# Top-Down Complementation of Automata on Finite Trees

Laurent Doyen[a]

[a] *CNRS & LMF, ENS Paris-Saclay, 4 av. des sciences, 91190 Gif-sur-Yvette, France*

**Abstract**

We present a new complementation construction for nondeterministic automata on finite trees. The traditional complementation involves determinization of the corresponding bottom-up automaton (recall that top-down deterministic automata are less powerful than nondeterministic automata, whereas bottom-up deterministic automata are equally powerful).

The construction works directly in a top-down fashion, therefore without determinization. The main advantages of this construction are: (*i*) in the special case of finite words it boils down to the standard subset construction (which is not the case of the traditional bottom-up complementation construction), and (*ii*) it illustrates the core argument of the complementation lemma for infinite trees, central in the proof of Rabin's tree theorem, in a simpler setting where issues related to acceptance conditions over infinite words and determinacy of infinite games are not present.

*Keywords:* Tree automata, Complementation, Games, Determinacy

## 1. Introduction

Finite automata running on finite trees [1, 2] generalize finite word automata and enjoy similar expressiveness, closure, and decidability properties [3]. Regular languages of finite trees are those recognizable by nondeterministic tree automata, and their closure properties under union, intersection, and complement enable a tight connection with logical formalisms, prominently monadic second-order logic (MSO) on finite trees.

In this context, the most critical closure property is complementation, as closure under union is straightforward with nondeterministic automata, and closure under intersection then follows from closure under complementation. As for finite words, the traditional approach to complementation of tree automata is to first determinize the given automaton, then to complement it by dualizing the acceptance condition [3, 4].

A crucial remark is that the notion of determinism in tree automata has two different flavours that are not (expressively) equivalent. In the first flavour, determinism is understood in a *bottom-up* fashion: given the states of the automaton after processing the subtrees of a given position $p$ in the input tree and given the letter at the position $p$, there is a unique transition that can be used to assign a state to $p$. In the second flavour, determinism is understood in a dual *top-down* fashion: given the state of the automaton at a given position $p$ in the input tree and given the letter at the position $p$, there is a unique transition that can be used to assign a state to each child of $p$, from which to process the corresponding subtree.

Bottom-up determinization is always possible, as deterministic bottom-up tree automata are as expressive as nondeterministic automata. In contrast, deterministic top-down tree automata are strictly less expressive than nondeterministic automata, leading to several research directions [5] such as deciding whether a regular tree language is recognizable by a deterministic top-down automaton, which was solved in polynomial time only recently [6].

However, in the special case of words, the traditional approach to complementation is akin to the top-down flavour of determinization (assuming the input word, viewed as a particular tree, starts at the root). Of course, bottom-up determinization could also be applied to word automata, but this corresponds to first reversing all transitions and exchanging initial and final states, then determinizing (using the classical approach by subset construction), and finally reversing again the resulting automaton. This does not yield a deterministic automaton in the usual sense, but is indeed sufficient for complementation. A natural question is then to find a simple complementation procedure for tree automata that does boil down to the classical approach in the special case of word automata, yielding a top-down deterministic word automaton. We present such a procedure in this paper. By the above remarks, the construction yields a nondeterministic tree automaton, that becomes deterministic only on words.

A more fundamental reason to look for top-down complementation is the connection with automata on infinite trees, for which complementation is the key ingredient to Rabin's decidability theorem (for the logic MSO on infinite trees), considered to be one of the most important (and difficult) decidability result in the area of logics [3, 7, 8]. Over infinite trees, the complementation procedure involves a combination of deep insights, prominently the determinacy of infinite-state parity games and equivalence of various acceptance conditions for infinite trees, notably due to the strictly lower expressive power of both deterministic parity automata and nondeterministic Büchi automata, as compared to nondeterministic parity automata [8]. We remark that in the case of finite trees, only the difference in expressive power between deterministic and nondeterministic automata remains an issue for top-down complementation. Acceptance conditions are easy to dualize over finite trees, and determinacy is required in a much simpler form, namely over finite-duration games, which can be established using backward induction without difficulty. Our complementation construction follows a modern approach [8], using games and determinacy to circumvent determinization [9, 10]. It is therefore a special case of the complementation for automata on infinite trees, that illustrates the role of games and the power of determinacy in a more accessible way thanks to the simpler setting of finite trees.

