisfying the following constraints: for any non-empty word* $n = m \cdot s$ *in* T *with* $m \in S^*$ *and* $s \in S$*, the word* m *is also in* T*;*
- $l \colon T \to \Sigma$ *is a labelling function.*

The unwinding of a finite-state Kripke structure $\mathcal{S} = \langle Q, R, \ell \rangle$ from a state $q \in Q$ is the (finitely-branching) 2^{AP}-labelled Q-tree $\mathcal{T}_{\mathcal{S}}(q) = \langle \mathsf{Exec}^{\mathsf{f}}(q), \ell_{\mathcal{T}} \rangle$ with $\ell_{\mathcal{T}}(q_0 \cdots q_i) = \ell(q_i)$. Note that $\mathcal{T}_{\mathcal{S}}(q) = \langle \mathsf{Exec}^{\mathsf{f}}(q), \ell_{\mathcal{T}} \rangle$ can be seen as an (infinite-state) Kripke structure where the set of states is $\mathsf{Exec}^{\mathsf{f}}(q)$, labelled according to $\ell_{\mathcal{T}}$, and with transitions $(m, m \cdot s)$ for all $m \in \mathsf{Exec}^{\mathsf{f}}(q)$ and $s \in Q$ s.t. $m \cdot s \in \mathsf{Exec}^{\mathsf{f}}(q)$.

[4] In what follows, the input Kripke structures will always be finite. But it will be convenient to see the execution trees of such Kripke structures as an infinite-state Kripke structures, hence our considering countable state spaces.

Definition 3. *For $P \subseteq \mathsf{AP}$, 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 $\mathcal{S} \equiv_P \mathcal{S}'$) whenever $Q = Q'$, $R = R'$, and $\ell(q) \cap P = \ell'(q) \cap P$ for any $q \in Q$.*

In other terms, $\mathcal{S} \equiv_P \mathcal{S}'$ if $\mathcal{S}'$ can be obtained from $\mathcal{S}$ by modifying the labelling function of $\mathcal{S}$ for propositions in P.

2.2 CTL and quantified extensions

Definition 4. *The syntax of* QCTL^* *is defined by the following grammar:*

$$\varphi_{\mathsf{state}}, \psi_{\mathsf{state}} ::= p \mid \neg\, \varphi_{\mathsf{state}} \mid \varphi_{\mathsf{state}} \vee \psi_{\mathsf{state}} \mid \mathbf{E}\varphi_{\mathsf{path}} \mid \mathbf{A}\varphi_{\mathsf{path}} \mid \exists p.\ \varphi_{\mathsf{state}}$$
$$\varphi_{\mathsf{path}}, \psi_{\mathsf{path}} ::= \varphi_{\mathsf{state}} \mid \neg\, \varphi_{\mathsf{path}} \mid \varphi_{\mathsf{path}} \vee \psi_{\mathsf{path}} \mid \mathbf{X}\, \varphi_{\mathsf{path}} \mid \varphi_{\mathsf{path}}\, \mathbf{U}\, \psi_{\mathsf{path}}$$

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.*

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

Definition 5. *The syntax of* QCTL *is defined by the following grammar:*

$$\varphi_{\mathsf{state}}, \psi_{\mathsf{state}} ::= p \mid \neg\, \varphi_{\mathsf{state}} \mid \varphi_{\mathsf{state}} \vee \psi_{\mathsf{state}} \mid \exists p.\ \varphi_{\mathsf{state}} \mid$$
$$\mathbf{E}\varphi_{\mathsf{state}}\, \mathbf{U}\, \psi_{\mathsf{state}} \mid \mathbf{A}\varphi_{\mathsf{state}}\, \mathbf{U}\, \psi_{\mathsf{state}} \mid \mathbf{EX}\, \varphi_{\mathsf{state}} \mid \mathbf{AX}\, \varphi_{\mathsf{state}}.$$

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 $\varphi[(p_i \to \psi_i)_i]$ (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 for instance [Kup95,Fre01,RP03]). Here we consider two semantics: the *structure semantics* and the *tree semantics*.

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. This is defined as for CTL^*, with the following addition:

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

Intuitively, $\exists p.\varphi$ holds true at state q of structure $\mathcal{S}$ if it is possible to modify the p-labelling of $\mathcal{S}$ in such a way that φ holds at q.

Example 6. As an example, consider the formula

$$\mathsf{selfloop} = \forall z.(z \Rightarrow \mathbf{EX}\, z).$$

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

$$\mathsf{uniq}(\varphi) = \mathbf{EF}\,(\varphi) \wedge \forall z.\Big(\mathbf{EF}\,(\varphi \wedge z) \Rightarrow \mathbf{AG}\,(\varphi \Rightarrow z)\Big).$$

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 $\mathsf{uniq}(\varphi)$ holds in a state (under the structure semantics) if, and only if, exactly one reachable state satisfies φ.

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 $\mathcal{T}_S(q)$ as an infinite-state Kripke structure, we define:

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

Clearly enough, $\mathsf{selfloop}$ is always false under the tree semantics, while $\mathsf{uniq}(\varphi)$ holds if, and only if, φ holds at only one node of the execution tree.

Example 7. Formula

$$\mathsf{acyclic} = \mathbf{AG}\,\big(\exists z.\ (z \wedge \mathsf{uniq}(z) \wedge \mathbf{AX}\ \mathbf{AG}\ \neg z)\big) \tag{1}$$

expresses that all infinite paths (starting from the current state) are acyclic, which for *finite-state* Kripke structures is always false under the structure semantics and always true under the tree semantics.

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. Note also that for any state formula ψ, we have $\Phi[\psi] \equiv_{s,t} \exists p_\psi.\big(\Phi[p_\psi] \wedge \mathbf{AG}\,(p_\psi \Leftrightarrow \psi)\big)$ where p_ψ is a fresh atomic proposition (*i.e.*, a proposition not already appearing in the model).

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 logics[5]

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

- for any $\varphi \in \mathsf{CTL}$ and any $p \in \mathsf{AP}$, $\exists p.\varphi$ is an $\mathsf{EQ}^1\mathsf{CTL}$ formula, and $\forall p.\varphi$ is in $\mathsf{AQ}^1\mathsf{CTL}$;
- for any $\varphi \in \mathsf{EQ}^k\mathsf{CTL}$ and any $p \in \mathsf{AP}$, $\exists p.\varphi$ is in $\mathsf{EQ}^k\mathsf{CTL}$ and $\forall p.\varphi$ is in $\mathsf{AQ}^{k+1}\mathsf{CTL}$. Symmetrically, if $\varphi \in \mathsf{AQ}^k\mathsf{CTL}$, then $\exists p.\varphi$ is in $\mathsf{EQ}^{k+1}\mathsf{CTL}$ while $\forall p.\varphi$ remains in $\mathsf{AQ}^k\mathsf{CTL}$.

Using similar ideas, we define fragments of QCTL and QCTL*. Again, the definition is inductive: $\mathsf{Q}^1\mathsf{CTL}$ is the logic $\mathsf{CTL}[\mathsf{EQ}^1\mathsf{CTL}]$, and $\mathsf{Q}^{k+1}\mathsf{CTL} = \mathsf{Q}^1\mathsf{CTL}[\mathsf{Q}^k\mathsf{CTL}]$.

The corresponding extensions of CTL*, which we respectively denote with $\mathsf{EQ}^k\mathsf{CTL}^*$, $\mathsf{AQ}^k\mathsf{CTL}^*$ and $\mathsf{Q}^k\mathsf{CTL}^*$, are defined in a similar way.

Remark 8. Notice that $\mathsf{EQ}^k\mathsf{CTL}$ and $\mathsf{AQ}^k\mathsf{CTL}$ are (syntactically) included in $\mathsf{Q}^k\mathsf{CTL}$, and $\mathsf{EQ}^k\mathsf{CTL}^*$ and $\mathsf{AQ}^k\mathsf{CTL}^*$ are fragments of $\mathsf{Q}^k\mathsf{CTL}^*$.

3 Expressiveness

In this section we present several results about the expressiveness of our logics for both 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 13).

3.1 Prenex normal form

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

Proposition 9. *In both semantics,* QCTL *and* EQCTL *are equally expressive. (i.e., any* QCTL *formula can be translated into an equivalent formula in prenex normal form).*

[5] Notice that the logics named EQCTL and EQCTL* defined in [Kup95] are restrictions of our prenex-normal-form logics where only existential quantification is allowed. They correspond to our fragments $\mathsf{EQ}^1\mathsf{CTL}$ and $\mathsf{EQ}^1\mathsf{CTL}^*$.

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

In the following, we assume w.l.o.g. that every propositional quantification deals with fresh Boolean variables. We use $\mathcal{Q}$ to denote a sequence of quantifications, and write $\bar{\mathcal{Q}}$ for the dual sequence. Our translation is defined as a sequence of rewriting rules that are to be applied in a bottom-up manner, replacing innermost subformulas with s-equivalent ones first. As for CTL, we only consider the temporal modalities $\mathbf{EX}$, $\mathbf{E}\,\mathbf{U}$ and $\mathbf{EG}$ (as $\mathbf{AX} \equiv \neg\,\mathbf{EX}\,\neg$ 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:

$$\mathcal{Q}.p \equiv_s p \qquad \neg\,\mathcal{Q}.\varphi \equiv_s \bar{\mathcal{Q}}\,\neg\varphi \qquad \mathcal{Q}_1.\varphi_1 \vee \mathcal{Q}_2\varphi_2 \equiv_s \mathcal{Q}_1.\mathcal{Q}_2.(\varphi_1 \vee \varphi_2)$$

Extracting a bloc of quantifiers out of an $\mathbf{EX}$ operator can be done as follows:

$$\mathbf{EX}\,\mathcal{Q}.\varphi \equiv_s \exists z.\mathcal{Q}.\Big(\mathsf{uniq}(z) \wedge \mathbf{EX}\,(z \wedge \varphi)\Big)$$

Here variable z (which is assumed to not appear in $\mathcal{Q}$) is used to mark the immediate successor that has to satisfy $\mathcal{Q}.\varphi$, we also require z to be unique (allowing more than one successor would make the equivalence to be wrong). Note that the right-hand-side formula is not yet in prenex form, because $\mathsf{uniq}(z)$ involves a universal quantifier under a Boolean operator; applying the above rules for Boolean subformulas concludes the translation for this case.

For $\mathbf{E}(\mathcal{Q}_1.\varphi_1)\,\mathbf{U}\,(\mathcal{Q}_2.\varphi_2)$, we take the quantifiers out by labelling with z_0 a short witnessing path (up to the position witnessing $\mathcal{Q}_2.\varphi_2$, which we label with z_2). Quantifications in $\mathcal{Q}_2$ then only depend on z_2. Similarly, intermediate positions can be checked against $\mathcal{Q}_1.\varphi_1$ using a (universally quantified) variable z_1. In the end:

$$\mathbf{E}(\mathcal{Q}_1.\varphi_1)\,\mathbf{U}\,(\mathcal{Q}_2.\varphi_2) \equiv_s \exists z_0.z_2.\mathcal{Q}_2.\forall z_1.\mathcal{Q}_1.\\ \Big(\mathsf{dpath}(z_0,z_2) \wedge \mathbf{EF}\,(z_2 \wedge \varphi_2) \wedge \Big(\mathsf{uniq}(z_1) \Rightarrow \mathbf{AG}\,((z_1 \wedge z_0) \Rightarrow \varphi_1)\Big)\Big)$$

where $\mathsf{dpath}(z_0, z_2)$ ensures that z_0 labels a *direct* path leading to z_2, meaning that it contains no loop and there is at most one state labeled with z_0 having z_2 as immediate successor. Formally:

$$\mathsf{dpath}(x,y) \equiv_s \mathsf{uniq}(y) \wedge \mathbf{E}x\,\mathbf{U}\,y \;\wedge \\ \mathbf{AG}\,\Big(\Big((x \wedge \mathsf{selfloop}) \Rightarrow (\,\mathbf{EX}_2(x \vee y)\Big) \wedge \Big((x \wedge \neg\,\mathsf{selfloop}) \Rightarrow (\,\mathbf{EX}_1(x \vee y))\Big)\Big)$$

where $\mathbf{EX}_i\varphi$ specifies that exactly i of the immediate successors satisfy φ:

$$\mathbf{EX}_1(\varphi) = \mathbf{EX}\,\varphi \wedge \forall z.\Big(\mathbf{EX}\,(\varphi \wedge z) \;\Rightarrow\; \mathbf{AX}\,(\varphi \Rightarrow z)\Big)$$

$$\mathbf{EX}_2(\varphi) = \exists z.\,\mathbf{EX}\,(\varphi \wedge z) \wedge \mathbf{EX}\,(\varphi \wedge \neg\,z) \wedge \forall z'.\\ \Big((\,\mathbf{EX}\,(\varphi \wedge z \wedge z') \wedge \mathbf{EX}\,(\varphi \wedge \neg\,z \wedge z')) \Rightarrow\; \mathbf{AX}\,(\varphi \Rightarrow z')\Big)$$

Again, notice that $\mathsf{dpath}(x,y)$ can easily be written in prenex form as $\mathbf{AG}\,(\mathcal{Q}.\varphi) \equiv_s \forall z.\mathcal{Q}.(\mathsf{uniq}(z) \Rightarrow \mathbf{AG}\,(z \Rightarrow \varphi))$.

