Quantified CTL: expressiveness and model checking

Arnaud Da Costa, François Laroussinie,
Nicolas Markey

January 2012

Research report LSV-12-02

Quantified CTL: expressiveness and model checking

Arnaud Da Costa
LSV – ENS Cachan & CNRS
Email: dacosta@lsv.ens-caachan.fr

François Laroussinie
LIAFA – Univ. Paris Diderot & CNRS
Email:francoisl@liafa.jussieu.fr

Nicolas Markey
LSV – ENS Cachan & CNRS
Email: markey@lsv.ens-caachan.fr

Abstract—**While it was defined long ago, the extension of CTL with quantification over atomic propositions has never been studied extensively. Considering two different semantics (depending whether propositional quantification refers to the Kripke structure or to its unwinding tree), we study its expressiveness (showing in particular that QCTL coincides with Monadic Second-Order Logic for both semantics) and 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.**

I. Introduction

Temporal logics. Temporal logics extend propositional logics with modalities for specifying constraints on the order of events in time. Since [1], [2], [3], they have received much attention from the computer-aided-verification community, since they fit particularly well for expressing and automatically verifying (*model checking*) properties of reactive systems.

Two important families of temporal logics have been considered: linear-time temporal logics (*e.g.* LTL [1]) can be used to express properties of one single execution of the system under study, while branching-time temporal logics (*e.g.* CTL [2], [3] and CTL* [4]) consider the execution tree as a whole. Since the 90s, many extensions of these logics have been introduced, of which alternating-time temporal logics (such as ATL, ATL* [5]) extend CTL towards the study of open systems (in which several agents are involved).

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* [6], [7]. 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 [8], for the two-alternation fragment [9] and for the full extension [10]; 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[1]. 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 II-D).

	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 (especially for the verification of multi-agent systems), games are not purely antagonist, but involve several independent systems, each having its own objective. This has given rise for instance to different notions of *equilibria*, such as Nash equilibria [11]. Recently, several extensions of ATL have been defined to cope with this. Among those, our logic ATL$_{sc}$ [12] extends ATL with *strategy contexts*, which provides a way of expressing interactions between strategies. Other similar approaches include Strategy Logics [13], [14] or (B)SIL [15].

Interestingly, the model-checking problem for these extensions of ATL for non-zero-sum objectives (also for Strategy Logics) can be seen as a QCTL model-checking problem[2]: 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 V. Notice that while the

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

[2]Notice that the link between games and propositional quantification already emerges in QDμ [16], which extends the *decision μ-calculus* with some flavour of propositional quantification. Also, the main motivation of [9] for studying the two-alternation fragment of QCTL is a hardness result for the control and synthesis of open systems.

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 [6], [17], [18], where the model-checking problem for the k-alternation fragment was settled complete for $\mathsf{k\text{-}EXPSPACE}$. In the branching-time setting, CTL and CTL^* with existential quantification was studied in [8], where model-checking is shown NP- and PSPACE-complete resp. (for the structure semantics) and $\mathsf{EXPTIME}$- and $\mathsf{2\text{-}EXPTIME}$-complete resp. (for the tree semantics). The two-alternation fragment was then studied in [9] (only for the tree semantics): model-checking is $\mathsf{2\text{-}EXPTIME}$- and $\mathsf{3\text{-}EXPTIME}$-complete, respectively for CTL or CTL^*. Finally, the full extension (with arbitrary quantification) was studied in [10] for satisfiability.

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 [10]. Yet another semantics is considered in [19], where quantification is handled by taking a product with another structure. There is no obvious link (in terms of expressiveness) between all these semantics. Finally, model checking for a slightly different (and less expressive) version of QCTL in the structure semantics is studied in [20].

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) [21], [22].

II. Preliminaries

A. 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 (possibly infinite) set of states, $R \subseteq Q \times Q$ is a total[3] binary relation and $\ell\colon Q \to 2^{\mathsf{AP}}$ is a labelling function.

An execution (or a path) in $\mathcal{S}$ is an infinite sequence $\rho = (q_i)_{i\in\mathbb{N}}$ such that $(q_i, q_{i+1}) \in R$ 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 $\rho(i)$ the for 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 labeling function.

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

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 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. $m\cdot s \in \mathsf{Exec}^{\mathsf{f}}(q)$.

For a (labelling) function $\ell\colon Q \to \mathsf{AP}$ and $P \subseteq \mathsf{AP}$, we write $\ell \cap P$ for the (labelling) function defined as $(\ell \cap P)(q) = \ell(q) \cap P$ for all $q \in Q$.

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 $\mathcal{S} \equiv_P \mathcal{S}'$) if the Kripke structures $\langle Q, R, \ell \cap P\rangle$ and $\langle Q', R', \ell' \cap P\rangle$ are equal.

B. 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: $\top \stackrel{\mathsf{def}}{=} p \vee \neg\, p$, $\bot \stackrel{\mathsf{def}}{=} \neg\,\top$, $\mathbf{F}\,\varphi \stackrel{\mathsf{def}}{=} \top\,\mathbf{U}\,\varphi$, $\mathbf{G}\,\varphi \stackrel{\mathsf{def}}{=} \neg\,\mathbf{F}\,\neg\,\varphi$, and $\forall p\cdot\varphi \stackrel{\mathsf{def}}{=} \neg\,\exists p\cdot\neg\,\varphi$.

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 $L_1[L_2] = \{\varphi[(\psi_i)_i] \mid \varphi \in L_1,\ (\psi_i)_i \in L_2\}$.

C. Structure- and tree semantics

Formulas of the form $\exists p.\varphi$ can be interpreted in different manners (see [8], [10], [19]). Here we consider two semantics: the *structure semantics* and the *tree semantics*.

1) Structure semantics. Given a QCTL^* (or 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 formula φ holds at q under the structure semantics. This is defined as for CTL^*, with the following addition (see Appendix A for full definition):

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

Example 6: As an example, consider the formula

$$\mathsf{selfloop} \overset{\text{def}}{=} \forall z.(z \Rightarrow \mathbf{EX}\, z). \tag{1}$$

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

$$\mathsf{uniq}(\varphi) \overset{\text{def}}{=} \mathbf{EF}\,(\varphi) \wedge \forall z.\Big(\mathbf{EF}\,(\varphi \wedge z) \Rightarrow \mathbf{AG}\,(\varphi \Rightarrow z)\Big). \tag{2}$$

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.

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

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

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

3) Equivalences between QCTL^* *formulas.* We consider two kinds of equivalences depending 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).

D. 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[4]

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 $\mathsf{Q}^k\mathsf{CTL}^*$.

III. 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 12).

A. Prenex normal form

Proposition 9: Every QCTL formula is equivalent (for both semantics) to a formula in prenex normal form.

Sketch of proof: In [23], the existence of a prenex normal form for a branching-time modal logic with some quantification over propositions is proven for the tree semantics. This proof can be adapted to QCTL for both semantics (see Appendix B). Here we simply sketch the technique we use to extract a bloc of quantifiers $\mathcal{Q} = Q_1p_1.Q_2p_2 \ldots Q_kp_k$ out of a temporal modality (the case of Boolean operators is straightforward). First, considering modality $\mathbf{EX}$, we have:

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

Here variable z (which we assume does not appear in $\mathcal{Q}$) is used to mark the immediate successor that witnesses $\mathcal{Q}.\varphi$.

[4]Notice that the logics named EQCTL and EQCTL^* defined in [8] 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 $\mathsf{EQ}^1\mathsf{CTL}^*$.

