Arbitrary-Arity Tree Automata for QCTL

François Laroussinie
IRIF, Université Paris Cité, France

Nicolas Markey
IRISA – Inria, CNRS, Univ. Rennes 1, France

Abstract

We introduce a new class of automata (which we coin EU-automata) running on infinite trees of arbitrary (finite) arity. We develop and study several algorithms to perform classical operations (union, intersection, complement, projection, alternation removal) for those automata, and precisely characterise their complexities. We also develop algorithms for solving membership and emptiness for the languages of trees accepted by EU-automata. We then use EU-automata to obtain several algorithmic and expressiveness results for the temporal logics QCTL and QCTL* (which extends CTL and CTL* with quantification over atomic propositions) and for MSO.

2012 ACM Subject Classification Theory of computation → Logic and verification

Keywords and phrases Model-checking, Verification, Automata theory, Quantified CTL

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2025.28

Related Version *Full Version*: https://arxiv.org/abs/2410.18799 [17]

Acknowledgements We thank Igor Walukiewicz for answering our questions on MSO-automata, and the reviewers for their careful reading and constructive comments.

1 Introduction

Logics and automata. The very tight links between logics and automata on infinite words and trees date back to the early 1960's with the seminal works of Büchi, Elgot, Trakhtenbrot, McNaughton and Rabin [4, 8, 26, 21, 23]. These early results were mainly concerned with the Monadic Second-Order Logic (MSO), and have been further extended to many other logical formalisms such as modal, temporal and fixpoint logics [24, 28, 2, 13, 32]. Those tight links are embodied as translations back and forth between various logical languages and corresponding classes of automata; translations from logics to automata have allowed to derive efficient algorithms for satisfiability or model checking on the one hand [27, 9, 2]; with additional translations from automata to logics, we get effective ways for proving expressiveness or succinctness results for some of those logics [29, 31, 19, 16, 33]. In this paper, we introduce a new flavour of automata running over trees of arbitrary arity [29, 31, 16], develop and analyse tools to manipulate them, and derive algorithmic and expressiveness results for MSO and the temporal logic Quantified CTL (QCTL) [14, 15, 11, 7] and its fragments.

QCTL. QCTL extends the classical temporal logic CTL with quantification on atomic propositions. For instance, formula $\exists p.\phi$, where ϕ is a CTL formula, states that there exists a labelling of the model under scrutiny with proposition p under which ϕ holds. QCTL is (much) more expressive than CTL: as an example, formula $\exists p.\ (\mathbf{EF}(\phi \wedge p) \wedge \mathbf{EF}(\phi \wedge \neg p))$ expresses the fact that there are at least two reachable states where ϕ holds. The extension of CTL with only *existential* quantification was first studied in [10, 14]: contrary to CTL, the resulting logic is sensitive to unwinding and duplication of transitions; the semantics thus depends on whether the extra labelling refers to the Kripke structure under scrutiny, or on its computation tree. Our sample formula above expresses that there are at least two *different* reachable control states satisfying ϕ in the former case (which we call the *structure semantics*), while it only requires that two different paths lead to some ϕ-states (possibly two copies of the same control state) in the latter semantics (called the tree semantics hereafter). In this paper, we will only consider QCTL with the tree semantics. In that setting, it turns out that QCTL is as expressive as MSO over infinite trees [18].

Tree automata. We use (top-down[1]) tree-automata techniques to study QCTL. Several results already exist on this topic [10, 14, 18], but they all rely on fixed-arity tree automata.

The limitation has several drawbacks. When dealing with model checking, it implies that the compilation of the formula being checked into a tree automaton depends on the (size of the) structure under scrutiny. In particular, it cannot be used directly for evaluating the *program complexity* of QCTL model checking, as it requires bounding the size of the structures that the automaton can handle. An indirect solution to this problem is given in [18], by replacing nodes of arbitrary (finite) arity with binary-tree gadgets. A similar problem occurs when dealing with satisfiability: one has to use additional results to ensure that looking for a structure with bounded size is sufficient. More importantly, when deriving expressiveness results, using fixed-arity tree automata again restricts the results to trees or structures with bounded branching.

In order to handle trees of arbitrary branching degree, tree automata must have a *symbolic* way of expressing transitions, with a finite representation that can cope with any arity. Several solutions have been proposed [1, 13, 29, 31, 16]; we highlight two of them:

- Janin and Walukiewicz introduce MSO-automata [13, 29], in which transitions are defined as first-order formulas: quantification is over the successors of the current node, and predicates indicate in which states of the automaton those successors must be explored. These automata are shown to be as expressive as MSO, and several expressiveness results have been obtained from this construction [13, 30, 33]. However, to the best of our knowledge, the exact complexity of the operations for manipulating those automata has not been studied, and no bounds on the size and complexity of the translations can be derived without a more careful study.
- Wilke introduces $\{\Box, \Diamond\}$-automata [31], which are alternating tree automata with $\Box q$ and $\Diamond q$ as basic blocks for expressing transitions: the former requires that all successors be explored in state q, while the latter asks that some successor be explored in state q. Any CTL formula can be turned into an equivalent $\{\Box, \Diamond\}$-automaton of linear size; this is used to prove that the extension CTL^+ of CTL is exponentially more succinct that CTL. However, $\{\Box, \Diamond\}$-automata are not expressive enough to capture MSO or QCTL.

Our contribution. In this paper, we define a new class of symmetric arbitrary-arity alternating tree automata, develop effective operations for their manipulation, and study the complexity of those operations and the size of the resulting automata. Instead of using pairs (k, q) in the transition function to specify that the k-th successor of the current node has to be accepted by the automaton in state q, transitions of our automata are defined with pairs $\langle E; U\rangle$, where E is a multiset of states that have to occur among the set of states involved in the exploration of the successors of the current node, while U is a set of states indicating which states are allowed for exploring successor nodes that are not explored by

[1] There are several families of tree automata: top-down tree automata explore (finite or infinite) trees starting from the root; bottom-up tree automata explore finite trees from the leaves up to the root; tree-walking automata are a kind of two-way automata for trees. We refer to [6, 3] for more details.

states of E. For example, $\langle E = \{\!\{q, q, q'\}\!\}; U = \{q''\}\rangle$ requires the presence of at least three successors nodes; two successors will be explored in state q, one in state q', and the remaining ones (if any) in state q''. We name those automata EU-automata[2].

It is not hard to prove that such automata are closed under conjunction and disjunction, thanks to alternation. Closure under negation is harder to prove: while $\Box$ and $\Diamond$ are dual to each other, which provides an easy complementation procedure for $\{\Box, \Diamond\}$-automata, there is no obvious way of expressing the negation of EU-pairs in terms of EU-pairs. We develop such a translation, and obtain an exponential complementation procedure for EU-automata.

We show that non-alternating EU-automata are also closed under projection, which is the operation we need to encode quantification over atomic propositions of QCTL, and first- and second-order quantification of MSO. Finally we prove that any alternating EU-automaton can be turned into an equivalent non-alternating EU-automaton. For this, we adapt the simulation procedure developed in [29, 33] for MSO-automata to our setting, and evaluate its exact complexity. Note that we use a fine-tuned notion of size for our EU-automata, with different components (number of states, size of Boolean formulas in the transition function, size of EU-pairs, number of (parity) priorities); this allows us to precisely estimate the complexity of these operations.

Putting all the pieces together, we prove that any QCTL formula φ can be turned into an equivalent EU-automaton $\mathcal{A}_\varphi$. The size of the automaton is k-exponential in the size of φ, where k is the number of nested quantifier blocks in φ. This construction then yields optimal algorithms for model-checking and satisfiability for QCTL. Conversely, we prove that acceptance by any EU-automaton can be expressed as a simple QCTL formula. We obtain similar results for QCTL* and MSO. Therefore EU-automata, QCTL, and MSO all characterise exactly the same tree languages.

2 Definitions

2.1 Preliminary definitions

Sets and multisets. Let $\mathcal{S}$ be a finite set. A *multiset* over $\mathcal{S}$ is a mapping $\mu\colon \mathcal{S} \to \mathbb{N}$. Sets are seen as special cases of multisets taking values in $\{0, 1\}$. We use double-brace notation to distinguish between sets and multisets: $\{a, a, a\}$ is the same as the set $\{a\}$ with one element, while $\{\!\{a, a, a\}\!\}$ is the three-element multiset $a \mapsto 3$. The *empty multiset* is the multiset mapping all elements of $\mathcal{S}$ to zero; we denote it with $\varnothing$.

The *support of a multiset* μ is the set $\mathsf{supp}(\mu) = \{s \in \mathcal{S} \mid \mu(s) > 0\}$. The *size* $|\mu|$ of μ is the sum $\sum_{s\in\mathcal{S}} \mu(s)$. For two multisets μ and μ', we write $\mu \sqsubseteq \mu'$, and say that μ is a submultiset of μ', whenever $\mu(s) \leq \mu'(s)$ for all $s \in \mathcal{S}$. This defines a partial ordering over multisets. We write $\mu \sqsubset \mu'$ when $\mu \sqsubseteq \mu'$ and $\mu \neq \mu'$. We define the following operations on multisets: $\mu \uplus \mu'\colon s \in \mathcal{S} \mapsto \mu(s) + \mu'(s)$ and $\mu' \setminus \mu\colon s \in \mathcal{S} \mapsto \max(0, \mu'(s) - \mu(s))$.

Fix a second set $\mathcal{S}'$. For any $c = (s, s') \in \mathcal{S} \times \mathcal{S}'$, we define $\mathsf{proj}_1(c) = s$ and $\mathsf{proj}_2(c) = s'$.

Markings. Let $\mathcal{S}$ and $\mathcal{S}'$ be two finite sets. A *marking* of $\mathcal{S}'$ by $\mathcal{S}$ is a mapping $\nu\colon \mathcal{S}' \to 2^{\mathcal{S}} \setminus \{\varnothing\}$ decorating each element of $\mathcal{S}'$ with a (non-empty) subset of $\mathcal{S}$. A marking ν is a *submarking* of a marking ν', denoted $\nu \sqsubseteq \nu'$, whenever $\nu(s') \subseteq \nu'(s')$ for all $s' \in \mathcal{S}'$.

[2] In [16], Kupferman and Vardi define another variant of arbitrary-arity alternating tree automata in which transitions are based on pairs (U, E). Those automata are equivalent to Wilke's $\{\Box, \Diamond\}$-automata.

A marking ν is *unitary* when $|\nu(s')| = 1$ for all $s' \in \mathcal{S}'$; unitary markings can be seen as mappings from $\mathcal{S}'$ to $\mathcal{S}$. For a unitary marking ν and a subset T of $\mathcal{S}'$, we write $\nu(T)$ for the multiset μ over $\mathcal{S}$ defined as $\mu(s) = \#\{t \in T \mid \nu(t) = s\}$, which we may also write as $\{\!\{\nu(t) \mid t \in T\}\!\}$. We write $\mathsf{img}(\nu)$ for the multiset $\nu(\mathcal{S}')$.

Words and trees. Let Σ be a finite set. A *word* over Σ (or Σ-*word*) is a sequence $w = (w_i)_{0 \leq i < k}$ of elements of Σ, with $k \in \mathbb{N} \cup \{+\infty\}$. The *length* (or *size*) of w, denoted with $|w|$, is k. We write Σ^* for the set of finite words over Σ, and Σ^∞ for the set of infinite words over Σ. We write ε for the *empty word* (the only word of size 0).

