On the expressiveness of QCTL

Amélie David[1], François Laroussinie[2], and Nicolas Markey[1]

1 LSV – CNRS, ENS Cachan & University Paris-Saclay – France
2 IRIF – University Paris Diderot & CNRS – France

Abstract

QCTL extends the temporal logic CTL with quantification over atomic propositions. While the algorithmic questions for QCTL and its fragments with limited quantification depth are well-understood (e.g. satisfiability of $\mathsf{Q}^k\mathsf{CTL}$, with at most k nested blocks of quantifiers, is $(k+1)$-EXPTIME-complete), very few results are known about the expressiveness of this logic. We address such expressiveness questions in this paper. We first consider the *distinguishing power* of these logics (i.e., their ability to separate models), their relationship with behavioural equivalences, and their ability to capture the behaviours of finite Kripke structures with so-called characteristic formulas. We then consider their *expressive power* (i.e., their ability to express a property), showing that in terms of expressiveness the hierarchy $\mathsf{Q}^k\mathsf{CTL}$ collapses at level 2 (in other terms, any QCTL formula can be expressed using at most two nested blocks of quantifiers).

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2016.28

1 Introduction

Temporal logics have been introduced in computer science in the late 1970's [18]; they provide a powerful formalism for specifying and verifying (e.g. model checking [19, 3]) correctness properties of (finite-state models representing) evolving systems. Various kinds of temporal logics have been defined, with different expressiveness, succinctness and algorithmic properties. For instance, the *Computation Tree Logic* (CTL) expresses properties of the computation tree of the system under study; it has rather limited expressive power (it cannot express fairness), but enjoys PTIME-complete model-checking. The *Linear-time Temporal Logic* (LTL) expresses properties of a single execution at a time. It can express fairness along that single execution, but cannot express properties of other possible executions; LTL model checking is PSPACE-complete. The logic CTL^* combines CTL and LTL, offering better expressiveness than CTL with the same (theoretical) complexity as LTL.

In terms of expressiveness, CTL^* still has some limitations: in particular, it lacks the ability of *counting*. For instance, it cannot express that an event occurs (at least) at every even position along a path, or that a state has two successors. In order to cope with this, temporal logics have been extended with *propositional quantifiers* [20]: those quantifiers allow for adding fresh atomic propositions in the model before evaluating the truth value of a temporal-logic formula. That a state has at least two successors can then be expressed (in *quantified CTL*, hereafter written QCTL) by saying that it is possible to label the model with atomic proposition p in such a way there is a successor that is labelled with p and one that is not.

The algorithmic questions about QCTL have been extensively studied [14, 7, 17, 8, 4, 15], with various semantic assumptions (in particular, depending on whether the labellings refer to the finite-state model or to its execution tree). In the latter case, model-checking QCTL

Work supported by STREP project Cassting (FP7-601148) and ERC Stg Grant EQualIS (FP7-308087).

27th International Conference on Concurrency Theory (CONCUR 2016).
Editors: Josée Desharnais and Radha Jagadeesan; Article No. 28; pp. 28:1–28:14
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

with at most k nested propositional quantifiers (written $\mathsf{Q}^k\mathsf{CTL}$) has been shown k-EXPTIME-complete [15], which tends to indicate that propositional quantification substantially increases the expressiveness of the logic.

However, the expressiveness of QCTL has remained mostly unexplored, except for a few (rather straightforward) results: QCTL is as expressive as[1] MSO; QCTL and QCTL^* are equally expressive; QCTL formulas can be written in prenex normal form. To the best of our knowledge, no results are known about the relative expressiveness of QCTL and its fragments $\mathsf{Q}^k\mathsf{CTL}$ with limited quantification height.

In this paper, we focus on the so-called *tree semantics*, where quantification refers to the execution tree. The main contributions presented in this paper are the following:

- all logics from $\mathsf{Q}^1\mathsf{CTL}$ to full QCTL^* have the same *distinguishing* power. We define a bisimulation equivalence that precisely corresponds to the distinguishing power of these logics.
- given a regular tree $\mathcal{T}$, one can build a *characteristic formula* $\Phi_{\mathcal{T}}$ in $\mathsf{Q}^2\mathsf{CTL}$ such that any tree $\mathcal{T}'$ satisfying $\Phi_{\mathcal{T}}$ is isomorphic to T. This completes the result of [2], where a construction of characteristic formulas in CTL was presented for the bisimulation equivalence.
- all logics from $\mathsf{Q}^2\mathsf{CTL}$ to QCTL^* have the same expressiveness, but $\mathsf{Q}^1\mathsf{CTL}$ and $\mathsf{Q}^1\mathsf{CTL}^*$ are less expressive. In particular, any QCTL or QCTL^* formula can be translated into a formula in $\mathsf{Q}^2\mathsf{CTL}$ (i.e., with at most two nested blocks of propositional quantifiers)[2].

The outline of the paper is as follows: we begin with setting up the necessary formalism in order to define QCTL^* and its fragments. We then devote Section 3 to the study of the *distinguishing power* of $\mathsf{Q}^k\mathsf{CTL}$, showing in particular that if QCTL can distinguish between two finite-state models, then already $\mathsf{Q}^1\mathsf{CTL}$ can. We also develop *characteristic formulas* in this section. Finally, Section 4 focuses on expressiveness, with as main result the fact that any QCTL^* formula has an equivalent formula in $\mathsf{Q}^2\mathsf{CTL}$, but $\mathsf{Q}^2\mathsf{CTL}$ is strictly more expressive than $\mathsf{Q}^1\mathsf{CTL}^*$.

2 Definitions

2.1 Words and trees

Let Σ be a finite alphabet. A finite word over Σ is a finite sequence $w = (w_i)_{1\leq i\leq k}$. The integer k is the length of w, usually denoted by $|w|$. We write ε for the empty word, which is the unique word of size zero, and identify the alphabet Σ with the set of words of length 1 as long as it raises no ambiguity. For a non-empty word w, we let $\mathsf{last}(w) = w_{|w|}$. The concatenation of two words w and w', denoted $w \cdot w'$, is the word z of size $|w| + |w'|$ defined as

$$z_i = w_i \quad \text{when } 1 \leq i \leq |w| \qquad\qquad z_i = w'_{i-|w|} \quad \text{when } |w| + 1 \leq i \leq |w| + |w'|.$$

For a finite word w and a finite set of words S, we let $w \cdot S = \{w \cdot w' \mid w' \in S\}$. A word w is a prefix of a word z if there exists a word w' such that $z = w \cdot w'$. This defines a partial order $\leq$ over words. An infinite word is the limit of an infinite increasing sequence of finite

[1] This requires adequate definitions, since a temporal logic formula may only deal with the reachable part of the model, while MSO has a more *global* point of view.

[2] Notice that a similar result exists for MSO over trees: one alternation of *second-order quantifiers* is enough to express any MSO property. But while it relies on similar tree-automata techniques, our result does not directly follow from the result for MSO: the translated MSO formula may contain *first-order quantifiers*, which involves extra propositional quantifiers when translated to QCTL.

words; infinite words can equivalently be seen as infinite sequences of letters of Σ. The size of an infinite word w is $|w| = +\infty$. The notions of concatenation of a finite word with an infinite word, and of prefix of an infinite word, are easily obtained from the definitions for finite words. We write Σ^* for the set of finite words, and Σ^ω for the set of infinite words. Given an infinite word w, we write $\mathsf{Inf}(w) \subseteq \Sigma$ for the set of letters that appear infinitely many times in w.

▶ **Definition 1.** Let D be a set and Σ be a finite alphabet. A *Σ-labelled D-tree* is a pair $\mathcal{T} = \langle T, l\rangle$, where

- $T \subseteq D^*$ is a non-empty set of finite words on D satisfying the following constraints: for any non-empty word $x \in T$, which can be written unequivocally as $x = y \cdot c$ with $y \in D^*$ and $c \in D$, the word y is in T. Moreover, we require that for every word $y \in T$, there exists $c \in D$ such that $y \cdot c \in T$.
- $l \colon T \to \Sigma$ is a labelling function.