Finally, note that our construction involves an exponential blow up, which is unavoidable already for words [11, 12].

## 2. Definitions

A (finite) tree is a finite nonempty prefix-closed set $t \subseteq \mathbb{N}^*$. The elements of $t$ are called *positions* and the root is the empty sequence $\varepsilon$. The *children* of position $p \in t$ are the positions $p \cdot c \in t$ where $c \in \mathbb{N}$. We refer to $p \cdot c$ as the child of $p$ in *direction* $c$. A position with no children is a *leaf*.

Given a finite ranked alphabet $\Sigma$ where each letter $a \in \Sigma$ has a fixed arity denoted by $\text{arity}(a) \in \mathbb{N}$, a $\Sigma$-labeled tree is a pair $\langle t, \mu \rangle$ where $t$ is a tree and $\mu : t \to \Sigma$ is a mapping that respects the arity, that is for every position $p \in t$, if the letter $\mu(p)$ is of arity $n = \text{arity}(\mu(p)) \geq 0$, then the children of $p$ are $p \cdot 1, \dots, p \cdot n$. In particular, the positions labeled by a constant (a letter of arity 0) are leaves.

A (nondeterministic) tree automaton over alphabet $\Sigma$ is a tuple $\mathcal{A} = \langle Q, Q_\varepsilon, (\Delta_a)_{a \in \Sigma} \rangle$ consisting of a finite set $Q$ of states, a set $Q_\varepsilon \subseteq Q$ of initial states, and for each letter $a \in \Sigma$ a transition relation $\Delta_a \subseteq Q \times Q^n$ where $n$ is the arity of $a$ and $Q^n = Q \times Q^{n-1}$ where by convention $Q^0 = \{\varepsilon\}$. Given a state $q \in Q$, we denote by $\mathcal{A}_q$ the tree autmoaton obtained from $\mathcal{A}$ by replacing the set $Q_\varepsilon$ of initial states by $\{q\}$. We view transition relations as tiling systems where the elements of $\Delta_a$ are tiles, and a tiling associates states to positions in the input tree $\langle t, \mu \rangle$, such that if $q$ is a state associated with position $p$ and $q_1, \dots, q_n$ are states associated with the children of $p$, then $(q, q_1, \dots, q_n) \in \Delta_a$ is a tile where $a = \mu(p)$ is the letter at position $p$

in $t$. We say that $(q, q_1, \ldots, q_n)$ is a tile from $q$ with children $q_1, \ldots, q_n$. We may denote tiles associated with letter $a$ by $a^q(q_1, \ldots, q_n)$.

A *run* (or tiling) of $\mathcal{A}$ on an input tree $\langle t, \mu \rangle$ is a $Q$-labeled tree $\langle t_r, r \rangle$ where $t_r = t$ and $r : t \to Q$ satisfies the following: for all positions $p \in t$, if $\mu(p) = a$, then $(r(p), r(p \cdot 1), \ldots, r(p \cdot n)) \in \Delta_a$ where $n = \text{arity}(a)$. A run is *initialized* if $r(\varepsilon) \in Q_\varepsilon$. An input tree $\langle t, \mu \rangle$ is accepted by $\mathcal{A}$ if there exists an initialized run on it. The *language* of $\mathcal{A}$ is the set $L(\mathcal{A})$ of all trees accepted by $\mathcal{A}$. Note that $L(\mathcal{A}) = \bigcup_{q \in Q_\varepsilon} L(\mathcal{A}_q)$. We denote by $\overline{L}(\mathcal{A})$ the complement of $L(\mathcal{A})$, that is the set of all $\Sigma$-labeled trees that are not accepted by $\mathcal{A}$. The languages accepted by nondeterministic tree automata are called *regular* [3, Chapter 2]. While word automata have accepting states, and a word is accepted if the state that labels its last position is accepting, the symbol in the last position of a branch of a tree must be of arity 0, which can be used in tree automata to mimic the acceptance mechanism of word automata. For instance, we may say that a state $q$ is *accepting at a leaf labeled by $a$* (thus $\text{arity}(a) = 0$) if $(q, \varepsilon) \in \Delta_a$.