Finally, we give the translation for $\mathbf{EG}\,(\mathcal{Q}.\varphi)$; the idea again is to label a witnessing (lasso-shaped) path with z, ensuring that $\mathcal{Q}.\varphi$ always holds along that path:

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

Before we prove correctness of the above equivalences, we introduce a useful characterization: consider a Kripke structure $\mathcal{S} = \langle Q, R, \ell\rangle$, a state q and a QCTL formula $\mathcal{Q}.\varphi$ with $\mathcal{Q} = \mathcal{Q}_1 z_1 \cdots \mathcal{Q}_k z_k$. We have $\mathcal{S}, q \models_s \mathcal{Q}.\varphi$ iff 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 $\mathsf{AP}\backslash\{z_1, \ldots, z_k\}$, and:
2. for any $\mathcal{S}' = \langle Q, R, \ell'\rangle$ in ξ, and for any i with $\mathcal{Q}_i = \forall$, we have: for every $\mathsf{lab}_{z_i} : Q \to 2^{\{z_i\}}$, there exists $\langle Q, R, \ell''\rangle \in \xi$ such that $\ell'' \cap \{z_i\} = \mathsf{lab}_{z_i}$, and ℓ'' and ℓ' coincide over $\mathsf{AP}\backslash\{z_i \cdots z_k\}$;
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 is said to be $(\mathcal{Q}, \mathcal{S})$-compatible.

We now proceed to the proof of the previous equivalences. We omit the cases of propositional and Boolean formulas, and focus on $\mathbf{EX}$, $\mathbf{EG}$ and $\mathbf{E}\,\mathbf{U}$:

– $\mathbf{EX}\,(\mathcal{Q}.\varphi)$: 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:

$$\xi' = \Big\{\mathcal{S}' = \langle Q, R, \ell'\rangle \;\Big|\; \exists \langle Q, R, \ell''\rangle \in \xi \text{ and } \ell' = \ell'' \oplus \{q' \mapsto z\}\Big\}$$

with:

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

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

Now assume $\mathcal{S}, q \models_s \exists z.\mathcal{Q}.\big(\mathsf{uniq}(z) \wedge \mathbf{EX}\,(z \wedge \varphi)\big)$. Then there exists a Kripke structure $\mathcal{S}' \equiv_{\mathsf{AP}\backslash\{z\}} \mathcal{S}$ such that $\mathcal{S}', q \models_s \mathcal{Q}.\big(\mathsf{uniq}(z) \wedge \mathbf{EX}\,(z \wedge \varphi)\big)$. 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 $\mathcal{S}, q \models \mathbf{EX}\,(\mathcal{Q}.\varphi)$.

– $\mathbf{EG}\,(\mathcal{Q}.\varphi)$: Assume $\mathcal{S}, q \models_s \mathbf{EG}\,(\mathcal{Q}.\varphi)$. There exists a path $\rho = q_0 q_1 q_2 \dots q_l$ with $q_0 = q$ and $q_l = q_j$ for some $j < l$. We assume that ρ is a direct path: q_l is the only repeated state and $\mathcal{S}$ does not contain a transition (q_l, q_m) unless $m = l+1$. Thus labeling ρ-states by z will make the formula $(z \wedge \mathbf{AG}\,(z \Rightarrow \mathbf{EX}_1 z)$ hold at q. Now we know that for every $i < l$, $\mathcal{S}, q_i \models_s \mathcal{Q}.\varphi$, so that there exists a set ξ_i of Kripke structures that are $(\mathcal{Q}, \mathcal{S})$-compatible and such that $\mathcal{S}', q_i \models_s \varphi$ for every $\mathcal{S}' \in \xi_i$. Now, let ξ be the following set of Kripke structures:

$$\xi = \Big\{ \mathcal{S}' = \langle Q, R, \ell' \rangle \;\Big|\; \exists i < l, \exists \langle Q, R, \ell'' \rangle \in \xi_i, \text{ and } \ell' = \ell'' \oplus \{q_j \mapsto z\}_{j=0,\dots,l} \oplus \{q_i \mapsto z'\} \Big\}$$

For every $\mathcal{S}' \in \xi$, we have $\mathcal{S}', q \models_s z \wedge \mathbf{AG}\,(z \Rightarrow \mathbf{EX}_1 z) \wedge (\mathsf{uniq}(z') \Rightarrow \mathbf{AG}\,((z \wedge z' \Rightarrow \varphi)))$. But the set ξ is not $(\exists z \forall z' \mathcal{Q}, \mathcal{S})$-compatible because it contains only Kripke structures with a z'-labeling for a single state along ρ (the universal quantification requires that we consider all labelings). But ξ can easily be completed with arbitrary Kripke structures with other form of z'-labeling to obtain a compatible set $\hat{\xi}$. Note that the additional Kripke structures still satisfy $(\mathsf{uniq}(z') \Rightarrow \mathbf{AG}\,((z \wedge z' \Rightarrow \varphi)))$. Thus $\mathcal{S}, q$ satisfies the right-hand-side formula of the equivalence.
Now assume that

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

Let $\mathcal{S}'$ be the structure labeled with z such that

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

The first property ensures that the z-labeling describes an infinite 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 $\mathcal{S}'', q \models_s \mathsf{uniq}(z') \Rightarrow \mathbf{AG}\,((z \wedge z') \Rightarrow \varphi)$. Clearly it entails that for any position i along the z-path, there exists a $(\mathcal{Q}, \mathcal{S}'')$-compatible set in which $q_i \models_s \varphi$. This proves the result.

– $\mathbf{E}(\mathcal{Q}_1\varphi_1)\,\mathbf{U}\,(\mathcal{Q}_2\varphi_2)$: Assume $\mathcal{S}, q \models_s \mathbf{E}(\mathcal{Q}_1\varphi_1)\,\mathbf{U}\,(\mathcal{Q}_2\varphi_2)$. Then there exists a path $\rho = q_0 q_1 x \dots q_i$ such that $q_0 = q$ and $\mathcal{S}, q_i \models_s \mathcal{Q}_2\varphi_2$ and $\mathcal{S}, q_j \models_s \mathcal{Q}_1\varphi_1$ for $j < i$. We can assume that ρ is *direct*: it is never necessary to loop or postpone the arrival in a state satisfying $\mathcal{Q}_2.\varphi_2$. Thus labeling the final state q_i with z_2 and every q_j (with $j < i$) with z_0 makes ρ satisfy $\mathsf{dpath}(z_0, z_2)$. Now as $\mathcal{S}, q_i \models_s \mathcal{Q}_2\varphi_2$, there exists a set ξ_2 of Kripke structures that is $(\mathcal{Q}_2, \mathcal{S})$-compatible and such that $\mathcal{S}', q_i \models_s \varphi_2$ for every $\mathcal{S}' \in \xi_2$. Now we define ξ_2' as follows:

$$\xi_2' = \Big\{ \mathcal{S}' = \langle Q, R, \ell' \rangle \;\Big|\; \exists \langle Q, R, \ell'' \rangle \in \xi_2 \text{ and } \ell' = \ell'' \oplus \{q_i \mapsto z_2, q_j \mapsto z_0 (j = 0, \dots, i-1)\} \Big\}$$

Clearly ξ_2' is $(\exists z_0.\exists z_2.\mathcal{Q}_2, \mathcal{S})$-compatible and for any $\mathcal{S}' \in \xi_2'$, we have $\mathcal{S}', q \models_s \mathsf{dpath}(z_0, z_2) \wedge \mathbf{EF}\,(z_2 \wedge \varphi_2)$.
Moreover for any $j = 0, \ldots, i-1$, we have $\mathcal{S}, q_j \models_s \mathcal{Q}_1.\varphi_1$. As $\mathcal{Q}_1.\varphi_1$ does not depend on z_0, z_2 and variables in $\mathcal{Q}_2$, it holds: for every $\mathcal{S}' \in \xi_2'$, $\mathcal{S}', q_j \models_s \mathcal{Q}_1.\varphi_1$. Therefore there exists $\xi_1^{\mathcal{S}'}$ that is $(\mathcal{Q}_1, \mathcal{S}')$-compatible s.t. for every $\mathcal{S}'' \in \xi_1^{\mathcal{S}'}$, we have $\mathcal{S}'', q_j \models_s \varphi_1$. Now consider:

$$\xi = \Big\{ \mathcal{S}'' = \langle Q, R, \ell'' \rangle \;\Big|\; \exists \mathcal{S}' \in \xi_2'.\ \exists j < i.\ \exists \langle Q, R, \ell''' \rangle \in \xi_1^{\mathcal{S}'} \text{ and } \ell'' = \ell''' \oplus \{q_j \mapsto z_1\} \Big\}$$

As in the previous case, ξ is not $(\exists z_0 \exists z_2 \mathcal{Q}_2 \forall z_1 \mathcal{Q}_1, \mathcal{S})$-compatible because it only contains structures with a unique state labeled with z_1. But it can be completed by arbitrary structures to get a compatible set $\widehat{\xi}$. This entails the result.
Now assume

$$\mathcal{S}, q \models_s \exists z_0.z_2.\mathcal{Q}_2.\forall z_1.\mathcal{Q}_1.(\mathsf{dpath}(z_0, z_2) \wedge \mathbf{EF}\,(z_2 \wedge \varphi_2) \wedge (\mathsf{uniq}(z_1) \Rightarrow \mathbf{AG}\,((z_1 \wedge z_0) \Rightarrow \varphi_1))).$$

Then it entails that there is a path labeled with $z_0\,\mathbf{U}\,z_2$, that $\mathcal{Q}_2\varphi_2$ holds true in the final state (by using the same arguments as for the $\mathbf{EX}$ case), and that for any z_1-labeling of a unique state q' along the path, that state q' has to satisfy $\mathcal{Q}_1.\varphi_1$. We clearly have the property.

Finally note that we didn't pay attention to the complexity of the translation but it is easy to see that the DAG-size of the resulting formula is linear in the size of the original formula. The translation increases also the number of quantifications. Note that in the case of $\mathbf{E}\,\mathbf{U}$, quantifications $\mathcal{Q}_1$ and Q_2 are *independent* and the quantification $\mathcal{Q}_1.\mathcal{Q}_2$ can be reordered so as to minimize the number of quantifier alternations (of course the orders inside the blocs have to be preserved). □

Remark 10. As already stated, the above translation into prenex normal form is still correct when considering the tree semantics. But in that framework, we could define a simpler transformation (in particular, we can get rid of the $\mathsf{dpath}(z_0, z_2)$ formula); also, for the tree semantics, another translation is given in [Fre06] (in a slightly different setting).

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.*, labeled finite graphs). In both cases, we use constant monadic predicates P_a for $a \in \mathsf{AP}$, and a relation Edge either for the immediate successor relation in an S-tree $\langle T, l \rangle$ or for the relation R in a finite Kripke structure $\langle Q, R, \ell \rangle$.

MSO is built with first-order (or individual) variables for nodes or vertices (denoted with lowercase letters $x, y, ...$), monadic second-order variables for sets of nodes (denoted with uppercase letters $X, Y, ...$). Atomic formulas are of the form $x = y$, $\mathsf{Edge}(x, y)$, $x \in X$, $\mathsf{P}_a(x)$. Formulas are constructed from atomic formulas using the Boolean connectives and the first- and second-order quantifier $\exists$. We write $\varphi(x_1, ..., x_n, X_1, ..., X_k)$ to state that $x_1, ..., x_n$ and $X_1, ..., X_k$ 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 $\mathcal{M}, s_1, ..., s_n, S_1, ..., S_k \models \varphi(x_1, ..., x_n, X_1, ..., X_k)$ when φ holds on $\mathcal{M}$ when s_i (resp. S_j) is assigned to the variable x_i (resp. X_j) for $i = 1, ..., n$ (resp. $j = 1, ..., k$).

In the following, we compare the expressiveness of our logics 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 directly over 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 that a graph is connected.

Formally, for the tree semantics, we say that $\varphi(x) \in \mathsf{MSO}$ 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 $\mathcal{T}_\mathcal{S}(q), q \models \psi$. 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 $\mathcal{S}_q, q \models \psi$, where $\mathcal{S}_q$ is the reachable part of $\mathcal{S}$ from q. For these definitions, we have:

Proposition 11. *Under both semantics,* MSO *and* QCTL *are equally expressive.*

Proof. One inclusion is straightforward: CTL is easily translated into MSO, and propositional quantification (for both semantics) can be encoded using second-order quantification.