Let $\mathcal{T} = \langle T, l\rangle$ be a Σ-labelled tree. The elements of T are the *nodes* of $\mathcal{T}$ and the empty word ε, which is easily shown to necessarily belong to T, is the *root* of $\mathcal{T}$. Given a node $x \in T$, we use $\mathsf{Succ}_{\mathcal{T}}(x)$ (or $\mathsf{Succ}(x)$ when the underlying tree is clear) to denote the set of successors of x, defined as $x \cdot \Sigma \cap T$. The *degree* of $x \in T$, denoted $\mathsf{d}_{\mathcal{T}}(x)$ (or $\mathsf{d}(x)$), is the cardinality of $\mathsf{Succ}(x)$. A tree has *bounded branching* if the degree of all its nodes is bounded. Given a node $x \in T$, we denote with $\mathcal{T}_x$ the *subtree* $\langle T_x, l_x\rangle$ rooted at x, defined by $T_x = \{y \in D^* \mid x \cdot y \in T\}$.

An (infinite) *branch* in T is an infinite increasing (for the prefix relation) sequence of nodes. A branch can be identified with an infinite word $\rho = (x_i)_{i\in\mathbb{N}}$ over D; it can be associated with the infinite word $l(\rho) = (l(x_i))_{i\in\mathbb{N}}$ over Σ. A branch ρ contains a node x whenever x is a prefix of ρ.

2.2 Kripke structures

Fix a finite set AP of atomic propositions.

▶ **Definition 2.** A *Kripke structure* is a tuple $\mathcal{K} = \langle V, E, \ell\rangle$ where V is a finite set of vertices, $E \subseteq V \times V$ is a set of edges (requiring that for any $v \in V$, there exists $v' \in V$ s.t. $(v, v') \in E$), and $\ell \colon V \to 2^{\mathsf{AP}}$ is a labelling function.

A path in a Kripke structure is a finite or infinite word w over V such that $(w_i, w_{i+1}) \in E$ for all $i < |w|$. We write $\mathsf{Path}^*_{\mathcal{K}}$ and $\mathsf{Path}^\omega_{\mathcal{K}}$ for the sets of finite and infinite paths of $\mathcal{K}$, respectively. Given a vertex $v \in V$, the execution tree of $\mathcal{K}$ from v is the 2^{AP}-labelled V-tree $\mathcal{T}_{\mathcal{K},v} = \langle T_{\mathcal{K},v}, \hat{\ell}\rangle$ with $T_{\mathcal{K},x} = \{w \in V^* \mid v \cdot w \in \mathsf{Path}^*_{\mathcal{K}}\}$ and $\hat{\ell}(v \cdot w) = \ell(\mathsf{last}(v \cdot w))$. Notice that two nodes w and w' of $\mathcal{T}_{\mathcal{K},v}$ for which $\mathsf{last}(w) = \mathsf{last}(w')$ give rise to the same subtrees. A tree is said *regular* when it corresponds to the execution tree of some finite Kripke structure.

It will be convenient in some situations to allow Kripke structures to have infinitely many states. For instance, a tree can be seen as an infinite-state Kripke structure.

2.3 QCTL* and its fragments

This section is devoted to the definition of the logic QCTL* and its fragments, and to the semantics of these logics.

2.3.1 Syntax and (tree) semantics

▶ **Definition 3.** The syntax of QCTL^* over a finite set AP of atomic propositions is defined by the following grammar:

$$\begin{aligned}\mathsf{QCTL}^* \ni \varphi_s, \psi_s &::= q \mid \neg\varphi_s \mid \varphi_s \vee \psi_s \mid \mathbf{E}\varphi_p \mid \mathbf{A}\varphi_p \mid \exists p.\ \varphi_s \\ \varphi_p, \psi_p &::= \varphi_s \mid \neg\varphi_p \mid \varphi_p \vee \psi_p \mid \mathbf{X}\,\varphi_p \mid \varphi_p \,\mathbf{U}\, \psi_p\end{aligned}$$

where q and p range over AP. Formulas φ_s and ψ_s are called *state formulas*, while φ_p and ψ_p are *path formulas*. The size of a formula $\varphi \in \mathsf{QCTL}^*$, denoted $|\varphi|$, is the number of steps needed to built φ. The logic CTL^* is obtained from QCTL^* by disallowing rule $\exists p.\ \varphi_s$. The logics QCTL and CTL are obtained by using path formulas built with the following grammar:

$$\varphi_p, \psi_p ::= \mathbf{X}\,\varphi_s \mid \varphi_s \,\mathbf{U}\, \psi_s.$$

Finally, LTL is the fragment of CTL^* containing exactly one path quantifier[3] ($\mathbf{E}$ or $\mathbf{A}$); it is easily seen that any LTL formula can be written as $\mathbf{E}\varphi$ or $\mathbf{A}\varphi$, where φ contains no path quantifier.

QCTL^* formulas are evaluated over (execution trees of) finite Kripke structures. We begin with defining the semantics of CTL^* (and CTL). Given a CTL^* formula φ, an infinite 2^{AP}-labelled D-tree $\mathcal{T} = \langle T, l\rangle$, a branch ρ and a node (*i.e.*, a prefix) x of ρ, we write $\mathcal{T}, \rho, x \models \varphi$ to denote that φ holds at x along ρ. This is defined inductively as follows:

$$\begin{aligned}
\mathcal{T}, \rho, x \models p &\text{ iff } p \in l(x) \\
\mathcal{T}, \rho, x \models \neg\varphi &\text{ iff } \mathcal{T}, \rho, x \not\models \varphi \\
\mathcal{T}, \rho, x \models \varphi \vee \psi &\text{ iff } \mathcal{T}, \rho, x \models \varphi \text{ or } \mathcal{T}, \rho, x \models \psi \\
\mathcal{T}, \rho, x \models \mathbf{E}\varphi_p &\text{ iff } \exists \rho' \text{ containing } x.\ \mathcal{T}, \rho', x \models \varphi_p \\
\mathcal{T}, \rho, x \models \mathbf{A}\varphi_p &\text{ iff } \forall \rho' \text{ containing } x.\ \mathcal{T}, \rho', x \models \varphi_p \\
\mathcal{T}, \rho, x \models \mathbf{X}\,\varphi_p &\text{ iff } \exists a \in D.\ x \cdot a \leq \rho \text{ and } \mathcal{T}, \rho, x \cdot a \models \varphi_p \\
\mathcal{T}, \rho, x \models \varphi_p \,\mathbf{U}\, \psi_p &\text{ iff } \exists w \in D^*.\ x \cdot w \leq \rho \text{ and } \\
&\qquad \mathcal{T}, \rho, x \cdot w \models \psi_p \text{ and } \forall w' \lneq w.\ \mathcal{T}, \rho, x \cdot w' \models \varphi_p
\end{aligned}$$

In order to extend this definition to QCTL^*, we first introduce some extra definitions. For a function $l \colon T \to 2^{\mathsf{AP}}$ and $P \subseteq \mathsf{AP}$, we write $l \cap P$ for the function defined as $(l \cap P)(q) = l(q) \cap P$ for all $x \in T$. Now, for $P \subseteq \mathsf{AP}$, two trees $\mathcal{T} = \langle T, l\rangle$ and $\mathcal{T}' = \langle T', l'\rangle$ are said *P-equivalent* (denoted by $\mathcal{T} \equiv_P \mathcal{T}'$) if $T = T'$, and $l \cap P = l' \cap P$. Then:

$$\mathcal{T}, \rho, x \models \exists p.\ \varphi_s \text{ iff } \exists \mathcal{T}' \equiv_{\mathsf{AP}\setminus\{p\}} \mathcal{T} \text{ s.t. } \mathcal{T}', \rho, x \models \varphi_s.$$

It is easily noticed that for any state formula φ_s of QCTL^* and any two paths ρ and ρ' containing node x, we have $\mathcal{T}, \rho, x \models \varphi_s$ if, and only if, $\mathcal{T}, \rho', x \models \varphi_s$. In view of this, we define $\mathcal{T}, x \models \varphi_s$ in the natural way. Finally, for a Kripke structure $\mathcal{K}$ and one of its states v, we write $\mathcal{K}, v \models \varphi_s$ whenever $\mathcal{T}_{\mathcal{K},v}, \varepsilon \models \varphi_s$.