It then remains to extract the universal quantifier occurring in $\mathsf{uniq}(z)$ from the conjunction.

We use a similar approach for $\mathbf{E}\,\mathbf{U}$ and $\mathbf{EG}$. Here, instead of labeling a single successor state, we have to label a finite path, which we do as follows:

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

where $\mathsf{dpath}(z_0, z_2)$ ensures that z_0 labels a path ending in a state labeled with z_2. Note that there is no QCTL formula (under the structure semantics) to constrain a variable to label a general path, but here we can restrict ourselves to *direct* paths (with no loop, and ending in z_2 as soon as possible). ■

B. 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 case, 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 KS $\langle Q, R, \ell\rangle$.

MSO is built with first-order (or individual) variables for nodes or vertices (which we denote with lowercase letters $x, y, \ldots$), monadic second-order variables for sets of nodes (denoted with uppercase letters $X, Y, \ldots$). 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 $\varphi(x_1,\ldots,x_n,X_1,\ldots,X_k)$ to state that $x_1,\ldots,x_n$ and $X_1,\ldots,X_k$ may appear free (*i.e.* not within the scope of a quantifier) in φ. A sentence (or closed formula) contains no free variable. We use the standard semantics for MSO and write $\mathcal{M}, s_1,\ldots,s_n,S_1,\ldots,S_k \models \varphi(x_1,\ldots,x_n,X_1,\ldots,X_k)$ when φ holds on $\mathcal{M}$ when s_i (resp. S_j) is assigned to the variable x_i (resp. X_j) for $i = 1,\ldots,n$ (resp. $j = 1,\ldots,k$).

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 will represent 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 will 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.

Proposition 10: Under both semantics, MSO and QCTL have the same expressive power.

Sketch of 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, every MSO formula $\Phi(x)$ can be translated into an equivalent QCTL formula $\widehat{\Phi}$. The complete definition of $\widehat{\Phi}$ is given in Appendix C, the main idea is to use QCTL propositional quantifications to encode both first-order and second-order quantification in Φ (but in the first-order case, we require that only one state is labeled by the dedicated proposition). Then an MSO subformula of the form $x_i \in X_j$ will be rewritten in QCTL by $\mathbf{EF}\,(\mathsf{p}_{x_i} \wedge \mathsf{p}_{X_j})$ where p_{x_i} (resp. p_{X_j}) is the proposition associated with x_i (resp. X_j). For formula of the form $\mathsf{Edge}(x_i, x_j)$, we use $\mathbf{EF}\,(\mathsf{p}_{x_i} \wedge \mathbf{EX}\,\mathsf{p}_{x_j})$. And a formula $x_i = x_j$ will be replaced by $\mathbf{EF}\,(\mathsf{p}_{x_i} \wedge \mathsf{p}_{x_j})$. Other cases use the same ideas. ■

Remark 11: One can also notice that it is easy to express fixpoint operators with QCTL in both semantics, thus μ-calculus can be translated into QCTL. This provides another proof of the previous result for the tree semantics, since the μ-calculus extended with counting capabilities has the same expressiveness as MSO on trees [24].

C. QCTL *and* QCTL^*

Finally, we show that QCTL^* and QCTL are equally expressive for both semantics:

Proposition 12: In the tree and structure semantics, every QCTL^* formula is equivalent to some QCTL formula.

Proof: The result for the tree semantic has been shown in [10]. Here we give a translation that is correct for both semantics. Consider a QCTL^* formula Φ. The proof is done by induction over the number k of subformulas of Φ that are not in QCTL. If $k = 0$, Φ already belongs to QCTL. Otherwise let ψ be one of the smallest Φ-subformulas in $\mathsf{QCTL}^* \setminus \mathsf{QCTL}$. Let α_is with $i = 1,\ldots,m$ be the largest ψ-subformulas belonging to QCTL. These are state formulas, so that ψ is equivalent (w.r.t. both semantics) to:

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

Let Ω be $\psi[(\alpha_i \leftarrow p_i)_{i=1,\ldots,m}]$. Then Ω is a CTL^* formula: every subformula of the form $\exists p.\xi$ in ψ belongs to some QCTL formula α_i, since ψ is one of the smallest $\mathsf{QCTL}^* \setminus \mathsf{QCTL}$ subformula. And every CTL^* formula is equivalent to some μ-calculus formula, so that Ω is equivalent to some QCTL formula $\widetilde{\Omega}$ (see Remark. 11). Hence

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

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 $\mathsf{QCTL}^* \setminus \mathsf{QCTL}$, so that the induction hypothesis applies. ■

From Propositions 9 to 12, we get:

Corollary 13: In both semantics, EQCTL, QCTL and QCTL^* and MSO are equally expressive.

Remark 14: In [10], Tim French considers a variant of QCTL^* (which we call FQCTL^*), with propositional quantification within path formulas: $\exists p.\ \varphi_{\mathsf{path}}$ is added in the definition of path formulas. The semantics is defined as follows:

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

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

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

This formula expresses the existence of an (infinite) path along which, between any two occurrences of the same state, all the other reachable states will be visited. This precisely corresponds to the existence of a Hamilton cycle, which is known not to be expressible in MSO [25, Cor. 6.3.5]. However, note that the existence of a Hamilton cycle can be expressed in Guarded Second Order Logic GSO[5], in which quantification over sets of *edges* is allowed (in addition to quantification over sets of states). But still, FQCTL^* is strictly more expressive than GSO, as it is easy to modify the above formula in order to express the existence of *Euler* cycles:

$$\begin{aligned}\mathbf{EG}\Big(\exists x.\exists y.\forall x'.\forall y'.\Big[&\mathsf{trans}(x,y) \wedge \mathsf{trans}(x',y') \wedge \\ &\mathsf{next\_trans}(x,y) \wedge \neg\, \mathsf{next\_trans}(x',y')\Big] \\ &\Rightarrow \mathbf{X}\,(\neg\, \mathsf{next\_trans}(x,y)\,\mathbf{U}\,\mathsf{next\_trans}(x',y'))\Big)\end{aligned}$$

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

Theorem 15: Under the structure semantics, FQCTL^* is more expressive than QCTL^* and MSO.

Nevertheless FQCTL^* model checking (see next section) is decidable: for the tree semantics, it suffices to translate FQCTL^* to QCTL, as proposed by French [10]. The problem in the structure semantics can then be encoded in the tree semantics: for this we need to first extend the labelling of the Kripke structure $\mathcal{S}$ with fresh propositions, one per state. Let $\mathcal{S}'$ be such an extension (notice that the existence of an Euler path in such a Kripke structure can be expressed in CTL). Then any quantification $\exists p.\varphi$ in some FQCTL^* formula Φ (for the structure semantics) is considered in the tree semantics, with the requirement that any two copies of the same state receive the same labelling. We then have: $\mathcal{S}, q \models_s \Phi$ iff $\mathcal{S}', q \models_t \widehat{\Phi}^{\mathcal{S}'}$.

[5]This logic is called MS_2 in [26].

IV. 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 [6] 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[8]. Hardness results for $\mathsf{EQ}^2\mathsf{CTL}$ and $\mathsf{EQ}^2\mathsf{CTL}^*$ under the tree semantics can be found in [9]. Here we extend these results to all the fragments of QCTL^* we have defined.

A. Structure semantics

1) Formulas in prenex normal form. We first consider the simpler case of prenex-normal-form formulas.