Conversely, given $\varphi(x) \in \mathsf{MSO}$, we define $\widehat{\varphi} \in \mathsf{QCTL}$ as follows:

$$\begin{aligned}
\widehat{\exists x_i.\varphi} &= \exists \mathsf{p}_{x_i}.\mathsf{uniq}(\mathsf{p}_{x_i}) \wedge \widehat{\varphi} & \widehat{\exists X_i.\varphi} &= \exists \mathsf{p}_{X_i}.\widehat{\varphi} \\
\widehat{x_i \in X_j} &= \mathbf{EF}\,(\mathsf{p}_{x_i} \wedge \mathsf{p}_{X_j}) & \widehat{x \in X_i} &= \mathsf{p}_{X_i} \\
\widehat{\mathsf{P}_a(x_i)} &= \mathbf{EF}\,(\mathsf{p}_{x_i} \wedge a) & \widehat{\varphi \wedge \psi} &= \widehat{\varphi} \wedge \widehat{\psi} \\
\widehat{\mathsf{Edge}(x, x_i)} &= \mathbf{EX}\,\mathsf{p}_{x_i} & \widehat{\neg \varphi} &= \neg\,\widehat{\varphi} \\
\widehat{\mathsf{Edge}(x_i, x)} &= \bot & \widehat{\mathsf{P}_a(x)} &= a \\
\widehat{x_i = x_j} &= \mathbf{EF}\,(\mathsf{p}_{x_i} \wedge \mathsf{p}_{x_j}) & \widehat{x = x_i} &= \mathsf{p}_{x_i} \\
\widehat{\mathsf{Edge}(x_i, x_j)} &= \mathbf{EF}\,(\mathsf{p}_{x_i} \wedge \mathbf{EX}\,\mathsf{p}_{x_j}) &&
\end{aligned}$$

We prove correctness of this translation under the structure semantics (the tree semantics can be handled using similar arguments). Consider a Kripke structure $\mathcal{S}$

and a state q: we have to prove that $\mathcal{S}_q, q \models \varphi(x)$ is equivalent to $\mathcal{S}_q, q \models_s \widehat{\varphi}$. For this, we inductively prove the following stronger equivalence:

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

where $\mathcal{S}'_q$ is obtained from $\mathcal{S}_q$ by labelling state s with p_{x_i} (resp. with p_{X_i}) iff $s = s_i$ (resp. $s \in S_i$). Each case of the induction follows easily. □

Remark 12. One can also notice that it is easy to express (least and greatest) fixpoint operators with QCTL in both semantics. For instance,

$$\mu X.\ \varphi(X) \equiv \exists p.\, [\, \mathbf{AG}\,(p \Leftrightarrow \varphi(p)) \wedge \forall p'.[(\, \mathbf{AG}\,(p' \Leftrightarrow \varphi(p'))) \Rightarrow\ \mathbf{AG}\,(p \Rightarrow p').]$$

Hence QCTL can express the μ-calculus. In the proof of Prop. 9, we observed that QCTL also has the ability to count the number of successors of a state. This provides another proof of Prop. 11 for the tree semantics, since the μ-calculus extended with counting capabilities has the same expressiveness as MSO on trees [MR03].

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.

Proposition 13. *In the tree and structure semantics, every* QCTL* *formula is equivalent to some* QCTL *formula.*

Proof. This was shown in [Fre01] for the tree semantic. We give another translation, which is correct for both semantics. Consider a QCTL* formula Φ, and write k for the number of subformulas of Φ that are not in QCTL. If $k = 0$, Φ already belongs to QCTL. Otherwise let ψ be one of the inner-most Φ-subformulas in QCTL* \ QCTL. Let $(\alpha_i)_{1 \leq i \leq m}$ be the largest ψ-subformulas belonging to QCTL. These are state formulas, so that ψ is equivalent (for both semantics) to:

$$\exists p_1 ... \exists p_m . \Big(\psi[(\alpha_i \leftarrow p_i)_{i=1,...,m}] \wedge \bigwedge_{i=1,...,m} \mathbf{AG}\,(p_i \ \Leftrightarrow\ \alpha_i) \Big)$$

Let Ω be $\psi[(\alpha_i \leftarrow p_i)_{i=1,...,m}]$. Then Ω is a CTL* formula: every subformula of the form $\exists p.\xi$ in ψ appears in some QCTL formula α_i, since ψ is one of the smallest QCTL* \ QCTL subformula. As every CTL* formula is equivalent to some μ-calculus formula, Ω is equivalent to some QCTL formula $\widetilde{\Omega}$ (see Remark 12). Hence

$$\psi \equiv_{s,t} \exists p_1 ... \exists p_m . \Big(\widetilde{\Omega} \wedge \bigwedge_{i=1,...,m} \mathbf{AG}\,(p_i \ \Leftrightarrow\ \alpha_i) \Big)$$

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 QCTL* \ QCTL, so that the induction hypothesis applies. □

From Propositions 9, 11 and 13, we get:

Corollary 14. *Under both the structure semantics and the tree semantics,* EQCTL*,* QCTL *and* QCTL* *and* MSO *are equally expressive.*

Remark 15. In [Fre01], French considers a variant of QCTL* (which we call FQCTL* here), where propositional quantification can also be used within path formulas: $\exists p.\ \varphi_{\mathsf{path}}$ is a valid path formula, meaning that φ_{path} holds along ρ after modifying the labelling with p:

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

For the tree semantics, FQCTL* as expressive as QCTL [Fre01]: for a formula $\exists\varphi$ (with φ an FQCTL* path formula), the translation consists in first marking the witness path with some atomic proposition y, and then changing subformulas $\mathbf{G}\,\psi$ into $\mathbf{AG}\,(y \Rightarrow \psi)$ (and similarly for other unary modalities; binary modalities have to be removed first, again using propositional quantification).

For the structure semantics, we show that FQCTL* is strictly more expressive than MSO. Formula

$$\mathsf{Hamilton} = \mathbf{EG}\left(\exists z.\forall z'.[\mathsf{uniq}(z) \wedge \mathsf{uniq}(z') \wedge z \wedge \neg\, z'] \Rightarrow \mathbf{X}\,(\neg\, z\,\mathbf{U}\, z')\right).$$

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 characterizes the existence of a Hamilton cycle. This is known not to be expressible in MSO [EF95, Cor. 6.3.5]. However, the existence of a Hamilton cycle can be expressed in Guarded Second Order Logic GSO (also called MS_2 in [CE11]), 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:

$$\mathsf{Euler} = \mathbf{EG}\Big(\exists x.\exists y.\forall x'.\forall y'.\Big[\mathsf{tr}(x,y) \wedge \mathsf{tr}(x',y') \wedge \mathsf{next\_tr}(x,y) \wedge \neg\, \mathsf{next\_tr}(x',y')\Big] \Rightarrow \mathbf{X}\,(\neg\, \mathsf{next\_tr}(x,y)\,\mathbf{U}\, \mathsf{next\_tr}(x',y'))\Big)$$

where $\mathsf{tr}(x,y) = \mathsf{uniq}(x) \wedge \mathsf{uniq}(y) \wedge \mathbf{EF}\,(x \wedge \mathbf{X}\,y)$ states that x and y mark the source and target of a reachable transition, and $\mathsf{next\_tr}(x,y) = x \wedge \mathbf{X}\,y$ 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 [Cou90].

Proposition 16. *Under the structure semantics,* FQCTL* *is more expressive than* QCTL* *and* MSO*.*

Still, FQCTL* model checking (see next section) is decidable: for the tree semantics, it suffices to translate FQCTL* to QCTL [Fre01], for which model checking is decidable (as we prove below). The problem in the structure semantics can then be encoded in the tree semantics: we first assume that each state of the

input Kripke structure $\mathcal{S}$ is labelled with its name (so that any two different states can be distinguished). Then any quantification $\exists p.\varphi$ in the structure semantics is considered in the tree semantics, with the additional requirement that any two copies of the same state receive the same p-labelling.

4 QCTL 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? Some results already exist, *e.g.* for $\mathsf{EQ}^1\mathsf{CTL}$ and $\mathsf{EQ}^1\mathsf{CTL}^*$ under both semantics [Kup95]. Hardness results for $\mathsf{EQ}^2\mathsf{CTL}$ and $\mathsf{EQ}^2\mathsf{CTL}^*$ 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 model- and formula-complexities [Var82] of model-checking for these fragments[6].

4.1 Model checking for the structure semantics

Formulas in prenex normal form. Prenex-normal-form formulas are (technically) easy to handle: a formula in $\mathsf{EQ}^k\mathsf{CTL}$ can be model-checked by non-determinisitically guessing a labelling and applying a model-checking procedure for $\mathsf{AQ}^{k-1}\mathsf{CTL}$. We easily derive the following results.

Theorem 17. *Under the structure semantics, model checking* $\mathsf{EQ}^k\mathsf{CTL}$ *is* $\Sigma^{\mathsf{P}}_{\mathsf{k}}$*-complete, model checking* $\mathsf{AQ}^k\mathsf{CTL}$ *is* $\Pi^{\mathsf{P}}_{\mathsf{k}}$*-complete, and model checking* $\mathsf{EQ}^k\mathsf{CTL}^*$, $\mathsf{AQ}^k\mathsf{CTL}^*$, EQCTL *and* EQCTL^* *is* PSPACE*-complete.*

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

Problem: $\Sigma^{\mathsf{P}}_{\mathsf{k}}\mathbf{SAT}$
Input: k families of variables $U_i = \{u^i_1, \ldots, u^i_n\}$, and a propositional formula $\Phi(U_1, \ldots, U_k)$ over $\bigcup_i U_i$;
Question: what is the truth value of the quantified Boolean formula $\mathcal{Q}_1 U_1 \mathcal{Q}_2 U_2 \ldots \mathcal{Q}_k U_k.\Phi(U_1, \ldots, U_k)$ where $\mathcal{Q}_i$ is $\exists$ (resp. $\forall$) when i is odd (resp. even)?

Membership in $\Sigma^{\mathsf{P}}_{\mathsf{k}}$ is proved inductively: an $\mathsf{EQ}^1\mathsf{CTL}$ instance $\exists u^1_1 \ldots \exists u^1_k.\ \varphi$ can be solved in $\mathsf{NP} = \Sigma^{\mathsf{P}}_1$ by non-deterministically picking a labelling of the Kripke structure under study with atomic propositions u^1_1 to u^1_k, and then checking (in

[6] For standard notions of size for $\mathcal{S}$ and φ, unless specified otherwise (see Theorems 20 and 19)

polynomial time) whether the CTL formula φ holds true in the resulting Kripke structure. Similarly, an $\mathsf{EQ}^k\mathsf{CTL}$ formula $\exists u_1^1 \dots \exists u_k^1.\ \varphi$, where φ is in $\mathsf{AQ}^{k-1}\mathsf{CTL}$, 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 $\mathsf{AQ}^{k-1}\mathsf{CTL}$ 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}.

The algorithm for $\mathsf{EQ}^k\mathsf{CTL}^*$ and $\mathsf{AQ}^k\mathsf{CTL}^*$ is the same as for $\mathsf{EQ}^k\mathsf{CTL}$, with the CTL model-checking algorithm replaced with a CTL^* model-checking algorithm running in polynomial space. Since $\mathsf{NP}^{\mathsf{PSPACE}} = \mathsf{PSPACE}$, the resulting algorithms are in PSPACE. Hardness is straightforward (CTL^* model checking is already PSPACE-hard).

Finally, any formula in EQCTL (resp. EQCTL^*) is in $\mathsf{EQ}^k\mathsf{CTL}$ (resp. $\mathsf{EQ}^k\mathsf{CTL}^*$) for some k. The algorithms above can be applied, and use polynomial space. Hardness is straightforward after noticing that any instance of QBF is a special case of a model-checking problem for EQCTL. □

General case. We now handle the general case, not assuming that formulas are in prenex normal form.

Notice that for those fragments, the parameter we use here is the *quantifier depth* of the formula, instead of its *alternation depth*. Actually, by possibly inserting negations, it is easily checked that in $\mathsf{Q}^k\mathsf{CTL}$, we can assume that all propositional quantifiers are existential.

Theorem 18. *For the structure semantics, model checking is* $\Delta_{k+1}^{\mathsf{P}}[O(\log n)]$*-complete for* $\mathsf{Q}^k\mathsf{CTL}$*, and* PSPACE*-complete for* $\mathsf{Q}^k\mathsf{CTL}^*$*,* QCTL *and* QCTL^**.*

Proof. We define the algorithm for $\mathsf{Q}^k\mathsf{CTL}$ inductively: when $k = 0$, we just have a CTL model-checking problem, which is complete for $\mathsf{PTIME} = \Delta_1^{\mathsf{P}}[O(\log n)]$. Assume that we have a $\Delta_{k+1}^{\mathsf{P}}[O(\log n)]$ algorithm for the $\mathsf{Q}^k\mathsf{CTL}$ model-checking problem, and consider a formula $\varphi \in \mathsf{Q}^{k+1}\mathsf{CTL}$: it can be written under the form $\varphi = \Phi[(q_i \to \exists P_i.\ \psi_i)_i]$ with Φ being a CTL formula involving fresh atomic propositions q_i, and $\exists P_i.\ \psi_i$ are subformulas [7] of φ. The existential quantifiers in these subformulas are the outermost propositional quantifiers in φ, and ψ_i belongs to $\mathsf{Q}^k\mathsf{CTL}$, 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 $\mathsf{NP}^{\Delta_{k+1}^{\mathsf{P}}[O(\log n)]} = \Sigma_{k+1}^{\mathsf{P}}$. 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 $\Delta_{k+1}^{\mathsf{P}}[O(\log n)]$, since $\Delta_{k+1}^{\mathsf{P}}[O(\log n)] = \Delta_{k+1}^{\mathsf{P}}{}_{||}$ [Wag90].

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

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

Problem: PARITY (Σ_k^P)
Input: m instances of Σ_k^PSAT $Q_1^i U_1^i \dots Q_k^i U_k^i . \Phi^i(U_1^i, \dots, U_k^i)$, 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 as follows: for each $1 \leq i \leq m$, the instance $Q_1^i U_1^i \dots Q_k^i U_k^i . \Phi^i(U_1^i, \dots, U_k^i)$ of Σ_k^PSAT is encoded as in the previous reduction, using a one-state Kripke structure that will be labelled with atomic propositions $u_{j,k}^i$; the formula to be checked is then exactly Φ_i. We label the unique state of that Kripke structure with a fresh atomic proposition x_i, that will be used in the sequel of the reduction.