Let $\mathcal{D}$ be a finite set. A *tree structure* over $\mathcal{D}$ (or $\mathcal{D}$-*tree*) is a subset $t \subseteq \mathcal{D}^*$ that is closed under prefix. The empty word ε is its *root*. The elements of a tree are called nodes. A node m in t is a *successor* of a node n if $m = n \cdot d$ for some $d \in \mathcal{D}$. In that case, n is the (unique) predecessor of m. We write $\mathsf{succ}(n)$ for the set of successors of node n. Notice that in a $\mathcal{D}$-tree, any node may have at most $|\mathcal{D}|$ successors; this integer $|\mathcal{D}|$ is the *arity* of the tree. A *branch* of a tree is a (finite of infinite) sequence $b = (n_i)_{0 \leq i < k}$ of nodes of the tree such that $n_0 = \varepsilon$, n_{i+1} is a successor of n_i for all $0 \leq i < k-1$, and if k is finite, n_{k-1} is a leaf. The value of $k \in \mathbb{N} \cup \{+\infty\}$ is the length of b, denoted with $|b|$.

A Σ-*labelled* $\mathcal{D}$-*tree* is a pair $\mathcal{T} = (t, l)$ where t is a $\mathcal{D}$-tree and $l \colon t \to \Sigma$ labels each node of t with a letter in Σ. With any branch $b = (n_i)_{0 \leq i < k}$ of t in a Σ-labelled $\mathcal{D}$-tree $\mathcal{T} = (t, l)$, we associate its *word* $w(b)$ over Σ as the word $(w_i)_{0 \leq i < k}$ defined as $w_i = l(n_i)$ for all $0 \leq i < k$.

2.2 Automata over trees of arbitrary arity

In this section, we introduce our automata running over trees of arbitrary arity. The core element of their transition functions are EU-pairs and EU-constraints.

EU-pairs and EU-constraints. Let $\mathcal{S}$ be a countable set. An *EU-pair* over $\mathcal{S}$ is a pair $\langle E; U\rangle \in \mathbb{N}^{\mathcal{S}} \times 2^{\mathcal{S}}$, where E is a multiset over $\mathcal{S}$ and U is a subset of $\mathcal{S}$. A multiset μ over $\mathcal{S}$ satisfies the EU-pair $\langle E; U\rangle$, denoted $\mu \Vdash \langle E; U\rangle$, whenever $E \sqsubseteq \mu$ and $\mathsf{supp}(\mu \setminus E) \subseteq U$. We write $\mathsf{EU}(\mathcal{S}) = \mathbb{N}^{\mathcal{S}} \times 2^{\mathcal{S}}$ for the set of EU-pairs over $\mathcal{S}$.

▶ **Example 1.** Consider a set $\mathcal{S} = \{q_1, q_2, q_3, q_4\}$. The EU-pair $\langle q_1 \mapsto 3, q_2 \mapsto 1; \{q_1, q_3\}\rangle$ characterises all multisets containing *at least* three occurrences of q_1, *exactly* one occurrence of q_2, an arbitrary number of occurrences of q_3, and no occurrences of q_4. ⌟

For a finite set $\mathcal{B}$ of variables, we write $\mathsf{PBF}(\mathcal{B})$ for the set of *positive boolean combinations* over $\mathcal{B}$: $\mathsf{PBF}(\mathcal{B}) \ni \phi ::= \top \mid \bot \mid v \mid \phi \wedge \phi \mid \phi \vee \phi$ where v ranges over $\mathcal{B}$. The set of *disjunctions* over $\mathcal{B}$, denoted $\mathsf{DBF}(\mathcal{B})$, is the subset of $\mathsf{PBF}(\mathcal{B})$ where conjunctions are not allowed.

An *EU-constraint* is a positive boolean formula over EU-pairs (that is an element of $\mathsf{PBF}(\mathsf{EU}(\mathcal{S}))$) whose semantics is defined as follows:

▶ **Definition 2.** *Let $\mathcal{S}$ and $\mathcal{S}'$ be two countable sets. Let $\langle E; U\rangle$ be an EU-pair over $\mathcal{S}$, and ν be a marking of $\mathcal{S}'$ by $\mathcal{S}$. Then ν satisfies $\langle E; U\rangle$, denoted $\nu \models\!\!\!\equiv \langle E; U\rangle$, if there exists a unitary marking $\nu' \sqsubseteq \nu$, such that $\mathsf{img}(\nu') \Vdash \langle E; U\rangle$.*

This definition extends to EU-constraints inductively as follows:

- *$\nu \models\!\!\!\equiv \phi_1 \vee \phi_2$ if, and only if, $\nu \models\!\!\!\equiv \phi_1$ or $\nu \models\!\!\!\equiv \phi_2$;*
- *$\nu \models\!\!\!\equiv \phi_1 \wedge \phi_2$ if, and only if, $\nu \models\!\!\!\equiv \phi_1$ and $\nu \models\!\!\!\equiv \phi_2$.*

Notice that $\nu \models\!\!\!\equiv \phi$ is *not* equivalent to having an unitary marking $\nu' \sqsubseteq \nu$ satisfy ϕ, since different submarkings ν' may be needed for different EU-pairs. However, the equivalence holds if ϕ is a disjunction of EU-pairs, since in that case a single EU-pair has to be fulfilled.

EU tree automata. We can now define our class of automata:

▶ **Definition 3.** *Let Σ be a finite alphabet. An* alternating EU parity tree automaton *(*AEUPTA *for short) over Σ is a 4-tuple $\mathcal{A} = (Q, q_{init}, \delta, \omega)$ with*

- *Q is a finite set of states, and $q_{init} \in Q$ is the initial state;*
- *$\delta\colon Q \times \Sigma \to \mathit{PBF}(\mathit{EU}(Q))$ is the set of transitions;*
- *$\omega\colon Q \to \mathbb{N}$ is a* priority *function defining the acceptance condition (as a* parity condition*).*

An AEUPTA is non-alternating[3] *(and is thus an* EU parity tree automaton*,* EUPTA *for short) if δ takes values in $\mathit{DBF}(\mathit{EU}(Q))$.*

We can then define the notion of execution tree of an AEUPTA:

▶ **Definition 4.** *Let $\mathcal{A} = (Q, q_{init}, \delta, \omega)$ be an AEUPTA over Σ and $\mathcal{T} = (t, l)$ be a Σ-labelled $\mathcal{D}$-tree, for some finite set $\mathcal{D}$. An* execution tree *of $\mathcal{A}$ over $\mathcal{T}$ is a $(t \times Q)$-labelled $(\mathcal{D} \times Q)$-tree $\mathcal{U} = (u, \ell)$ such that*

- *the root ε_u of u is labelled with ε_t and q_{init} (formally, $\ell(\varepsilon_u) = (\varepsilon_t, q_{init})$);*
- *any non-root node $n_u = (d_i, q_i)_{0 \leq i < |n_u|}$ of u is labelled with $\ell(n_u) = ((d_i)_{0 \leq i < |n_u|}, q_{|n_u|-1})$;*
- *for any node n_u of the form $(d_i, q_i)_{0 \leq i < |n_u|}$ of u with $\ell(n_u) = (m_t, q)$, letting ν_{n_u} be the marking of $\mathsf{succ}(m_t)$ by Q such that $\nu_{n_u}(m_t \cdot d) = \{q' \in Q \mid n_u \cdot (d, q') \in \mathsf{succ}(n_u)\}$, we have $\nu_{n_u} \models\!\!\equiv \delta(q, l(m_t))$. We name this marking ν_{n_u} the marking of $\mathsf{succ}(m_t)$* induced *by $\mathcal{U}$.*

The tree $\mathcal{T}$ is accepted *by $\mathcal{A}$ if there exists an execution tree $\mathcal{U}$ of $\mathcal{A}$ over $\mathcal{T}$ such that any infinite branch b is* accepting*, i.e., the least priority $\omega_{\min}(b)$ appearing infinitely many times along b is even. Such an execution tree is said to be* accepting*. The* language *of $\mathcal{A}$, denoted by $\mathcal{L}(\mathcal{A})$, is the set of all trees accepted by $\mathcal{A}$.*

Clearly, if a marking ν satisfies an EU-constraint, then so does any marking ν' containing ν. that $\mathcal{D}$ is not constrained by the definition of AEUPTAs, so that AEUPTAs may accept trees of arbitrary (finite) arity. However, the multisets in the "existential part" of EU-pairs can be used to impose a lower bound on the number of successors for the EU-pair to be satisfied, and an upper bound can be imposed by letting the universal part be empty.

▶ **Example 5.** Consider a node m of some input tree $\mathcal{T}$, with three successors $m \cdot d_1$, $m \cdot d_2$ and $m \cdot d_3$; assume that this node m is labelled with some letter σ. Consider an AEUPTA $\mathcal{A}$ visiting node m in state q, giving rise to a node n in an execution tree $\mathcal{U}$ with $\ell(n) = (m, q)$. The successors of n in the execution tree give rise to the marking ν_n such that $\nu_n(m \cdot d_1) = \{q_1, q_3\}$, $\nu_n(m \cdot d_2) = \{q_2, q_4\}$, and $\nu_n(m \cdot d_3) = \{q_1, q_4\}$, as depicted in Fig. 1.

Assume that $\delta(q, \sigma)$ is satisfied by the following set W of EU-pairs:

$$W = \Big\{ \underbrace{\langle q_1 \mapsto 2; \{q_2\}\rangle}_{\langle E_1; U_1\rangle},\ \underbrace{\langle q_1 \mapsto 1, q_2 \mapsto 1, q_3 \mapsto 1; \{q_4\}\rangle}_{\langle E_2; U_2\rangle},\ \underbrace{\langle q_3 \mapsto 1; \{q_4\}\rangle}_{\langle E_3; U_3\rangle} \Big\}.$$

Figure 1 displays a possible set of successors $\mathsf{succ}(n)$ of n in the execution tree $\mathcal{U}$. Using the following three submarkings ν_i of ν_n, we are able to fulfill all three EU-pairs of W: $\nu_1\colon n \cdot d_1 \mapsto q_1, n \cdot d_2 \mapsto q_2, n \cdot d_3 \mapsto q_1$, $\nu_2\colon n \cdot d_1 \mapsto q_3, n \cdot d_2 \mapsto q_2, n \cdot d_3 \mapsto q_1$, and $\nu_3\colon n \cdot d_1 \mapsto q_3, n \cdot d_2 \mapsto q_4, n \cdot d_3 \mapsto q_4$. ⌟

[3] Such automata are usually said to be *non-deterministic* in the litterature. We prefer to name them *non-alternating* since they are the class of automata we get with our alternation-removal procedure, and also because we do not have a notion of being deterministic for our tree automata (an EUPTA has in general more than one execution trees on an input tree).

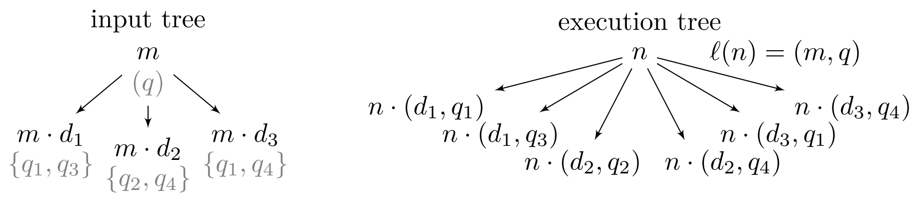

Figure 1 Example of a transition of the automaton when exploring node m in some state q.

▶ **Example 6.** To illustrate the use of AEUPTA, we display an example of an automaton accepting all trees satisfying the following two conditions: exactly two infinite branches contain infinitely many occurrences of a, and at least one branch is fully labelled with b.