Theorem 16: Under the structure semantics, $\mathsf{EQ}^k\mathsf{CTL}$ model checking is Σ_k^{P}-complete, and $\mathsf{AQ}^k\mathsf{CTL}$ model checking is Π_k^{P}-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. It thus suffices to 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 Σ_k^{P}-complete [27]:

Problem:	Σ_k^{P}**SAT**
Input:	k families of variables $U_i = \{u_1^i, \ldots, u_n^i\}$, 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 Σ_k^{P} is proved inductively: an $\mathsf{EQ}^1\mathsf{CTL}$ instance $\exists u_1^1 \ldots \exists u_k^1.\ \varphi$ can be solved in $\mathsf{NP} = \Sigma_1^{\mathsf{P}}$ by non-deterministically picking a labelling of the Kripke structure under study with atomic propositions u_1^1 to u_k^1, and then checking (in polynomial time) whether the CTL formula φ holds true in the resulting Kripke structure. Similarly, an $\mathsf{EQ}^k\mathsf{CTL}$ formula $\exists u_1^1 \ldots \exists u_k^1.\ \varphi$, 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}. ■

Theorem 17: Model checking $\mathsf{EQ}^k\mathsf{CTL}^*$ under the structure semantics is PSPACE-complete.

Proof: The algorithm 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). ■

[6]We use the classical notions of size for $\mathcal{S}$ and φ.

Theorem 18: Model checking EQCTL and EQCTL^* under the structure semantics is PSPACE-complete.

Proof: 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. ■

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

Theorem 19: Model checking $\mathsf{Q}^k\mathsf{CTL}$ under the structure semantics is $\Delta^{\mathsf{P}}_{\mathsf{k+1}}[O(\log n)]$-complete.

Proof: We define the algorithm inductively: when $k = 0$, we just have a CTL model-checking problem, which is complete for $\mathsf{PTIME} = \Delta^{\mathsf{P}}_{1}[O(\log n)]$. Assume that we have a $\Delta^{\mathsf{P}}_{\mathsf{k+1}}[O(\log n)]$ 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

$$\varphi \stackrel{\text{def}}{=} \Phi[(q_i \to \exists P_i.\ \psi_i)_i]$$

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 $\mathsf{NP}^{\Delta^{\mathsf{P}}_{\mathsf{k+1}}[O(\log n)]} = \Sigma^{\mathsf{P}}_{\mathsf{k+1}}$. 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 $\Delta^{\mathsf{P}}_{\mathsf{k+1}}[O(\log n)]$, since $\Delta^{\mathsf{P}}_{\mathsf{k+1}}[O(\log n)] = \Delta^{\mathsf{P}}_{\mathsf{k+1}\ ||}$ [28].

In order to prove hardness, we use the family of problems $\mathsf{PARITY}\ (\Sigma^{\mathsf{P}}_{\mathsf{k}})$, defined as follows:

Problem: $\mathsf{PARITY}\ (\Sigma^{\mathsf{P}}_{\mathsf{k}})$
Input: m instances of $\Sigma^{\mathsf{P}}_{\mathsf{k}}\mathsf{SAT}$ $Q^i_1 U^i_1 \dots Q^i_k U^i_k.\ \Phi^i(U^i_1, \dots, U^i_k)$, where $Q^i_j = \exists$ when j is odd and $Q^i_j = \forall$ otherwise;
Question: is the number of positive instances even?

This problem is $\Delta^{\mathsf{P}}_{\mathsf{k+1}}[O(\log n)]$-complete [29]. We encode it into a $\mathsf{Q}^k\mathsf{CTL}$ model-checking problem as follows: for each $1 \le i \le m$, the instance $Q^i_1 U^i_1 \dots Q^i_k U^i_k.\Phi^i(U^i_1, \dots, U^i_k)$ of $\Sigma^{\mathsf{P}}_{\mathsf{k}}\mathsf{SAT}$ is encoded as in the previous reduction, using a one-state Kripke structure that will be labelled with atomic propositions $u^i_{j,k}$; 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 strcutres above, with an extra state x_{m+1} and transitions (x_i, x_{i+1}), for each $1 \le i \le m$. We define

$$\varphi \stackrel{\text{def}}{=} \bigvee_{1 \le i \le m} (x_i \wedge \Phi_i).$$

This formula holds true in those states x_i of B whose corresponding $\Sigma^{\mathsf{P}}_{\mathsf{k}}\mathsf{SAT}$ instance is positive. It remains to build a formula for "counting" these sets: we let

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

It is easily seen that ψ_s holds true in state x_1 of B iff exactly s of the m instances of $\Sigma^{\mathsf{P}}_{\mathsf{k}}\mathsf{SAT}$ are positive. Moreover, each ψ_i has quantifier height at most k. Concluding the reduction is then obvious. ■

Theorem 20: Model checking QCTL, $\mathsf{Q}^k\mathsf{CTL}^*$ and QCTL^* under the structure semantics is PSPACE-complete.

Proof: The arguments are the same as for the proof of Theorems 17 and 18. ■

B. QCTL *model checking under the tree semantics*

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

Sketch of proof: Since $\mathsf{EQ}^k\mathsf{CTL}$ and $\mathsf{AQ}^k\mathsf{CTL}$ are dual and contained in $\mathsf{Q}^k\mathsf{CTL}$, it suffices to prove hardness for $\mathsf{EQ}^k\mathsf{CTL}$ and membership for $\mathsf{Q}^k\mathsf{CTL}$. We briefly sketch the proof here, and refer to Appendix E for a detailed proof.

► ***Hardness in*** k-$\mathsf{EXPTIME}$. The reduction uses the ideas of [9], [17]: we encode an alternating Turing machine $\mathcal{M}$ whose tape is has size k-exponential. 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 an $\mathsf{EQ}^k\mathsf{CTL}$ formula φ such that φ holds true in K (for the tree semantics) iff $\mathcal{M}$ accepts y. The encoding is depicted on Fig. 1.

The main tool in this proof is a set of (polynomial-size) formulas of $\mathsf{EQ}^k\mathsf{CTL}$ that are able to relate two states that are at distance k-exponential. 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.

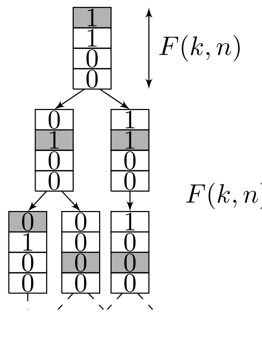

Fig. 1. A run of $\mathcal{M}$

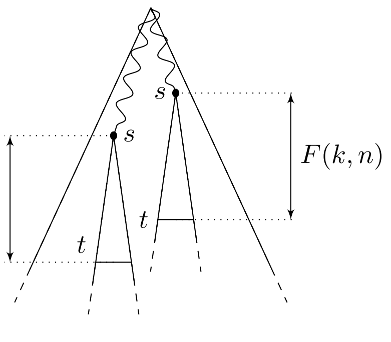

Fig. 2. Chunks of height $F(k,n)$

Our set of formulas will ensure the following (see Fig. 2): given a tree labeled with propositions s and t (among others),

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

both s and t appear exactly once along each branch, and the distance between them is $F(k,n)$, defined as

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

The formulas for $k = 0$ are easy to write. Then, given a formula for level k, we build the formula for level $k+1$ as follows: we add a new proposition r, which is required to holds at s and t and at various positions inbetween, at distance $F(k,n)$ from each other. We then implement a counter, using existential quantification over another proposition, in order to enforce that there are exactly $2^{F(k,n)}$ occurrences of r between s and t.