In the sequel, we use standard abbreviations such 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 $\forall p.\ \varphi_s = \neg\exists p.\ \neg\varphi_s$. Also note that $\mathbf{A}\varphi_p = \neg\,\mathbf{E}\neg\varphi_p$.

[3] It is more usual to assume that LTL formulas contain no path quantifier at all; our definition allows for a more uniform presentation, and fits better in our branching-time setting, making it clear how LTL formulas are to be evaluated (existentially or universally) in a tree.

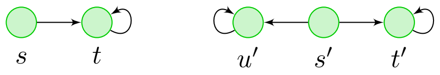

Figure 1 Two Kripke structures (with empty labelling functions) with s and s' being bisimilar

2.3.2 Discussion on the semantics.

Several other natural semantics coexist in the literature for propositional quantifiers. Above we have introduced the *tree semantics*: $\exists p.\ \varphi$ holds true when there exists a p-labelling of the *execution tree* of the Kripke structure under which φ holds. Therefore two nodes (in the tree) corresponding to the same state of the Kripke structure may be labelled differently with the newly-quantified propositions. For example, in the Kripke structure depicted at Fig. 1, we have: $s \models \exists p.(\mathbf{EX}\, p \wedge \mathbf{EX}\ \mathbf{EX}\ \neg\, p)$, because it is possible to label only the first occurrence of t with p in the execution tree of this Kripke structure.

Another classical semantics (called the *structure semantics*) consists in labelling the Kripke structure directly. With this semantics, $\exists p.(\mathbf{EX}\, p \wedge \mathbf{EX}\ \mathbf{EX}\ \neg\, p)$ would not hold true in state s of Fig. 1: all the occurrences of state t in the execution tree would be labelled the same way.

These two semantics have very different properties (see [15] for a deeper study of these semantics). But none of them make QCTL bisimulation-invariant[4]: as we exemplify in the next section, under both semantics, QCTL can *count* the number of successors of a given state (and thus distinguish between states s and s' in Fig. 1, even though they are bisimilar).

Finally, let us mention the *amorphous semantics* [7], where $\exists p.\ \varphi$ holds true at a state s in some Kripke structure $\mathcal{K}$ if, and only if, there exists some Kripke structure $\mathcal{K}'$ with a state s' such that s and s' are bisimilar and for which there exists a p-labeling making φ hold true at s'. With this semantics, the logic is insensitive to unwinding, and more generally it is bisimulation-invariant (for example, states s and s' of Fig. 1 satisfy the same formulas). This semantics corresponds to bisimulation quantification as studied in [5, 9].

2.3.3 Fragments of QCTL*.

The central topic of this paper is the hierarchy of temporal logics defined by restricting quantifications in QCTL formulas. We define this hierarchy here. Given QCTL* (state) formulas φ and $(\psi_i)_i$, and atomic propositions $(p_i)_i$ that appear free in φ (*i.e.*, not 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 $L_1[L_2] = \{\varphi[(\psi_i)_i] \mid \varphi \in L_1,\ (\psi_i)_i \in L_2\}$.

For a set $P = \{p_i \mid 1 \leq i \leq k\} \subseteq \mathsf{AP}$, we define *blocks of existential quantifiers* $\exists P.\ \varphi$ as a shorthand for $\exists p_1.\ \exists p_2 \ldots \exists p_k.\ \varphi$. We write $\mathsf{EQ}^1\mathsf{CTL}$ for the set of formulas of the form $\exists P.\ \varphi$ for $\varphi \in \mathsf{CTL}$, and define $\mathsf{Q}^1\mathsf{CTL} = \mathsf{CTL}[\mathsf{Q}^1\mathsf{CTL}]$ and $\mathsf{Q}^{k+1}\mathsf{CTL} = \mathsf{Q}^1\mathsf{CTL}[\mathsf{Q}^k\mathsf{CTL}]$.

▶ **Example 4.** *Consider formula*

$$\mathbf{E}_1\mathbf{X}\,\varphi = \mathbf{EX}\,\varphi \wedge \neg\,\exists p.\ \big(\mathbf{EX}\,(p \wedge \varphi) \wedge \mathbf{EX}\,(\neg\, p \wedge \varphi)\big)$$

(where we assume that p does not appear in φ). This $\mathsf{Q}^1\mathsf{CTL}$ formula states that there is exactly one successor satisfying φ: if there were two of them, then labelling only one of them with p

[4] The notion of bisimulation is formally defined in Section 2.4.

would falsify the formula. Now, for $P = \{p_i \mid 1 \leq i \leq k\} \subseteq \mathsf{AP}$, consider the following formula

$$\varphi_{\mathit{dupl}(P)} = \exists\{q_1, q_2\}.\ \mathbf{E}_1\mathbf{X}\,(q_1) \wedge \mathbf{E}_1\mathbf{X}\,(q_2) \wedge \mathbf{AX}\,(\neg\, q_1 \vee \neg\, q_2) \wedge \bigwedge_{p\in P} \big(\mathbf{EX}\,(q_1 \wedge p) \Leftrightarrow \mathbf{EX}\,(q_2 \wedge p)\big).$$

where $P \cap \{q_1, q_2\} = \varnothing$. This formula in $\mathsf{Q}^2\mathsf{CTL}$ holds true whenever there are two different successor nodes that carry the same atomic propositions of P. Formula $\varphi_{\mathit{exp}} = \forall P.\ \varphi_{\mathit{dupl}(P)}$ in $\mathsf{Q}^3\mathsf{CTL}$ is then true in nodes that have at least $2^k + 1$ successors: it states that any labelling with P gives rise to at least two successors with the exact same labelling. Of course, a simpler formula could be obtained for expressing the existence of exactly k successors satisfying a given property; for instance:

$$\mathbf{E}_k\mathbf{X}\,\varphi = \exists P.\ \Big[\bigwedge_{1\leq i\leq k} \mathbf{E}_1\mathbf{X}\,p_i \wedge \bigwedge_{1\leq i\neq j\leq k} \mathbf{AX}\,\neg(p_i \wedge p_j) \wedge \mathbf{AX}\,\Big(\big(\bigvee_{1\leq i\leq k} p_i\big) \Leftrightarrow \varphi\Big)\Big].$$

This formula is in $\mathsf{Q}^2\mathsf{CTL}$. We can express that at least *k successors satisfy φ in $\mathsf{Q}^1\mathsf{CTL}$ as follows:*

$$\mathbf{E}_{\geq k}\mathbf{X}\,\varphi = \exists P.\ \Big(\bigwedge_{1\leq i\leq k} \mathbf{EX}\,(p_i \wedge \bigwedge_{i'\neq i} \neg\, p_{i'}) \wedge \mathbf{AX}\,\Big(\big(\bigvee_{1\leq i\leq k} p_i\big) \Rightarrow \varphi\Big)\Big).$$

2.4 Expressive power and distinguishing power

In the sequel, we compare the relative expressiveness of the fragments $\mathsf{Q}^k\mathsf{CTL}$. Several criteria are classically used to compare the expressiveness of temporal logics: one can compare their ability to distinguish between models (the *distinguishing power*), their ability to express properties (the *expressive power*), or their *succinctness*. In this paper, we only consider the former two notions, which we now formally define.

Distinguishing power. A logic $\mathcal{L}$ is said to be *at least as distinguishing* as another logic $\mathcal{L}'$ over a class $\mathcal{M}$ of models, denoted $\mathcal{L} \geq_{\mathcal{M}} \mathcal{L}'$ (we may omit to mention $\mathcal{M}$ when it is clear from the context), whenever any two states s and s' of any two structures $\mathcal{K}$ and $\mathcal{K}'$ in $\mathcal{M}$ that are $\mathcal{L}$-equivalent (i.e., for all $\varphi \in \mathcal{L}$, it holds $\mathcal{K}, s \models \varphi$ if, and only if, $\mathcal{K}', s' \models \varphi$) are also $\mathcal{L}'$-equivalent. Both logics are said *equally distinguishing*, written $\mathcal{L} \equiv_{\mathcal{M}} \mathcal{L}'$, if $\mathcal{L} \geq_{\mathcal{M}} \mathcal{L}'$, and $\mathcal{L}' \geq_{\mathcal{M}} \mathcal{L}$; finally, $\mathcal{L}$ is *strictly more distinguishing* than $\mathcal{L}'$, denoted $\mathcal{L} >_{\mathcal{M}} \mathcal{L}'$, whenever $\mathcal{L} \geq_{\mathcal{M}} \mathcal{L}'$, and $\mathcal{L}' \not\geq_{\mathcal{M}} \mathcal{L}$. In our setting, $\mathcal{M}$ is the class of all finite Kripke structures.

