Quantified CTL: expressiveness and model checking (Extended abstract)

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

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

Abstract. While it was defined long ago, the extension of CTL with quantification over atomic propositions has never been studied extensively. Considering two different semantics (depending whether propositional quantification refers to the Kripke structure or to its unwinding tree), we study its expressiveness (showing in particular that QCTL coincides with Monadic Second-Order Logic for both semantics) and characterize the complexity of its model-checking problem, depending on the number of nested propositional quantifiers (showing that the structure semantics populates the polynomial hierarchy while the tree semantics populates the exponential hierarchy). We also show how these results apply to model checking ATL-like temporal logics for games.

1 Introduction

Temporal logics. Temporal logics extend propositional logics with modalities for specifying constraints on the order of events in time. Since [25,5,26], they have received much attention from the computer-aided-verification community, since they fit particularly well for expressing and automatically verifying (*e.g.* via *model checking*) properties of reactive systems. Two important families of temporal logics have been considered: linear-time temporal logics (*e.g.* LTL [25]) can be used to express properties of one single execution of the system under study, while branching-time temporal logics (*e.g.* CTL [5,26] and CTL* [10]) consider the execution tree. Since the 90s, many extensions of these logics have been introduced, of which alternating-time temporal logics (such as ATL, ATL* [1]) extend CTL towards the study of open systems (involving several agents).

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

Our contributions. We are interested in the present paper in the extension of CTL (and CTL*) with *propositional quantification* [28,11]. 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 [15], for the two-alternation fragment [16] and for the full extension [12]; but expressiveness issues, as well as a complete study of model checking for the whole hierarchy, have been mostly overlooked. We answer these questions in the present paper: in terms of expressiveness, we prove that QCTL and QCTL* are equally expressive, and coincide with Monadic Second-Order Logic. Regarding model checking, we consider both prenex-normal-form formulas (EQCTL) and general formulas (QCTL), and our results are summarized in the table below (where k in EQ^{k}CTL and Q^{k}CTL refers to some measure of quantification height of formulas, see Section 2.4). Finally, we also characterize the model- and formula-complexities of our problems, when one of the inputs to the model-checking problem is fixed. By lack of spaces, most proofs are omitted. They can be found in [8].

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

Applications to alternating-time temporal logics. ATL also has several flaws in terms of expressiveness: namely, it can only focus on (some) zero-sum properties, *i.e.*, on purely antagonist games, in which two coalitions fight with opposite objectives. In many situations, games are not purely antagonist, but involve several independent systems, each having its own objective. Recently, several extensions of ATL have been defined to express properties of such non-zero-sum games. Among those, our logic ATL$_{sc}$ [7] extends ATL with *strategy contexts*, which provides a way of expressing interactions between strategies. Other similar approaches include Strategy Logics [4,19] or (B)SIL [32].

Interestingly, the model-checking problem for these extensions of ATL$_{sc}$ (and also for Strategy Logics) can be seen as a QCTL model-checking problem[3]: strategy quantification in ATL is naturally encoded using propositional quantification of QCTL; since this labelling is *persistent*, it can encode interactions between strategies. We give the full encoding in Section 5. Notice that while the *tree semantics* of QCTL encodes plain strategies, the *structure semantics* also finds a meaning in that translation, as it may correspond to *memoryless strategies*.

Related works. Propositional quantification was also defined and studied on LTL [28,29,14], where the model-checking problem for the k-alternation fragment

[3] Notice that the link between games and propositional quantification already emerges in Q$_{\mathrm{D}}\mu$ [24], which extends the *decision μ-calculus* with some flavour of propositional quantification. Also, the main motivation of [16] for studying the two-alternation fragment of QCTL is a hardness result for the control and synthesis of open systems.

was settled complete for k-EXPSPACE. In the branching-time setting, the results are more sparse: CTL and CTL* with only existential quantification was studied in [15], where model checking is shown NP- and PSPACE-complete resp. (for the structure semantics) and EXPTIME- and 2-EXPTIME-complete resp. (for the tree semantics). The two-alternation fragment was then studied in [16] (only for the tree semantics): model checking is 2-EXPTIME- and 3-EXPTIME-complete, respectively for CTL or CTL*. Finally, satisfiability of the full extension (with arbitrary quantification) was studied in [12].