▶ ***Membership in*** k-EXPTIME. Our algorithm for $\mathsf{Q}^k\mathsf{CTL}$ model checking uses alternating parity tree automata[30], [31]. The construction is inductive: we begin with building an automaton for the innermost CTL formula [32], and then use projection to encode existential quantification. This requires turning the alternating automata into non-deterministic ones, which comes with an exponential blowup [33]. We apply this procedure recursively, until the last propositional quantifier. We end up with a non-deterministic parity tree automaton with size k-exponential and index $(k-1)$-exponential, for which emptiness is then solved in k-exponential [34]. It then remains to apply a CTL model-checking algorithm to handle the possible outermost CTL operators. In the end, the algorithm runs in k-EXPTIME. ■

Theorem 22: 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)-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. Regarding hardness, we can take advantage of using CTL^* in order to have $\mathsf{yardstick}_0^n(s,t)$ enforce that the distance between s and t is 2^n. ■

V. Using QCTL for specifying 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) $\mathcal{C}$ and a property Φ written in ATL_{sc} (an extension of ATL [35] geared towards the expression of non-zero-sum objectives [36], [12], see below) or Strategy Logic (a different extension of ATL with similar ideas [13], [14]) can be reduced to a model-checking problem $\mathcal{S}_\mathcal{C} \models \widehat{\Phi}$, where $\mathcal{S}_\mathcal{C}$ is the Kripke structure underlying $\mathcal{C}$, and $\widehat{\Phi}$ is a QCTL formula.

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.

In the following section, we introduce basic definitions of this setting, namely concurrent game structures and classical notions such as strategies and outcomes, and the syntax and semantics of ATL_{sc}. The reductions are given in Sections V-B and V-C.

A. Basic definitions

1) Concurrent game structures. Concurrent game structures [35] are a multi-player extension of classical Kripke structures. Their definition is as follows:

Definition 23: 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 finite-state Kripke structure,
- $\mathsf{Agt} = \{A_1, ..., A_p\}$ is a finite set of *agents* (or *players*);
- $\mathcal{M}$ is a finite, non-empty set of moves;
- $\mathsf{Mov}\colon Q \times \mathsf{Agt} \to \mathcal{P}(\mathcal{M}) \smallsetminus \{\varnothing\}$ 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; with each state q and each set of moves of the agents, it associates the resulting transition, which is required to depart 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 $\mathsf{Edge}(q, (m_1, ..., m_p))$. 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 the move m_j. We extend Mov and Next to coalitions (*i.e.*, sets of agents) in the natural way.

A (finite or infinite) *path* of $\mathcal{C}$ is a sequence $\rho = q_0 q_1 \ldots$ of states such that for any i, $q_{i+1} \in \mathsf{Next}(q_i)$. Finite paths are also 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 ρ' s.t. $\mathsf{last}(\rho) = \mathsf{first}(\rho')$, the concatenation of ρ and ρ' is the path $\tau = \rho \cdot \rho' = r_0 r_1 \ldots r_m$ 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 $f_i(q_0 \ldots q_m) \in \mathsf{Mov}(q_m, A_i)$. A strategy for a coalition A of agents is a mapping assigning a strategy to each agent in the coalition. The set of strategies for A is denoted $\mathsf{Strat}(A)$. The *domain* of $F_A \in \mathsf{Strat}(A)$ (denoted $\mathsf{dom}(F_A)$) is A. Given a coalition B, the strategy $(F_A)_{|B}$ (resp. $(F_A)_{\smallsetminus B}$) denotes the restriction of F_A to the coalition $A \cap B$ (resp. $A \smallsetminus B$).

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 (or from) 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 exists a set of moves $(m_k^i)_{A_k \in \mathsf{Agt}}$ such that $m_k^i \in \mathsf{Mov}(q_i, A_k)$ for all $A_k \in \mathsf{Agt}$, $m_k^i = f_{A_k}(\pi_{|\rho|+i})$ if $A_k \in A$, and $q_{i+1} \in \mathsf{Next}(q_i, \mathsf{Agt}, (m_k^i)_{A_k \in \mathsf{Agt}})$. It is also possible to *combine* two strategies $F \in \mathsf{Strat}(A)$ and $F' \in \mathsf{Strat}(B)$, resulting in a strategy $F \circ F' \in \mathsf{Strat}(A \bigcup B)$ defined as follows: $(F \circ F')_{|A_j}(\rho)$ is $F_{|A_j}(\rho)$ (resp. $F'_{|A_j}(\rho)$) if $A_j \in A$ (resp. $A_j \in B \smallsetminus A$). 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 $F^\rho(\pi) = F(\rho \cdot \pi)$.

2) Alternating-time temporal logics. We now introduce the extension of ATL with strategy contexts [36], [12]:

Definition 24: The syntax of ATL_{sc} is defined by the following grammar:

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

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 (*i.e.*, a preselected strategy for some of the players, hence belonging to some $\mathsf{Strat}(A)$ for a coalition A), denoted $\mathcal{C}, q \models_F \varphi$, is defined as follows [8]:

$$\begin{array}{lll} \mathcal{C}, q \models_F \rangle\!\cdot A \cdot\!\langle\; \varphi_{\mathsf{state}} & \text{iff} & \mathcal{C}, q \models_{F_{\smallsetminus A}} \varphi_{\mathsf{state}} \\ \mathcal{C}, q \models_F \langle\!\cdot A \cdot\!\rangle\; \varphi_{\mathsf{path}} & \text{iff} & \exists F_A \in \mathsf{Strat}(A). \\ & & \forall \rho' \in \mathsf{Out}(q, F_A \circ F).\ \mathcal{C}, \rho' \models_{F_A \circ F} \varphi_{\mathsf{path}} \end{array}$$

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

Example 25: 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

$$\langle\!\cdot S \cdot\!\rangle\, \mathbf{G} \Big[\bigwedge_{i \neq j} \neg(\mathsf{grant}_i \wedge \mathsf{grant}_j) \wedge \bigwedge_i \langle\!\cdot A_i \cdot\!\rangle\, \mathbf{F}\, \mathsf{grant}_i\Big] \qquad (3)$$

B. 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: $\mathsf{P}_Q \stackrel{\text{def}}{=} \{\mathsf{p}_q \mid q \in Q\}$, $\mathsf{P}^a_{\mathcal{M}} \stackrel{\text{def}}{=} \{\mathsf{m}^a_1, \ldots, \mathsf{m}^a_k\}$ for every $a \in \mathsf{Agt}$, and $\mathsf{P}_{\mathcal{M}} \stackrel{\text{def}}{=} \bigcup_a \mathsf{P}^a_{\mathcal{M}}$.

Let $\mathcal{S}_{\mathcal{C}}$ be the Kripke structure $\langle Q, R, \ell_+\rangle$ where for any state q, we have: $\ell_+(q) \stackrel{\text{def}}{=} \ell(q) \cup \{\mathsf{p}_q\}$. $\mathcal{S}_{\mathcal{C}}$ 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 can be seen as a function labelling the execution tree of $\mathcal{S}_{\mathcal{C}}$ with $\mathsf{P}^a_{\mathcal{M}}$. More precisely, a strategy for a is a labelling function $f_a \colon \mathsf{Exec}^{\mathsf{f}}(\ell) \to \mathsf{P}^a_{\mathcal{M}}$. A memoryless strategy for a corresponds to a labelling function $f_a \colon Q \to \mathsf{P}^a_{\mathcal{M}}$, *i.e.*, a labelling of the Kripke structure $\mathcal{S}_{\mathcal{C}}$.