Let $\Sigma = \{a, b, c\}$. We let $\mathcal{A} = (Q, q_{init}, \delta, \omega)$, where the set Q of states is $\{q_{init}, q_a^{2i}, q_a^{1i}, r_a^i, q_b^\infty, q_a^f, r_a^f, q_\top\}$: the automaton uses state q_a^{2i} to visit a prefix of a branch having exactly two infinite sub-branches where a occurs infinitely many times; it uses states q_a^{1i} and r_a^i to visit subtrees in which exactly one branch has infinitely many occurrences of a: state r_a^i is used when letter a is read, and the acceptance condition requires that it be visited infinitely many times along some branch; similarly, states q_a^f and r_a^f are used to explore subtrees in which all branches have finitely many occurrences of a. Finally, we use q_b^∞ to explore branches fully labelled with b. Following this intuition, the transition function is defined as:

$$\delta(q_{init}, \sigma) = \begin{cases} \Big(\langle q_a^{2i} \mapsto 1; \{q_a^f\}\rangle \vee \langle q_a^{1i} \mapsto 2; \{q_a^f\}\rangle\Big) \wedge \langle q_b^\infty \mapsto 1; \{q_\top\}\rangle & \text{if } \sigma = b \\ \bot & \text{otherwise} \end{cases}$$

$$\delta(q_a^{2i}, \sigma) = \langle q_a^{2i} \mapsto 1; \{q_a^f\}\rangle \vee \langle q_a^{1i} \mapsto 2; \{q_a^f\}\rangle \quad \text{for any } \sigma \in \Sigma$$

$$\delta(q_a^{1i}, \sigma) = \delta(r_a^i, \sigma) = \begin{cases} \langle r_a^i \mapsto 1; \{q_a^f\}\rangle & \text{if } \sigma = a \\ \langle q_a^{1i} \mapsto 1; \{q_a^f\}\rangle & \text{otherwise} \end{cases}$$

$$\delta(q_a^f, \sigma) = \delta(r_a^f, \sigma) = \begin{cases} \langle \varnothing; \{r_a^f\}\rangle & \text{if } \sigma = a \\ \langle \varnothing; \{q_a^f\}\rangle & \text{otherwise} \end{cases}$$

$$\delta(q_b^\infty, \sigma) = \begin{cases} \langle q_b^\infty \mapsto 1; \{q_\top\}\rangle & \text{if } \sigma = b \\ \bot & \text{otherwise} \end{cases}$$

Finally, $\delta(q_\top, \sigma) = \top$ for any $\sigma \in \Sigma$, so that any subtree is accepted when explored in $q_\top$. Notice that $\delta(q_{init}, \sigma)$ contains a conjunction, so that the automaton is alternating. Notice also how the disjunctive EU-constraint in $\delta(q_{init}, \sigma)$ and $\delta(q_a^{2i}, \sigma)$ will either look for a single successor from which two branches will have infinitely many occurrences of a, or for two successors from each of which there will be such a branch. All other branches will be checked to have finitely many occurrences of a by exploring them in state q_a^f. The priority function is defined so as to check that a occurs infinitely many times along branches where it has to:

$$\omega(r_a^i) = \omega(q_b^\infty) = 0 \qquad \omega(q_a^{1i}) = \omega(q_a^{2i}) = \omega(r_a^f) = 1 \qquad \omega(q_a^f) = 2$$

Figures 2 and 3 display (part of) an input tree and a corresponding execution tree for the automaton built above The root of the execution tree is labelled with (ε, q_{init}) and the marking ν induced by the execution tree for the successors of the root of the input tree is $\{d_1 \mapsto \{q_\top, q_a^{1i}\}, d_2 \mapsto \{q_a^{1i}, q_b^\infty\}, d_3 \mapsto \{q_\top, q_a^f\}\}$, which satisfies $\delta(q_{init}, b)$: indeed, the unitary marking $\{d_1 \mapsto \{q_a^{1i}\}, d_2 \mapsto \{q_a^{1i}\}, d_3 \mapsto \{q_a^f\}\}$ fulfills $\langle q_a^{1i} \mapsto 2; \{q_a^f\}\rangle$, and the unitary marking $\{d_1 \mapsto \{q_\top\}, d_2 \mapsto \{q_b^\infty\}, d_3 \mapsto \{q_\top\}\}$ fulfills $\langle q_b^\infty \mapsto 1; \{q_\top\}\rangle$. ⌟

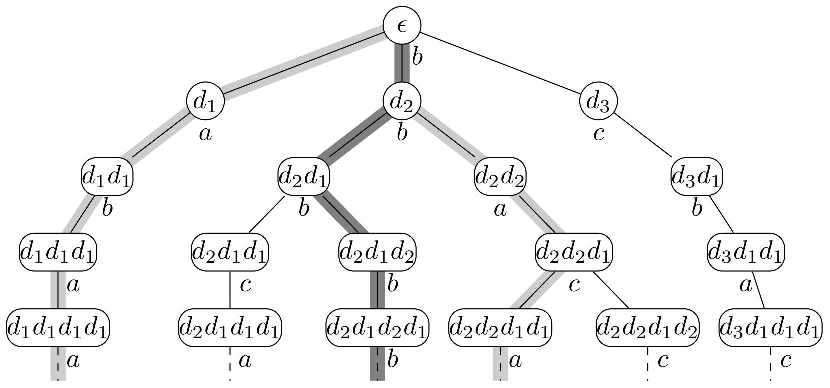

Figure 2 An input tree with exactly two branches (light grey) having infinitely many occurrecnes of a and at least one branch (dark grey) fully labelled with b.

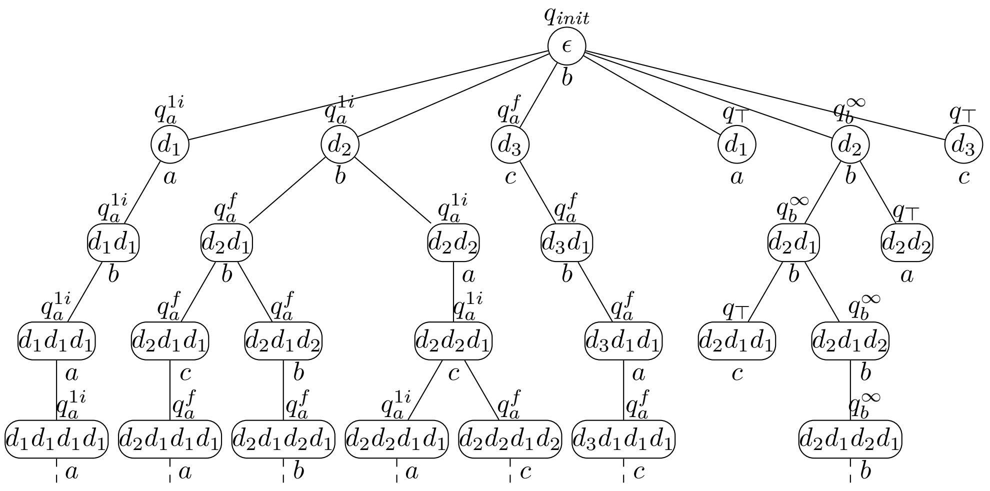

Figure 3 An execution tree for our automaton $\mathcal{A}$ of Example 6 on the input tree of Fig. 2 (notice that, for the sake of readability, we keep the names of the nodes as in the input tree, using words over $\mathcal{D}$ instead of words over $\mathcal{D}\times Q$). Because of the conjunction in $\delta(q_{init}, b)$, the automaton explores the input tree twice, so that the execution tree contains two copies of the input tree: the subtree to the left of the root corresponds to the part looking for two branches with infinitely many occurrences of a, while the subtree to the right looks for a branch fully labelled with b.

Size of an AEUPTA. The *size of an AEUPTA*, denoted with $|\mathcal{A}|$, is a 5-tuple $(|Q|, |\delta|_{\mathsf{Bool}}, |\delta|_{\mathsf{E}}, |\delta|_{\mathsf{U}}, |\omega|)$, where $|\delta|_{\mathsf{Bool}}$ is the maximum size of the boolean formulas in δ, $|\delta|_{\mathsf{E}}$ (resp. $|\delta|_{\mathsf{U}}$) is the size of the largest existential part E (resp. universal part U) in some EU-pair in δ, and $|\omega| = |\{\omega(q) \mid q \in Q\}|$ is the number of priorities used in the automaton. Note that contrary to classical, fixed-arity tree automata, we explicitly consider the size of the transition function δ in the size of an AEUPTA; this is motivated by the fact that EU-constraints may succinctly encode very complex transitions, regardless of the size of Q.

In the sequel, we say that the size of an AEUPTA is at most $(s_Q, s_B, s_E, s_U, s_\omega)$ when $|Q| \leq s_Q$, $|\delta|_{\mathsf{Bool}} \leq s_B$, $|\delta|_{\mathsf{E}} \leq s_E$, $|\delta|_{\mathsf{U}} \leq s_U$ and $|\omega| \leq s_\omega$. The fact that we use five different parameters in the size of AEUPTAs will allow us to have more precise bounds on the complexities of our operations for manipulating them.

▶ Remark 7. It is possible to modify any EU-automaton in such a way that every EU-pair it involves uses only a singleton or the empty set as its universal part (*i.e.*, with $|\delta|_{\mathsf{U}} \leq 1$). This can be performed by replacing any general EU-pair $\langle E; U\rangle$ with $\langle E; \{q_U\}\rangle$, where q_U is a fresh state s.t. $\delta(q_U, \sigma) = \bigvee_{q\in U} \delta(q, \sigma)$, and keeping track of the state being chosen in order to encode the acceptance condition.

This transformation produces an automaton whose size remains polynomial in the size of the original automaton. Moreover, as we will see in the sequel, all the constructions we develop for manipulating AEUPTAs preserve this property of having only singletons or empty sets as the universal part of EU-pairs. ◀

Game-based semantics. Some of our results rely on the tight links between automata and games, which we can extend to our class of AEUPTA. More precisely, given a Σ-labelled $\mathcal{D}$-tree $\mathcal{T}$ and an AEUPTA $\mathcal{A}$, we can build a two-player turn-based parity game $\mathcal{G}_{\mathcal{A},\mathcal{T}}$ enjoying the following property:

▶ **Proposition 8.** *The Σ-labelled $\mathcal{D}$-tree $\mathcal{T}$ is accepted by the AEUPTA $\mathcal{A}$ if, and only if, Player 0 has a winning strategy from the initial state $(\varepsilon_t, q_{init})$ in the parity game $\mathcal{G}_{\mathcal{A},\mathcal{T}}$.*

When $\mathcal{T}$ is regular and corresponds to the execution tree of some finite Kripke structure $\mathcal{K} = (V, E, \ell)$, the construction above can be adapted to give rise to a *finite* game $\mathcal{G}_{\mathcal{A},\mathcal{K}}$ with the same property. The sizes of the state space of $\mathcal{G}_{\mathcal{A},\mathcal{K}}$ and of its transition relation both are in $O(|V| \cdot (|Q| \cdot |\delta|_{\mathsf{Bool}} + |Q|^{|V|}))$. The term $|Q|^{|V|}$ comes from the fact we have to consider all possible unitary markings of the successors of the nodes in $\mathcal{K}$ by Q. It is worth noticing that for AEUPTA involving only simple constraints of the form $\langle q \mapsto 1; \{q_\top\}\rangle$ (where from $q_\top$ any tree is accepted) or $\langle \varnothing; \{q\}\rangle$ those sizes are in $O(|V| \cdot |Q| \cdot (|\delta|_{\mathsf{Bool}} + |V|))$. Such automata actually correspond to $\{\Box, \Diamond\}$-automata.

3 Operations on AEUTAs

This section is the main technical part of our paper: we develop algorithms for performing various operations on AEUPTAs (namely union and intersection, projection, complementation and alternation removal), and carefully study the size of the AEUPTAs we obtain. We will see that there is no surprise with union, intersection and projection: the size of the resulting automata is linear in the sizes of the inputs.

However, projection assumes that the input automaton is non-alternating; hence projection will be used in combination with alternation removal, and this combination involves an exponential blow-up. Thanks to our careful computation of the various parameters of the size of automata (see Table 1), we notice that while complement and alternation removal both involve an exponential blow-up, their combination only yields a single-exponential blow-up. This will allow us to obtain optimal complexity in our applications to QCTL and MSO.

3.1 Union, intersection

Union and intersection are straightforward for AEUPTAs, thanks to alternation.