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 [12]. Alternative semantics are proposed and studied in [27,23].

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) [3].

2 Preliminaries

2.1 Kripke structures and trees

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

Definition 1. *A* Kripke structure $\mathcal{S}$ *is a* 3*-tuple* $\langle Q, R, \ell \rangle$ *where* Q *is a countable set of states,* $R \subseteq Q^2$ *is a total*[4] *relation and* $\ell \colon Q \to 2^{\mathsf{AP}}$ *is a labelling function.*

An execution (or path) in $\mathcal{S}$ is an infinite sequence $\rho = (q_i)_{i \in \mathbb{N}}$ s.t. $(q_i, q_{i+1}) \in R$ for all i. We use $\mathsf{Exec}(q)$ to denote the set of executions issued from q and $\mathsf{Exec}^{\mathsf{f}}(q)$ for the set of all finite prefixes of executions of $\mathsf{Exec}(q)$. Given $\rho \in \mathsf{Exec}(q)$ and $i \in \mathbb{N}$, we write ρ^i for the path $(q_{i+k})_{k \in \mathbb{N}}$ of $\mathsf{Exec}(q_i)$ (the i-th suffix of ρ), ρ_i for the finite prefix $(q_k)_{k \leq i}$ (the i-th prefix), and $\rho(i)$ for the i-th state q_i.

Definition 2. *Let* Σ *and* S *be two finite sets. A* Σ-labelled S-tree *is a pair* $\mathcal{T} = \langle T, l \rangle$*, where* $T \subseteq S^*$ *is a non-empty set of finite words on* S *s.t. 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*; and* $l \colon T \to \Sigma$ *is a labelling function.*

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

Definition 3. *For* $P \subseteq \mathsf{AP}$*, two (possibly infinite-state) Kripke structures* $\mathcal{S} = \langle Q, R, \ell \rangle$ *and* $\mathcal{S}' = \langle Q', R', \ell' \rangle$ *are* P-equivalent *(denoted by* $\mathcal{S} \equiv_P \mathcal{S}'$*) whenever* $Q = Q'$*,* $R = R'$*, and* $\ell(q) \cap P = \ell'(q) \cap P$ *for any* $q \in Q$*.*

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

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

2.2 CTL and quantified extensions

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

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

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

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

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

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

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

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

2.3 Structure- and tree semantics

Formulas of the form $\exists p.\varphi$ can be interpreted in different manners (see [15,12,27]). Here we consider two semantics: the *structure semantics* and the *tree semantics*.

Structure semantics. Given a QCTL^* state formula φ, a (possibly infinite-state) Kripke structure $\mathcal{S} = \langle Q, R, \ell \rangle$ and a state $q \in Q$, we write $\mathcal{S}, q \models_s \varphi$ to denote that φ holds at q under the structure semantics. This is defined as for CTL^*, with the following addition:

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

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

Example 6. As an example, consider the formula $\mathsf{selfloop} = \forall z.(z \Rightarrow \mathbf{EX}\, z)$. If a state q in $\mathcal{S}$ satisfies this formula, then the particular labelling in which only q is labelled with z implies that q has to carry a self-loop. Conversely, any state that carries a self-loop satisfies this formula (for the structure semantics).

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

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

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

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

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

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

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

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

2.4 Fragments of QCTL^*.

In the sequel, besides QCTL and QCTL^*, we study several interesting fragments. The first one is the fragment of QCTL in *prenex normal form*, *i.e.*, in which propositional quantification must be external to the CTL formula. We write EQCTL and EQCTL^* for the corresponding logics[5]

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

[5] Notice that the logics named EQCTL and EQCTL^* defined in [15] 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 $\mathsf{EQ}^1\mathsf{CTL}^*$.