Now, consider the Kripke structure B obtained as the "union" of the one-state Kripke structures above, with an extra state x_{m+1} and transitions (x_i, x_{i+1}), for each $1 \leq i \leq m$. We define $\varphi = \bigvee_{1 \leq i \leq m} (x_i \wedge \Phi_i)$. This formula holds true in those states x_i of B whose corresponding Σ_k^PSAT instance is positive. It remains to build a formula for "counting" these sets: we let

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

It is easily seen that ψ_s holds true in state x_1 of B iff exactly s of the m instances of Σ_k^PSAT are positive. Moreover, each ψ_i has quantifier height at most k. Concluding the reduction is then obvious.

The proofs for the other cases are similar to those of Theorem 17. □

Formula- and program-complexity. Most of the proofs above can be adapted to use a fixed formula or a fixed model. One notable exception is QCTL: 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 model checking thus lies in the small gap between PH and PSPACE (unless the polynomial-time hierarchy collapses).

Theorem 19. *Under the structure semantics, the formula-complexity (i.e., when the model is fixed) of model checking is Σ_k^P-complete for EQ^kCTL, and Π_k^P-complete for AQ^kCTL; it is $\Delta_{k+1}^P[O(\log n)]$-complete for Q^kCTL when considering the* DAG*-size of the formula. It is* PSPACE*-complete for EQ^kCTL*, AQ^kCTL*, Q^kCTL*, EQCTL, QCTL, EQCTL*, and QCTL*.*

Proof. Our proofs for EQ^kCTL and AQ^kCTL already consider a fixed model, entailing the first two results.

Regarding Q^kCTL, we 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^PSAT. 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 $\{\Psi_1, \ldots, \Psi_i\}$, 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.

Note if we do not consider DAG-size, one can easily see that formula complexity of $\mathsf{Q}^k\mathsf{CTL}$ model checking is Σ_k^P-hard and Π_k^P-hard.

PSPACE-hardness results follow from those results and from the PSPACE-hardness of the fixed-model model-checking problem for CTL^* [Sch03]. □

Theorem 20. *Under the structure semantics, the program-complexity (i.e., when the formula is fixed) of model checking is Σ_k^P-complete for $\mathsf{EQ}^k\mathsf{CTL}$ and $\mathsf{EQ}^k\mathsf{CTL}^*$, Π_k^P-complete for $\mathsf{AQ}^k\mathsf{CTL}$ and $\mathsf{AQ}^k\mathsf{CTL}^*$, and $\Delta_{k+1}^\mathsf{P}[O(\log n)]$-complete for $\mathsf{Q}^k\mathsf{CTL}$ and $\mathsf{Q}^k\mathsf{CTL}^*$ (for positive k). It is PH-hard but not in PH (unless the polynomial-time hierarchy collapses), and in PSPACE but not PSPACE-hard for EQCTL, QCTL, EQCTL^* and QCTL^*.*

Proof. Membership in Σ_k^P for $\mathsf{EQ}^k\mathsf{CTL}$ and $\mathsf{EQ}^k\mathsf{CTL}^*$, and in Π_k^P for $\mathsf{AQ}^k\mathsf{CTL}$ and $\mathsf{AQ}^k\mathsf{CTL}^*$, directly follows from the general algorithms (Theorem 17), and from the fact that the program-complexity of CTL^* model checking is in PTIME.

We now prove hardness in Σ_k^P for $\mathsf{EQ}^k\mathsf{CTL}$ (the results for $\mathsf{EQ}^k\mathsf{CTL}^*$, $\mathsf{AQ}^k\mathsf{CTL}$ and $\mathsf{AQ}^k\mathsf{CTL}^*$ 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) formula, while the model encodes the SAT formula to be checked. We begin with an NP-hardness proof for $\mathsf{EQ}^1\mathsf{CTL}$, and then explain how it can be extended to $\mathsf{EQ}^k\mathsf{CTL}$.

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

Write $\varphi = \bigwedge_{1\leq i\leq m} \bigvee_{1\leq j\leq n} \ell_{i,j}$, where $l_{i,j}$ 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, m states named C_i for $1 \leq i \leq m$, and $3|P|$ states named p_k, $\neg\, p_k$ and $\mathsf{test}p_k$ for each $p_k \in P$.
- there is a transition from φ to each C_i and to each $\mathsf{test}p_k$, a transition from each $\mathsf{test}p_k$ 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 $\mathsf{test}(p_k)$ are labelled with an atomic proposition test, which is the only atomic proposition in the model.

The intuition is as follows: one of the states p_k and $\neg p_k$ will be labelled (via the EQCTL formula) by 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:

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

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 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 $\mathsf{EQ}^k\mathsf{CTL}$. Consider an instance of $\Sigma_k^{\mathsf{P}}\mathsf{SAT}$ of the form $\exists P_1 \ldots \mathcal{Q}_k P_k.\, \varphi(P_1, \ldots, P_k)$, 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 $\mathcal{Q}_k$ is universal, and φ is a disjunction of conjunctive clauses—the dual case being similar), formula Φ_k then writes as follows:

$$\begin{aligned}\Phi_k = \exists \oplus_1 \ldots \forall \oplus_k .[\, &\mathbf{AX}\,(\mathsf{test}_{2i+1} \Rightarrow (\,\mathbf{EX}\ \oplus_{2i+1} \wedge \mathbf{EX}\,\neg\oplus_{2i+1}))\ \wedge \\ &[\,\mathbf{AX}\,(\mathsf{test}_{2i+2} \Rightarrow (\,\mathbf{EX}\ \oplus_{2i+2} \wedge \mathbf{EX}\,\neg\oplus_{2i+2}))] \Rightarrow (\,\mathbf{AX}\,(\neg\,\mathsf{test} \Rightarrow\ \mathbf{EX}\,\oplus))].\end{aligned}$$

Using similar ideas, we prove hardness in $\Delta_{k+1}^{\mathsf{P}}[O(\log n)]$ for the fixed-formula model-checking problem for $\mathsf{Q}^k\mathsf{CTL}$. Fix some k, and consider of $\mathsf{PARITY}\ (\Sigma_k^{\mathsf{P}})$, made of m instances of $\Sigma_k^{\mathsf{P}}\mathsf{SAT}$, which we write $\Phi^i(U_1^i, \ldots, U_k^i)$ (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 $+$ or $-$ (to indicate the validity of the i-th instance). Transitions are defined as follows: from init, there is a transition to $(1,0,+)$ and $(1,0,-)$; from $(i,0,+)$ and $(i,1,-)$, there are transitions to $(i+1,1,+)$ and to $(i+1,1,-)$; from $(i,1,+)$ and $(i,0,+)$, there are transitions to $(i+1,0,+)$ and $(i+1,0,-)$. Finally, there is a transition from $(m,0,-)$ and $(m,1,+)$ to final, and self-loops on final, $(m,0,+)$ and $(m,1,-)$. Consider a path in such a Kripke structure, and assume that we can enforce that the path visits a $+$-state if, and only if, the corresponding $\Sigma_k^{\mathsf{P}}\mathsf{SAT}$ 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 $(m,0,+)$ or in $(m,1,-)$. In other words, formula $\mathbf{EF}\ \mathsf{final}$ 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 $\Sigma_k^{\mathsf{P}}\mathsf{SAT}$. This is achieved using the reduction of the proof of Theorem 20: 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 20. Now, the formula to be checked in the resulting structure has to be reinforced as follows:

$$\Psi_k = \mathbf{E}(+ \Leftrightarrow \tilde{\Phi}_k)\,\mathbf{U}\,\mathsf{final}$$

where $\tilde{\Phi}_k$ is (a slightly modified version of) the formula built in the proof of Theorem 20.

From the previous results, model-checking a fixed formula in EQCTL or EQCTL^* is PH-hard. Membership in PSPACE follows from Theorem 17. 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^{k}CTL or EQ^{k}CTL* for some k) could be used to encode any instance of QSAT, again collapsing the polynomial-time hierarchy. □

4.2 Model checking for the tree semantics

Theorem 21. *Model checking* EQ^{k}CTL, AQ^{k}CTL *and* Q^{k}CTL *under the tree semantics is* k-EXPTIME*-complete (for positive k).*

Proof. Since EQ^{k}CTL and AQ^{k}CTL are dual and contained in Q^{k}CTL, it suffices to prove hardness for EQ^{k}CTL and membership for Q^{k}CTL.

▶ *Hardness in* k-EXPTIME. The reduction uses the ideas of [KMTV00,SVW87]: we encode an alternating Turing machine $\mathcal{M}$ whose tape is bounded by the following recursively-defined function:

$$E(0,n) = n \qquad\qquad E(k+1,n) = 2^{E(k,n)}.$$

An execution of $\mathcal{M}$ on an input word y of length n is then a tree. Our reduction consists in building a Kripke structure K and a Q^{k}CTL formula φ such that φ holds true in K (for the tree semantics) iff $\mathcal{M}$ accepts y.

As a first step, we design a set of (polynomial-size) formulas of EQ^{k}CTL that are able to relate two states that are at distance $E(k,n)$ (actually, a slightly different value). This will be used in our reduction to ensure that the content of one cell of the Turing machine is preserved from one configuration to the next one, unless the tape head is around. Define

$$F(0,n) = n \qquad\qquad F(k+1,n) = F(k,n) \cdot 2^{F(k,n)},$$

and assume we are given a tree labelled with atomic propositions s and t (among others). We first require that s and t appear exactly once along any branch, by means of the following formula

$$\mathsf{once}(\varphi) = \mathbf{AF}\,\varphi \wedge \mathbf{AG}\,(\varphi \Rightarrow \mathbf{AX}\,\mathbf{AG}\,\neg\varphi).$$

Our formula for requiring one occurrence of s and t (in that order) along each branch then reads

$$\mathsf{delimiters}(s,t) = \mathsf{once}(s) \wedge \mathsf{once}(t) \wedge \mathbf{AG}\,(s \Rightarrow \mathbf{AF}\,t). \tag{2}$$

We now inductively build our "yardstick" formulas enforcing that, along any branch, the distance between the occurrence of s and that of t is precisely $F(k,n)$ (see Fig. 1). When $k=0$, this is easy[8]:

$$\mathsf{yardstick}_0^n(s,t) = \mathbf{AG}\left(s \Rightarrow \big((\,\mathbf{AX}\,)^n t \wedge \bigwedge_{0 \le k < n} (\,\mathbf{AX}\,)^k \neg t\big)\right). \tag{3}$$

[8] There is a bit of redundancy with delimiter, but this will be needed as this formula will sometimes be used alone.

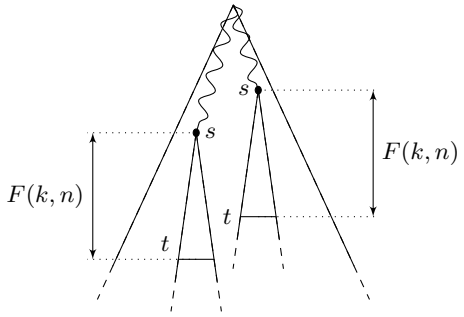

Fig. 1. Chunks of height $F(k, n)$

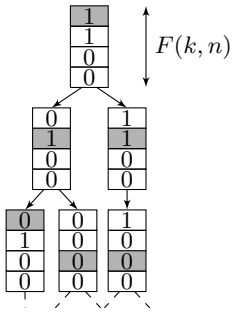

Fig. 2. Encoding runs of $\mathcal{M}$

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:

$$\mathsf{yardstick}_k^n = \exists r.\exists c.\ (\mathsf{graduation}_k(r,s,t) \wedge \mathsf{counter}_k(c,r,s,t)). \quad (4)$$

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

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

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:

$$\begin{aligned}
\mathsf{counter}_1(c,r,s,t) &= \mathsf{zeros}_1(c,r,s,t) \wedge \mathsf{ones}_1(c,r,s,t) \wedge \mathsf{increment}_1(c,r,s,t)\\
\mathsf{zeros}_1(c,r,s,t) &= \mathbf{AG}\,(s \Leftrightarrow (r \wedge \neg c \wedge \mathbf{AX}\ \mathbf{A}(\neg c\,\mathbf{U}\,r)))\\
\mathsf{ones}_1(c,r,s,t) &= \mathbf{AG}\,((r \wedge \mathbf{AX}\ \mathbf{A}(\neg r\,\mathbf{U}\,t)) \Rightarrow \mathbf{A}(c\,\mathbf{U}\,t))\\
\mathsf{increment}_1(c,r,s,t) &= \mathbf{AG}\,(s \Rightarrow (\,\mathbf{AG}\,((c \Leftrightarrow (\,\mathbf{AX}\,)^n c) \Leftrightarrow \\
&\qquad \mathbf{AX}\ \mathbf{A}(\neg r\,\mathbf{U}\,(\neg c \wedge \neg r))))).
\end{aligned}$$

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, $\mathsf{increment}_1$ 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, $\mathsf{yardstick}_1$ is an $\mathsf{EQ}^1\mathsf{CTL}$ formula.