▶ **Theorem 9.** *Let $\mathcal{A} = (Q, q_{init}, \delta, \omega)$ and $\mathcal{A}' = (Q', q'_{init}, \delta', \omega')$ be two AEUPTAs. There exist AEUPTAs $\mathcal{A}_\cup$ and $\mathcal{A}_\cap$, respectively accepting the union and the intersection of the language of $\mathcal{A}$ and $\mathcal{A}'$, and having size at most $(|Q| + |Q'| + 1, |\delta|_{Bool} + |\delta'|_{Bool} + 1, \max(|\delta|_E, |\delta'|_E), \max(|\delta|_U, |\delta'|_U), \max(|\omega|, |\omega'|) + 1)$. If the minimum priorities of ω and ω' are equal, then the number of priorities in $\mathcal{A}_\cup$ and $\mathcal{A}_\cap$ can be bounded by $\max(|\omega|, |\omega'|)$.*

Table 1 Summary of the size of the automata produced by our algorithms.

	projection	complement	alternation removal
$\lvert Q_{res}\rvert$	$\leq \lvert Q\rvert$	$\in O(\lvert Q\rvert \cdot \lvert\delta\rvert_{\mathsf{Bool}} \cdot \lvert\Sigma\rvert \cdot \lvert\delta\rvert_{\mathsf{E}} \cdot 2^{\lvert\delta\rvert_{\mathsf{E}}})$	$\in 2^{O(\lvert Q\rvert^2 \cdot \log(\lvert Q\rvert))}$
$\lvert\delta_{res}\rvert_{\mathsf{Bool}}$	$\leq \lvert\Sigma'\rvert \cdot \lvert\delta\rvert_{\mathsf{Bool}}$	$\in O(\lvert Q\rvert \cdot \lvert\delta\rvert_{\mathsf{Bool}} \cdot {\lvert\delta\rvert_{\mathsf{E}}}^3 \cdot 4^{\lvert\delta\rvert_{\mathsf{E}}})$	$\in (2\lvert\delta\rvert_{\mathsf{E}} \cdot \lvert\delta\rvert_{\mathsf{U}})^{O(\lvert Q\rvert^2 \cdot {\lvert\delta\rvert_{\mathsf{Bool}}}^2 \cdot \lvert\delta\rvert_{\mathsf{E}})}$
$\lvert\delta_{res}\rvert_{\mathsf{E}}$	$\leq \lvert\delta\rvert_{\mathsf{E}}$	$\leq \lvert\delta\rvert_{\mathsf{E}} + 1$	$\leq \lvert Q\rvert \cdot \lvert\delta\rvert_{\mathsf{Bool}} \cdot \lvert\delta\rvert_{\mathsf{E}}$
$\lvert\delta_{res}\rvert_{\mathsf{U}}$	$\leq \lvert\delta\rvert_{\mathsf{U}}$	$\leq \max(\lvert\delta\rvert_{\mathsf{U}}, 1)$	$\leq {\lvert\delta\rvert_{\mathsf{U}}}^{\lvert Q\rvert \cdot \lvert\delta\rvert_{\mathsf{Bool}}}$
$\lvert\omega_{res}\rvert$	$\leq \lvert\omega\rvert$	$\leq \lvert\omega\rvert + 1$	$\leq 2(\lvert Q\rvert \cdot \lvert\omega\rvert + 1)$

3.2 Projection

Given an AEUPTA $\mathcal{A}$ over alphabet $\Sigma_1 \times \Sigma_2$, *projection* consists in building another AEUPTA $\mathcal{A}_1$, over alphabet Σ_1, accepting all Σ_1-labelled trees whose labelling can be extended on $\Sigma_1 \times \Sigma_2$ to make the tree accepted by $\mathcal{A}$. This is a classical construction, and it can be performed easily on *non-alternating* automata [22].

Formally, two $\Sigma_1 \times \Sigma_2$-labelled trees $\mathcal{T} = (t, l)$ and $\mathcal{T}' = (t', l')$ are said Σ_1-*equivalent*, denoted $\mathcal{T} \equiv_{\Sigma_1} \mathcal{T}'$, whenever $t = t'$ and for any node n of these trees, it holds $\mathsf{proj}_1(l(n)) = \mathsf{proj}_1(l'(n))$. Σ_2-equivalence is defined analogously.

▶ **Theorem 10.** *Let $\mathcal{A} = (Q, q_{init}, \delta, \omega)$ be an EUPTA over $\Sigma = \Sigma_1 \times \Sigma_2$. For each $i \in \{1, 2\}$, we can build an EUPTA $\mathcal{A}_i$ over Σ such that, for any Σ-labelled tree $\mathcal{T}$, it holds: $\mathcal{T} \in \mathcal{L}(\mathcal{A}_i)$ if, and only if, there is a Σ-labelled tree $\mathcal{T}'$ in $\mathcal{L}(\mathcal{A})$ such that $\mathcal{T} \equiv_{\Sigma_i} \mathcal{T}'$. The size of $\mathcal{A}_i$ is at most $(\lvert Q\rvert, \lvert\Sigma_{3-i}\rvert \cdot \lvert\delta\rvert_{Bool}, \lvert\delta\rvert_E, \lvert\delta\rvert_U, \lvert\omega\rvert)$.*

3.3 Complementation

Complementation is usually easy for alternating parity automata: it suffices to dualise the transition function (swapping disjunctions and conjunctions) and shifting the priorities. For our AEUTA however, we have to express the negation of any EU-pair $\langle E; U\rangle$ as an EU-constraint.

The question then is to characterise those nodes that *fail to satisfy* an EU-pair $\langle E; U\rangle$. There can be two reasons for this:

1. either we cannot find $\lvert E\rvert$ successors of the current node n to associate with the $\lvert E\rvert$ states of the existential part,
2. or for every way to satisfy E with nodes in $\mathsf{succ}(n)$, there remain successors that are accepted by no states in U. Equivalently, there is no way to satisfy the existential part E with (at least) all nodes that are accepted by no states in the universal part U. This includes as a special case the situations where we have more than $\lvert E\rvert$ successors that are accepted by no states in U.

Failing to satisfy the existential part of $\langle E; U\rangle$. We first address the case $\langle E; \{q_\top\}\rangle$. When such an EU-pair cannot be satisfied from a tree node x, there exists some pair (F, g) (called a blocking pair hereafter) where $F \sqsubset E$ is a submultiset and $g \in E \setminus F$, such that the constraint $\langle F; \{q_\top\}\rangle$ is satisfied from x and the constraint $\langle F \uplus \{g\}; \{q_\top\}\rangle$ is not satisfied from x. More precisely, one can prove (see Prop. 11, 13 and 14 in [17]) that there exists a blocking pair (F, g) s.t. for any node $y \in \mathsf{succ}(x)$ that is not involved in the satisfaction of F, the subtree rooted at y does not belong to $\mathcal{L}(\mathcal{A}_{g'})$ for any $g' \in \mathsf{supp}(F) \cup \{g\}$, where $\mathcal{A}_{g'}$ denotes the automaton obtained from $\mathcal{A}$ by taking g' as the initial state. Applying this result, the negation of $\langle E; \{q_\top\}\rangle$ can then be expressed as follows:

$$\Phi_E = \bigvee_{F \sqsubset E} \bigvee_{g \in E \setminus F} \langle F; \{(\overline{\mathsf{supp}(F)} \cup \{\overline{g}\}, \wedge)\}\rangle \qquad (1)$$

where $(\overline{\mathsf{supp}(F)} \cup \{\overline{g}\}, \wedge)$ is a (single) new state of our automaton from which a subtree will be accepted if, and only if, it does not belong to any $\mathcal{L}(\mathcal{A}_{g'})$ with $g' \in \mathsf{supp}(F) \cup \{g\}$. Notice that for the special case where E is empty, we end up with an empty disjunction, which is equivalent to false (indeed $\langle \varnothing; \{q_\top\}\rangle$ is satisfied by any tree).

Failing to satisfy $\langle E; U\rangle$. We can now address the general case. If an EU-pair $\langle E; U\rangle$ is not satisfied by a node x, then either it has at least $|E| + 1$ successors y that are recognized by no automata states $u \in U$, or for some $0 \leq k \leq |E|$, it has at least k successors recognized by no states $u \in U$ and it does not have $|E|$ successors witnessing the satisfaction of $\langle E;$ $\{q_\top\}\rangle$ and of which k successors are accepted by no states $u \in U$ (see Prop. 15 in [17]). This explains the definition of $\overline{\langle E; U\rangle}$ below.

Construction of the complement automaton. We now define the complement automaton $\mathcal{A}^c$ of $\mathcal{A} = (Q, q_{init}, \delta, \omega)$; we let $\mathcal{A}^c = (Q^c, q^c_{init}, \delta^c, \omega^c)$ be such that:

- Q^c is a subset of $D \cup (2^D \times \{\wedge\}) \cup (2^{(2^D \times \{\vee\})} \times \{\wedge\})$, where $D = Q \cup \overline{Q}$, and $\overline{Q} = \{\overline{q} \mid q \in Q\}$ is a set of fresh states. This defines an operator $\overline{\cdot}\colon q \mapsto \overline{q}$ over Q, which we extend to states $\overline{q}$ of $\overline{Q}$ by letting $\overline{\overline{q}} = q$, to subsets X of D by letting $\overline{X} = \{\overline{x} \mid x \in X\}$, to states $(P, \wedge)$ of $2^D \times \{\wedge\}$ by letting $\overline{(P, \wedge)} = (\overline{P}, \vee)$. We finally extend it to $\mathsf{PBF}(Q^c)$ by letting $\overline{\top} = \bot$, $\overline{\bot} = \top$, $\overline{\psi \wedge \phi} = \overline{\psi} \vee \overline{\phi}$ and $\overline{\psi \vee \phi} = \overline{\psi} \wedge \overline{\phi}$.
  Intuitively, from a state $q \in Q$, automaton $\mathcal{A}^c$ will accept the same language as from the same state in $\mathcal{A}$, while from a state $\overline{q} \in \overline{Q}$, it will accept its complement. The language accepted by $\mathcal{A}$ from a state $(P, \wedge) \in 2^D \times \{\wedge\}$ will be the intersection of all languages accepted by $\mathcal{A}$ from all states in P, whereas the language accepted from states of the form $(\{(P_i, \vee) \mid 1 \leq i \leq k\}, \wedge) \in 2^{(2^D \times \{\vee\})} \times \{\wedge\}$ will be the intersection (over i) of the unions of the languages accepted from all states in P_i. Notice that the indications of $\wedge$ and $\vee$ are mainly used for the sake of clarity.
- accordingly, $q^c_{init} = \overline{q_{init}}$;
- δ^c is defined as follows, for $q \in Q$, $P \in 2^D \cup 2^{2^D}$, and $\sigma \in \Sigma$: we let

$$\delta^c(q, \sigma) = \delta(q, \sigma) \qquad \delta^c((P, \wedge), \sigma) = \bigwedge_{r \in P} \delta^c(r, \sigma)$$

$$\delta^c(\overline{q}, \sigma) = \overline{\delta(q, \sigma)} \qquad \delta^c((P, \vee), \sigma) = \bigvee_{r \in P} \delta^c(r, \sigma)$$

  and, following the developments above:

$$\overline{\langle E; U\rangle} = \langle (\overline{U}, \wedge) \mapsto |E| + 1; \{q_\top\}\rangle \vee \bigvee_{0 \leq k \leq |E|} \Big(\langle (\overline{U}, \wedge) \mapsto k; \{q_\top\}\rangle \wedge \bigwedge_{\substack{m \sqsubseteq E \\ |m| = k}} \Phi_{(E \setminus m) \uplus m^{\overline{U}}} \Big)$$

  where $m^{\overline{U}}$ is the multiset defined by $\{(\{x\} \cup \overline{U}, \wedge) \mapsto m(x)\}_{x \in \mathsf{supp}(m)}$, and Φ_G (defined as above in equation 1) characterises the failure to satisfy $\langle G; \{q_\top\}\rangle$. Notice also that Φ_G contains EU-pair $\langle F; \{(\overline{\mathsf{supp}(F)} \cup \{\overline{g}\}, \wedge)\}\rangle$ where F may contain states of the form $(P, \wedge)$ with $P \subseteq D$, and then $\overline{F}$ may contain states of the form $\overline{(P, \wedge)}$, which we rewrite as $(\overline{P}, \vee)$. This way, δ^c is defined for any state of Q^c and only involves states of Q^c.
  Finally, observe that, as claimed in Remark 7, this construction only introduces new EU-pairs of the form $\langle E; \{u\}\rangle$.
- priorities are defined as follows: $\omega^c(q) = \omega(q)$, $\omega^c(\overline{q}) = \omega(q) + 1$, and $\omega^c(P, op) = \max\{\omega(q) + 1 \mid q \in Q\}$ for all $(P, op) \in Q^c$.