[8]We omit the cases of Boolean operators and path modalities.

Let F be a strategy context for some coalition $C \subseteq \mathsf{Agt}$. Let Φ be an ATL_{sc} formula. 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 as follows. For non-temporal constructs, we let

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

For a formula of the form $\langle\!\cdot A \cdot\!\rangle\, \mathbf{X}\, \varphi$ with $A = \{a_1, \ldots, a_l\}$, we let:

$$\widehat{\langle\!\cdot A \cdot\!\rangle\, \mathbf{X}\, \varphi}^{\,C} \stackrel{\text{def}}{=} \exists \mathsf{m}^{a_1}_1 \ldots \mathsf{m}^{a_1}_k \ldots \mathsf{m}^{a_l}_1 \ldots \mathsf{m}^{a_l}_k.$$
$$\bigwedge_{a \in A} \mathbf{AG}\, (\Phi_{\mathsf{strat}}(a)) \wedge \mathbf{A}\Big(\Phi^{[A \cup C]}_{\mathsf{out}} \Rightarrow \mathbf{X}\, \widehat{\varphi}^{C \cup A}\Big)$$

where:

- $\Phi_{\mathsf{strat}}(a)$ ensures that, the labeling of propositions m^a_is describes a feasible trategy for a (*i.e.* exactly one possible move for a in any reachable state):

$$\Phi_{\mathsf{strat}}(a) \stackrel{\text{def}}{=} \bigvee_{q \in Q} \Big(\mathsf{p}_q \wedge \bigvee_{m_i \in \mathsf{Mov}(q,a)} (\mathsf{m}^a_i \wedge \bigwedge_{j \neq i} \neg\, \mathsf{m}^a_j)\Big)$$

- Given a coalition A, formula $\Phi^{[A]}_{\mathsf{out}}$ characterizes the outcomes of the strategy for A that is described by the atomic propositions in the model (see Lemma 26). It is defined as follows:

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

  where m is a move $(m^a)_{a \in A} \mathsf{Mov}(q, A)$ for A and P_m is the propositional formula $\bigwedge_{a \in A} m^a$ which describes m. Note that $\Phi^{[A]}_{\mathsf{out}}$ is based on the transition table Edge of $\mathcal{C}$.

Formula $\widehat{\langle\!\cdot A \cdot\!\rangle\, \mathbf{X}\, \Phi}^{\,C}$ states that there exists a way of fixing a strategy for A (by an appropriate labelling of the execution tree with moves of A) such that any execution corresponding to this strategy and the (previously selected) strategy for C satisfies $\mathbf{X}\, \widehat{\varphi}^{C \cup A}$

For $\langle\!\cdot A \cdot\!\rangle\, \varphi\, \mathbf{U}\, \psi$, we use the same idea:

$$\widehat{\langle\!\cdot A \cdot\!\rangle\, (\varphi\, \mathbf{U}\, \psi)}^{\,C} \stackrel{\text{def}}{=} \exists \mathsf{m}^{a_1}_1 \ldots \mathsf{m}^{a_1}_k \ldots \mathsf{m}^{a_l}_1 \ldots \mathsf{m}^{a_l}_k.$$
$$\bigwedge_{a \in A} \mathbf{AG}\, (\Phi_{\mathsf{strat}}(a)) \wedge \mathbf{A}\Big(\Phi^{[A \cup C]}_{\mathsf{out}} \Rightarrow (\widehat{\varphi}^{C \cup A}\, \mathbf{U}\, \widehat{\psi}^{C \cup A})\Big)$$

Other temporal modalities can be handled similarly.

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 26: 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 $a \in A$, $\ell'(\pi) \cap \mathsf{P}^a_{\mathcal{M}}$ is a singleton. Let ρ be an execution in $\mathcal{T}'$. Then

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

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.

In the end, we have the following result (a proof sketch is given in Appendix D):

Theorem 27: 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 $\pi \in \mathsf{Exec}^\mathsf{f}(q)$ and $a \in C$, $\ell'(\pi) \cap \mathsf{P}^a_\mathcal{M} = \mathsf{m}^a_j$ iff $F(\pi)_{|a} = m_j$. Then $\mathcal{C}, q \models_F \Phi$ iff $\mathcal{T}', q \models \widehat{\Phi}^C$.

Example 28: Consider the client-server example and the ATL_{sc} formula of Equation (3). The corresponding QCTL^* formula is:

$$\exists \mathsf{m}_1^S ... \mathsf{m}_k^S.\, \mathbf{AG}\,(\Phi_{\mathsf{strat}}(S)) \;\wedge\; \mathbf{A}\Bigg(\Phi_{\mathsf{out}}^{[S]} \Rightarrow \mathbf{G}\,\Big[\bigwedge_{i \neq j} \neg(\mathsf{grant}_i \wedge \mathsf{grant}_j) \wedge \bigwedge_i \exists \mathsf{m}_1^{A_i} ... \mathsf{m}_k^{A_i}.\, \mathbf{AG}\,(\Phi_{\mathsf{strat}}(A_i)) \wedge \mathbf{A}(\Phi_{\mathsf{out}}^{[S \cup A_i]} \Rightarrow \mathbf{F}\,\mathsf{grant}_i)\Big]\Bigg)$$

and every $\Phi_{\mathsf{out}}^{[S \cup A_i]}$ is defined as above.

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 [12]).

Remark 29: We have two remarks about our reduction:

a) Memoryless strategies. 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 theorem similar to Theorem 27 can be stated by considering the structure semantics for QCTL and memoryless strategies for ATL_{sc}.

b) Translation into **QCTL.** Our reduction above is into QCTL^* but we can use Proposition 12 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.

C. 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 $\mathsf{AP}_\mathcal{Q}(\Phi) = \{P_1, \ldots, P_k\}$. 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 $\mathsf{Agt} = \{A_0, \ldots, A_k\}$: 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 $Q' = Q \cup \{c_{q,i} \mid i = 1, \ldots, k\} \cup \{p_i \mid i = 0, \ldots, k\}$: 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 $i = 1, \ldots, k$ 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:

$$\widetilde{\exists P_i . \varphi} \overset{\text{def}}{=} \langle\cdot A_i \cdot\rangle\, \widetilde{\varphi} \qquad \widetilde{\varphi \wedge \psi} \overset{\text{def}}{=} \widetilde{\varphi} \wedge \widetilde{\psi}$$

$$\widetilde{P_i} \overset{\text{def}}{=} \langle\cdot A_0 \cdot\rangle\, \mathbf{X}\, \langle\cdot A_0 \cdot\rangle\, \mathbf{X}\, P_i \qquad \widetilde{\neg \psi} \overset{\text{def}}{=} \neg\, \widetilde{\varphi}$$

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

$$\widetilde{\mathsf{E}\mathbf{G}\, \varphi} \overset{\text{def}}{=} \langle\cdot A_0 \cdot\rangle\, \mathbf{G}\, (P_Q \wedge \widetilde{\varphi})$$

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 30: 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:

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

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

D. 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 [13][9], [14]. In this framework we can assign strategies to some variables and

[9]Notice that Strategy Logic in [13] 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.

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 $\Phi_{\mathsf{out}}^{\square}$ 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 14, FQCTL^* can be translated into QCTL when considering the tree semantics.

The converse reduction from QCTL (under the tree semantics) to SL model checking also holds. This entails the following result, correcting a wrong claim in [14, Theorem 4.2]:

Theorem 32: The model-checking problems for QCTL, ATL_{sc} and SL are inter-reducible (in logarithmic space). They all are non-elementary.