**Example.** Let $\Sigma = \{f, a, b\}$ be a ranked alphabet where $f$ has arity 2, and $a, b$ have arity 0. Given $k \geq 1$, consider the set $L_k$ of all $\Sigma$-labeled trees having two (distinct) leaves labeled by $a$ that are $k$-cousin (where two positions are 0-cousin if they are equal, and inductively two positions are $(k+1)$-cousin if their parents are $k$-cousin). An example of a tree automaton accepting $L_k$ is defined by the state space $Q_k = \{q_0, \ldots, q_k\} \cup \{q_\perp\}$, the initial state $Q_\varepsilon = \{q_0\}$, and transitions $\Delta_a = \{q_k\}$, $\Delta_b = \{q_\perp\}$, and $\Delta_f$ as the following tiles, for all states $z \in Q_k$:

$$\begin{array}{lll}
f^{q_0}(q_1, q_1) & f^{q_i}(q_{i+1}, z) \ (1 \leq i < k) & f^{q_\perp}(q_\perp, q_\perp) \\
f^{q_0}(q_0, z) & f^{q_i}(z, q_{i+1}) \ (1 \leq i < k) & f^{q_\perp}(q_1, z) \\
f^{q_0}(z, q_0) & & f^{q_\perp}(z, q_1).
\end{array}$$

Intuitively, from a position labeled by $q_i$ there is a branch of length $k - i$ to a leaf labeled by $a$, and the automaton checks that there exist two positions labeled by $q_i$ that are $i$-cousin. The state $q_\perp$ is used for unsuccessful labeling of a position, discarding information about the length of branches from that position. Figure 1 shows a partial tiling of an input tree in $L_3$. $\square$

A tree automaton is *bottom-up deterministic* (or *backward deterministic*, specially in the case of words) if the following holds: if $(q, q_1, \ldots, q_n) \in \Delta_a$ and $(q', q_1, \ldots, q_n) \in \Delta_a$, then $q = q'$, that is the states at the children of a position $p$ determine the state at $p$ (given the letter at $p$);

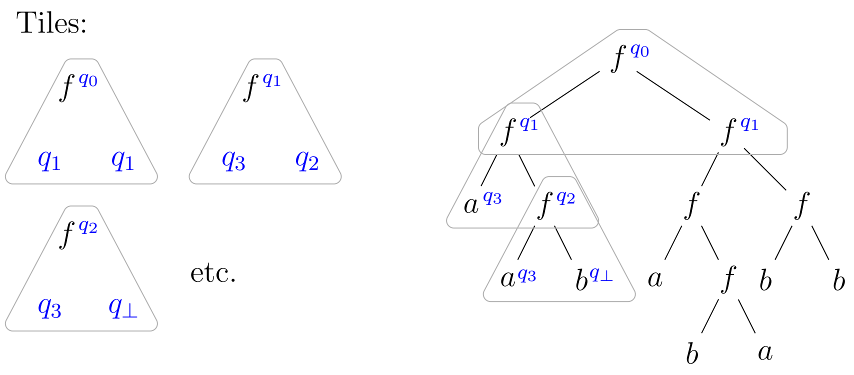

Figure 1: Tiles and a (partial) tiling of an input tree.

A tree automaton is *top-down deterministic* (or *forward deterministic*, specially in the case of words) if $|Q_\varepsilon| = 1$ and the following holds: if $(q, q_1, \dots, q_n) \in \Delta_a$ and $(q, q'_1, \dots, q'_n) \in \Delta_a$, then $q_i = q'_i$ for all $1 \leq i \leq n$, that is the state at a position $p$ determines the states at the children of $p$ (given the letter at $p$);

In the sequel we denote by $\Delta(q, a)$ the set of all tuples $u$ such that $(q, u) \in \Delta_a$. Note that specifying the sets $\Delta(q, a)$ for each state $q \in Q$ and letter $a \in \Sigma$ is equivalent to specifying the transition relations $(\Delta_a)_{a \in \Sigma}$. In particular, if $a$ is of arity 0, then either $\Delta(q, a) = \{\varepsilon\}$ or $\Delta(q, a) = \varnothing$.