For any $k \geq 2$, $\mathsf{yardstick}_k$ is obtained using similar ideas, with slightly more involved formulas.

$$\begin{aligned}\mathsf{graduation}_k(r,s,t) = \ & \mathbf{AG}\,((s \vee t) \Rightarrow r) \wedge \\ & \forall u.\forall v.\ [(\mathsf{delimiters}(u,v) \wedge \mathsf{yardstick}^n_{k-1}(u,v)) \Rightarrow \\ & \quad (\,\mathbf{AG}\,(u \Rightarrow \mathbf{AF}\,(r \wedge \mathbf{AF}\,v)) \wedge \\ & \quad\quad \mathbf{AG}\,((r \wedge \mathbf{AF}\,v \wedge \neg\,\mathbf{AF}\,u) \Rightarrow \mathbf{AX}\,\mathbf{A}(\neg\, r\,\mathbf{U}\,v)))].\end{aligned}$$

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 $\mathsf{yardstick}^n_{k-1}$ appears negated in $\mathsf{graduation}_k$. 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 $\mathsf{graduation}_k$:

$$\begin{aligned}\mathsf{increment}_k(c,r,s,t) = \forall u.\forall v.[(\mathsf{delimiters}(u,v) \wedge \mathsf{yardstick}^n_{k-1}(u,v)) \Rightarrow \\ (\,\mathbf{AG}\,((s \wedge \mathbf{AF}\,u) \Rightarrow (\,\mathbf{AG}\,(((u \wedge c) \Leftrightarrow \mathbf{AG}\,(v \Rightarrow c)) \Leftrightarrow \\ (\,\mathbf{AX}\,\mathbf{A}\neg\, r\,\mathbf{U}\,(\neg\, c \wedge \neg\, r))))))]\end{aligned}$$

This formula is a mix between $\mathsf{increment}_1$, 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 $\mathsf{yardstick}^n_{k-1}$ is, by induction hypothesis, in $\mathsf{EQ}^{k-1}\mathsf{CTL}$, formula $\mathsf{yardstick}^n_k$ is in $\mathsf{EQ}^k\mathsf{CTL}$ (notice that $\mathsf{yardstick}^n_{k-1}$ appears negated after the universal quantifiers on u and v).

We now explain how we encode the acceptance problem for a k-1-EXPSPACE alternating Turing machine into an $\mathsf{EQ}^k\mathsf{CTL}$ 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. 2, with one configuration being encoded as a sequence of cells, and branching occurring only between two consecutive configurations.

With $\mathcal{M}$ and y, we associate a Kripke structure $K = \langle S, s_0, R, \ell\rangle$ where $S = (Q \cup \{\epsilon\}) \times \Sigma \cup \{\#\}$, $R = S \times S$ is the complete transition relation, $s_0 = \#$, and ℓ labels each state with its name (hence the set of "original" atomic propositions is S).

The execution tree of K contains as branches any word in $s_0 \cdot S^\omega$, not all of which are needed in order to represent an accepting execution of $\mathcal{M}$. Hence we will label this execution tree with an (existentially-quantified) proposition a, having in mind that a will label exactly the branches that participate in the encoding of an accepting execution of $\mathcal{M}$.

We can easily enforce that a labels exactly branches of execution tree of K, by requiring that

$$\mathbf{AG}\,(\neg\, a \Rightarrow \mathbf{AG}\,\neg\, a).$$

Then along those branches, we will require that # appears as a delimiter, exactly at any levels multiple of $F(k-1,n)$. This can be expressed as

$$\#\wedge\forall u.\forall v.\ [(\mathsf{delimiters}(u,v)\wedge\mathsf{yardstick}^n_{k-1}(u,v))\Rightarrow \mathbf{AG}\,((u\wedge\#\wedge a)\Rightarrow\mathbf{AX}\,\mathbf{A}(\neg\#\wedge a)\,\mathbf{U}\,((v\wedge\#)\vee\neg a))]$$

Notice that the latter formula is in $\mathsf{AQ}^{k-1}\mathsf{CTL}$, since the $\mathsf{yardstick}^n_{k-1}$ formula is in $\mathsf{EQ}^{k-1}\mathsf{CTL}$. Using the same idea, one easily comes up with formulas expressing that

- each configuration contains exactly one occurrence of the tape head,
- the content of the tape is preserved from one configuration to the next one, except that one transition of the Turing machine has been applied (with sufficiently many executions being forked),
- each a-branch is reaches an accepting state.

We have thus reduced the acceptance problem for an alternating Turing machine running in $k-1$-exponential space to a model-checking problem for $\mathsf{EQ}^k\mathsf{CTL}$, which entails that the latter is k-EXPTIME-hard.

► *Membership in* k-EXPTIME. Our algorithm uses alternating parity tree automata. We use $\langle Q, 2^{\mathsf{AP}}\rangle$-APT to denote alternating parity tree automata running on 2^{AP}-labelled Q-trees. Similarly, $\langle Q, 2^{\mathsf{AP}}\rangle$-NPT denotes non-deterministic parity tree automata. With each $\mathsf{Q}^k\mathsf{CTL}$ formula φ, Kripke structure K and state q_0, we associate such a $\langle Q, 2^{\mathsf{AP}}\rangle$-APT whose language is non-empty if, and only if $K, q_0 \models \varphi$. We will not recall the definitions of parity tree automata, and better refer to [MS87,Tho97] for more details. Our construction uses classical techniques, already present *e.g.* in [Tho97,CHP07,Pin07,DLM10,MMV10]. Our proof uses the following lemmas as building blocks. In those lemmas, the size of an automaton is its number of states, and the index is the number of priorities in the parity condition.

Lemma 22. *[KVW00] Given a* CTL *formula* φ *over* AP *and a set* Q *of directions, we can construct a* $\langle Q, 2^{\mathsf{AP}}\rangle$*-APT* $\mathcal{A}_\varphi$ *accepting exactly the* 2^{AP}*-labelled* Q*-trees satisfying* φ*.* $\mathcal{A}_\varphi$ *has size linear in the size of* φ*, and uses a constant number of priorities.*

Lemma 23. *[MS87,MS95] Let* $\mathcal{A}$ *and* $\mathcal{B}$ *be two* $\langle Q, 2^{\mathsf{AP}}\rangle$*-APTs that respectively accept languages* A *and* B*. We can build two* $\langle Q, 2^{\mathsf{AP}}\rangle$*-APTs* $\mathcal{C}$ *and* $\mathcal{D}$ *that respectively accept the languages* $A\cap B$ *and* $\overline{A}$ *(the complement of* A *in the set of* 2^{AP}*-labelled* Q*-trees). The size and index of* $\mathcal{C}$ *are at most* $(|\mathcal{A}|+|\mathcal{B}|)$ *and* $\max(\mathsf{idx}(\mathcal{A}),\mathsf{idx}(\mathcal{B}))+1$*, while those of* $\mathcal{D}$ *are* $|A|$ *and* $\mathsf{idx}(\mathcal{A})$*.*

Lemma 24. *[MS95] Let* $\mathcal{A}$ *be a* $\langle Q, 2^{\mathsf{AP}}\rangle$*-APT. We can build an* $\langle Q, 2^{\mathsf{AP}}\rangle$*-NPT* $\mathcal{N}$ *accepting the same language as* $\mathcal{A}$*, and such that* $|\mathcal{N}| \in 2^{O(|\mathcal{A}|\mathsf{idx}(\mathcal{A})\cdot\log(|\mathcal{A}|\mathsf{idx}(\mathcal{A})))}$ *and* $\mathsf{idx}(\mathcal{N}) \in O(|\mathcal{A}|\cdot\mathsf{idx}(\mathcal{A}))$*.*

Lemma 25. *[MS85] Let $\mathcal{A}$ be a $\langle Q, 2^{\mathsf{AP}}\rangle$-NPT, with $\mathsf{AP} = \mathsf{AP}_1 \cup \mathsf{AP}_2$. For all $i \in \{1,2\}$, we can build a $\langle Q, 2^{\mathsf{AP}}\rangle$-NPT $\mathcal{B}_i$ such that, for any 2^{AP}-labelled Q-tree $\mathcal{T}$, it holds: $\mathcal{T} \in \mathcal{L}(\mathcal{B}_i)$ iff $\exists \mathcal{T}' \in \mathcal{L}(\mathcal{A})$. $\mathcal{T} \equiv_{\mathsf{AP}_i} \mathcal{T}'$. The size and index of $\mathcal{B}_i$ are those of $\mathcal{A}$.*

Lemma 26. *[DLM10] Let $\mathcal{A}$ be a $\langle Q, 2^{\mathsf{AP}\cup\{p\}}\rangle$-APT s.t. for any two $2^{\mathsf{AP}\cup\{p\}}$-labelled Q-trees $\mathcal{T}$ and $\mathcal{T}'$ with $\mathcal{T} \equiv_p \mathcal{T}'$, we have $\mathcal{T} \in \mathcal{L}(\mathcal{A})$ iff $\mathcal{T}' \in \mathcal{L}(\mathcal{A})$. Then we can build a $\langle Q, 2^{\mathsf{AP}\cup\{p\}}\rangle$-APT $\mathcal{B}$ s.t. for all $2^{\mathsf{AP}\cup\{p\}}$-labelled Q-tree $\mathcal{T} = \langle T, l\rangle$, it holds: $\mathcal{T} \in \mathcal{L}(\mathcal{B})$ iff $\forall n \in T$. $(p \in l(n)$ iff $\mathcal{T}_n \in \mathcal{L}(\mathcal{A}))$. Then $\mathcal{B}$ has size $O(|\mathcal{A}|)$ and index $\mathsf{idx}(\mathcal{A}) + 1$.*

Lemma 27. *[KV98] Given an $\langle Q, 2^{\mathsf{AP}}\rangle$-NPT $\mathcal{A}$, whether $\mathcal{L}(\mathcal{A}) = \varnothing$ can be checked in time linear in $|\mathcal{A}|^{O(\mathsf{idx}(\mathcal{A}))}$.*

We now sketch our transformation: first, given $\varphi \in \mathsf{Q}^k\mathsf{CTL}$, we extract its maximal subformulas $(\psi_i)_i$ beginning with a propositional quantifier. Then $\varphi = \Phi[(\psi_i)_i]$ with $\Phi \in \mathsf{CTL}$. Our model-checking procedure will first compute the sets of states satisfying ψ_i, for each i, and then apply a CTL mode-checking algorithm. Hence we have reduced our problem to formulas of the form $\exists p.\psi$ of $\mathsf{Q}^k\mathsf{CTL}$. We solve this simplified problem inductively.

When $k = 1$, we then consider a formula φ in $\mathsf{EQ}^1\mathsf{CTL}$: from Lemma 22, we obtain a polynomial-size alternating automaton for the inner CTL formula. Applying Lemma 24 and Lemma 25, we get an NPT $\mathcal{A}_\varphi$ for φ, with size exponential and index linear in $|\varphi|$. by checking emptiness of the product of this automaton with the automaton generating the execution tree of K, we can decide in exponential time whether a given state satisfies φ. Our global algorithm thus also runs in exponential time.

For $k > 1$, formula φ has the form $\exists P.\ \Phi[(\psi_i)_i]$, where formulas ψ_i belong to $\mathsf{EQ}^{k-1}\mathsf{CTL}$ and start with an (existential) propositional quantifier, and Φ is in CTL. Applying the induction hypothesis, we get NPTs $\mathcal{A}_{\psi_i}$ for these formulas, having size at most $(k-1)$-exponential and index at most $(k-2)$-exponential in $|\psi|$. Using Lemma 26, and Lemma 22 for formula Φ, we can build an APT of size at most $(k-1)$-exponential and index at most $(k-2)$-exponential in $|\psi|$ for $\Phi[(\psi_i)_i]$. Then Lemmas 24 and 25 can be used to obtain an APT for φ, having size k-exponential and index $(k-1)$-exponential in $|\varphi|$, which can be used to decide whether a given state satisfies φ. □

Theorem 28. *Model checking $\mathsf{EQ}^k\mathsf{CTL}^*$, $\mathsf{AQ}^k\mathsf{CTL}^*$ and $\mathsf{Q}^k\mathsf{CTL}^*$ under the tree semantics are* (k+1)-$\mathsf{EXPTIME}$*-complete (for positive k).*

Proof. The proof techniques are the same as in the previous proof. Membership requires that we build an automaton for a CTL^* formula, which entails an additional exponential blowup. Hardness is proven by using CTL^* to have $\mathsf{yardstick}_0^n(s,t)$ enforce that the distance between s and t is 2^n. □

Formula- and program-complexity. The reductions above can be made to work with a fixed model. When fixing the formula, the problem becomes much easier (in terms of theoretical complexity):

Theorem 29. *Under the tree semantics, the formula-complexity of model-checking is* k-EXPTIME*-complete for* $\mathsf{EQ}^k\mathsf{CTL}$ *and* $\mathsf{Q}^k\mathsf{CTL}$*, and* k+1-EXPTIME*-complete for* $\mathsf{EQ}^k\mathsf{CTL}^*$ *and* $\mathsf{Q}^k\mathsf{CTL}^*$ *(with* $k \geq 1$*). It is non-elementary for* EQCTL*,* QCTL*,* EQCTL^* *and* QCTL^**.*