Size of $\mathcal{A}^c$. We now evaluate the size of the complement automaton:

- *(over)approximating the number of states:* all states of $\mathcal{A}^c$ are in $D \cup (2^D \times \{\wedge\}) \cup (2^{(2^D\times\{\vee\})} \times \{\wedge\})$, but not all states of this set are reachable. The reachable states are either in D, or they appear in $\overline{\langle E;U\rangle}$ for some EU-pair $\langle E;U\rangle$ in δ.
  Take an EU-pair $\langle E;U\rangle$ in δ. Besides $(\overline{U},\wedge)$ and $q_\top$, $\overline{\langle E;U\rangle}$ contains, for each submultiset m of E, a formula of the form Φ_G, where $G = (E \setminus m) \uplus m^{\overline{U}}$. Formula Φ_G involves states of G, which are either in E (hence already in Q) or of the form $(\{x\}\cup\overline{U},\wedge)$ for $x \in E$, and states of the form $(\overline{\mathsf{supp}(F)} \cup \{\overline{g}\},\wedge)$ where F is a submultiset of G and $g \in G\setminus F$. In the end, using $|Q|\cdot|\Sigma|\cdot|\delta|_{\mathsf{Bool}}$ as an upper bound on the number of EU-pairs, we have:
  $$|Q^c| \le \underbrace{2\,|Q|}_{Q\cup\overline{Q}} + \underbrace{1}_{q_\top} + \underbrace{|Q|\cdot|\delta|_{\mathsf{Bool}}\cdot|\Sigma|}_{\text{number of }\mathsf{EU}\text{-pairs}} \cdot (\underbrace{1}_{\substack{\text{state}\\(\overline{U},\wedge)}} + \underbrace{|\delta|_{\mathsf{E}}}_{\substack{\text{states}\\(\{x\}\cup\overline{U},\wedge)}} + \underbrace{2^{|\delta|_{\mathsf{E}}}\cdot|\delta|_{\mathsf{E}}}_{\substack{\text{states}\\(\overline{\mathsf{supp}(F)}\cup\{\overline{g}\},\wedge)}})$$
  Hence $|Q^c| \le 1 + |Q|\cdot(2 + |\Sigma|\cdot|\delta|_{\mathsf{Bool}}\cdot|\delta|_{\mathsf{E}}\cdot(1+2^{|\delta|_{\mathsf{E}}}))$.
- *bounding the size of transition function:* It is easy to see that $|\delta^c_F|_{\mathsf{E}} \le \max(|\delta|_{\mathsf{E}}, |\delta^c_{\overline{Q}}|_{\mathsf{E}})$ and $|\delta^c_F|_{\mathsf{U}} \le \max(|\delta|_{\mathsf{U}}, |\delta^c_{\overline{Q}}|_{\mathsf{U}})$. The boolean size is more complex to overapproximate. Intuitively, in the definition of $\overline{\langle E;U\rangle}$, we can bound the number of the subformulas $\Phi_{(E\setminus m)\uplus m^{\overline{U}}}$ by $2^{|\delta|_{\mathsf{E}}}$ and each of them gives rise to a disjunction with at most $|\delta|_{\mathsf{E}}\cdot 2^{|\delta|_{\mathsf{E}}}$ EU-pairs. From this we can see that the boolean size of δ^c from states in $\overline{Q}$ is at most $|\delta|_{\mathsf{Bool}}\cdot(1+|\delta|_{\mathsf{E}}\cdot(1+|\delta|_{\mathsf{E}}\cdot 2^{2|\delta|_{\mathsf{E}}}))$. Those formulas also appear in the transitions from states $(\overline{U},\wedge)$ and $(\overline{\mathsf{supp}(F)}\cup\{\overline{g}\},\wedge)$ and states in multisets F (in the definition of Φ_G). Finally we end up with $|\delta^c|_{\mathsf{Bool}} \le 4\cdot(1+|Q|)\cdot|\delta|_{\mathsf{Bool}}\cdot|\delta|_{\mathsf{E}}^3\cdot 2^{2|\delta|_{\mathsf{E}}}$.

To summarise our results:

▶ **Theorem 11.** *Given an AEUPTA $\mathcal{A} = (Q, q_{init}, \delta, \omega)$, we can build an AEUPTA $\mathcal{A}^c$ recognising the complement of $\mathcal{L}(\mathcal{A})$, with size at most $(1 + |Q|\cdot(2+|\Sigma|\cdot|\delta|_{Bool}\cdot(1+|\delta|_E + 2^{|\delta|_E}\cdot|\delta|_E)), 4\cdot(1+|Q|)\cdot|\delta|_{Bool}\cdot|\delta|_E^3\cdot 4^{|\delta|_E}, |\delta|_E+1, \max(|\delta|_U,1), |\omega|+1)$.*

3.4 Alternation removal (a.k.a. simulation)

Building a non-alternating automaton equivalent to a given alternating automaton is an important construction, e.g. in order to perform projection, or for algorithmic purposes. In this section, we present an *alternation-removal* (a.k.a. *simulation*) algorithm, based on ideas developed in [30, 33] for MSO-automata. For the rest of this section, we fix an AEUPTA $\mathcal{A} = (Q, q_{init}, \delta, \omega)$. The procedure can be decomposed into four steps:

1. pair each state of the automaton with its immediate ancestor in order to keep enough information in the infinite branches to decide whether they are accepting or not. This provides us with an AEUPTA $\mathcal{P}$ using EU-constraints over $Q\times Q$;
2. build an (alternating) powerset automaton involving a new satisfaction relation $\Vert\!\models$ and a new acceptance condition (which is *not* a parity condition). It gives an AEUTA $\mathcal{Q}$ using EU-constraints over $2^{Q\times Q}$ (and based on $\Vert\!\models$);
3. remove the conjunctions in the transition function of $\mathcal{Q}$ to get a non-alternating automaton. It gives an EUTA $\mathcal{R}$ using EU-constraints over $2^{Q\times Q}$ and relation $\Vert\!\models$;
4. adapt the acceptance condition to obtain an EUPTA $\mathcal{N}$, using EU-constraints over $Q_\omega\times 2^{Q\times Q}$ (where Q_ω is a new set of states introduced to encode the acceptance condition), and relation $|\!\models$.

We briefly describe each step below, and refer to [17, Section 4.4] for more details.

Keeping track of ancestor states. For any state $q' \in Q$, we define the mapping $\phi_{q'} : Q \to Q^2$ as $\phi_{q'}(q) = (q', q)$, and extend it to (multi-)sets of states, EU-pairs and EU-constraints in the natural way. We then define the AEUPTA $\mathcal{P} = (Q^2, (q_{init}, q_{init}), \gamma, \omega')$ with $\gamma((q, q'), \sigma) = \phi_{q'}(\delta(q', \sigma))$, and $\omega'(q, q') = \omega(q')$. Intuitively, state (q, q') in $\mathcal{P}$ corresponds to state q' in $\mathcal{A}$, with the extra information that this state originates from state q. Notice that both γ and ω' only depend only on the second state of the pair (q, q'). We have:

▶ **Proposition 12.** *The languages of $\mathcal{A}$ and $\mathcal{P}$ are equal. The size of $\mathcal{P}$ is $(|Q|^2, |\delta|_{Bool}, |\delta|_E, |\delta|_U, |\omega|)$.*

▶ **Example 13.** Let $\Sigma = \{a, b\}$. Consider an AEUPTA $\mathcal{A}$ with an initial state q_{init} with $\omega(q_{init}) = 1$, and a state q_1 with $\omega(q_1) = 0$, and $\delta(q_{init}, b) = \langle q_1 \mapsto 1; \varnothing \rangle$ and

$$\delta(q_{init}, a) = \langle q_{init} \mapsto 1; \varnothing \rangle \wedge \langle q_1 \mapsto 1; \varnothing \rangle \quad \delta(q_1, a) = \delta(q_1, b) = \langle q_{init} \mapsto 1; \varnothing \rangle.$$

Notice that this automaton only accepts trees with a single branch (i.e., words). It is easily seen that the word $a^3 \cdot b^\omega$ is accepted, while a^ω is not. If we perform a simple powerset construction, the sequence of sets of states along the (unique) computation for both words is $\{q_{init}\} \cdot \{q_{init}, q_1\}^\omega$. This does not keep enough information to decide if a run is accepting. Now, if each state is paired with its ancestor (arbitrarily pairing the initial state with itself), then the sequence of sets of pairs of states visited along $a \cdot b^\omega$ is $\{(q_{init}, q_{init})\} \cdot \{(q_{init}, q_{init}), (q_{init}, q_1)\} \cdot \{(q_1, q_{init}), (q_{init}, q_1)\}^\omega$, while along a^ω it is $\{(q_{init}, q_{init})\} \cdot \{(q_{init}, q_{init}), (q_{init}, q_1)\} \cdot \{(q_{init}, q_{init}), (q_{init}, q_1), (q_1, q_1)\}^\omega$. In the latter sequence, we can detect the presence of an infinite branch looping in q_{init}; we will introduce in the next section a new acceptance condition to capture this fact. ⌟

Building the powerset automaton. In this step, we build an (still alternating) automaton whose states are sets of states of $\mathcal{P}$ (*i.e.* elements of $K = 2^{Q^2}$) with the following changes:

- The satisfaction relation for EU-pairs is modified: instead of considering markings by K, we will consider markings by Q^2 (associating sets of Q-pairs with every successor of a node) from which we will extract submarkings viewed as unitary markings by K in order to satisfy some EU-pair. This is a natural change but it induces a new satisfaction relation denoted $\Vvdash$. The notion of execution tree is then modified accordingly, in particular the execution trees are then labeled by sets of pairs in Q^2.
- The acceptance condition Ω_ω is not a parity condition any more but is defined explicitly.

Formally we define the powerset AEUTA $\mathcal{Q} = (K, \{(q_{init}, q_{init})\}, \beta, \Omega_\omega)$ by letting:

- $K = 2^{Q^2}$ contains all the sets of states of $\mathcal{P}$, hence all the sets of pairs of states of $\mathcal{A}$;
- $\beta\big(\{(q_i, q'_i) \mid 1 \le i \le k\}, \sigma\big) = \bigwedge_{1 \le i \le k} \gamma^s((q_i, q'_i), \sigma)$, where $\gamma^s((q_i, q'_i), \sigma)$ is obtained from $\gamma((q_i, q'_i), \sigma)$ by replacing each pair of states (q, q') by the singleton $\{(q, q')\}$. For the time being, this powerset automaton still is alternating.
- the condition Ω_ω works as follows: given a an infinite sequence $\alpha = (\alpha_i)_{i \in \mathbb{N}}$ of subsets of pairs in Q^2 s.t. $\alpha_i = \{(q_{i,j}, q'_{i,j}) \mid 0 \le j \le z_i\}$ for each $i \in \mathbb{N}$, a sequence $(r_i)_{0 \le i < k}$ of states in Q is said to *appear* in α if for each $i \in \mathbb{N}$, there exists an index $0 \le j \le z_i$ such that $(r_{i-1}, r_i) = (q_{i,j}, q'_{i,j})$. The sequence α is *accepting* if all the sequences $(r_i)_{0 \le i < k}$ that appear in α satisfy the parity condition ω of $\mathcal{A}$ (such a sequence α corresponds to the labelling of an infinite branch in the execution trees associated with relation $\Vvdash$).