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

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

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

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

3 Expressiveness

In this section we present several results about the expressiveness of our logics for both semantics. We show that QCTL, QCTL^* and Monadic Second-Order Logic are equally expressive. First we show that any QCTL formula is equivalent to a formula in prenex normal form (which extends to QCTL^* thanks to Proposition 12).

3.1 Prenex normal form

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

Proposition 9. *In both semantics,* EQCTL *and* QCTL *are equally expressive.*

3.2 QCTL and Monadic Second-Order Logic

We briefly review Monadic Second-Order Logic (MSO) over trees and over finite Kripke structures (*i.e.*, labeled finite graphs). In both 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 (denoted with lowercase letters $x, y, ...$), monadic second-order variables for sets of nodes (denoted with uppercase letters $X, Y, ...$). Atomic formulas are of the form $x = y$, $\mathsf{Edge}(x, y)$, $x \in X$, $\mathsf{P}_a(x)$. Formulas are constructed from atomic formulas using the Boolean connectives and the first- and second-order quantifier $\exists$. We write $\varphi(x_1, ..., x_n, X_1, ..., X_k)$ to state that $x_1, ..., x_n$ and $X_1, ..., X_k$ may appear free (*i.e.* not within the scope of a quantifier) in φ. A closed formula contains no free variable. We use the standard semantics for MSO, writing $\mathcal{M}, s_1, ..., s_n, S_1, ..., S_k \models \varphi(x_1, ..., x_n, X_1, ..., X_k)$ when φ holds on $\mathcal{M}$ when s_i (resp. S_j) is assigned to the variable x_i (resp. X_j) for $i = 1, ..., n$ (resp. $j = 1, ..., k$).

In the following, we compare the expressiveness of our logics with MSO over the finite Kripke structures (the structure semantics) and the execution trees corresponding to a finite Kripke structure (tree semantics). First note that MSO formulas may express properties directly over trees or graphs, while our logics are interpreted over *states* of these structures. Therefore we use MSO formulas with one free variable x, which represents the state where the formula is evaluated. Moreover, we restrict the evaluation of MSO formulas to the *reachable* part of the model from the given state. This last requirement makes an important difference for the structure semantics, since MSO can express that a graph is connected.

Formally, for the tree semantics, we say that $\varphi(x) \in \mathsf{MSO}$ is t-equivalent to some QCTL^* formula ψ (written $\varphi(x) \equiv_t \psi$) when for any finite Kripke structure $\mathcal{S}$ and any state $q \in \mathcal{T}_\mathcal{S}$, it holds $\mathcal{T}_\mathcal{S}(q), q \models \varphi(x)$ iff $\mathcal{T}_\mathcal{S}(q), q \models \psi$. Similarly, for the structure semantics: $\varphi(x)$ is s-equivalent to ψ (written $\varphi(x) \equiv_s \psi$) iff for any finite Kripke structure $\mathcal{S}$ and any state $q \in \mathcal{S}$, it holds $\mathcal{S}_q, q \models \varphi(x)$ iff $\mathcal{S}_q, q \models \psi$, where $\mathcal{S}_q$ is the reachable part of $\mathcal{S}$ from q. For these definitions, we have:

Proposition 10. *Under both semantics,* MSO *and* QCTL *are equally expressive.*

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 $\widehat{\Phi}$. QCTL propositional quantifications are used 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 $x_i \in X_j$ is rewritten in QCTL as $\mathbf{EF}\,(\mathsf{p}_{x_i} \wedge \mathsf{p}_{X_j})$ where p_{x_i} (resp. p_{X_j}) is the proposition associated with x_i (resp. X_j). A formula of the form $\mathsf{Edge}(x_i, x_j)$ is rewritten as $\mathbf{EF}\,(\mathsf{p}_{x_i} \wedge \mathbf{EX}\,\mathsf{p}_{x_j})$, and $x_i = x_j$ is 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 [20].