For classical branching-time temporal logics, it is well known [10] that CTL^*, CTL, and the fragment $\mathsf{B}(\mathbf{X}\,)$ of CTL not involving the *Until* modality, all have the same distinguishing power. Note also that the distinguishing power is often related to some behavioral equivalence. Here we recall the classical notion of (strong) bisimulation: given two Kripke structures $\mathcal{K} = \langle V, E, \ell\rangle$ and $\mathcal{K}' = \langle V', E', \ell'\rangle$, a relation $R \subseteq V \times V'$ is a bisimulation when for any $(v, v') \in R$, the following properties hold:

- $\ell(v) = \ell'(v')$;
- for any transition $(v, w) \in E$, there is a transition $(v', w') \in E$ such that $(w, w') \in R$;
- for any transition $(v', w') \in E$, there is a transition $(v, w) \in E$ such that $(w, w') \in R$.

Two states v and v' are bisimilar (denoted by $v \sim v'$) whenever there exists a bisimulation relation R such that $(v, v') \in R$.

Bisimilarity characterizes the distinguishing power of CTL^*, CTL and $\mathsf{B}(\mathbf{X}\,)$ for finitely-branching Kripke structures: in particular, two bisimilar states cannot be distinguished by CTL [10]. For instance, CTL cannot distinguish between a finite Kripke structure and its execution tree.

Expressive power. A logic $\mathcal{L}$ is said to be *at least as expressive* as a logic $\mathcal{L}'$ over a class $\mathcal{M}$ of models, which we denote by $\mathcal{L} \succeq_{\mathcal{M}} \mathcal{L}'$ (omitting to mention $\mathcal{M}$ if it is clear from the context), whenever for any formula $\varphi' \in \mathcal{L}'$, there is a $\varphi \in \mathcal{L}$ such that φ and φ' are equivalent over $\mathcal{M}$. Both logics $\mathcal{L}$ and $\mathcal{L}'$ are *equally expressive*, denoted $\mathcal{L} \cong_{\mathcal{M}} \mathcal{L}'$, when $\mathcal{L} \succeq \mathcal{L}'$ and $\mathcal{L}' \succeq \mathcal{L}$; finally, $\mathcal{L}$ is strictly more expressive than $\mathcal{L}'$, written $\mathcal{L} \succ_{\mathcal{M}} \mathcal{L}'$, if $\mathcal{L} \succeq \mathcal{L}'$ and $\mathcal{L}' \not\succeq \mathcal{L}$.

One easily notices that expressive power is finer than distinguishing power: being more distinguishing implies being more expressive. The converse is not true: for instance, CTL^* is strictly more expressive than CTL [6], and CTL is strictly more expressive than $\mathsf{B}(\mathbf{X})$.

3 Distinguishing power of QCTL

In this section, we prove the following:

▶ **Theorem 5.** *Over finite Kripke structures, $\mathsf{CTL} < \mathsf{Q}^1\mathsf{CTL} \equiv \mathsf{Q}^k\mathsf{CTL} \equiv \mathsf{Q}^k\mathsf{CTL}^* \equiv \mathsf{QCTL} \equiv \mathsf{QCTL}^*$ for $k \geq 1$.*

That $\mathsf{CTL} < \mathsf{Q}^1\mathsf{CTL}$ is easily observed, for instance using the Kripke structures of Fig. 1: states s and s' are bisimilar, thus equivalent for CTL, but formula $\mathbf{E}_1\mathbf{X}\top$ ("*there is exactly one successor*") is true in s and false in s'. We now prove the equivalences of Theorem 5.

3.1 Characteristic formulas with QCTL

As said above, CTL has enough power to distinguish between two non-bisimilar states. More precisely, given a finite Kripke structure $\mathcal{K}$ and a state v, one can build a CTL formula $\alpha_{\mathcal{K},v}$ such that for any Kripke structure $\mathcal{K}'$ and any state v', we have: $\mathcal{K}', v' \models \alpha_{\mathcal{K},v}$ if, and only if, $v \sim v'$ [2].

For QCTL and QCTL^*, the appropriate behavioural equivalence is the *isomorphism of the execution tree.* Two trees $\mathcal{T} = \langle T, \ell\rangle$ and $\mathcal{T}' = \langle T', \ell'\rangle$ are said isomorphic if there exists a bijection $\varphi\colon T \to T'$ such that $\ell'(\varphi(t)) = \ell(t)$ for all t, and $\varphi(\varepsilon_{\mathcal{T}}) = \varepsilon_{\mathcal{T}'}$ and $\varphi(u)$ is a successor of $\varphi(t)$ in $\mathcal{T}'$ if, and only if, u is a successor of t in $\mathcal{T}$. As we now explain, QCTL can capture the behaviour of $\mathcal{K}$ up to *tree isomorphism*: there exists a $\mathsf{Q}^2\mathsf{CTL}$ formula $\beta_{\mathcal{K},v}$ such that for any tree $\mathcal{T}'$, it holds $\mathcal{T}' \models \beta_{\mathcal{K},v}$ if, and only if, $\mathcal{T}_{\mathcal{K},v}$ and $\mathcal{T}'$ are isomorphic.

Given a finite Kripke structure $\mathcal{K} = \langle V, E, \ell\rangle$, and a vertex $v \in V = \{v_0, \ldots, v_n\}$, we define the $\mathsf{Q}^2\mathsf{CTL}$ formula $\beta_{\mathcal{K},v}$ as follows:

$$\beta_{\mathcal{K},v} = \exists V.\Big[v \wedge \mathbf{AG} \bigwedge_{i=0}^{n} \Big(v_i \Rightarrow \big(\bigwedge_{j\neq i} \neg v_j \wedge \bigwedge_{p\in\ell(v_i)} p \wedge \bigwedge_{p\in\mathsf{AP}\setminus V\cup\ell(v_i)} \neg p \wedge \\ \neg\, \mathbf{E}_{\geq \mathsf{d}(v_i)+1}\mathbf{X}\top \wedge \bigwedge_{(v_i,v_j)\in E} \mathbf{E}_1\mathbf{X}\, v_j\Big)\Big]$$

Formula $\beta_{\mathcal{K},v}$ holds true at the root of a tree $\mathcal{T}'$ when it is possible to associate with every node of $\mathcal{T}$ a state v of $\mathcal{K}$ in such a way that this node will behave exactly as v (same labelling and same successors).

▶ **Lemma 6.** *Let $\mathcal{K} = (V, E, \ell)$ be a finite Kripke structure, and $v \in V$. For any tree $\mathcal{T}' = (T', \ell')$, we have: $\mathcal{T}', \varepsilon_{\mathcal{T}'} \models \beta_{\mathcal{K},v}$ if, and only if, $\mathcal{T}_{\mathcal{K},v}$ and $\mathcal{T}'$ are isomorphic.*

Lemma 6 shows that $\mathsf{Q}^2\mathsf{CTL}$ is powerful enough to distinguish between two finite Kripke structures that do not have isomorphic execution trees. Conversely:

▶ **Lemma 7.** *Two finite Kripke structures that have isomorphic execution trees cannot be separated by QCTL^*.*

3.2 $\mathsf{Q}^1\mathsf{CTL}$, QCTL and QCTL^* have the same distinguishing power

We show in this section that $\mathsf{Q}^1\mathsf{CTL}$ is sufficient to separate non-isomorphic trees.

▶ **Proposition 8.** $\mathsf{Q}^1\mathsf{CTL}$ has the same distinguishing power as QCTL and QCTL^* over finite Kripke structures.