VI. Conclusions and future works

We have characterized the complexity of model-checking QCTL, completing earlier results from [8], [10]. We have applied these results to model checking for recent extensions of ATL geared towards non-zero-sum properties. Our plans for future works include a finer study of the complexity of model checking, for a fixed model or a fixed formula, and the study of satisfiability.

Acknowledgement. We thank Thomas Colcombet and Olivier Serre for helpful comments during the redaction of this paper.

References

[1] A. Pnueli, "The temporal logic of programs," in FOCS'77. IEEE Comp. Soc. Press, 1977, pp. 46–57.

[2] E. M. Clarke and E. A. Emerson, "Design and synthesis of synchronization skeletons using branching-time temporal logic," in LOP'81, LNCS, vol. 131. Springer, 1982, pp. 52–71.

[3] J.-P. Queille and J. Sifakis, "Specification and verification of concurrent systems in CESAR," in SOP'82, LNCS, vol. 137. Springer, 1982, pp. 337–351.

[4] E. A. Emerson and J. Y. Halpern, ""Sometimes" and "not never" revisited: On branching versus linear time temporal logic," *Journal of the ACM*, vol. 33, no. 1, pp. 151–178, 1986.

[5] R. Alur, T. A. Henzinger, and O. Kupferman, "Alternating-time temporal logic," in FOCS'97. IEEE Comp. Soc. Press, 1997, pp. 100–109.

[6] A. P. Sistla, "Theoretical issues in the design and verification of distributed systems," Ph.D. dissertation, Harvard University, Cambridge, Massachussets, USA, 1983.

[7] E. A. Emerson and A. P. Sistla, "Deciding full branching time logic," *Information and Control*, vol. 61, no. 3, pp. 175–201, 1984.

[8] O. Kupferman, "Augmenting branching temporal logics with existential quantification over atomic propositions," in CAV'95, LNCS, vol. 939. Springer, 1995, pp. 325–338.

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

[10] T. French, "Decidability of quantified propositional branching time logics," in AJCAI'01, LNCS, vol. 2256. Springer, 2001, pp. 165–176.

[11] J. F. Nash, Jr., "Equilibrium points in n-person games," *Proceedings of the National Academy of Sciences*, vol. 36, no. 1, pp. 48–49, 1950.

[12] A. Da Costa, F. Laroussinie, and N. Markey, "ATL with strategy contexts: Expressiveness and model checking," in FSTTCS'10, LIPIcs, vol. 8. Leibniz-Z. Informatik, 2010, pp. 120–132.

[13] K. Chatterjee, T. A. Henzinger, and N. Piterman, "Strategy logic," in CONCUR'07, LNCS, vol. 4703. Springer, 2007, pp. 59–73.

[14] F. Mogavero, A. Murano, and M. Y. Vardi, "Reasoning about strategies," in FSTTCS'10, LIPIcs, vol. 8. Leibniz-Z. Informatik, 2010, pp. 133–144.

[15] F. Wang, C.-H. Huang, and F. Yu, "A temporal logic for the interaction of strategies," in CONCUR'11, LNCS, vol. 6901. Springer, 2011, pp. 466–481.

[16] S. Pinchinat, "A generic constructive solution for concurrent games with expressive constraints on strategies," in ATVA'07, LNCS, vol. 4762. Springer, 2007, pp. 253–267.

[17] A. P. Sistla, M. Y. Vardi, and P. Wolper, "The complementation problem for Büchi automata with applications to temporal logics," *Theoretical Computer Science*, vol. 49, pp. 217–237, 1987.

[18] Y. Kesten and A. Pnueli, "A complete proof systems for QPTL," in LICS'95. IEEE Comp. Soc. Press, 1995, pp. 2–12.

[19] S. Riedweg and S. Pinchinat, "Quantified μ-calculus for control synthesis," in MFCS'03, LNCS, vol. 2747. Springer, 2003, pp. 642–651.

[20] A. C. Patthak, I. Bhattacharya, A. Dasgupta, P. Dasgupta, and P. P. Chakrabarti, "Quantified computation tree logic," *Information Processing Letters*, vol. 82, no. 3, pp. 123–129, 2002.

[21] G. Bruns and P. Godefroid, "Model checking partial state spaces with 3-valued temporal logics," in CAV'99, LNCS, vol. 1633. Springer, 1999, pp. 274–287.

[22] A. Gurfinkel and M. Chechik, "How thorough is thorough enough?" in CHARME'05, LNCS, vol. 3725. Springer, 2005, pp. 65–80.

[23] T. French, "Bisimulation quantifiers for modal logics," PhD thesis, School of Computer Science & Software Engineering, University of Western Australia, 2006.

[24] F. Moller and A. Rabinovich, "Counting on CTL*: on the expressive power of monadic path logic," *Information and Computation*, vol. 184, no. 1, pp. 147–159, 2003.

[25] H.-D. Ebbinghaus and J. Flum, *Finite Model Theory*. Springer, 1995.

[26] B. Courcelle and J. Engelfriet, *Graph Structure and Monadic Second-Order Logic, a Language Theoretic Approach*. Cambridge U. Press, 2011.

[27] Ch. H. Papadimitriou, *Computational Complexity*. Addison-Wesley, 1994.

[28] K. W. Wagner, "Bounded query classes," *SIAM Journal on Computing*, vol. 19, no. 5, pp. 833–846, 1990.

[29] G. Gottlob, "NP trees and Carnap's modal logic," *Journal of the ACM*, vol. 42, no. 2, pp. 421–457, 1995.

[30] D. E. Muller and P. E. Schupp, "Alternating automata on infinite trees," *Theoretical Computer Science*, vol. 54, no. 2-3, pp. 267–276, 1987.

[31] W. Thomas, "Languages, automata and logics," in *Handbook of Formal Languages*, G. Rozenberg and A. Salomaa, Eds. Springer, 1997, vol. 3, pp. 389–455.

[32] O. Kupferman, M. Y. Vardi, and P. Wolper, "An automata-theoretic approach to branching-time model-checking," *Journal of the ACM*, vol. 47, no. 2, pp. 312–360, 2000.

[33] D. E. Muller and P. E. Schupp, "Simulating alternating tree automata by nondeterministic automata: New results and new proofs of the theorems of Rabin, McNaughton and Safra," *Theoretical Computer Science*, vol. 141, no. 1-2, pp. 69–107, 1995.

[34] O. Kupferman and M. Y. Vardi, "Weak alternating automata and tree automata emptiness," in STOC'98. ACM Press, 1998, pp. 224–233.

[35] R. Alur, T. A. Henzinger, and O. Kupferman, "Alternating-time temporal logic," *Journal of the ACM*, vol. 49, no. 5, pp. 672–713, 2002.

[36] Th. Brihaye, A. Da Costa, F. Laroussinie, and N. Markey, "ATL with strategy contexts and bounded memory," in LFCS'09, LNCS, vol. 5407. Springer, 2009, pp. 92–106.

Appendix

A. *Semantics of* QCTL

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

B. *Prenex normal form for* QCTL

Proposition 9: Every QCTL formula is equivalent (for both semantics) to a formula in prenex normal form.

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.