3.3 QCTL and QCTL*

Finally, we show that QCTL^* and QCTL are equally expressive for both semantics. The main idea of the proof is an inductive replacement of quantified subformulas with extra atomic propositions.

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

Proof. This was shown in [12] for the tree semantic. We give another translation, which is correct for both semantics. Consider a QCTL^* formula Φ, and write k

for the number of subformulas of Φ that are not in QCTL. If $k = 0$, Φ already belongs to QCTL. Otherwise let ψ be one of the inner-most Φ-subformulas in $\mathsf{QCTL}^* \setminus \mathsf{QCTL}$. Let $(\alpha_i)_{1 \leq i \leq m}$ be the largest ψ-subformulas belonging to QCTL. These are state formulas, so that ψ is equivalent (for both semantics) to:

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

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

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

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

From Propositions 9, 10 and 12, we get:

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

Remark 14. In [12], French considers a variant of QCTL^* (which we call FQCTL^*), with propositional quantification within path formulas: $\exists p.\ \varphi_{\mathsf{path}}$ is a valid path formula, meaning that φ_{path} holds along ρ after modifying the labelling with p:

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

For the tree semantics, QCTL is as expressive as FQCTL^* [12]. For the structure semantics, we show that FQCTL^* is strictly more expressive than MSO. Formula

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

expresses the existence of an (infinite) path along which, between any two occurrences of the same state, all the other reachable states will be visited. This precisely characterizes the existence of a Hamilton cycle. This is known not to be expressible in MSO [9, Cor. 6.3.5], it can be expressed in Guarded Second Order Logic GSO (also called MS_2 in [6]), in which quantification over sets of *edges* is allowed (in addition to quantification over sets of states). Still, FQCTL^* is strictly more expressive than GSO, as it is easy to modify the above formula to express the existence of *Euler* cycles:

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

where $\mathsf{tr}(x,y) = \mathsf{uniq}(x) \wedge \mathsf{uniq}(y) \wedge \mathbf{EF}\,(x \wedge \mathbf{X}\,y)$ states that x and y mark the source and target of a reachable transition, and $\mathsf{next\_tr}(x,y) = x \wedge \mathbf{X}\,y$ states that the next transition along the current path jumps from x to y.

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

Still, FQCTL^* model checking (see next section) is decidable: for the tree semantics, it suffices to translate FQCTL^* to QCTL [12]. The problem in the structure semantics can then be encoded in the tree semantics: we first assume that each state of the input Kripke structure $\mathcal{S}$ is labelled with its name (so that any two different states can be distinguished). Then any quantification $\exists p.\varphi$ in the structure semantics is considered in the tree semantics, with the additional requirement that any two copies of the same state receive the same p-labelling.

4 QCTL model checking

We now consider the model-checking problem for QCTL^* and its fragments under both semantics: given a finite Kripke structure $\mathcal{S}$, a state q and a formula[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 [15]. Hardness results for $\mathsf{EQ}^2\mathsf{CTL}$ and $\mathsf{EQ}^2\mathsf{CTL}^*$ under the tree semantics can be found in [16]. Here we extend these results to all the fragments of QCTL^* we have defined. We also characterize the model- and formula-complexities [31] of model-checking for these fragments.

4.1 Model checking for the structure semantics

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

Theorem 16. *Under the structure semantics, model checking* $\mathsf{EQ}^k\mathsf{CTL}$ *is* Σ^{P}_k*-complete, model checking* $\mathsf{AQ}^k\mathsf{CTL}$ *is* Π^{P}_k*-complete, and model checking* $\mathsf{EQ}^k\mathsf{CTL}^*$, $\mathsf{AQ}^k\mathsf{CTL}^*$, EQCTL *and* EQCTL^* *is* PSPACE*-complete.*

General case. If we drop the prenex-normal-form restriction, we get

Theorem 17. *For the structure semantics, model checking is* $\Delta^{\mathsf{P}}_{k+1}[O(\log n)]$*-complete for* $\mathsf{Q}^k\mathsf{CTL}$*, and* PSPACE*-complete for* $\mathsf{Q}^k\mathsf{CTL}^*$, QCTL *and* QCTL^*.