Proof. Membership in k-EXPTIME (or k+1-EXPTIME) follows from the general case. We can adapt the hardness proof for $\mathsf{EQ}^k\mathsf{CTL}$ (Theorem 21) to work with a fixed Kripke structure K. Let $\mathcal{M}$ be a (k-1)-EXPSPACE alternating Turing machine $\langle Q, q_0, \delta, F\rangle$ with $Q = \{q_1, \ldots, q_l\}$ and let y be an input word over Σ. Assume (w.l.o.g.) that $\Sigma = \{\alpha, \beta\}$ and $|y| = n$. Now consider the Kripke structure $K = \langle S, s_0, R, \ell\rangle$ with $S = \{s_0, s_1\}$, $R = \{(s_0, s_0), (s_0, s_1), (s_1, s_0), (s_1, s_1)\}$ and $\ell(s_i) = \varnothing$ for $i = 0, 1$. The execution tree $\mathcal{T}_S$ of S is a binary tree. In order to reuse the reduction of Theorem 21, we need to generate an execution labeled with $(Q \times \Sigma) \cup \Sigma \cup \{\#\}$ with constant branching degree $b = 1 + |Q| \cdot (1 + |\Sigma|)$. For this, we label the binary tree $\mathcal{T}_K$ with an atomic proposition p_i to simulate branching degree b with intermediary states: every "main" node will have exactly b successors after a path labeled with p_i. These "main" states are labeled with a proposition in $(Q \times \Sigma) \cup \Sigma \cup \{\#\}$. This encoding is achieved with the following formula where $\Phi_{\mathcal{M},y}$ is a slight variant of the $\mathsf{EQ}^k\mathsf{CTL}$ formula used in the proof of Theorem 21:

$$\Phi = \exists p_i \Big(\Phi_b \ \wedge \ \exists q_1 . \ldots \exists q_l . \exists \alpha . \exists \beta . \Phi_{\mathcal{M},y} \Big)$$

with Φ_b defined as follows:

$$\begin{aligned}\Phi_b = \neg p_i \wedge \mathbf{AG}\,(\neg p_i \Rightarrow \mathbf{AX}\ \mathbf{A} p_i\, \mathbf{U}\ \neg p_i) \wedge \exists n_1 \ldots \exists n_b .\\ \Big[\bigwedge_{j=1\ldots b} \Big(\mathsf{uniq}(n_j) \wedge \mathbf{AG}\,(n_j \Rightarrow \bigwedge_{j' \neq j} \neg n_{j'}) \wedge \mathbf{EX}\ \mathbf{E} p_i\, \mathbf{U}\,(n_j \wedge \neg p_i) \Big) \wedge \\ \forall n_{b+1} . (\,\mathbf{EX}\ \mathbf{E} p_i\, \mathbf{U}\,(n_{b+1} \wedge \neg p_i)) \Rightarrow (\bigvee_{j=1\ldots b} \mathbf{EF}\,(n_j \wedge n_{b+1})) \Big]\end{aligned}$$

In $\Phi_{\mathcal{M},y}$, the only change we have to do is for $\mathsf{yardstick}_0^n(s,t)$: we replace $\mathbf{AX}^n$ by a sequence of n nested formulae of the form "$\mathbf{AX}\,(\,\mathbf{A} p_i\, \mathbf{U}\,(\neg p_i \wedge \ldots))$" in order to find the n-th cell without considering intermediary states.

Clearly Φ is still in $\mathsf{EQ}^k\mathsf{CTL}$ and we have $y \in \mathcal{L}(\mathcal{M})$ iff $K, s_0 \models \Phi$. □

Theorem 30. *Under the tree semantics, the program-complexity of model-checking is* PTIME*-complete for all the fragments of* QCTL^* *containing* $\mathsf{EQ}^1\mathsf{CTL}$ *or* $\mathsf{AQ}^1\mathsf{CTL}$*.*

Proof. Let φ be a QCTL formula and $K = \langle S, s_0, R, \ell\rangle$ be a Kripke structure. The algorithm we used for proving Theorem 21 is based on the construction of a tree automaton $\mathcal{A}_\varphi$ to recognize the S-trees satisfying φ. In such automata, the

transition function is defined with positive Boolean combinations of terms (q, d) (where $d \in S$ is a direction in the tree, and q is a state of the automaton); such a transition forks an execution in state q in the d-successor of the current node. When considering *program complexity*, we aim at building an automaton $\mathcal{A}'_\varphi$ that does not depend on K: it has to recognize any tree satisfying φ, independently of its branching degree or of its set of directions. Such a construction exists for CTL [Wil99], using so-called *symmetric tree automata*; in such automata, the transition function is defined with positive Boolean combinations of terms of the form $\Diamond q$ and $\Box q$; these terms respectively require continuing the run of the automaton at state q in some successor (resp. all successors) of the current node. The use of $\Diamond$ and $\Box$ allows to avoid explicitly considering the possible directions of the tree. On the negative side, symmetric tree automata are inherently alternating, as several executions might be forked in the same branch of the input tree; while this is ok for CTL, it prevents applying the projection construction, which is central to our approach.

To solve the problem for QCTL, we use a weaker variant of symmetric tree automata (using ideas from [JW96,Wal02]): here the transition function associates with every (q, σ) (where q is the current state of the automaton and σ is the labeling of the current node s of the input tree) a set of pairs $(Q, Q') \subseteq Q^2_{\mathcal{A}'_\varphi}$. Writing $Q = \{q_1, \ldots, q_k\}$, the pair (Q, Q') specifies that

- the current node s has at least k successors $s_1, \ldots, s_k$, and the execution in the successor node s_i will continue in state q_i (for $i = 1, \ldots, k$),
- in all the other successors of s, the execution of the automaton will continue in one of the states in Q'.

Such symmetric tree automata are inherently non-deterministic, since given a pair (Q, Q'), only one execution is launched in each direction of the input tree. This kind of tree automata supports the projection operation.

We now explain how we build such an automaton $\mathcal{A}'_\varphi$ for $\varphi \in$ CTL. For this we adapt the standard construction of LTL: $\mathcal{A}'_\varphi$ states are sets of φ-subformulas that are consistent w.r.t. propositional logic, maximal, and locally-consistent with $\mathbf{E}\,\mathbf{U}$ and $\mathbf{A}\,\mathbf{U}$. The transition function for $\mathcal{A}'_\varphi$ ensures consistency between states: for example, if q contains $\mathbf{EX}\,\varphi_1$, $\mathbf{EX}\,\varphi_2$ and $\neg\,\mathbf{EX}\,\varphi_3$, the function $\delta(q, \sigma)$ will contain all possible pairs (Q_1, Q_2) s.t. states in Q_1 ensure $\varphi_1 \wedge \neg\,\varphi_3$ and $\varphi_2 \wedge \neg\,\varphi_3$, or $\varphi_1 \wedge \varphi_2 \wedge \neg\,\varphi_3$, and states in Q_2 ensure $\neg\,\varphi_3$. This incurs an exponential blow up in the size of $\mathcal{A}'_\varphi$, but since we consider program complexity, the size of φ is fixed, and so is the size of $\mathcal{A}'_\varphi$ as it only depends on φ. The complete construction for QCTL can be done with this kind of automaton. Finally it remains to check whether the execution tree of a Kripke structure K is accepted by $\mathcal{A}'_\varphi$, which can be done in polynomial time (in $|K|$). □

5 Using QCTL for specifying properties of multi-agent systems

Extending CTL with propositional quantification has already found several applications for reasoning about complex systems. In this section, we show how it can be used to express important properties of multi-agent systems. More precisely, We show how a model-checking problem involving a multi-agent system (typically a concurrent game) and a property written in ATL_{sc} or Strategy Logic (see below for an introduction to these formalisms) is logspace-reducible to a QCTL model-checking problem.

The reduction is rather natural: we use propositional quantification to label the execution tree with the moves chosen by agents, which thus provides a simple way to describe their strategies. Considering QCTL under the tree semantics allows us to represent general strategies (depending on the whole history of the computation), while considering the structure semantics restricts quantification to memoryless strategies.

Moreover we show that the converse reduction is also possible: a model-checking problem for QCTL can be reduced to a model-checking problem for ATL_{sc}. Finally, we notice that both translations also apply for Strategy Logic.

5.1 Basic definitions

Concurrent game structures Concurrent game structures are a multi-player extension of classical Kripke structures. Their definition is as follows:

Definition 31 ([AHK02]). *A* Concurrent Game Structure *(*CGS *for short)* $\mathcal{C}$ *is a* 7*-tuple* $\langle Q, R, \ell, \mathsf{Agt}, \mathcal{M}, \mathsf{Mov}, \mathsf{Edge}\rangle$ *where:*

- $\langle Q, R, \ell\rangle$ *is a Kripke structure,*
- $\mathsf{Agt} = \{A_1, ..., A_p\}$ *is a finite set of* agents*, (or* players*);*
- $\mathcal{M}$ *is a non-empty set of moves,*
- $\mathsf{Mov}\colon Q \times \mathsf{Agt} \to \mathcal{P}(\mathcal{M}) \setminus \{\varnothing\}$ *defines the set of available moves of each agent in each state,*
- $\mathsf{Edge}\colon Q \times \mathcal{M}^{\mathsf{Agt}} \to R$ *is a transition table associating, with each state* q *and each set of moves of the agents, the resulting transition departing from* q*.*

The size of a CGS $\mathcal{C}$ is $|Q| + |\mathsf{Edge}|$.

The intended behaviour of a CGS is as follows: in a location q, each player A_i in Agt chooses one among her possible moves m_i in $\mathsf{Mov}(\ell, A_i)$; the next transition to be fired is given by $\mathsf{Edge}(q, (m_1, ..., m_p))$. For a state $q \in Q$, we write $\mathsf{Next}(q)$ for the set of all transitions corresponding to possible moves from q, and $\mathsf{Next}(q, A_j, m_j)$, with $m_j \in \mathsf{Mov}(q, A_j)$, for the restriction of $\mathsf{Next}(q)$ to possible transitions from q when player A_j plays move m_j.

We extend Mov and Next to coalitions (*i.e.*, sets of agents) in the natural way:

- given $A \subseteq \mathsf{Agt}$ and $\ell \in \mathsf{Loc}$, $\mathsf{Mov}(\ell, A)$ denotes the set of possible moves for coalition A from ℓ. Those moves m are composed of one single move per agent of the coalition, *i.e.*, $m = (m_a)_{a \in A}$.

– Given $m = (m_a)_{a \in A} \in \mathsf{Mov}(\ell, A)$, we let $\mathsf{Next}(\ell, A, m)$ denote the restriction of $\mathsf{Next}(\ell)$ to locations reachable from ℓ when every player $A_j \in A$ makes the move m_{A_j}.

A *path* of $\mathcal{C}$ is an infinite sequence $\rho = q_0 q_1 \ldots$ of states such that for any i, $q_{i+1} \in \mathsf{Next}(q_i)$. Finite sequences satisfying that property are called *histories*. Let $\rho = q_0 q_1 \ldots q_n$ be a history. The *length* of ρ (denoted with $|\rho|$) is n, and we write $\mathsf{first}(\rho)$ for q_0 and $\mathsf{last}(\rho)$ for q_n. Given a path (resp. history) ρ' s.t. $\mathsf{last}(\rho) = \mathsf{first}(\rho')$, the concatenation of ρ and ρ' is the path (resp. history) $\tau = \rho \cdot \rho' = r_0 r_1 \ldots r_m$ s.t. $\rho = r_0 r_1 \ldots r_n$ and $\rho' = r_n r_{n+1} \ldots r_m$.

A *strategy* for some player $A_i \in \mathsf{Agt}$ is a function f_i that maps any history to a possible move for A_i, *i.e.*, satisfying $f_i(\pi) \in \mathsf{Mov}(\mathsf{last}(\pi), A_i)$. A strategy for a coalition A is a mapping assigning a strategy to each agent in A. The set of strategies for A is denoted $\mathsf{Strat}(A)$. The *domain* $\mathsf{dom}(F_A)$ of $F_A \in \mathsf{Strat}(A)$ is A. Given a coalition B, the strategy $(F_A)_{|B}$ (resp. $(F_A)_{\setminus B}$) denotes the restriction of F_A to the coalition $A \cap B$ (resp. $A \setminus B$). Given two strategies $F \in \mathsf{Strat}(A)$ and $F' \in \mathsf{Strat}(B)$, we define $F \circ F' \in \mathsf{Strat}(A \bigcup B)$ as $(F \circ F')_{|A_j}(\rho) = F_{|A_j}(\rho)$ (resp. $F'_{|A_j}(\rho)$) if $A_j \in A$ (resp. $A_j \in B \setminus A$).

Let ρ be a history. A strategy $F_A = (f_j)_{A_j \in A}$ for some coalition A induces a set of paths from ρ, called the *outcomes* of F_A after ρ, and denoted $\mathsf{Out}(\rho, F_A)$: an infinite path $\pi = \rho \cdot q_1 q_2 \ldots$ is in $\mathsf{Out}(\rho, F_A)$ iff, writing $q_0 = \mathsf{last}(\rho)$, for all $i \geq 0$ there is a set of moves $(m^i_k)_{A_k \in \mathsf{Agt}}$ such that $m^i_k \in \mathsf{Mov}(q_i, A_k)$ for all $A_k \in \mathsf{Agt}$, $m^i_k = f_{A_k}(\pi_{|\rho|+i})$ if $A_k \in A$, and $q_{i+1} \in \mathsf{Next}(q_i, \mathsf{Agt}, (m^i_k)_{A_k \in \mathsf{Agt}})$. Finally, given a strategy F and a history ρ, we define the strategy F^ρ corresponding to the behaviour of F after prefix ρ: it is defined, for any history π with $\mathsf{last}(\rho) = \mathsf{first}(\pi)$, as $F^\rho(\pi) = F(\rho \cdot \pi)$.