For propositional and Boolean subformulas, we have:

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

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:

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

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:

$$
\begin{aligned}
\mathsf{dpath}(x, y) \equiv_s\ & \mathsf{uniq}(y) \wedge \mathbf{E}x\,\mathbf{U}\,y\ \wedge\\
&\mathbf{AG}\,\Big(\big((x \wedge \mathsf{selfloop}) \Rightarrow (\mathbf{EX}_2(x \vee y)\big) \wedge\\
&\quad \big((x \wedge \neg\,\mathsf{selfloop}) \Rightarrow (\mathbf{EX}_1(x \vee y))\big)\Big)
\end{aligned}
$$

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

$$
\begin{aligned}
\mathbf{EX}_1(\varphi) &\overset{\text{def}}{=} \mathbf{EX}\,\varphi \wedge \forall z.\Big(\mathbf{EX}\,(\varphi \wedge z) \ \Rightarrow\ \mathbf{AX}\,(\varphi \Rightarrow z)\Big)\\
\mathbf{EX}_2(\varphi) &\overset{\text{def}}{=} \exists z.\,\mathbf{EX}\,(\varphi \wedge z) \wedge \mathbf{EX}\,(\varphi \wedge \neg\, z) \wedge \forall z'.\\
&\quad \Big((\mathbf{EX}\,(\varphi \wedge z \wedge z') \wedge \mathbf{EX}\,(\varphi \wedge \neg\, z \wedge z')) \Rightarrow \mathbf{AX}\,(\varphi \Rightarrow z')\Big)
\end{aligned}
$$

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:

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

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 $\mathsf{AP}\setminus\{z_1, \ldots, z_k\}$, 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 $\mathsf{lab}_{z_i} \colon Q \to 2^{\{z_i\}}$, 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 $\mathsf{AP}\setminus\{z_i \cdots z_k\}$;
3) for all $\mathcal{S}' \in \xi$, it holds $\mathcal{S}', q \models_s \varphi$.

A non-empty set ξ satisfying the first two properties 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:

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

with:

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

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 $\rho = q_0 q_1 q_2 \ldots q_l$ 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:

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

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 $\widehat{\xi}$. 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 $\rho = q_0 q_1 x \ldots q_i$ 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:

$$\xi'_2 \stackrel{\text{def}}{=} \Big\{\mathcal{S}' = \langle Q, R, \ell'\rangle \;\Big|\; \exists \langle Q, R, \ell''\rangle \in \xi_2 \text{ and } \ell' \stackrel{\text{def}}{=} \ell'' \oplus \{q_i \mapsto z_2, q_j \mapsto z_0 (j = 0, \ldots, i-1)\}\Big\}$$

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:

$$\xi \stackrel{\text{def}}{=} \Big\{\mathcal{S}'' = \langle Q, R, \ell''\rangle \;\Big|\; \exists \mathcal{S}' \in \xi'_2.\ \exists j < i.\ \exists \langle Q, R, \ell'''\rangle \in \xi_1^{\mathcal{S}'} \text{ and } \ell'' \stackrel{\text{def}}{=} \ell''' \oplus \{q_j \mapsto z_1\}\Big\}$$

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 $\mathbf{EU}$, 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). ■

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

C. From MSO to QCTL

We present the translation from MSO to QCTL mentionned in the proof of Proposition 10. Given $\varphi(x) \in \mathsf{MSO}$, we define $\widehat{\varphi} \in \mathsf{QCTL}$ as follows:

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

This translation is correct for both semantics, as for any $\mathcal{S}$ and q, we have that $\mathcal{T}'_{\mathcal{S}}(q), q \models_s \widehat{\varphi}$ iff

$$\mathcal{T}_{\mathcal{S}}(q), q, s_1, ..., s_n, S_1, ..., S_k \models \varphi(x, x_1, ..., x_n, X_1, ..., X_k)$$

and that $\mathcal{S}'_q, q \models_s \widehat{\varphi}$ iff

$$\mathcal{S}_q, q, s_1, ..., s_n, S_1, ..., S_k \models \varphi(x, x_1, ..., x_n, X_1, ..., X_k)$$

where $\mathcal{T}_{\mathcal{S}}$ and $\mathcal{T}'_{\mathcal{S}}$ (resp. $\mathcal{S}_q$ and $\mathcal{S}'_q$) only differ in the labeling of propositions p_{x_i} and p_{X_i}: in $\mathcal{T}_{\mathcal{S}}$ (resp. $\mathcal{S}_q$), no state is labeled with these propositions, while in $\mathcal{T}'_{\mathcal{S}}$ (resp. $\mathcal{S}'_q$) we have (1) $\mathsf{p}_{x_i} \in \ell(s)$ iff $s = s_i$ and (2) $\mathsf{p}_{X_i} \in \ell(s)$ iff $s \in S_i$. From this we deduce $\varphi(x) \equiv_t \widehat{\varphi}$ and $\varphi(x) \equiv_s \widehat{\varphi}$.

D. From ATL_{sc} *to* QCTL^*

Sketch of proof of Theorem 27: The proof is done by structural induction over Φ. Boolean cases are trivial.

Consider $\Phi \stackrel{\text{def}}{=} \langle\cdot A\cdot\rangle\, \mathbf{X}\, \varphi$. 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 $\ell''(\pi) \stackrel{\text{def}}{=} \ell'(\pi) \cup \{\mathsf{m}^a_j\}$ if $F_A(\pi)_{|a} = m_j$ [10]. Now let ρ' be an execution in $\mathcal{T}''$. Assume we have $\mathcal{T}'', \rho' \models \Phi^{[A \cup C]}_{\mathsf{out}}$.

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

From Lemma 26, 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^a_is 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 $\Phi \stackrel{\text{def}}{=} \langle\cdot A\cdot\rangle\, (\varphi\, \mathbf{U}\, \psi)$.

Finally assume $\Phi \stackrel{\text{def}}{=} \cdot\rangle A \langle\cdot\, \varphi$. Then $\widehat{\Phi}^{C}$ is $\widehat{\varphi}^{C \smallsetminus A}$. Indeed in $\widehat{\varphi}^{C \smallsetminus A}$, 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. ■

E. Proof of Theorem 21

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

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

b) ▶ 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 $\mathsf{Q}^k\mathsf{CTL}$ 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 $\mathsf{EQ}^k\mathsf{CTL}$ 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

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

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

$$\mathsf{delimiters}(s, t) = \mathsf{once}(s) \wedge \mathsf{once}(t) \wedge \mathbf{AG}\,(s \Rightarrow \mathbf{AF}\,t). \quad (4)$$

We now inductively build our "yardstick" formulas enforcing that, along any branch, the distance between the occurrence

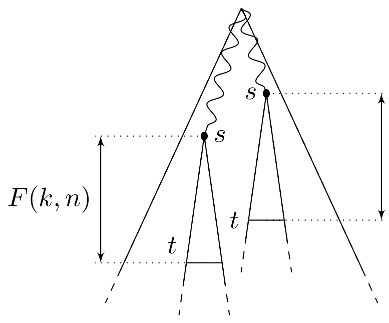

Fig. 3. Chunks of height $F(k,n)$

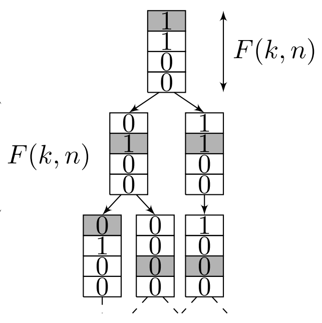

Fig. 4. Encoding runs of $\mathcal{M}$

of s and that of t is precisely $F(k,n)$ (see Fig. 3). When $k=0$, this is easy[11]:

$$\mathsf{yardstick}_0^n(s,t) = \mathbf{AG}\Big(s \Rightarrow \big((\mathbf{AX})^n t \wedge \bigwedge_{0\le k<n} (\mathbf{AX})^k \neg t\big)\Big). \tag{5}$$