Sketch of proof. The algorithm in $\Delta^{\mathsf{P}}_{k+1}[O(\log n)]$ is obtained by first noticing that a formula $\varphi \in \mathsf{Q}^{k+1}\mathsf{CTL}$ can be written as $\Phi[(q_i \to \exists P_i.\ \psi_i)_i]$ with Φ being a CTL formula involving fresh atomic propositions q_i, and $\exists P_i.\ \psi_i$ (with $\exists P_i$ denoting a sequence of existential quantifications) are subformulas of φ with $\psi_i \in \mathsf{Q}^k\mathsf{CTL}$. The algorithm then consists in asking independant oracles for the sets of states satisfying $\exists P_i.\ \psi_i$, and applying a CTL model-checking algorithm. Hardness is proved by encoding PARITY (Σ^{P}_k), which aims at deciding whether the number of positive instances of Σ^{P}_k in a given set of instances is even [13]. □

[6] For standard notions of size for $\mathcal{S}$ and φ, unless specified otherwise (see Theorem 18).

Formula- and program-complexity. Most of the proofs above can be adapted to use a fixed formula or a fixed model. One notable exception is QCTL: when model checking a fixed formula of QCTL (hence with fixed alternation depth), there is no hope of being able to encode arbitrary alternation: the program complexity of QCTL model checking thus lies in the small gap between PH and PSPACE (unless the polynomial-time hierarchy collapses).

Theorem 18. *Under the structure semantics, the formula-complexity (i.e., when the model is fixed) of model checking is* Σ_k^P*-complete for* $\mathsf{EQ}^k\mathsf{CTL}$*,* Π_k^P*-complete for* $\mathsf{AQ}^k\mathsf{CTL}$*; it is* $\Delta_{k+1}^P[O(\log n)]$*-complete for* $\mathsf{Q}^k\mathsf{CTL}$ *when considering the* DAG-size *of the formula. It is* PSPACE*-complete for* $\mathsf{EQ}^k\mathsf{CTL}^*$*,* $\mathsf{AQ}^k\mathsf{CTL}^*$*,* $\mathsf{Q}^k\mathsf{CTL}^*$*,* EQCTL*,* QCTL*,* EQCTL^**, and* QCTL^**.*

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

4.2 Model checking for the tree semantics

Theorem 19. *Model checking* $\mathsf{EQ}^k\mathsf{CTL}$*,* $\mathsf{AQ}^k\mathsf{CTL}$ *and* $\mathsf{Q}^k\mathsf{CTL}$ *under the tree semantics is* k-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.

▶ *Hardness in* k-EXPTIME. The reduction uses the ideas of [16,29]: we encode an alternating Turing machine $\mathcal{M}$ whose tape 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 is 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.

Our set of formulas will ensure the following (see Fig. 2): given a tree labeled with propositions s and t (among others), both s and t appear exactly once along each branch, and the distance between them is $F(k,n)$, defined as

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

The formulas for $k = 0$ are easy to write. 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 hold at s and t, and at distance $F(k,n)$ from each other inbetween. We then use existential quantification over another proposition in order to implement a counter enforcing that there are exactly $2^{F(k,n)}$ occurrences of r between s and t.