Alternating-time temporal logics. We now introduce the extension of ATL with strategy contexts [BDLM09,DLM10]:

Definition 32. *The syntax of* ATL_{sc} *is defined by the following grammar:*

$$\varphi_{\mathsf{state}}, \psi_{\mathsf{state}} ::= p \mid \neg \varphi_{\mathsf{state}} \mid \varphi_{\mathsf{state}} \vee \psi_{\mathsf{state}} \mid \rangle A \langle \cdot \varphi_{\mathsf{state}} \mid \langle \cdot A \cdot \rangle \varphi_{\mathsf{path}}$$
$$\varphi_{\mathsf{path}}, \psi_{\mathsf{path}} ::= \mathbf{X} \varphi_{\mathsf{state}} \mid \varphi_{\mathsf{state}} \mathbf{U} \psi_{\mathsf{state}} \mid \varphi_{\mathsf{state}} \mathbf{W} \psi_{\mathsf{state}}$$

where p ranges over AP and A over 2^{Agt}.

That a formula φ in ATL_{sc} is satisfied by a state q of a CGS $\mathcal{C}$ under a strategy context $F \in \mathsf{Strat}(B)$ (for some coalition B), denoted $\mathcal{C}, q \models_F \varphi$, is defined as follows (omitting Boolean operators and path modalities):

$\mathcal{C}, q \models_F \rangle A \langle \cdot \varphi_{\mathsf{state}}$ iff $\mathcal{C}, q \models_{F_{\setminus A}} \varphi_{\mathsf{state}}$

$\mathcal{C}, q \models_F \langle \cdot A \cdot \rangle \varphi_{\mathsf{path}}$ iff $\exists F_A \in \mathsf{Strat}(A). \forall \rho' \in \mathsf{Out}(q, F_A \circ F).\ \mathcal{C}, \rho' \models_{F_A \circ F} \varphi_{\mathsf{path}}$

In the following we will use $\langle \cdot A \cdot \rangle \varphi_{\mathsf{state}}$ as a shorthand for $\langle \cdot A \cdot \rangle \bot \mathbf{U} \varphi_{\mathsf{state}}$.

Example 33. Consider a system made of several clients $(C_i)_i$ trying to get access to some shared resource. A server S is in charge of ensuring mutual exclusion, and granting access to the resource. The correct functioning of the whole system (in which each client will eventually be able to access the resource if it decides to) can be expressed as

$$\langle\!\cdot S\cdot\!\rangle\, \mathbf{G}\left[\bigwedge_{i\neq j} \neg(\mathsf{grant}_i \wedge \mathsf{grant}_j) \wedge \bigwedge_i \langle\!\cdot A_i\cdot\!\rangle\, \mathbf{F}\, \mathsf{grant}_i\right] \tag{5}$$

5.2 From ATL_{sc} to QCTL* and QCTL model checking

Let $\mathcal{C} = \langle Q, R, \ell, \mathsf{Agt}, \mathcal{M}, \mathsf{Mov}, \mathsf{Edge}\rangle$ be a CGS, and $\mathcal{M}$ be $\{m_1, \ldots, m_k\}$. We consider the following sets of fresh atomic propositions: $\mathsf{P}_Q = \{\mathsf{p}_q \mid q \in Q\}$, $\mathsf{P}^j_{\mathcal{M}} = \{\mathsf{m}^j_1, \ldots, \mathsf{m}^j_k\}$ for every $A_j \in \mathsf{Agt}$, and $\mathsf{P}_{\mathcal{M}} = \bigcup_{A_j\in\mathsf{Agt}} \mathsf{P}^j_{\mathcal{M}}$.

Let $\mathcal{S}_{\mathcal{C}}$ be the Kripke structure $\langle Q, R, \ell_+\rangle$ where for any state q, we have: $\ell_+(q) = \ell(q) \cup \{\mathsf{p}_q\}$. $\mathcal{S}_{\mathcal{C}}$ is the Kripke structure underlying $\mathcal{C}$, in which every state q is labelled with its own atomic proposition p_q. In the following, every labelling function we consider coincides with ℓ_+ on $\mathsf{AP}\backslash\mathsf{P}_{\mathcal{M}}$.

A strategy for an agent A_j can be seen as a function labelling the execution tree of $\mathcal{S}_{\mathcal{C}}$ with $\mathsf{P}^j_{\mathcal{M}}$. More precisely, a strategy for A_j is a labelling function $f_j \colon \mathsf{Exec}^{\mathsf{f}}(\ell) \to \mathsf{P}^j_{\mathcal{M}}$. A memoryless strategy for A_j corresponds to a labelling function $f_j \colon Q \to \mathsf{P}^j_{\mathcal{M}}$, *i.e.*, a labelling of the Kripke structure $\mathcal{S}_{\mathcal{C}}$.

Let $F \in \mathsf{Strat}(C)$ be a strategy context and $\Phi \in \mathsf{ATL}_{sc}$. We reduce the question whether $\mathcal{C}, q \models_F \Phi$ to a model-checking instance for QCTL* over $\mathcal{S}_{\mathcal{C}}$. For this, we define a QCTL* formula $\widehat{\Phi}^C$ inductively; for non-temporal formulas,

$$\widehat{\varphi\wedge\psi}^C = \widehat{\varphi}^C \wedge \widehat{\psi}^C \qquad \widehat{\langle\!\cdot A\cdot\!\rangle\,\varphi}^C = \widehat{\varphi}^{C\smallsetminus A}$$
$$\widehat{\neg\psi}^C = \neg\,\widehat{\varphi}^C \qquad \widehat{P}^C = P$$

For a formula of the form $\langle\!\cdot A\cdot\!\rangle\, \mathbf{X}\,\varphi$ ("until" formulas would be handled similarly) with $A = \{A_{j_1}, \ldots, A_{j_l}\}$, we let:

$$\widehat{\langle\!\cdot A\cdot\!\rangle\, \mathbf{X}\,\varphi}^{\,C} = \exists \mathsf{m}^{j_1}_1 \ldots \mathsf{m}^{j_1}_k \ldots \mathsf{m}^{j_l}_1 \ldots \mathsf{m}^{j_l}_k .\bigwedge_{A_j\in A} \mathbf{AG}\,(\Phi_{\mathsf{strat}}(A_j)) \wedge \mathbf{A}\Big(\Phi^{[A\cup C]}_{\mathsf{out}} \Rightarrow \mathbf{X}\,\widehat{\varphi}^{C\cup A}\Big)$$

where:

$$\Phi_{\mathsf{strat}}(A_j) = \bigvee_{q\in Q}\Big(\mathsf{p}_q \wedge \bigvee_{m_i\in\mathsf{Mov}(q,A_j)} (\mathsf{m}^j_i \wedge \bigwedge_{l\neq i} \neg\,\mathsf{m}^j_l)\Big)$$

$$\Phi^{[A]}_{\mathsf{out}} = \mathbf{G} \bigwedge_{\substack{q\in Q\\ m\in\mathsf{Mov}(q,A)}} \Big((\mathsf{p}_q \wedge P_m) \Rightarrow \mathbf{X}\,\big(\bigvee_{q'\in\mathsf{Next}(q,A,m)} \mathsf{p}_{q'}\big)\Big)$$

where m is a move $(m^j)_{A_j\in A} \in \mathsf{Mov}(q, A)$ for A and P_m is the propositional formula $\bigwedge_{A_j\in A} m^j$ characterizing m. Formula $\Phi_{\mathsf{strat}}(A_j)$ ensures that, the labelling

of propositions m_i^js describes a feasible strategy for A_j. Formula $\Phi_{\mathsf{out}}^{[A]}$ characterizes the outcomes of the strategy for A that is described by the atomic propositions in the model. Note that $\Phi_{\mathsf{out}}^{[A]}$ is based on the transition table Edge of $\mathcal{C}$.

The correctness of the reduction is based on the following lemma where we identify an execution in a structure with its corresponding path in the execution tree in order to simplify the notation:

Lemma 34. *Let $A \subseteq \mathsf{Agt}$ be a coalition and q be a state in Q. Let $\mathcal{T}'$ be the execution tree $\mathcal{T}_{\mathcal{S}_\mathcal{C}}(q)$ with a labelling function ℓ' s.t. for every $\pi \in \mathsf{Exec}^{\mathsf{f}}(q)$ and $a \in A$, $\ell'(\pi) \cap \mathsf{P}_{\mathcal{M}}^a$ is a singleton. Let ρ be an execution in $\mathcal{T}'$. Then*

$$\mathcal{T}', \rho \models \Phi_{\mathsf{out}}^{[A]} \quad \Leftrightarrow \quad \rho \in \mathsf{Out}(q, F_A)$$

where F_A is the strategy labelled in $\mathcal{T}'$, obtained as follows: for every $a \in \mathsf{A}$ and $i \geq 0$, if $\ell'(\rho_i) \ni \mathsf{m}_j^a$, then $F_a(\rho_i)$ is m_j.

Then:

Theorem 35. *Let q be a state in $\mathcal{C}$. Let Φ be an ATL_{sc} formula and F be a strategy context for some coalition C. Let $\mathcal{T}'$ be the execution tree $\mathcal{T}_{\mathcal{S}_\mathcal{C}}(q)$ with a labelling function ℓ' s.t. for every $\pi \in \mathsf{Exec}^{\mathsf{f}}(q)$ of length i and any $A_j \in C$, $\ell'(\pi) \cap \mathsf{P}_{\mathcal{M}}^j = \mathsf{m}_i^j$ iff $F(\pi)_{|A_j} = m_i$. Then $\mathcal{C}, q \models_F \Phi$ iff $\mathcal{T}', q \models \widehat{\Phi}^C$.*

Proof. The proof is done by structural induction over Φ. Boolean cases are trivial.

Consider $\Phi = \langle\!\cdot A \cdot\!\rangle \mathbf{X}\, \varphi$. Assume $\mathcal{C}, q \models_F \langle\!\cdot A \cdot\!\rangle \mathbf{X}\, \varphi$. Therefore there exists $F_A \in \mathsf{Strat}(A)$ s.t. for any $\rho \in \mathsf{Out}(q, F_A \circ F)$, we have $\mathcal{C}, \rho(1) \models_{F_A \circ F^{(q)}} \varphi$.

Let $\mathcal{T}''$ be the execution tree $\mathcal{T}_{\mathcal{S}}(q)$ with a new labeling ℓ'' that extends ℓ' in the following way: for every $\pi \in \mathsf{Exec}^{\mathsf{f}}_{\mathcal{S}_\mathcal{C}}\mathcal{T}(q)$, we have $\ell''(\pi) = \ell'(\pi) \cup \{\mathsf{m}_j^a\}$ if $F_A(\pi)_{|a} = m_j$ [9]. Now let ρ' be an execution in $\mathcal{T}''$. Assume we have $\mathcal{T}'', \rho' \models \Phi_{\mathsf{out}}^{[A \cup C]}$. From Lemma 34, we deduce $\rho' \in \mathsf{Out}(q, F_A \circ F)$, and then $\mathcal{C}, \rho'(1) \models_{F_A \circ F^{(q)}} \varphi$ (see above). It remains to apply the induction hypothesis to obtain $\mathcal{T}'', \rho'_{\leq 1} \models \widehat{\varphi}^{A \cup C}$ (interpreting a formula in the subtree with root $\rho'(1)$ or in $\mathcal{T}''$ from $\rho'_{\leq 1}$ does not matter here). And clearly it gives that $\mathcal{T}', q \models \widehat{\Phi}^C$.

For the other direction, we build a strategy from the labeling and it works in the same way: the first part of formula $\widehat{\Phi}^C$ ensures that the labeling for m_i^as describes a correct strategy and the second part ensures that every outcome has to verify $\mathbf{X}\, \varphi$. We use the same method to prove the result for $\Phi = \langle\!\cdot A \cdot\!\rangle (\varphi \mathbf{U}\, \psi)$.

Finally assume $\Phi = \cdot\rangle A \langle\cdot\, \varphi$. Then $\widehat{\Phi}^C$ is $\widehat{\varphi}^{C \setminus A}$. Indeed in $\widehat{\varphi}^{C \setminus A}$, we do not take into account the value of atomic propositions encoding the moves for agents in A to characterize the outcomes of the current strategy context: this is precisely the meaning of $[A]$ operator. □

Thus a model checking problem $\mathcal{C}, q \models \Phi$ can be reduced to the problem $\mathcal{S}_\mathcal{C}, q \models_t \Phi'$ with $\Phi \in \mathsf{QCTL}^*$. This reduction, together with the algorithm developed earlier in this paper, provides a non-elementary model-checking algorithm for ATL_{sc} (which is similar to the algorithm we proposed in [DLM10]).

[9] Remember we identify the execution in a structure and in its execution tree

Remark 36. The translation above assumes the tree semantics. However, it also makes sense in the structure semantics, where quantification then corresponds to the selection of a *memoryless* strategy. A variant of Theorem 35 can be stated for the structure semantics for QCTL and memoryless strategy quantification for ATL_{sc}.

Remark 37. Our reduction above is into QCTL^* but we can use Proposition 13 to get an equivalent QCTL formula. This may increase the quantifier height of the formula. For the tree semantics, a direct translation into QCTL exists: instead of using $\Phi_{\mathsf{out}}^{[A]}$, we can use an extra atomic proposition p_{out} for labelling outcomes. This yields a QCTL formula with the same quantifier height.