▶ **Proposition 14.** *A Σ-labelled $\mathcal{D}$-tree $\mathcal{T}$ is accepted by $\mathcal{P}$ if, and only if, it is accepted by the $\mathcal{A}$-powerset AEUTA $\mathcal{Q}$. The size of $\mathcal{Q}$ is $(2^{|Q|^2}, |Q|^2 \cdot |\delta|_{Bool}, |\delta|_E, |\delta|_U, -)$.*

Removing conjunctions. We now remove conjunctions from the transition function β of $\mathcal{Q}$. As a first step, we turn each formula $\beta(P,\sigma)$ in disjunctive normal form. We can bound the number of different EU-pairs appearing in any given $\beta(P,\sigma)$ by $|Q|\cdot|\delta|_{\mathsf{Bool}}$: indeed, while it is built as a conjunction of up to Q^2 transition formulas, any two pairs (q',q) and (q'',q) give rise to the same EU-pairs. It follows that $\beta(P,\sigma)$ can be written as the disjunction of at most $2^{|Q|\cdot|\delta|_{\mathsf{Bool}}}$ conjunctions of at most $|Q|\cdot|\delta|_{\mathsf{Bool}}$ EU-pairs.

We now turn those conjunctions into disjunctions. We proceed inductively, by replacing any conjunction $\langle E_1;U_1\rangle\wedge\langle E_2;U_2\rangle$ of two EU-pairs over $2^{Q\times Q}$ with an "equivalent" disjunction of EU-pairs over $2^{Q\times Q}$.

Write m_1, n_1, m_2 and n_2 for the sizes of E_1, U_1, E_2 and U_2, respectively. The disjunction we build ranges over the possible ways the "existential" and "universal" parts of the EU-pairs overlap (see Fig. 4). For each combination, we write an EU-pair whose existential part contains the "existential" overlaps and the two "mixed" overlaps, and whose universal part handles the "universal" overlap.

The disjunction of EU-pairs can then be written as follows:

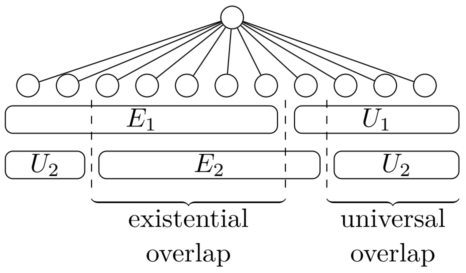

Figure 4 Representation of the overlaps in an execution tree when satisfying a conjunction of two EU-constraints $\langle E_1;U_1\rangle$ and $\langle E_2;U_2\rangle$.

$$C(\langle E_1;U_1\rangle,\langle E_2;U_2\rangle)=\bigvee_{\substack{J_1\sqsubseteq E_1,J_2\sqsubseteq E_2\\|J_1|=|J_2|}}\ \bigvee_{\substack{\tau\text{ permutation}\\\text{of }[1;|J_1|]}}\ \bigvee_{\substack{g_1\colon E_1\setminus J_1\to U_2\\g_2\colon E_2\setminus J_2\to U_1}}\left(E'=\biguplus\begin{array}{l}\{\!\!\{j^1_k\cup j^2_{\tau(k)}\mid 1\le k\le|J_1|\}\!\!\}\\\{\!\!\{e^1_k\cup g_1(e^1_k)\mid 1\le k\le n_1-|J_1|\}\!\!\}\\\{\!\!\{g_2(e^2_k)\cup e^2_k\mid 1\le k\le n_2-|J_2|\}\!\!\}\end{array};U'=U_1\otimes U_2\right)$$

where we use the notations

$$J_1=\{\!\!\{j^1_k\mid 1\le k\le o\}\!\!\}\qquad J_2=\{\!\!\{j^2_k\mid 1\le k\le o\}\!\!\}$$
$$E_1\setminus J_1=\{\!\!\{e^1_k\mid 1\le k\le n_1-o\}\!\!\}\qquad E_2\setminus J_2=\{\!\!\{e^2_k\mid 1\le k\le n_2-o\}\!\!\}$$
$$U_1\otimes U_2=\{u_1\cup u_2\mid u_1\in U_1,u_2\in U_2\}.$$

Note that the sizes of existential and universal parts of any EU-pair in $C(\langle E_1;U_1\rangle,\langle E_2;U_2\rangle)$ are bounded by n_1+n_2 and $m_1\cdot m_2$, respectively. Note that when U_1 is empty, the only possible overlaps are between E_1 and $\langle E_2;U_2\rangle$ (and we have $J_2=E_2$ and $U_1\otimes U_2$ is also empty). The correctness of the construction is stated as follows (for a proof, see [17]):

▶ **Lemma 15.** *Let $\mathcal{S}$ and $\mathcal{S}'$ be two finite sets, and $\langle E_1;U_1\rangle\wedge\langle E_2;U_2\rangle$ be a conjunction of two EU-pairs on $2^{\mathcal{S}}$. For any unitary marking ν of $\mathcal{S}'$ by $2^{\mathcal{S}}$, it holds $\nu\Vvdash\langle E_1;U_1\rangle\wedge\langle E_2;U_2\rangle$ if, and only if, $\nu\Vvdash C(\langle E_1;U_1\rangle,\langle E_2;U_2\rangle)$.*

Let $\mathcal{R} = (2^{Q\times Q}, \{(q_{init}, q_{init})\}, \zeta, \Omega_\omega)$ be an $\mathcal{A}$-powerset AEUTA obtained from $\mathcal{Q}$ by replacing all conjunctions $\langle E_1; U_1\rangle \wedge \langle E_2; U_2\rangle$ (in no specific order). We can prove its equivalence with $\mathcal{Q}$ and since $\mathcal{R}$ is non-alternating, it only has to visit each node of the input tree in one of the states given by the transition function, so that both notions of execution trees (with $\models$ and $\Vert\!\models$) coincide.

We now evaluate the size of $\mathcal{R} = (2^{Q\times Q}, \{(q_{init}, q_{init})\}, \zeta, \Omega_\omega)$. In order to evaluate the size of the transition function ζ of $\mathcal{R}$, we first focus on the size of $C(\langle E_1; U_1\rangle, \langle E_2; U_2\rangle)$. For this, we define the size of an EU-pair $\langle E; U\rangle$ as the pair $(|E|, |U|)$. In the following, a set of EU-pairs is said to have a size at most (n, m) if all its EU-pairs have existential parts of size at most n and universal parts of size at most m.

Consider a conjunction $\langle E_1; U_1\rangle \wedge \langle E_2; U_2\rangle$ of two EU-pairs of size (n_1, m_1) and (n_2, m_2), respectively, and assume w.l.o.g. that $n_1 \le n_2$. Then the formula $C(\langle E_1; U_1\rangle, \langle E_2; U_2\rangle)$ is a disjunction of N EU-pairs of size at most $(n_1 + n_2, m_1 \cdot m_2)$, with:

$$N \quad\le\quad \sum_{l=0}^{n_1} \binom{n_1}{l} \cdot \binom{n_2}{l} \cdot l! \cdot m_2^{n_1-l} \cdot m_1^{n2-l} \quad\le\quad (n_2+1)^{n_1} \cdot m_2^{n_1} \cdot m_1^{n2}.$$

Then we can prove that any conjunction of k EU-pairs of sizes at most (n, m) can be turned into disjunctions of at most $((k-1)! \cdot (n+1)^{k-1} \cdot m^{k^2})^n$ EU-pairs, each of size at most $(k \cdot n, m^k)$ (by induction over k).

We now evaluate the size of the transition function ζ: as explained at the beginning of the present section, ζ is obtained from the transition function β of $\mathcal{Q}$ by first putting each formula $\beta(P, \sigma)$ into disjunctive normal form, as the disjunction of at most $2^{|Q|\cdot|\delta|_{\mathsf{Bool}}}$ conjunctions of at most $|Q| \cdot |\delta|_{\mathsf{Bool}}$ EU-pairs, with EU-pairs of size at most $(|\delta|_{\mathsf{E}}, |\delta|_{\mathsf{U}})$.

Applying our formula above, we get a disjunctive expression for $\beta(P, \sigma)$ involving at most

$$2^{|Q|\cdot|\delta|_{\mathsf{Bool}}} \cdot \Big((|Q| \cdot |\delta|_{\mathsf{Bool}} - 1)! \cdot (|\delta|_{\mathsf{E}} + 1)^{|Q|\cdot|\delta|_{\mathsf{Bool}}-1} \cdot (|\delta|_{\mathsf{U}})^{(|Q|\cdot|\delta|_{\mathsf{Bool}})^2} \Big)^{|\delta|_{\mathsf{E}}}$$

EU-pairs of size at most $(|Q| \cdot |\delta|_{\mathsf{Bool}} \cdot |\delta|_{\mathsf{E}}, (|\delta|_{\mathsf{U}})^{|Q|\cdot|\delta|_{\mathsf{Bool}}})$. In the end:

▶ **Proposition 16.** *The languages of the original AEUPTA $\mathcal{A}$ and of the resulting EUTA $\mathcal{R}$ are the same. The size of $\mathcal{R}$ is at most $(2^{|Q|^2}, 2^{\mathfrak{K}} \cdot ((\mathfrak{K}-1)! \cdot (|\delta|_E + 1)^{\mathfrak{K}-1} \cdot (|\delta|_U)^{\mathfrak{K}^2})^{|\delta|_E}, \mathfrak{K} \cdot |\delta|_E, (|\delta|_U)^{\mathfrak{K}}, -)$ where $\mathfrak{K} = |Q| \cdot |\delta|_{Bool}$ (we omit the size of the acceptance condition of $\mathcal{R}$ as it is not a parity condition).*

Adapting the acceptance condition. It remains to turn the acceptance condition of $\mathcal{R}$ into a parity condition. The transformation is the same as in [33]: we first build a non-deterministic parity word automaton $\mathcal{W}$ accepting all words on the alphabet $2^{Q\times Q}$ that contain an infinite sequence of states $(r_i)_{i\in\mathbb{N}}$ of $\mathcal{A}$ (in the sense of the definition of $\mathcal{Q}$) *not satisfying* the parity condition of $\mathcal{A}$; we then turn it into a deterministic parity word automaton, take its complement, and run it in parallel with $\mathcal{R}$. The detailed construction is given in [17, Section 4.4.4]. Summarising our results:

▶ **Theorem 17.** *Given an AEUPTA $\mathcal{A} = (Q, q_{init}, \delta, \omega)$, we can build an EUPTA $\mathcal{N}$ recognising the same language. The size of $\mathcal{N}$ can be bounded by $(2^{1+|Q|^2+2(|Q|\cdot|\omega|+1)\cdot\log(|Q|\cdot|\omega|+1)}, 2^{\mathfrak{K}} \cdot ((\mathfrak{K}-1)! \cdot (|\delta|_E + 1)^{\mathfrak{K}-1} \cdot (|\delta|_U)^{\mathfrak{K}^2})^{|\delta|_E}, \mathfrak{K} \cdot |\delta|_E, |\delta|_U{}^{\mathfrak{K}}, 2(|Q| \cdot |\omega| + 1))$, where $\mathfrak{K} = |Q| \cdot |\delta|_{Bool}$.*

4 Algorithms for AEUTAs

Given some AEUTA $\mathcal{A}$, we are interested in two decision procedures: deciding whether a regular tree belongs to $\mathcal{L}(\mathcal{A})$ and deciding whether $\mathcal{L}(\mathcal{A}) = \varnothing$. Both consist in building a parity game and deciding whether Player 0 has a winning strategy and this can be done [4] in time $O(n^d)$ where n is the number of states of the game, and d is the number of priorities [20].