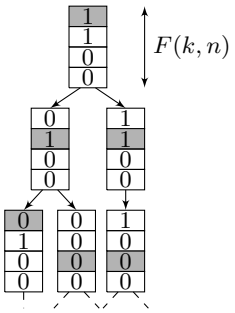

Fig. 1. A run of $\mathcal{M}$

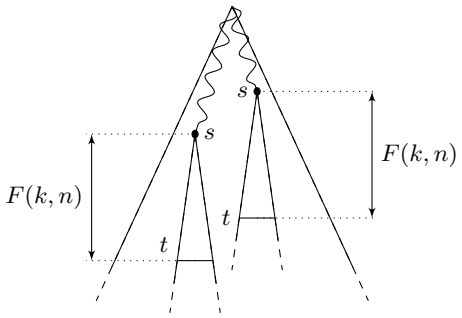

Fig. 2. Chunks of height $F(k, n)$

▶ *Membership in* k-EXPTIME. Our algorithm for $\mathsf{Q}^k\mathsf{CTL}$ model checking uses alternating parity tree automata[21,30]. The construction is inductive: we begin with building automata for the innermost CTL formulas [18], and then use projection to encode existential quantification. This requires turning the alternating automata into non-deterministic ones, which comes with an exponential blowup [22]. 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; emptiness is then solved in time k-exponential [17]. We apply a CTL model-checking algorithm to handle the possible outermost CTL operators. This whole algorithm runs in k-EXPTIME. □

Theorem 20. *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. Hardness is proven by using CTL^* to have $\mathsf{yardstick}_0^n(s,t)$ enforce that the distance between s and t is 2^n. □

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

Theorem 21. *Under the tree semantics, the formula-complexity of model-checking is* k-EXPTIME*-complete for* $\mathsf{EQ}^k\mathsf{CTL}$, $\mathsf{Q}^k\mathsf{CTL}$, $\mathsf{EQ}^k\mathsf{CTL}^*$ *and* $\mathsf{Q}^k\mathsf{CTL}^*$ *with* $k \geq 1$*. It is non-elementary for* EQCTL, QCTL, EQCTL^* *and* QCTL^**.*

The program-complexity of model-checking is PTIME*-complete for all those fragments of* QCTL^**.*

5 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 a

model-checking problem involving a multi-agent system (typically a concurrent game) and a property written in ATL_{sc} (see below) is logspace-reducible to a QCTL model-checking problem.

5.1 Basic definitions

Definition 22 ([1]). *A* Concurrent Game Structure *(*CGS*) $\mathcal{C}$ is a 7-tuple $\langle Q, R, \ell, \mathsf{Agt}, \mathcal{M}, \mathsf{Mov}, \mathsf{Edge}\rangle$ where: $\langle Q, R, \ell\rangle$ is a Kripke structure, $\mathsf{Agt} = \{A_1, ..., A_p\}$ is a finite set of* agents*, $\mathcal{M}$ is a non-empty set of moves, $\mathsf{Mov}\colon Q \times \mathsf{Agt} \to \mathcal{P}(\mathcal{M}) \setminus \{\varnothing\}$ defines the set of available moves of each agent in each state, and $\mathsf{Edge}\colon Q \times \mathcal{M}^{\mathsf{Agt}} \to R$ is a transition table associating, with each state q and each set of moves of the agents, the resulting transition departing from q.*

The size of a CGS $\mathcal{C}$ is $|Q| + |\mathsf{Edge}|$. For a state $q \in Q$, we write $\mathsf{Next}(q)$ for the set of all transitions corresponding to possible moves from q, and $\mathsf{Next}(q, A_j, m_j)$, with $m_j \in \mathsf{Mov}(q, A_j)$, for the restriction of $\mathsf{Next}(q)$ to possible transitions from q when player A_j plays move m_j. We extend Mov and Next to coalitions (*i.e.*, sets of agents) in the natural way. A path in $\mathcal{C}$ is a path in its underlying Kripke structure. For a finite prefix π of a path, we write $\mathsf{last}(\pi) = \pi_i$ for its last state.

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

Let ρ be a history. A strategy $F_A = (f_j)_{A_j \in A}$ for some coalition A induces a set of paths from ρ, called the *outcomes* of F_A after ρ, and denoted $\mathsf{Out}(\rho, F_A)$: an infinite path $\pi = \rho \cdot q_1 q_2 \ldots$ is in $\mathsf{Out}(\rho, F_A)$ iff, writing $q_0 = \mathsf{last}(\rho)$, for all $i \geq 0$ there is a set of moves $(m_k^i)_{A_k \in \mathsf{Agt}}$ 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 $q_{i+1} \in \mathsf{Next}(q_i, \mathsf{Agt}, (m_k^i)_{A_k \in \mathsf{Agt}})$.