5.3 From QCTL to ATL_{sc} model checking

Let Φ be a QCTL formula and $\mathcal{S} = \langle Q, R, \ell \rangle$ be a Kripke structure. W.l.o.g., we assume that every proposition quantifier in Φ deals with a fresh proposition. We use $\mathsf{AP}_f(\Phi)$ (resp. $\mathsf{AP}_{\mathcal{Q}}(\Phi)$) to denote the set of free (resp. quantified) proposition in Φ. Assume $\mathsf{AP}_{\mathcal{Q}}(\Phi) = \{P_1, \dots, P_k\}$. Now we define a turn-based CGS $\mathcal{C}_{\mathcal{S}}$ and an ATL_{sc} formula $\widetilde{\Phi}$ such that $\mathcal{S}, q \models_t \Phi$ iff $\mathcal{C}_{\mathcal{S}}, q \models \widetilde{\Phi}$.

The CGS $\mathcal{C}_{\mathcal{S}} = \langle Q', R', \ell', \mathsf{Agt}, \mathcal{M}, \mathsf{Mov}, \mathsf{Edge} \rangle$ is defined as follows. The set of agents is $\mathsf{Agt} = \{A_0, \dots, A_k\}$: A_0 is in charge of selecting the transitions in $\mathcal{S}$, and each A_i with $i \geq 1$ has to decide the truth value of P_i. The set of states is $Q' = Q \cup \{c_{q,i} \,|\, i = 1, \dots, k\} \cup \{p_i \,|\, i = 0, \dots, k\}$: every state $q \in Q$ is controlled by A_0 while every state $c_{q,i}$ is controlled by A_i. States p_i only carry a self-loop, and then are not explicitly controlled. The transition set R' contains every transition $(q, q') \in R$, and we also add $(q, c_{q,i})$, $(c_{q,i}, p_i)$ and $(c_{q,i}, p_0)$ for $i = 1, \dots, k$ and $q \in Q$. The labelling ℓ' is the following one: $\ell'(q) = \ell(q) \cup \{P_Q\}$ if $q \in Q$ (P_Q is assumed to be a fresh atomic proposition), $\ell'(c_{q,i}) = \ell'(p_0) = \varnothing$, and $\ell'(p_i) = P_i$ if $i \geq 1$. In a state $q \in Q$, A_0 can choose either a successor state q' (*i.e.* an $\mathcal{S}$ transition (q, q')) or some $c_{q,i}$, the latter choice being used to check whether P_i holds true in q. Indeed in $c_{q,i}$, A_i has two available moves: move m_1 goes to P_i, and while mode m_0 goes to P_0. Thus as soon as A_i has fixed his strategy, $c_{q,i}$ has a unique successor and this will encode the labelling for P_i. Note also that for any path in $\mathcal{C}_{\mathcal{S}}$ of the form $\rho \cdot c_{q,i}$, ρ is a path in $\mathcal{S}$ ending in q. Finally note that as $\mathcal{C}_{\mathcal{S}}$ is a turn-based CGS, its size is in $O(|Q| \cdot |\Phi| + |R|)$, *i.e.* in $O(|\mathcal{S}| \cdot |\Phi|)$. Now we define $\widetilde{\Phi}$ as follows:

$$\widetilde{\exists P_i.\varphi} = \langle\!\langle A_i \rangle\!\rangle \widetilde{\varphi} \qquad \widetilde{\varphi \wedge \psi} = \widetilde{\varphi} \wedge \widetilde{\psi}$$

$$\widetilde{P_i} = \langle\!\langle A_0 \rangle\!\rangle \mathbf{X} \, \langle\!\langle A_0 \rangle\!\rangle \mathbf{X} \, P_i \qquad \widetilde{\neg \psi} = \neg \widetilde{\varphi}$$

$$\widetilde{\mathsf{E}\varphi \, \mathbf{U} \, \psi} = \langle\!\langle A_0 \rangle\!\rangle (P_Q \wedge \widetilde{\varphi}) \, \mathbf{U} \, (P_Q \wedge \widetilde{\psi}) \qquad \widetilde{P} = P$$

$$\widetilde{\mathsf{E}\mathbf{G}\varphi} = \langle\!\langle A_0 \rangle\!\rangle \mathbf{G} \, (P_Q \wedge \widetilde{\varphi})$$

where we assume $P \notin \mathsf{AP}_{\mathcal{Q}}(\Phi)$. The size of $\widetilde{\Phi}$ is in $O(|\Phi|)$. We state the correctness of the reduction as follows:

Proposition 38. *Let Φ be a* QCTL *formula with* $\mathsf{AP}_{\mathcal{Q}}(\Phi) = \{P_1, \ldots, P_k\}$ *and ψ be a Φ-subformula. Let $\mathcal{I}$ be the indexes of propositions in* $\mathsf{AP}_f(\psi) \cap \mathsf{AP}_{\mathcal{Q}}(\Phi)$*. Let* $\mathcal{S} = \langle Q, R, \ell \rangle$ *be a KS and* $q_0 \in Q$*. Let $\mathcal{T}$ be an unwinding* $\mathcal{T}_{\mathcal{S}}(q_0)$ *with a labelling function $\ell_{\mathcal{T}}$ that extends ℓ for* $\{P_i \mid i \in \mathcal{I}\}$*. Let F be the strategy context* $(f_{A_i})_{i \in \mathcal{I}}$ *such that:* $F_{A_i}(\rho \cdot c_{q,i}) = \mathsf{m}_1$ *iff* $\ell_{\mathcal{T}}(\rho) \ni P_i$ *for every $\mathcal{S}$-path ρ. Then we have:*

$$\mathcal{T}, q \models_s \psi \quad \textit{iff} \quad \mathcal{C}_{\mathcal{S}}, q \models_F \widetilde{\psi}$$

Proof. The proof is based on the fact that any strategy for agent A_i from a given state q in $\mathcal{C}_{\mathcal{S}}$ corresponds to a (unique) P_i labelling of $\mathcal{T}_{\mathcal{S}}(q)$. Indeed such a strategy is a mapping from paths of the form $\rho \cdot c_{q,i}$ with $\rho \in Q^+$ (remember $\mathcal{C}$ is a turn-based game where A_i only plays in $c_{q,i}$ states) and as noticed above, we have: for any such a $\mathcal{C}_{\mathcal{S}}$ path $\rho \cdot c_{q,i}$, ρ is a path in $\mathcal{S}$ ending in q which implies that ρ is a state of $\mathcal{T}_{\mathcal{S}}(q)$. Given this observation, the proof is direct. □

The above reduction involves $k+1$ players, but quantification on strategies of A_0 can be changed to $\langle\cdot\varnothing\cdot\rangle$. In the end, we get:

Theorem 39. *Model-checking the fragment of* ATL_{sc} *with at most k non-trivial nested strategy quantifiers is* k-EXPTIME*-complete.*

5.4 Extension to Strategy Logics.

The reduction from ATL_{sc} model-checking to QCTL^* model-checking can be adapted for other formalisms. Especially it can be done for Strategy Logics, introduced in [CHP07][10] and further extended in [MMV10]. In this framework we can assign strategies to some variables and then associate them with agents.This can be done in FQCTL^* by quantifying over propositions m_i^x (such a labelling will define the strategy x) and then use a slightly modified version of $\Phi^{\square}_{\mathsf{out}}$ to ensure that agent a plays accordingly to strategy x. The use of FQCTL^* is due to SL's ability to quantify over strategies within path formulas. As explained in Remark 15, FQCTL^* can be translated into QCTL when considering the tree semantics. This entails the following result (correcting a wrong claim in [MMV10, Theorem 4.2] which states that SL model checking can be done in 2-EXPTIME):

Theorem 40. *The model-checking problems for* QCTL*,* ATL_{sc} *and* SL *are inter-reducible (in logarithmic space). They all are non-elementary.*

6 Conclusions and future works

We have proposed a complete picture of CTL extended with propositional quantifiers w.r.t. expressiveness and model-checking. On the expressiveness side, we proved how adding quantification on top of CTL fills in the gap between

[10] Notice that Strategy Logic in [CHP07] requires formulas with a temporal modality to be closed (they are not allowed to involve strategies quantified earlier). Under that restriction, our reduction does *not* apply.

temporal logics and monadic second-order logic. As for model checking, we exhaustively characterized the complexity of QCTL and its variants, completing the earlier results from [Kup95,Fre01]. Finally, we provided an application (which was our original motivation) of QCTL for reasonning about multi-agent systems. Satisfiability of fragments of QCTL^* is part of our future work.

References

[AHK02] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *J. ACM*, 49(5):672–713, 2002.

[BDLM09] Th. Brihaye, A. Da Costa, F. Laroussinie, and N. Markey. ATL with strategy contexts and bounded memory. In LFCS'09, LNCS 5407, p. 92–106. Springer, 2009.

[BG99] G. Bruns and P. Godefroid. Model checking partial state spaces with 3-valued temporal logics. In CAV'99, LNCS 1633, p. 274–287. Springer, 1999.

[CE82] E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In 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.

[CHP07] K. Chatterjee, T. A. Henzinger, and N. Piterman. Strategy logic. In CONCUR'07, LNCS 4703, p. 59–73. Springer, 2007.

[Cou90] B. Courcelle. The monadic second-order logic of graphs. i: recognizable sets of finite graphs. *Inf.&Comp.*, 85(1):12–75, 1990.

[DLM10] A. Da Costa, F. Laroussinie, and N. Markey. ATL with strategy contexts: Expressiveness and model checking. In FSTTCS'10, LIPIcs 8, p. 120–132. Leibniz-Zentrum für Informatik, 2010.

[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. *J. ACM*, 33(1):151–178, 1986.

[ES84] E. A. Emerson and A. P. Sistla. Deciding full branching time logic. *Inf.&Cont.*, 61(3):175–201, 1984.

[Fre01] T. French. Decidability of quantified propositional branching time logics. In AJCAI'01, LNCS 2256, p. 165–176. Springer, 2001.

[Fre06] T. French. *Bisimulation Quantifiers for Modal Logics*. Phd thesis, School of Computer Science & Software Engineering, University of Western Australia, 2006.

[Got95] G. Gottlob. NP trees and Carnap's modal logic. *J. ACM*, 42(2):421–457, 1995.

[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.

[JW96] D. Janin and I. Walukiewicz. On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic. In CONCUR'96, LNCS 1119, p. 263–277. Springer, 1996.

[KMTV00] O. Kupferman, P. Madhusudan, P. S. Thiagarajan, and M. Y. Vardi. Open systems in reactive environments: Control and synthesis. In CONCUR'00, LNCS 1877, p. 92–107. Springer, 2000.

[KP95] Y. Kesten and A. Pnueli. A complete proof systems for QPTL. In LICS'95, p. 2–12. IEEE Comp. Soc. Press, 1995.

[Kup95] O. Kupferman. Augmenting branching temporal logics with existential quantification over atomic propositions. In CAV'95, LNCS 939, p. 325–338. Springer, 1995.

[KV98] O. Kupferman and M. Y. Vardi. Weak alternating automata and tree automata emptiness. In STOC'98, p. 224–233. ACM Press, 1998.

[KVW00] O. Kupferman, M. Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model-checking. *J. ACM*, 47(2):312–360, 2000.

[MMV10] F. Mogavero, A. Murano, and M. Y. Vardi. Reasoning about strategies. In 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. *Inf.&Comp.*, 184(1):147–159, 2003.

[MS85] D. E. Muller and P. E. Schupp. Alternating automata on infinite objects, determinacy and Rabin's theorem. In EPIT'84, LNCS 192, p. 99–107. Springer, 1985.

[MS87] D. E. Muller and P. E. Schupp. Alternating automata on infinite trees. *TCS*, 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. *TCS*, 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 ATVA'07, LNCS 4762, p. 253–267. Springer, 2007.

[Pnu77] A. Pnueli. The temporal logic of programs. In 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 SOP'82, LNCS 137, p. 337–351. Springer, 1982.

[RP03] S. Riedweg and S. Pinchinat. Quantified μ-calculus for control synthesis. In MFCS'03, LNCS 2747, p. 642–651. Springer, 2003.

[Sch03] Ph. Schnoebelen. The complexity of temporal logic model checking. In AIML'02, p. 481–517. King's College Publications, 2003.

[Sis83] A. P. Sistla. *Theoretical Issues in the Design and Verification of Distributed Systems*. PhD thesis, Harvard University, Cambridge, Massachussets, USA, 1983.

[SVW87] A. P. Sistla, M. Y. Vardi, and P. Wolper. The complementation problem for Büchi automata with applications to temporal logics. *TCS*, 49:217–237, 1987.

[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 STOC'82, p. 137–146. ACM Press, 1982.

[Wag90] K. W. Wagner. Bounded query classes. *SIAM J. Computing*, 19(5):833–846, 1990.

[Wal02] I. Walukiewicz. Monadic second order logic on tree-like structures. *TCS*, 275(1-2):311–346, 2002.

[WHY11] F. Wang, C.-H. Huang, and F. Yu. A temporal logic for the interaction of strategies. In CONCUR'11, LNCS 6901, p. 466–481. Springer, 2011.

[Wil99] T. Wilke. CTL$^+$ is exponentially more succinct than CTL. In FSTTCS'99, LNCS 1738, p. 110–121. Springer, 1999.