Let $\mathcal{K} = (V, E, \ell)$ be a Kripke structure. Deciding whether $\mathcal{T}_\mathcal{K} \in \mathcal{L}(\mathcal{A})$ is equivalent to deciding whether Player 0 has a winning strategy in the parity game $\mathcal{G}_{\mathcal{A},\mathcal{K}}$ defined after Prop. 8, where the number of states is in $O(|V| \cdot (|Q| \cdot |\delta|_{\mathsf{Bool}} + |Q|^{|V|}))$ and the number of priorities is the same as $\mathcal{A}$. Then we have:

▶ **Theorem 18.** *Given a finite Kripke structure $\mathcal{K} = (V, E, \ell)$ and an AEUPTA $\mathcal{A} = (Q, q_0, \delta, \omega)$, whether $\mathcal{T}_K$ belongs to $\mathcal{L}(\mathcal{A})$ can be decided in time $O((|V| \cdot (|Q| \cdot |\delta|_{\mathsf{Bool}} + |Q|^{|V|}))^{|\omega|})$.*

Emptiness checking is performed as for classical tree automata: we consider a non-alternating automaton $\mathcal{A}$, and transform it into a parity game $\mathcal{G}_\mathcal{A}$ such that Player 0 has winning strategy in $\mathcal{G}_\mathcal{A}$ if, and only if, $\mathcal{L}(\mathcal{A}) \neq \varnothing$. It follows:

▶ **Theorem 19.** *Let $\mathcal{A} = (Q, q_0, \delta, \omega)$ be an EUPTA. Checking emptiness of $\mathcal{L}(\mathcal{A})$ can be performed in time $O((|Q| \cdot (1 + |\delta|_{\mathsf{Bool}} \cdot |\Sigma|))^{|\omega|})$.*

Combining the previous result with the simulation theorem, we get:

▶ **Corollary 20.** *Let $\mathcal{A} = (Q, q_0, \delta, \omega)$ be an AEUPTA. Checking emptiness of $\mathcal{L}(\mathcal{A})$ can be performed in time $2^{O(|Q|^3 \cdot |\omega| \cdot (\log |Q| + |\delta|_{\mathsf{Bool}}{}^2 \cdot |\delta|_E \cdot \log |\delta|_E) + |Q| \cdot |\omega| \cdot \log |\Sigma|)}$.*

5 Application to QCTL and MSO

QCTL extends the temporal logic CTL with quantifications over atomic propositions: subformulas of the form $\exists p.\ \psi$, where p is an atomic proposition, are added to the syntax of CTL (in addition to the usual **EX**, **AX**, **EU** and **AU** modalities, and abbreviations like **EF**, **EG**, **AF** and **AG**). QCTL formulas are interpreted over nodes of 2^{AP}-labelled infinite trees. Informally a node n satisfies $\exists p.\ \psi$ if, and only if, there exists a labelling of the subtree rooted at n for which ψ holds true. This is the so-called *tree semantics* [5]. Given a set $P = \{p_1, \ldots, p_m\} \subseteq \mathsf{AP}$, we write $\exists P.\ \phi$ for $\exists p_1 \ldots \exists p_n.\ \phi$. Propositional quantification can also be added on top of CTL*, yielding the logic QCTL*. We also introduce the sublogics Q$^\mathsf{k}$CTL with $k \geq 0$, such that Q^{0}CTL is CTL and Q$^{\mathsf{k+1}}$CTL contains formulas that can be written as $\psi[\exists P_1.\ \phi_1, \ldots, \exists P_n.\ \phi_n]$ where ψ is a CTL formula with placeholders, $P_i \subseteq \mathsf{AP}$, and ϕ_i belongs to Q$^\mathsf{k}$CTL for any $1 \leq i \leq n$. The model-checking and satisfiability problems for Q$^\mathsf{k}$CTL are known to be k-EXPTIME-complete (resp. $(k+1)$-EXPTIME-complete) [18]. Regarding expressiveness, QCTL (and QCTL*) is as expressive as MSO over infinite trees, hence it is strictly more expressive than CTL (and CTL*).

As we will now explain, our AEUPTA is a type of automata perfectly suited for QCTL (and QCTL*): both formalisms have exactly the same expressiveness, and the automata can be used to derive optimal decision procedures for satisfiability and model-checking.

[4] Better complexity results have been obtained recently for parity games [5], but they would make complexity results even harder to read, without significantly improving them.

[5] It is worth noticing that there exist several semantics for QCTL in the literature [14, 12, 18], with different algorithmic and expressiveness properties.

From QCTL or QCTL* to AEUPTA. We write k-exp(n) to denote the family of sets of functions of one variable n defined inductively as follows: 0-exp(n) is the set of functions bounded by a polynomial in n, and $(k+1)$-exp(n) contains all functions f such that $f \in O(2^g)$ with $g \in k$-exp(n). We have:

▶ **Theorem 21.** *Given a Q^{k}CTL (resp. Q^{k}CTL*) formula ϕ over AP with $k > 0$, we can construct an AEUPTA $\mathcal{A}_\phi$ over 2^{AP} accepting exactly the trees satisfying ϕ. The automaton $\mathcal{A}_\phi$ has size $(k\text{-exp}(|\phi|), k\text{-exp}(|\phi|), (k-1)\text{-exp}(|\phi|), 1, (k-1)\text{-exp}(|\phi|))$ (resp. $((k+1)\text{-exp}(|\phi|), (k+1)\text{-exp}(|\phi|), k\text{-exp}(|\phi|), 1, k\text{-exp}(|\phi|))$).*

We just sketch the main ingredients of the construction for QCTL. First note that any CTL formula ϕ can be turned into an $\{\Box, \Diamond\}$-automata [31]; this construction can easily be adapted to our EU-constraints: $\Diamond q$ can be replaced with $\langle q \mapsto 1; \{q_\top\}\rangle$ and $\Box q$ can be replaced with $\langle \varnothing; \{q\}\rangle$. We then get an AEUPTA $\mathcal{A}_\phi$ whose size is in $(O(|\phi|), O(|\phi|), 1, 1, 2)$.

Now consider a Q^{1}CTL formula ϕ; it can be seen as $\psi[\exists P_1.\ \psi_1, \ldots, \exists P_m.\ \psi_m]$ where ψ is a CTL formula and subformulas ψ_i belong to CTL. We assume w.l.o.g. that negations occur only in front of atomic propositions or $\exists$-quantifiers. From the remark above, we can build an AEUPTA $\mathcal{A}_i$ for each ψ_i with $1 \le i \le m$. We handle quantification over P_i in the logic using the projection operation on the automata; this first requires to transform $\mathcal{A}_i$ into an non-alternating automaton $\mathcal{N}_i$ using alternation removal. The size of the resulting automaton (after projection) is in $(1\text{-exp}(|\psi_i|), 1\text{-exp}(|\psi_i|), O(|\psi_i|^2), 1, O(|\psi_i|))$.

Negations in front of $\exists P_i.\ \psi_i$ in the logic are handled with the complementation operation on AEUPTA. It must be noticed that while complementation induces an exponential blow-up in general, this blow-up does not add up with the exponential blow-up for alternation removal. The size of the resulting automaton still is in $(1\text{-exp}(|\psi_i|), 1\text{-exp}(|\psi_i|), O(|\psi_i|^2), 1, O(|\psi_i|))$, Now we can combine all these automata $\mathcal{A}_{\exists P_i.\psi_i}$ (or $\mathcal{A}_{\neg\exists P_i.\psi_i}$) with the automaton for the CTL formula ψ, and get an alternating automaton $\mathcal{A}_\phi$ whose size is the sum of $|\mathcal{A}_\psi|$ and the sizes of automata $|\mathcal{A}_{\exists P_i.\psi_i}|$ (or $|\mathcal{A}_{\neg\exists P_i.\psi_i}|$). In the end, the size of $\mathcal{A}_\phi$ is in $(1\text{-exp}(|\phi|), 1\text{-exp}(|\phi|), O(|\phi|^2), 1, O(|\phi|))$.

The same construction can be applied to any formula ϕ in Q^{k+1}CTL. By induction, we get automata of size $(k\text{-exp}(|\phi|), k\text{-exp}(|\phi|), (k-1)\text{-exp}(|\phi|), 1, (k-1)\text{-exp}(|\phi|))$ for subformulas ψ_i, and end up with an automaton $\mathcal{A}_\phi$ for ϕ whose size is in $((k+1)\text{-exp}(|\phi|), (k+1)\text{-exp}(|\phi|), k\text{-exp}(|\phi|), 1, k\text{-exp}(|\phi|))$.

And back. From an EUPTA $\mathcal{A}$, we can build a QCTL formula $\Phi_\mathcal{A}$ whose size is polynomial in the size of $\mathcal{A}$ and Σ, and such that the models of $\Phi_\mathcal{A}$ are precisely the trees accepted by $\mathcal{A}$. The formula $\Phi_\mathcal{A}$ contains one alternation of quantifiers: it starts wit an existential one, which we use to associate a state of $\mathcal{A}$ with each node of the input tree (as $\mathcal{A}$ is non-alternating, the input tree can be used as accepting tree) and universal quantification is used to ensure the parity condition (see Section 6.3 in [17]). This provides us with an effective translation of any formula $\phi \in$ Q^{k}CTL in Q^{2}CTL, with a $(k+1)$-exp$(|\phi|)$ blow-up in the size of the formula.

Optimal decision procedures. Given a QCTL (res. QCTL*) formula ϕ, one can use the automaton $\mathcal{A}_\phi$ described above in order to decide satisfiability or model-checking. In both cases, the algorithms presented in Section 4 based on the parity games provide optimal algorithms for QCTL and QCTL*. Thus we get the following results (the hardness part is proven in [18]):

▶ **Proposition 22.** *Satisfiability for Q^{k}CTL (resp. Q^{k}CTL*) can be solved in $(k+1)$-EXPTIME (resp. $(k+2)$-EXPTIME) by using the AEUPTA construction and the emptiness checking. Model-checking for Q^{k}CTL (resp. Q^{k}CTL*) can be done in k-EXPTIME (resp. $(k+1)$-EXPTIME) by using the AEUPTA construction and the membership checking.*

Application to MSO. We briefly review *Monadic Second-Order Logic* (*MSO*) over finite or infinite 2^{AP}-labelled trees [25]. MSO is built with first-order variables for vertices (denoted with lowercase letters $x, y, ...$), and monadic second-order variables for sets of vertices (denoted with uppercase letters $X, Y, ...$). Atomic formulas are of the form $x = y$, $\mathsf{Edge}(x, y)$ (to represent the immediate successor relation), $x < y$ (the transitive closure of Edge) $x \in X$, and $\mathsf{P}_a(x)$ for $a \in \mathsf{AP}$. General MSO formulas are constructed from atomic formulas using the boolean connectives and the first- and second-order quantifiers $\exists^1$ and $\exists^2$. We write $\phi(x_1, ..., x_n, X_1, ..., X_k)$ to indicate that $x_1, ..., x_n$ and $X_1, ..., X_k$ may appear free in ϕ. A closed formula contains no free variables. We use the standard semantics for MSO and we write $\mathcal{T}, s_1, ..., s_n, S_1, ..., S_k \models \phi(x_1, ..., x_n, X_1, ..., X_k)$ to indicate that ϕ holds on $\mathcal{T}$ when variables x_1 to x_n in ϕ are replaced with s_1 to s_n, and variables X_1 to X_k are replaced with S_1 to S_k.

In [18], it is proved that MSO and QCTL are equally expressive over trees. This could be used to define translations between MSO and EU-automata, but we can get more efficient constructions as described below. Actually, for any (non-closed) MSO formula $\phi(x_1, ..., x_n, X_1, ..., X_k)$, we build an automaton $\mathcal{A}_\phi$ such that, for any nodes s_1 to s_n and any sets S_1 to S_k, it holds $\mathcal{T}, s_1, ..., s_n, S_1, ..., S_k \models \phi(x_1, ..., x_n, X_1, ..., X_k)$ if, and only if, the $2^{\mathsf{AP} \cup \{x_1,...,x_n,X_1,...,X_k\}}$-labelled tree $\mathcal{T}'$, obtained from $\mathcal{T}$ by labelling any node t with x_i if $t = s_i$, and with X_j if $t \in S_j$, belongs to $\mathcal{L}(\mathcal{A}_\phi)$. Given a closed formula $\Phi \in$ MSO, a 2^{AP}-labelled tree $\mathcal{T}$ belongs to $\mathcal{L}(\mathcal{A}_\Phi)$ if, and only if, $\mathcal{T} \models \Phi$.