We now introduce the extension of ATL with strategy contexts [2,7]:

Definition 23. *The syntax of ATL_{sc} is defined by the following grammar (where p ranges over AP and A over 2^{Agt}):*

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

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

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

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

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

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

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

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

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

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

For a formula of the form $\langle\!\langle A\rangle\!\rangle \mathbf{X}\, \varphi$ with $A = \{A_{j_1}, \ldots, A_{j_l}\}$, we let:

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

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

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

where m is a move $(m^j)_{A_j\in A} \in \mathsf{Mov}(q, A)$ for A and P_m is the propositional formula $\bigwedge_{A_j\in A} m^j$ characterizing m. Formula $\Phi_{\mathsf{strat}}(A_j)$ ensures that, the labelling of propositions m^j_is describes a feasible strategy for A_j. Formula $\Phi^{[A]}_{\mathsf{out}}$ characterizes the outcomes of the strategy for A that is described by the atomic propositions in the model. Note that $\Phi^{[A]}_{\mathsf{out}}$ is based on the transition table Edge of $\mathcal{C}$. Then:

Theorem 24. *Let q be a state in $\mathcal{C}$. Let Φ be an ATL_{sc} formula and F be a strategy context for some coalition C. Let $\mathcal{T}'$ be the execution tree $\mathcal{T}_{\mathcal{S}_{\mathcal{C}}}(q)$ with a labelling function ℓ' s.t. for every $\pi \in \mathsf{Exec}^{\mathsf{f}}(q)$ of length i and any $A_j \in C$, $\ell'(\pi) \cap \mathsf{P}^j_{\mathcal{M}} = \mathsf{m}^j_i$ iff $F(\pi)_{|A_j} = m_i$. Then $\mathcal{C}, q \models_F \Phi$ iff $\mathcal{T}', q \models \widehat{\Phi}^C$.*

We get a non-elementary model-checking algorithm for ATL_{sc}, similar to [7].

Remark 25. The translation above assumes the tree semantics. However, it also makes sense in the structure semantics, where quantification then corresponds to the selection of a *memoryless* strategy. A variant of Theorem 24 can be stated for the structure semantics for QCTL and memoryless strategies for ATL_{sc}.

Remark 26. 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 $\Phi^{[A]}_{\mathsf{out}}$, we can use an extra atomic proposition p_{out} for labelling outcomes. This yields a QCTL formula with the same quantifier height.

Using a converse translation, from QCTL to ATL_{sc}, we can prove:

Theorem 27. *Model-checking the fragment of* ATL_{sc} *with at most* k *non-trivial nested strategy quantifiers is* $\mathsf{k\text{-}EXPTIME}$*-complete.*

Strategy logic (SL) [4,19] is another temporal logic for non-zero-sum games, which has explicit first-order quantification over strategies. Our results above can be adapted to SL, correcting a wrong claim in [19, Theorem 4.2]:

Theorem 28. *The model-checking problems for* QCTL, ATL_{sc} *and* SL *are inter-reducible (in logarithmic space). They all are non-elementary.*

6 Conclusions and future works

We have proposed a complete picture of CTL extended with propositional quantifiers w.r.t. expressiveness and model-checking. On the expressiveness side, we proved how adding quantification on top of CTL fills in the gap between temporal logics and monadic second-order logic. As for model checking, we exhaustively characterized the complexity of QCTL and its variants, completing the earlier results from [15,12]. Finally, we provided an application (which was our original motivation) of QCTL for reasonning about multi-agent systems. Satisfiability of fragments of QCTL^* is part of our future work.

References

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

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

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

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

[5] E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In LOP'81, LNCS 131, p. 52–71. Springer, 1982.

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

[7] A. Da Costa, F. Laroussinie, and N. Markey. ATL with strategy contexts: Expressiveness and model checking. In FSTTCS'10, LIPIcs 8, p. 120–132. LZI, 2010.