Proof. Given a *finite* tree $\mathcal{U}$, we can easily define a $\mathsf{Q}^1\mathsf{CTL}$ formula $\gamma_\mathcal{U}$ expressing that $\mathcal{U}$ is embedded as a subtree, in the following sense: any infinite tree $\mathcal{T}$ satisfying $\gamma_\mathcal{U}$ contains a subtree that is isomorphic to $\mathcal{U}$. Write $U = \{p_i \mid 0 \le i \le k\}$ for the set of nodes of $\mathcal{U}$ (with $p_0 = \varepsilon_\mathcal{U}$ being the root of $\mathcal{U}$). Formula $\gamma_\mathcal{U}$ first existentially quantifies over $p_0, \dots, p_k$ (seen here as atomic propositions), labelling nodes of $\mathcal{T}$ with names of nodes of $\mathcal{U}$. Then $\gamma_\mathcal{U}$ checks that

- p_0 holds true initially;
- at most one p_i can be true at a time;
- if p_j is a successor of p_i in $\mathcal{U}$, then $\gamma_\mathcal{U}$ enforces $\mathbf{AG}\,(p_i \Rightarrow \mathbf{EX}\,p_j)$;
- the labelling function of $\mathcal{T}$ matches that of $\mathcal{U}$.

Notice that we do not prevent any of the p_i to hold true at several places, which would require an extra universal quantification. Obviously, any tree $\mathcal{T}$ containing a subtree isomorphic to $\mathcal{U}$ satisfies $\gamma_\mathcal{U}$. The converse also holds: assuming $\mathcal{T}$ has been labelled with $\{p_i \mid 0 \le i \le k\}$, we extract a subtree $\mathcal{V}$ as follows: the root of $\mathcal{T}$ (labelled with p_0) is in $\mathcal{V}$, and for each node n in $\mathcal{V}$ labelled with some p_i, for each successor p_j of p_i in $\mathcal{U}$, we insert into $\mathcal{V}$ exactly one successor of n labelled with p_j ($\gamma_\mathcal{U}$ enforces the existence of such a node). It is easily seen that $\mathcal{V}$ is isomorphic to $\mathcal{U}$.

Now, if two regular trees are not isomorphic, there must exist a finite subtree $\mathcal{U}$ of one of them that cannot be embedded into the second one. Then $\gamma_\mathcal{U}$ will distinguish between these two trees. It follows that $\mathsf{Q}^1\mathsf{CTL}$ and QCTL (and QCTL^*) have the same distinguishing power over finite Kripke structures. ◀

3.3 Behavioural equivalences for $\mathsf{Q}^k\mathsf{CTL}$

We conclude our study of the fragments of QCTL by defining intermediary notions of bisimulations which we prove characterize each level of the $\mathsf{Q}^k\mathsf{CTL}$ hierarchy. We begin with defining those refined bisimulations. In this definition, for two labelling functions $\ell\colon T \to 2^{\mathsf{AP}}$ and $\nu\colon T \to 2^P$ with $P \subseteq \mathsf{AP}$, we write $\ell \circ \nu\colon T \to 2^{\mathsf{AP}}$ for the labelling function mapping each $t \in T$ to $[\ell(t) \cap (\mathsf{AP} \setminus P)] \cup \nu(t)$.

▶ **Definition 9.** Consider two regular trees $\mathcal{T} = \langle T, \ell\rangle$ and $\mathcal{T}' = \langle T', \ell'\rangle$. A relation $R \subseteq T \times T'$ is a *k-labelling bisimulation* if R is a bisimulation and either $k = 0$, or $k > 0$ and for any $(t, t') \in R$, for any $P \subseteq \mathsf{AP}$ and any regular labelling $\nu\colon T \to 2^P$ (resp. $\nu'\colon T \to 2^P$), there exists a regular labelling $\nu'\colon T' \to 2^P$ (resp. $\nu\colon T \to 2^P$) such that there exists a $(k-1)$-labelling bisimulation in the trees $\mathcal{U} = \langle T, \ell \cup \nu\rangle$ and $\mathcal{U}' = \langle T', \ell' \cup \nu'\rangle$ containing (t, t'). We write $t \approx_k t'$ when there exists a k-labelling bisimulation containing (t, t').

Notice that for all k, it holds $\approx_{k+1} \subseteq \approx_k$. The relation $\approx_\infty$ can thus be defined as the limit of these sequences of relations.

In this definition, with *regular labelling* of a regular tree $\langle T, \ell\rangle$, we mean a labelling ν such that $\langle T, \ell \circ \nu\rangle$ is still regular. We let $\exists_r p.\varphi$ be a new QCTL^* modality where the quantification ranges only over regular labellings. Formally:

$$\mathcal{T}, \rho, x \models \exists_r p.\varphi \text{ iff } \exists \nu\colon T \to 2^{\{p\}}.\ \nu \text{ is regular and } \langle T, \ell \circ \nu\rangle, \rho, x \models \varphi.$$

Quantifying over regular labelling does not restrict the expressive power of our logics:

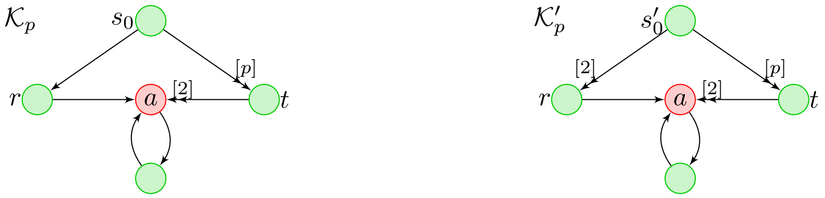

Figure 2 The states s_0 and s'_0 can be distinguished by $\mathbf{E}_1\mathbf{X}\,(\mathbf{E}_1\mathbf{X}\,a)$ (we use double arrows labelled with $[k]$ to indicate the presence of k arrows).

▶ **Lemma 10.** *Given a finite Kripke structure $\mathcal{K}$, a state s, and a QCTL* formula $\exists p.\varphi$,*

$$\mathcal{K}, s \models \exists p.\varphi \text{ iff } \mathcal{K}, s \models \exists_r p.\varphi$$

Fix two Kripke structures $\mathcal{K}$ and $\mathcal{K}'$, and a logic $\mathcal{L}$ (intended to range over Q^{k}CTL and Q^{k}CTL*). For any two states s and s', in $\mathcal{K}$ and $\mathcal{K}'$ respectively, we write $s \equiv_{\mathcal{L}} s'$ when s and s' are $\mathcal{L}$-equivalent (i.e., when they cannot be distinguished by $\mathcal{L}$). It is easily noticed that if $\mathcal{L} \subseteq \mathcal{L}'$, then $\equiv_{\mathcal{L}'} \subseteq \equiv_{\mathcal{L}}$.

▶ **Lemma 11.** *Over finite Kripke structures, for any $k \geq 0$, the relations $\approx_k$, $\equiv_{\mathsf{Q}^k\mathsf{CTL}}$ and $\equiv_{\mathsf{Q}^k\mathsf{CTL}^*}$ coincide. More precisely, for any two states s and s',*

$$s \approx_k s' \text{ iff } s \equiv_{\mathsf{Q}^k\mathsf{CTL}} s' \text{ iff } s \equiv_{\mathsf{Q}^k\mathsf{CTL}^*} s'.$$

As a corollary of Lemmas 6, 7 and 11, we get:

▶ **Corollary 12.** *For every $k, k', k'' \geq 2$, the relations $\equiv_{\mathsf{Q}^k\mathsf{CTL}^*}$, $\equiv_{\mathsf{Q}^{k'}\mathsf{CTL}}$, $\equiv_{\mathsf{QCTL}}$, $\approx_{k''}$, and $\approx_\infty$ coincide.*

4 Expressive power of QCTL

We now focus on the relative *expressive power* of the Q^{k}CTL hierarchy. Notice that being more distinguishing implies being more expressive. Hence we already have CTL $\prec$ Q^{1}CTL. In this section, we prove the following:

▶ **Theorem 13.** *Over finite Kripke structures, CTL $\prec$ Q^{1}CTL $\preceq$ Q^{1}CTL* $\prec$ Q^{2}CTL $\approxeq$ Q^{k}CTL $\approxeq$ Q^{k}CTL* $\approxeq$ QCTL $\approxeq$ QCTL* for $k \geq 2$.*

4.1 Q^{2}CTL is strictly more expressive than Q^{1}CTL and Q^{1}CTL*

In order to prove this, we have to exhibit a formula of Q^{2}CTL with no equivalent formula in Q^{1}CTL. First consider the Kripke structures depicted at Fig. 2. Those structures depend on an integer parameter p. As stated in the following lemma, these structures cannot be distinguished by any Q^{1}CTL* formula of size less than p:

▶ **Lemma 14.** *For the Kripke structures $\mathcal{K}_p$ and $\mathcal{K}'_p$ of Fig. 2, and for any $\varphi \in$ Q^{1}CTL* of size less than p it holds $\mathcal{K}_p, s_0 \models \varphi$ if, and only if, $\mathcal{K}'_p, s'_0 \models \varphi$.*

▶ **Theorem 15.** *We have Q^{2}CTL $\succ$ Q^{1}CTL and Q^{2}CTL $\succ$ Q^{1}CTL*.*

Proof. Q^{1}CTL is syntactically contained in Q^{2}CTL, so that Q^{2}CTL $\succeq$ Q^{1}CTL. Moreover, from Theorem 16, every QCTL* formula can be expressed in Q^{2}CTL, i.e. Q^{2}CTL $\succeq$ QCTL*, which in particular entails Q^{2}CTL $\succeq$ Q^{1}CTL*.

Moreover the $\mathsf{Q}^2\mathsf{CTL}$ formula $\mathbf{E}_1\mathbf{X}\,(\mathbf{E}_1\mathbf{X}\,a)$ (which states that there exists a unique path leading to a state where there is a unique successor verifying a) allows us to distinguish the trees of Fig. 2 (for any p): indeed $\mathbf{E}_1\mathbf{X}\,(\mathbf{E}_1\mathbf{X}\,a)$ holds true in s_0, but not in s'_0.

Now assume that $\mathbf{E}_1\mathbf{X}\,(\mathbf{E}_1\mathbf{X}\,a)$ have an equivalent formula in $\mathsf{Q}^1\mathsf{CTL}^*$. On the one hand, for any p, this formula would holds true in s_0 and not in s'_0; on the other hand, it would have a given size p_0, and according to Lemma 14 it could not distinguish between s_0 and s'_0. Hence $\mathbf{E}_1\mathbf{X}\,(\mathbf{E}_1\mathbf{X}\,a)$ has no $\mathsf{Q}^1\mathsf{CTL}^*$ formula. ◀

4.2 QCTL and Q^{2}CTL are equally expressive

In this section we prove that the hierarchies $\mathsf{Q}^k\mathsf{CTL}$ and $\mathsf{Q}^k\mathsf{CTL}^*$ also collapse in terms of expressive power. We propose an effective translation, using symmetric tree automata, which proves the following result:

▶ **Theorem 16.** *Any QCTL* formula can be translated into an equivalent formula in Q^{2}CTL.*

4.2.1 Symmetric tree automata

We consider here so-called *symmetric* tree automata (*i.e.*, automata over *unranked* trees of arbitrary branching), borrowing formalism from MSO-automata of [12, 13, 24], in order to prove that the expressiveness hierarchy of $\mathsf{Q}^k\mathsf{CTL}$ collapses: the proof consists in first translating any QCTL formula into a (symmetric) tree automaton, and then expressing acceptance of such a tree automaton as a $\mathsf{Q}^2\mathsf{CTL}$ formula. Using "classical" tree automata (as in [16]), the second step would not be possible (at least not easily), as QCTL cannot distinguish the different successors in a ranked tree; moreover, it would only provide an equivalent formula for a limited branching degree.

In the literature, a similar construction is done for MSO [23, 11, 1, 24]: MSO-automata are powerful tree automata whose transition functions are defined with first-order-logic formulas (where the states of the automata are used as unary predicates); this provides a powerful way of describing transition functions for unranked trees with arbitrary branching. Then any MSO formula φ can be turned into a MSO-automaton $\mathcal{A}_\varphi$ recognizing exactly the trees satisfying φ. Here we use an slightly different (but equally expressive) model of tree automata, which correspond to MSO-automata where transition functions are in a so-called *basic form* (see [24] for full details). Formally:

▶ **Definition 17.** Fix an alphabet Σ. A *symmetric parity tree automaton* over Σ is a tuple $\mathcal{A} = \langle Q, q_0, \delta, \Omega\rangle$ where

- Q is the finite set of states of the automaton, and q_0 is the initial state;
- $\delta\colon Q \times \Sigma \to 2^{(\mathbb{N}^{2^Q} \times 2^{2^Q})}$ is the transition function;
- $\Omega\colon Q \to \mathbb{N}$ defines the parity acceptance condition.

An execution of such a tree automaton $\langle Q, q_0, \delta, \Omega\rangle$ over a Σ-labelled D-tree $\mathcal{T} = \langle T, l\rangle$ is a $Q \times T$-labelled $Q \times D$-tree $\mathcal{T}' = \langle T', l'\rangle$ satisfying the following requirements:

- $l'(\varepsilon_{T'}) = (q_0, \varepsilon_T)$, and for any node $n = (q_i, d_i)_{1\le i\le k}$, it holds $l'(n) = (q_k, (d_i)_{1\le i\le k})$;
- for any node $n = (q_i, d_i)_{1\le i\le k}$ of T', there is a tuple $(E, U) \in \delta(l'(n))$ such that, writing $e = \sum_{s\in 2^Q} E(s)$ and viewing E as a multiset $\{E_j \mid 1 \le j \le e\}$, there exists a set $\Delta = \{d'_j \mid 1 \le j \le e\}$ of e distinct directions in D such that
  - for all $1 \le j \le e$, it holds $(d_i)_{1\le i\le k} \cdot d'_j \in T$,
  - for all $1 \le j \le e$, and for all $q \in E_j$, node $n \cdot (q, d'_j)$ is in T',

- for all node $(d_i)_{1\leq i\leq k+1}$ in T such that $d_{k+1} \notin \Delta$, for there exists $\overline{q} \in U$, such that for all $q \in \overline{q}$, node $n \cdot (q, d_{k+1})$ is in T'.

A branch of an execution tree is accepting if its associated sequence of states of $\mathcal{A}$ satisfies the parity condition (the least priority appearing infinitely often is even). An execution tree is accepting whenever all its branches are. The language $\mathcal{L}(\mathcal{A})$ of such an automaton $\mathcal{A}$ is the set of trees over which $\mathcal{A}$ has an accepting execution tree.

▶ **Example 18.** *Let* $\mathsf{AP} = \{a, b\}$ *and* $\Sigma = 2^{\mathsf{AP}}$*. Let* $\mathcal{A}$ *be the automaton with states* $\{q_0, q_1, q_2\}$*, with* q_0 *being the initial state, and with*

$$\delta(q_0,\sigma) = \begin{cases} (q_1 \mapsto 1; q_0) & \textit{if } a \in \sigma \\ (\varnothing; q_0) & \textit{otherwise} \end{cases} \qquad \delta(q_1,\sigma) = \begin{cases} (\varnothing; q_2) & \textit{if } b \notin \sigma \\ (q_1 \mapsto 1; q_0) & \textit{if } a, b \in \sigma \\ (\varnothing; q_0) & \textit{otherwise} \end{cases}$$

and $\delta(q_2, \sigma) = (\varnothing; q_2)$ *for any* σ*. The transition* $\delta(q_0, a) = (q_1 \mapsto 1; q_0)$ *means that when the automaton is visiting some node* n *in state* q_0*, one successor of* n *will be visited in state* q_1*, and all the other nodes will be visited in state* q_0*. Similarly,* $\delta(q_2, \sigma) = (\varnothing; q_2)$ *indicates that when the automaton is in state* q_2*, it will remain in state* q_2 *when visiting all the successors of the node being visited; in this case,* $\varnothing$ *represents the empty multiset, or equivalently the constant function* $\mathbf{0}$*. In the end, assuming that* q_0 *and* q_1 *are accepting (Büchi condition), this automaton accepts those tree in which any node labelled with* a *has a successor labelled with* b*.*

As defined above, symmetric tree automata are alternating, in the sense that they may launch several computations along the same subtree of the input tree. A symmetric tree automaton is said *non-alternating*[5] when the transition function takes values in $2^{(\mathbb{N}^Q \times 2^Q)}$. One may notice that in this case, any execution tree can be seen as a D-tree, instead of a $Q \times D$-tree. The automaton of Example 18 is non-alternating.