Consider a non-closed formula $\phi(x_1, ..., x_n, X_1, ..., X_k)$ where the negation can only be followed by an existential quantifier or a atomic formula. The automaton $\mathcal{A}_\phi$ is built inductively on the structure of ϕ. Boolean connectives $\wedge$ and $\vee$ are handled with the corresponding operations of AEUTA. For quantifications $\exists^1 x.\ \psi$ or $\exists^2 X.\ \psi$, we use projection (after having removed alternation), and check that propositions involved in first-order quantification labels exactly one node of the tree.

Consider a formula in negated normal form (where negations occur only before atomic formula), the complexity of the automata construction depends on the number of (first-order or second-order) quantifier alternations in the formula:

▶ **Theorem 23.** *Given a closed MSO formula ϕ over AP with at most k quantifier alternations (with $k > 0$), we can construct a AEUPTA $\mathcal{A}_\phi$ over 2^{AP} accepting exactly the trees satisfying ϕ. The automaton $\mathcal{A}_\phi$ has size $((k+1)\text{-}exp(|\phi|), (k+1)\text{-}exp(|\phi|), k\text{-}exp(|\phi|), 1, k\text{-}exp(|\phi|))$.*

This allows us to translate any MSO formula with k (first- and second-order) quantifier alternations into a formula with at most four quantifier alternations (and at most one second-order-quantifier alternation) and size in $(k+2)\text{-}\mathsf{exp}(|\phi|)$. Note also that Theorem 23 also provides us with decision procedures in $(k+1)$-EXPTIME for model-checking (and $(k+2)$-EXPTIME for satisfiability).

6 Conclusion

We have introduced a new class of tree automata (AEUPTA) for trees of arbitrary branching degrees. We showed that these automata have exactly the same expressive power as the temporal logics QCTL and QCTL*, and as MSO.

In order to prove those results, we have developed algorithms for manipulating our AEUPTA, and have carefully studied their complexities. This has allowed us to obtain decision procedures for satisfiability and model checking for QCTL* with optimal complexities. It also allowed us to obtain an effective translation from QCTL to Q^2CTL, and similarly, from MSO to its fragment with only one second-order-quantifier alternation.

References

1 Orna Bernholtz and Orna Grumberg. Branching time temporal logic and $\mathcal{A}$m$\mathbf{o}$r$\mathrm{P}\mathcal{H}$Ous tree automata. In Eike Best, editor, *Proceedings of the 4th International Conference on Concurrency Theory (CONCUR'93)*, volume 715 of *Lecture Notes in Computer Science*, pages 262–277. Springer-Verlag, 1993.

2 Orna Bernholtz, Moshe Y. Vardi, and Pierre Wolper. An automata-theoretic approach to branching-time model checking (extended abstract). In David L. Dill, editor, *Proceedings of the 6th International Conference on Computer Aided Verification (CAV'94)*, volume 818 of *Lecture Notes in Computer Science*, pages 142–155. Springer-Verlag, 1994. doi:10.1007/3-540-58179-0_50.

3 Mikołaj Bojańczyk. Tree-walking automata. In Carlos Martín-Vide, Friedrich Otto, and Henning Fernau, editors, *Revised Papers of the 2nd International Conference on Language and Automata Theory and Applications (LATA'08)*, volume 5196 of *Lecture Notes in Computer Science*, pages 1–2. Springer-Verlag, 2008. doi:10.1007/978-3-540-88282-4_1.

4 Julius R. Büchi. On a decision method in restricted second order arithmetic. In Ernest Nagel, Patrick Suppes, and Alfred Tarski, editors, *Proceedings of the 1960 International Congress on Logic, Methodology and Philosophy of Science (LMPS'60)*, pages 1–11. Stanford University Press, 1962.

5 Cristian S. Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li, and Frank Stephan. Deciding parity games in quasipolynomial time. In Hamed Hatami and Pierre McKenzie, editors, *Proceedings of the 49th Annual ACM Symposium on the Theory of Computing (STOC'17)*, pages 252–263. ACM Press, 2017. doi:10.1145/3055399.305540.

6 Hubert Comon, Max Dauchet, Rémi Gilleron, Florent Jacquemard, Denis Lugiez, Christof Löding, Sophie Tison, and Marc Tommasi. Tree automata techniques and applications. Technical Report hal-03367725, H.A.L., 2008.

7 Arnaud Da Costa, François Laroussinie, and Nicolas Markey. Quantified CTL: Expressiveness and model checking. In Maciej Koutny and Irek Ulidowski, editors, *Proceedings of the 23rd International Conference on Concurrency Theory (CONCUR'12)*, volume 7454 of *Lecture Notes in Computer Science*, pages 177–192. Springer-Verlag, 2012. doi:10.1007/978-3-642-32940-1_14.

8 Calvin C. Elgot. Decision problems of finite automata design and related arithmetics. *Transactions of the American Mathematical Society*, 98(1):21–51, 1961. doi:10.2307/1993511.

9 E. Allen Emerson and Charanjit S. Jutla. Tree automata, mu-calculus and determinacy. In *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science (FOCS'91)*, pages 368–377. IEEE Comp. Soc. Press, 1991. doi:10.1109/SFCS.1991.185392.

10 E. Allen Emerson and A. Prasad Sistla. Deciding full branching time logic. *Information and Control*, 61(3):175–201, 1984. doi:10.1016/S0019-9958(84)80047-9.

11 Tim French. Decidability of quantified propositional branching time logics. In Markus Stumptner, Dan Corbett, and Mike Brooks, editors, *Proceedings of the 14th Australian Joint Conference on Artificial Intelligence (AJCAI'01)*, volume 2256 of *Lecture Notes in Computer Science*, pages 165–176. Springer-Verlag, 2001. doi:10.1007/3-540-45656-2_15.

12 Tim French. *Bisimulation Quantifiers for Modal Logics.* PhD thesis, School of Computer Science & Software Engineering, University of Western Australia, 2006.

13 David Janin and Igor Walukiewicz. Automata for the modal μ-calculus and related results. In Jirí Wiedermann and Petr Hájek, editors, *Proceedings of the 20th International Symposium on Mathematical Foundations of Computer Science (MFCS'95)*, volume 969 of *Lecture Notes in Computer Science*, pages 552–562. Springer-Verlag, 1995. doi:10.1007/3-540-60246-1_160.

14 Orna Kupferman. Augmenting branching temporal logics with existential quantification over atomic propositions. In Pierre Wolper, editor, *Proceedings of the 7th International Conference on Computer Aided Verification (CAV'95)*, volume 939 of *Lecture Notes in Computer Science*, pages 325–338. Springer-Verlag, 1995. doi:10.1007/3-540-60045-0_60.

15 Orna Kupferman, Parthasarathy Madhusudan, P. S. Thiagarajan, and Moshe Y. Vardi. Open systems in reactive environments: Control and synthesis. In Catuscia Palamidessi, editor, *Proceedings of the 11th International Conference on Concurrency Theory (CONCUR'00)*, volume 1877 of *Lecture Notes in Computer Science*, pages 92–107. Springer-Verlag, 2000. doi:10.1007/3-540-44618-4_9.

16 Orna Kupferman and Moshe Y. Vardi. $\Pi_2 \cap \Sigma_2 \equiv$ AFMC. In Jos C. M. Baeten, Jan Karel Lenstra, Joachim Parrow, and Gerhard J. Woeginger, editors, *Proceedings of the 30th International Colloquium on Automata, Languages and Programming (ICALP'03)*, volume 2719 of *Lecture Notes in Computer Science*, pages 697–713. Springer-Verlag, 2003.

17 François Laroussinie and Nicolas Markey. Arbitrary-arity tree automata and QCTL, 2024. doi:10.48550/arXiv.2410.18799.

18 François Laroussinie and Nicolas Markey. Quantified CTL: expressiveness and complexity. *Logical Methods in Computer Science*, 10(4), 2014. doi:10.2168/LMCS-10(4:17)2014.

19 François Laroussinie, Nicolas Markey, and Philippe Schnoebelen. Temporal logic with forgettable past. In *Proceedings of the 17th Annual Symposium on Logic in Computer Science (LICS'02)*, pages 383–392. IEEE Comp. Soc. Press, 2002. doi:10.1109/LICS.2002.1029846.

20 Christof Löding. Automata on infinite trees. In Jean-Éric Pin, editor, *Handbook of automata theory*, volume 1, chapter 8, pages 265–302. EMS Press, 2021. doi:10.4171/AUTOMATA-1/8.

21 Robert McNaughton. Testing and generating infinite sequences by a finite automaton. *Information and Control*, 9(6):521–530, 1966. doi:10.1016/S0019-9958(66)80013-X.

22 David E. Muller and Paul E. Schupp. Alternating automata on infinite objects, determinacy and Rabin's theorem. In Maurice Nivat and Dominique Perrin, editors, *Automata on Infinite Words – École de Printemps d'Informatique Théorique (EPIT'84)*, volume 192 of *Lecture Notes in Computer Science*, pages 99–107. Springer-Verlag, 1985. doi:10.1007/3-540-15641-0_27.

23 Michael O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 141:1–35, 1969.

24 A. Prasad Sistla, Moshe Y. Vardi, and Pierre Wolper. The complementation problem for Büchi automata, with applications to temporal logic. In Wilfried Brauer, editor, *Proceedings of the 12th International Colloquium on Automata, Languages and Programming (ICALP'85)*, volume 194 of *Lecture Notes in Computer Science*, pages 465–474. Springer-Verlag, 1985.

25 Wolfgang Thomas. Languages, automata and logics. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages*, volume 3, pages 389–455. Springer-Verlag, 1997.

26 Boris A. Trakhtenbrot. Finite automata and the logic of one-place predicates. *Siberskii Matematicheskii Zhurnal*, 3(1):103–131, 1962.

27 Moshe Y. Vardi and Pierre Wolper. An automata-theoretic approach to automatic program verification. In *Proceedings of the 1st Annual Symposium on Logic in Computer Science (LICS'86)*, pages 332–344. IEEE Comp. Soc. Press, 1986.

28 Moshe Y. Vardi and Pierre Wolper. Automata theoretic techniques for modal logics of programs. *Journal of Computer and System Sciences*, 32(2):183–221, 1986. doi:10.1016/0022-0000(86)90026-7.

29 Igor Walukiewicz. Monadic second order logic on tree-like structures. In Claude Puech and Rüdiger Reischuk, editors, *Proceedings of the 13th Symposium on Theoretical Aspects of Computer Science (STACS'96)*, volume 1046 of *Lecture Notes in Computer Science*, pages 401–413. Springer-Verlag, 1996. doi:10.1007/3-540-60922-9_33.

30 Igor Walukiewicz. Monadic second order logic on tree-like structures. *Theoretical Computer Science*, 275(1-2):311–346, 2002. doi:10.1016/S0304-3975(01)00185-2.

31 Thomas Wilke. CTL$^+$ is exponentially more succinct than CTL. In C. Pandu Rangan, Venkatesh Raman, and R. Ramanujam, editors, *Proceedings of the 19th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'99)*, volume 1738 of *Lecture Notes in Computer Science*, pages 110–121. Springer-Verlag, 1999. doi:10.1007/3-540-46691-6_9.

32 Thomas Wilke. Alternating tree automata, parity games, and modal μ-calculus. *Bulletin of the Belgian Mathematical Society – Simon Stevin*, 8(2):359–391, 2001.

33 Fabio Zanasi. Expressiveness of monadic second-order logics on infinite trees of arbitrary branching degrees. Master's thesis, Amsterdam University, the Netherlands, 2012.