[8] A. Da Costa, F. Laroussinie, and N. Markey. Quantified CTL: expressiveness and model checking. Research Report LSV-12-02, Laboratoire Spécification et Vérification, ENS Cachan, France, 2012.
[9] H.-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer, 1995.
[10] E. A. Emerson and J. Y. Halpern. "Sometimes" and "not never" revisited: On branching versus linear time temporal logic. *J. ACM*, 33(1):151–178, 1986.
[11] E. A. Emerson and A. P. Sistla. Deciding full branching time logic. *Inf.&Cont.*, 61(3):175–201, 1984.
[12] T. French. Decidability of quantified propositional branching time logics. In AJCAI'01, LNCS 2256, p. 165–176. Springer, 2001.
[13] G. Gottlob. NP trees and Carnap's modal logic. *J. ACM*, 42(2):421–457, 1995.
[14] Y. Kesten and A. Pnueli. A complete proof systems for QPTL. In LICS'95, p. 2–12. IEEE Comp. Soc. Press, 1995.
[15] O. Kupferman. Augmenting branching temporal logics with existential quantification over atomic propositions. In CAV'95, LNCS 939, p. 325–338. Springer, 1995.
[16] 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.
[17] O. Kupferman and M. Y. Vardi. Weak alternating automata and tree automata emptiness. In STOC'98, p. 224–233. ACM Press, 1998.
[18] O. Kupferman, M. Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model-checking. *J. ACM*, 47(2):312–360, 2000.
[19] F. Mogavero, A. Murano, and M. Y. Vardi. Reasoning about strategies. In FSTTCS'10, LIPIcs 8, p. 133–144. LZI, 2010.
[20] F. Moller and A. Rabinovich. Counting on CTL*: on the expressive power of monadic path logic. *Inf.&Comp.*, 184(1):147–159, 2003.
[21] D. E. Muller and P. E. Schupp. Alternating automata on infinite trees. *TCS*, 54(2-3):267–276, 1987.
[22] D. E. Muller and P. E. Schupp. Simulating alternating tree automata by nondeterministic automata: New results and new proofs of the theorems of Rabin, McNaughton and Safra. *TCS*, 141(1-2):69–107, 1995.
[23] A. C. Patthak, I. Bhattacharya, A. Dasgupta, P. Dasgupta, and P. P. Chakrabarti. Quantified computation tree logic. *IPL*, 82(3):123–129, 2002.
[24] S. Pinchinat. A generic constructive solution for concurrent games with expressive constraints on strategies. In ATVA'07, LNCS 4762, p. 253–267. Springer, 2007.
[25] A. Pnueli. The temporal logic of programs. In FOCS'77, p. 46–57. IEEE Comp. Soc. Press, 1977.
[26] J.-P. Queille and J. Sifakis. Specification and verification of concurrent systems in CESAR. In SOP'82, LNCS 137, p. 337–351. Springer, 1982.
[27] S. Riedweg and S. Pinchinat. Quantified μ-calculus for control synthesis. In MFCS'03, LNCS 2747, p. 642–651. Springer, 2003.
[28] A. P. Sistla. *Theoretical Issues in the Design and Verification of Distributed Systems*. PhD thesis, Harvard University, Cambridge, Massachussets, USA, 1983.
[29] A. P. Sistla, M. Y. Vardi, and P. Wolper. The complementation problem for Büchi automata with applications to temporal logics. *TCS*, 49:217–237, 1987.
[30] W. Thomas. Languages, automata and logics. In *Handbook of Formal Languages*, p. 389–455. Springer, 1997.
[31] M. Y. Vardi. The complexity of relational query languages. In STOC'82, p. 137–146. ACM Press, 1982.
[32] F. Wang, C.-H. Huang, and F. Yu. A temporal logic for the interaction of strategies. In CONCUR'11, LNCS 6901, p. 466–481. Springer, 2011.