It is not difficult to notice that symmetric tree automata are closed under union: given two symmetric tree automata $\mathcal{A}$ and $\mathcal{B}$, there exists a tree automaton $\mathcal{C}$ accepting the union of the set of trees accepted by $\mathcal{A}$ and $\mathcal{B}$. Similarly, *non-alternating* symmetric tree automata are easily seen to be closed under projection.

Symmetric tree automata can also be proven to be closed under complement; this however involves dualizing the transition function and writing it back under the expected *basic form*. Finally, any symmetric tree automaton can be transformed into a non-alternating one accepting the same set of trees. This can be achieved by a refined powerset construction, as explained in [24]. In the end:

▶ **Theorem 19** ([24]). *Any symmetric tree automaton can be made non-alternating. Non-alternating symmetric tree automata are closed under union, intersection, projection and complement.*

4.2.2 From QCTL to symmetric tree automata

In this section, we briefly explain how a state formula of QCTL^* can be translated into a (non-alternating) symmetric tree automaton accepting the same tree language. This construction

[5] The classical terminology for this class of automata is *non-deterministic*, for historical reasons. We prefer using *non-alternating*, which better characterizes this class.

follows the same ideas as explained in [16]: in that paper however, the construction builds a "classical" tree automaton, running on bounded-branching ranked trees. The translation back to QCTL is not possible for such tree automata, as QCTL cannot distinguish between ranked successors of a node.

The construction is inductive, using Theorem 19 for the various rules defining state-formulas of QCTL*. The basic cases of atomic propositions and boolean operations are easy to handle. Existential quantification over atomic propositions is handled by projection: given a tree automaton $\mathcal{A}_\varphi$ corresponding to a QCTL* formula φ, the projection of $\mathcal{A}_\varphi$ from alphabet 2^{AP} to alphabet $2^{\mathsf{AP}\setminus\{p\}}$ yields a tree automaton characterizing formula $\exists p.\ \varphi$.

Finally, formulas of the form $\mathbf{E}\varphi_p$ and $\mathbf{A}\varphi_p$ are handled by considering word automata for the path formula φ_p: nested QCTL subformulas can be handled separately, by induction, and replaced by fresh atomic propositions. The resulting formula $\tilde{\varphi}_p$ is a pure LTL formula, and can be turned into a deterministic parity word automaton, which in turn is easily turned into a symmetric tree automaton for $\mathbf{E}\tilde{\varphi}_p$ or $\mathbf{A}\tilde{\varphi}_p$. The nested QCTL subformulas can then be included back by plugging the corresponding symmetric tree automata where needed.

Notice that this construction involves several exponential blowups in the size of the automaton and in the number of priorities. Since model checking Q^{k}CTL formulas is k-EXPTIME-complete, there is no hope of avoiding this non-elementary explosion in the construction of the automaton (because our translation back into Q^{2}CTL is linear in the size of the automaton, as we explain below).

4.2.3 From symmetric tree automata to Q^{2}CTL

In this section, we turn a non-alternating symmetric tree automaton $\mathcal{A} = (Q, q_0, \delta, \Omega)$ into a QCTL formula $\Phi_\mathcal{A}$ such that $\mathcal{T} \in \mathcal{L}(\mathcal{A}) \Leftrightarrow \mathcal{T}, \varepsilon \models \Phi_\mathcal{A}$ for any 2^{AP}-labelled tree $\mathcal{T}$.

We begin with a preliminary lemma, which we believe is interesting in itself but will be used in a special case in the sequel.

▶ **Lemma 20.** *For any LTL formula $\mathbf{E}\varphi$, there exists a Q^{1}CTL formula $\Psi_{\mathbf{E}\varphi}$ such that for all tree $\mathcal{T}$, it holds $\mathcal{T}, \varepsilon \models \mathbf{E}\varphi$ if, and only if, $\mathcal{T}, \varepsilon \models \Psi_{\mathbf{E}\varphi}$.*

Proof. Following classical techniques [22, 21], we associate with φ a Büchi (word) automaton $\mathcal{B}_\varphi = \langle Q, q_0, \delta, F\rangle$ accepting exactly the set of infinite words in which φ holds. The automaton $\mathcal{B}_\varphi$ is a non-deterministic word automaton, so that for any $q \in Q$ and any $\sigma \in 2^{\mathsf{AP}}$, it holds $\delta(q, \sigma) \subseteq Q$. The acceptance condition is defined in terms of a set $F \subseteq Q$ of states: an execution is accepting if some state of F is visited infinitely many times.

We use this automaton in order to write a Q^{1}CTL-formula characterizing those trees containing (at least) one branch accepted by $\mathcal{B}_\varphi$. The formula expresses that the tree can be (partially) labelled with states of $\mathcal{B}_\varphi$ in such a way that at least one branch is fully labelled with a sequence of states corresponding to an accepting run of $\mathcal{B}_\varphi$. Following this intuition, we use the states of $Q = \{q_i \mid 0 \le i \le n = |Q| - 1\}$ as new atomic propositions. For the sake of readability, we define the following two shorthand formulas: for a subset $S \subseteq Q$, formula λ_S is the propositional formula $\bigvee_{q\in S} q$, while for $P, P' \subseteq \mathsf{AP}$, we write $\chi_{P,P'}$ for $\bigwedge_{p\in P} p \wedge \bigwedge_{p'\in P'} \neg p'$. We then let $\Psi_{\mathbf{E}\varphi} = \exists q_0 \ldots \exists q_n.\ \widetilde{\Psi}_{\mathbf{E}\varphi}$, with $\widetilde{\Psi}_{\mathbf{E}\varphi}$ being defined as

$$q_0 \wedge \bigwedge_{i=0}^{n} \mathbf{AG}\left(q_i \Rightarrow \left(\neg\,\lambda_{Q\setminus\{q_i\}} \wedge \left[\bigvee_{P\subseteq \mathsf{AP}'} (\chi_{P,\mathsf{AP}'\setminus P} \wedge \mathbf{EX}\,\lambda_{\delta(q_i,P)} \wedge \neg\,\mathbf{EX}\,\lambda_{Q\setminus\delta(q_i,P)})\right]\right)\right)$$
$$\wedge\Big(\mathbf{AG}\ \mathbf{AF}\,(\neg\,\lambda_Q \vee \lambda_F)\Big)$$

where AP' stands for $\mathsf{AP} \setminus \{q_0, \ldots, q_n\}$. Formula $\Psi_{\mathbf{E}\varphi}$ reads as follows: it is possible to label the input tree with propositions $(q_i)_{0\le i\le n}$ in such a way that the root is labelled with q_0, and any node labelled with some q_i is not labelled with any other state and has a successor node labelled with a possible successor state of $\mathcal{B}_\varphi$. This in particular entails that at least one branch ρ is fully labelled with states of Q. Finally, the second part of the formula asserts that any branch has to fulfill the Büchi acceptance condition or to contain unlabelled nodes. In particular, the branch ρ identified above satisfies the Büchi condition.