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:

$$\mathsf{yardstick}_k^n = \exists r.\exists c.\ (\mathsf{graduation}_k(r,s,t) \wedge \mathsf{counter}_k(c,r,s,t)). \tag{6}$$

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:

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

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

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

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 $\mathsf{yardstick}_{k-1}^n$ 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 $\mathsf{graduation}_k$:

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

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 $\mathsf{yardstick}_{k-1}^n$ is, by induction hypothesis, in $\mathsf{EQ}^{k-1}\mathsf{CTL}$, formula $\mathsf{yardstick}_k^n$ is in $\mathsf{EQ}^k\mathsf{CTL}$ (notice that $\mathsf{yardstick}_{k-1}^n$ 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. 4, 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

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

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

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

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

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.

c) ▶ ***Membership in*** **k-EXPTIME.** Our algorithm uses alternating parity tree automata: with each $\mathsf{Q}^k\mathsf{CTL}$ formula φ, Kripke structure K and state q_0, we associate such an automaton which has non-empty language if, and only if $K, q_0 \models \varphi$. We won't recall the definitions of this classical setting, and better refer to [MS87], [Tho97] for more details. Our construction uses classical techniques, already present *e.g.* in [Tho97], [CHP07], [Pin07], [DLM10], [MMV10]. We assume we are given a Kripke structure $K = \langle Q, R, \ell\rangle$ on a set AP of atomic propositions. 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 33: [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 34: [MS87], [MS95] Let $\mathcal{A}$ and $\mathcal{B}$ be two $\langle Q, 2^{\mathsf{AP}}\rangle$-APTs[12] 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 35: [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 36: [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 37: [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 38: [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 33, we obtain a polynomial-size alternating automaton for the inner CTL formula. Applying Lemma 35 and Lemma 36, 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 37, and Lemma 33 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 35 and 36 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. ■

F. Results about Strategy Logic

In this section we present the reduction from QCTL model checking to SL model checking. It uses the same approach we use for ATL_{sc}: given a problem $\mathcal{S} \models \Phi$ with $\Phi \in \mathsf{QCTL}$, we reuse the same CGS $\mathcal{C}_\mathcal{S}$ with $k+1$ agents where k is the number of quantified propositions in Φ.

We use the version the logic SL introduced in [MMV10]. This logics contains Boolean operators, temporal modalities ($\mathbf{X}$ and $\mathbf{U}$) and new constructs to deal with strategies: existential and universal quantifiers over strategies ($\langle\!\langle x\rangle\!\rangle\,\varphi$ means "there exists a strategy x s.t. φ", and $[\![x]\!]$ is the universal quantification) and an agent binding operator to associate an agent with an existing strategy ($(A,x).\varphi$ means "when A plays according to the strategy x, φ holds true"). This formalism is very powerful because it allows us to deal explicitly with the strategies and for example, one can ensure some property φ

[12]An $\langle Q, 2^{\mathsf{AP}}\rangle$-APT is an alternating parity tree automaton running on 2^{AP}-labelled Q-trees. Similarly, $\langle Q, 2^{\mathsf{AP}}\rangle$-NPT denotes non-deterministic parity tree automata.

when two agents play according to the same given strategy (*i.e.* $(A,x).(B,x).\varphi$).

Let $\mathcal{S} \models \Phi$ be a model checking problem for QCTL. We assume that every proposition quantifier in Φ introduces a fresh variable. Let $\{P_1, \ldots, P_k\}$ be the set $\mathsf{AP}_{\mathcal{Q}}(\Phi)$. Now we define $\widetilde{\Phi} \in \mathsf{SL}$ in order to have: $\mathcal{S} \models \Phi$ iff $\mathcal{C}_{\mathcal{S}} \models \tilde{\Phi}$. A syntatic constraint in SL is that an SL specification (called a *sentence*) has to be *closed*: every agent has to associated with a strategy when temporal modalities are interpreted, and every strategy variable has to be in the scope of some quantifier. For this, we define $\widetilde{\Phi}$ as $\langle\!\langle x_0 \rangle\!\rangle .(A_0, x_0) \ldots \langle\!\langle x_k \rangle\!\rangle .(A_k, x_k).\widehat{\Phi}$ with:

$$\begin{aligned}
\widehat{\exists P_i.\varphi} &\stackrel{\text{def}}{=} \langle\!\langle x_i \rangle\!\rangle .(A_i, x_i).\widehat{\varphi} \\
\widehat{P_i} &\stackrel{\text{def}}{=} \langle\!\langle x_0 \rangle\!\rangle (A_0, x_0).\mathbf{X}\,\mathbf{X}\,P_i \\
\widehat{\mathsf{E}\varphi\,\mathbf{U}\,\psi} &\stackrel{\text{def}}{=} \langle\!\langle x_0 \rangle\!\rangle .(A_0, x_0).(P_Q \wedge \widehat{\varphi})\,\mathbf{U}\,(P_Q \wedge \widehat{\psi}) \\
\widehat{\mathsf{E}\mathbf{G}\,\varphi} &\stackrel{\text{def}}{=} \langle\!\langle x_0 \rangle\!\rangle .(A_0, x_0).\mathbf{G}\,(P_Q \wedge \widehat{\varphi})
\end{aligned}$$

And $\widehat{\varphi \wedge \psi} \stackrel{\text{def}}{=} \widehat{\varphi} \wedge \widehat{\psi}$, $\widehat{\neg\,\psi} \stackrel{\text{def}}{=} \neg\,\widehat{\varphi}$, and $\widehat{P} \stackrel{\text{def}}{=} P$ when $P \notin \mathsf{AP}_{\mathcal{Q}}(\Phi)$.

Note that the first choice for the strategies x_is does not matter for the truth value of $\widehat{\Phi}$ because every agent will be associated with another strategy when he will have to actually choose a move. In SL, a formula φ is interpreted over a state or a path in a CGS and over an assignment χ which provides a strategy to every free agent or variable in φ: we write $\mathcal{C}, \chi, q \models \varphi$ or $\mathcal{C}, \chi, \rho \models \varphi$.

We state the correctness of the reduction as follows:

Proposition 39: 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 $\{P_i \mid i \in \mathcal{I}\}$. For any $i \in \mathcal{I}$, let F_{A_i} be the strategy such that: $F_{A_i}(\rho \cdot c_{q,i}) = \mathsf{m}_1$ iff $\ell_{\mathcal{T}}(\rho) \ni P_i$ for every finite $\mathcal{S}$-path ρ.
Let χ be an assignment such that (1) for any $i \in \mathcal{I}$, χ maps the variable x_i and the agent A_i to the strategy F_{A_i}, and (2) for $j \in \{0, \ldots, k\} \smallsetminus \mathcal{I}$, χ maps x_j and A_j to an arbitrary feasible strategy for A_j. Then we have:

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

References

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

[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.

[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.

[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. of the 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.

[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. *Theor. Computer Science*, 54(2-3):267–276, 1987.

[MS95] D. E. Muller and P. E. Schupp. Simulating alternating tree automata by nondeterministic automata: New results and new proofs of the theorems of Rabin, McNaughton and Safra. *Theor. Computer Science*, 141(1-2):69–107, 1995.

[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.

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

[Tho97] W. Thomas. Languages, automata and logics. In *Handbook of Formal Languages*, p. 389–455. Springer, 1997.