It is now easy to prove equivalence of $\mathbf{E}\varphi$ and $\Psi_{\mathbf{E}\varphi}$:

- consider a tree $\mathcal{T} = \langle T, l\rangle$ in which one branch ρ satisfies φ: then $\mathcal{B}_\varphi$ has an accepting run on $l(\rho)$, and this run can be used to label $\mathcal{T}$ with states of $\mathcal{B}_\varphi$ so as to fulfill $\widetilde{\Psi}_{\mathbf{E}\varphi}$;
- conversely, if $\mathcal{T}$ can be labelled with states of Q in order to fulfill $\widetilde{\Psi}_{\mathbf{E}\varphi}$, then one branch has to be fully labelled, and each node along that branch will be labelled with exactly one state of Q. Formula $\Psi_{\mathbf{E}\varphi}$ then enforces that the labelling of consecutive nodes is coherent with the transition function of $\mathcal{B}_\varphi$, and that it satisfies the Büchi condition. ◀

We now describe our main construction: we consider a non-alternating symmetric parity tree automaton $\mathcal{A} = \langle Q, q_0, \delta, \Omega\rangle$, where $Q = \{q_i \mid 0 \le i \le n = |Q|-1\}$ and $\delta(q,\sigma) \subseteq \mathbb{N}^Q \times 2^Q$ is a set of pairs (E,U) with $E\colon Q \to \mathbb{N}$ and $U \subseteq Q$. For such a pair (E,U), we write $k(E) = \sum_{q\in Q} E(q)$, and we let $k_{\max}$ be the largest such value appearing in δ. We also see E as a multiset $\{E_i \mid 1 \le i \le k(E)\}$.

Reusing ideas (and notations) of the proof of Lemma 20, and with $\mathsf{AP}' = \mathsf{AP} \setminus \{q_0, \ldots, q_n, p_1, \ldots, p_{k_{\max}}\}$, our formula $\Phi_{\mathcal{A}}$ is written as $\exists q_0 \ldots \exists q_n . \exists p_1 \ldots \exists p_{k_{\max}}.\ \widetilde{\Phi}_{\mathcal{A}}$, where $\widetilde{\Phi}_{\mathcal{A}}$ is defined as

$$q_0 \wedge \bigwedge_{i=0}^{n} \mathbf{AG}\Big[q_i \Rightarrow \Big(\neg\lambda_{Q\setminus\{q_i\}} \wedge \bigvee_{P\subseteq \mathsf{AP}'} \Big(\chi_{P,\mathsf{AP}'\setminus P} \wedge \bigvee_{(E,U)\in\delta(q_i,P)} \Psi_{(E,U)}\Big)\Big)\Big] \wedge \neg\,\Psi_{\mathbf{E}\,\neg\,\mathsf{parity}(\Omega)}$$

where $\Psi_{(E,U)}$ encodes the transition (E,U) of $\mathcal{A}$ and $\Psi_{\mathbf{E}\,\neg\,\mathsf{parity}(\Omega)}$ encodes the parity acceptance condition. Using Lemma 20, the latter formula can be expressed as a $\mathsf{Q}^1\mathsf{CTL}$ formula (since parity acceptance condition can be expressed in LTL). Now, we let

$$\Psi_{(E,U)} = \bigwedge_{j=1}^{k(E)} \Big[\mathbf{E}_1\mathbf{X}\, p_j \wedge \mathbf{EX}\Big(p_j \wedge \bigwedge_{\substack{1\le j'\le k_{\max}\\ \wedge\, j'\neq j}} \neg p_{j'} \wedge E_j\Big) \wedge \mathbf{AX}\Big(\big(\bigwedge_{j=1}^{k} \neg p_j\big) \Rightarrow \bigvee_{q\in U} q\Big)\Big]$$

Note that $\Psi_{(E,U)}$ belongs to $\mathsf{Q}^1\mathsf{CTL}$ (because it uses $\mathbf{E}_1\mathbf{X}\, p_j$), so that $\Phi_{\mathcal{A}}$ is in $\mathsf{Q}^2\mathsf{CTL}$. It is not hard to see that $\Phi_{\mathcal{A}}$ characterizes the behaviour of $\mathcal{A}$, since the labelling of a tree $\mathcal{T}$ with $\{q_i \mid 0 \le i \le n\}$ corresponds to an execution tree of $\mathcal{A}$ on $\mathcal{T}$. This ends the proof of Theorem 16.

5 Concluding remarks

We see two main directions for future work. First it would be interesting to consider the expressiveness of QCTL^* fragments when the size of block of quantifiers (*i.e.* the number of atomic propositions used in a block) is bounded (several of our proofs use arbitrary many propositions). The second direction is to analyze the expressiveness of these logics in the context of the *structure semantics* (when the labellings apply to Kripke structures instead of execution trees) in order to see whether the hierarchy also collapses for this semantics.

References

1 D. Berwanger and A. Blumensath. The monadic theory of tree-like structures. In *Automata, Logics, and Infinite Games*, volume 2500 of *LNCS*, pages 285–302. Springer, 2002.

2 M. C. Browne, E. M. Clarke, and O. Grumberg. Characterizing finite Kripke structures in propositional temporal logic. *Theoretical Computer Science*, 59(1-2):115–131, 1988.

3 E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *LOP'81*, volume 131 of *LNCS*, pages 52–71. Springer, 1982.

4 A. Da Costa, F. Laroussinie, and N. Markey. Quantified CTL: Expressiveness and model checking. In *CONCUR'12*, volume 7454 of *LNCS*, pages 177–192. Springer, 2012.

5 G. D'Agostino and M. Hollenberg. Logical questions concerning the μ-calculus: Interpolation, lyndon and łoś-tarski. *Journal of Symbolic Logic*, 65(1):310–332, 2000.

6 E. A. Emerson and J. Y. Halpern. "Sometimes" and "not never" revisited: On branching versus linear time temporal logic. *Journal of the ACM*, 33(1):151–178, 1986.

7 T. French. Decidability of quantified propositional branching time logics. In *AJCAI'01*, volume 2256 of *LNCS*, pages 165–176. Springer, 2001.

8 T. French. Quantified propositional temporal logic with repeating states. In *TIME-ICTL'03*, pages 155–165. IEEE Comp. Soc. Press, 2003.

9 T. French. *Bisimulation Quantifiers for Modal Logics.* Ph.D. thesis, School of Computer Science & Software Engineering, University of Western Australia, 2006.

10 M. C. B. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32(1), 1985.

11 D. Janin and G. Lenzi. On the relationship between monadic and weak monadic second order logic on arbitrary trees. *Fundamenta Informaticae*, 61(3-4):247–265, 2004.

12 D. Janin and I. Walukiewicz. Automata for the modal μ-calculus and related results. In *MFCS'95*, volume 969 of *LNCS*, pages 552–562. Springer, 1995.

13 J. Johannsen and M. Lange. CTL^+ is complete for double exponential time. In *ICALP'03*, volume 2719 of *LNCS*, pages 767–775. Springer, 2003.

14 O. Kupferman. Augmenting branching temporal logics with existential quantification over atomic propositions. In *CAV'95*, volume 939 of *LNCS*, pages 325–338. Springer, 1995.

15 F. Laroussinie and N. Markey. Quantified CTL: expressiveness and complexity. *Logicical Methods in Computer Science*, 10(4), 2014.

16 F. Laroussinie and N. Markey. Augmenting ATL with strategy contexts. *Information and Computation*, 245:98–123, 2015.

17 A. C. Patthak, I. Bhattacharya, A. Dasgupta, P. Dasgupta, and P. P. Chakrabarti. Quantified computation tree logic. *Information Processing Letters*, 82(3):123–129, 2002.

18 A. Pnueli. The temporal logic of programs. In *FOCS'77*, pages 46–57. IEEE Comp. Soc. Press, 1977.

19 J.-P. Queille and J. Sifakis. Specification and verification of concurrent systems in CESAR. In *SOP'82*, volume 137 of *LNCS*, pages 337–351. Springer, 1982.

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

21 M. Y. Vardi. Nontraditional applications of automata theory. In *TACS'94*, volume 789 of *LNCS*, pages 575–597. Springer, 1994.

22 M. Y. Vardi and P. Wolper. Automata theoretic techniques for modal logics of programs. *Journal of Computer and System Sciences*, 32(2):183–221, 1986.

23 I. Walukiewicz. Monadic second order logic on tree-like structures. *Theoretical Computer Science*, 275(1-2):311–346, 2002.

24 F. Zanasi. *Expressiveness of Monadic Second-Order Logics on Infinite Trees of Arbitrary Branching Degrees.* Master's thesis, Amsterdam University, the Netherlands, 2012.