## 3. Complementation

We recall the classical (bottom-up) construction for complementing a tree automaton, and then we present our new construction, which follows a more top-down approach.

### *3.1. Classical construction: bottom-up*

The classical approach to complementation of a tree automaton $\mathcal{A}$ is to construct a bottom-up deterministic automaton that accepts the same trees as $\mathcal{A}$, which is then easy to complement [3, 4].

The complement of $\mathcal{A}$ is the automaton $\mathcal{B} = \langle S, S_\varepsilon, (\Delta^{\mathsf{d}}_a)_{a \in \Sigma} \rangle$ where $S = 2^Q$ contains all subsets of $Q$, the set of initial states is $S_\varepsilon = \{s \in S \mid s \cap Q_\varepsilon = \varnothing\}$, and the transition relation is defined, for all $a \in \Sigma$ and for all $s, s_1, \ldots, s_n \in S$ (where $n = \text{arity}(a)$), by:

$$(s_1, \ldots, s_n) \in \Delta^{\mathsf{d}}(s, a)$$
$$\text{iff}$$
$$s = \{q \in Q \mid \exists q_1 \in s_1 \cdots \exists q_n \in s_n : (q_1, \ldots, q_n) \in \Delta(q, a)\}.$$

Intuitively, the (unique) run of the automaton $\mathcal{B}$ on an input tree labels a position $p$ with the set of all states that may label the position $p$ in a run of the automaton $\mathcal{A}$ on the subtree rooted at $p$. The set $S_\varepsilon$ of initial states in $\mathcal{B}$ ensures that no run of $\mathcal{A}$ labels the root with an initial state of $\mathcal{A}$.

The automaton $\mathcal{B}$ is bottom-up deterministic. In the special case of words, it corresponds to a backward-deterministic automaton, obtained by first reversing the automaton (i.e., reversing all transitions and exchanging initial and final states), then determinizing (using the classical subset construction), and finally reversing the resulting automaton. Hence, this construction does not produce a deterministic (precisely, forward-deterministic) automaton in the special case of words.

**Theorem 1** ([1, Theorem 1 & 2])**.** $L(\mathcal{B}) = \overline{L}(\mathcal{A})$.

*3.2. New construction: top-down*

The *top-down complement* of $\mathcal{A}$ is the automaton $\mathcal{C} = \langle S, S_\varepsilon, (\Delta^{\mathsf{n}}_a)_{a \in \Sigma} \rangle$ where $S = 2^Q$ contains all subsets of $Q$, the set $S_\varepsilon = \{Q_\varepsilon\}$ is a singleton, and the transition relation is defined, for all $a \in \Sigma$ and for all $s, s_1, \ldots, s_n \in S$ (where $n = \text{arity}(a)$), by:

$$(s_1, \ldots, s_n) \in \Delta^{\mathsf{n}}(s, a)$$
$$\text{iff}$$
$$\forall q \in s \cdot \forall (q_1, \ldots, q_n) \in \Delta(q, a) \cdot \exists 1 \leq i \leq n : q_i \in s_i. \tag{1}$$

Given a letter $a \in \Sigma$, a tuple $(s, s_1, \ldots, s_n)$ is a tile associated with $a$ in $\mathcal{C}$ if for every state $q \in s$ and every tile $(q, q_1, \ldots, q_n)$ associated with $a$ in $\mathcal{A}$, some state $q_i$ in direction $i$ appears in $s_i$. Intuitively, the transitions in $\mathcal{C}$ from $s$ can "distribute" the states along the children, as long as every possible transition in $\mathcal{A}$ from a state $q \in s$ has one of its children $q_i$ that appears in the corresponding child $s_i$ of the transition in $\mathcal{C}$. Note that for a letter $a$ of arity 0, we have $\Delta^{\mathsf{n}}(s, a) \neq \varnothing$ (that is, $\Delta^{\mathsf{n}}(s, a) = \{\varepsilon\}$) if $\Delta(q, a) = \varnothing$ for all $q \in s$.

We prune the transition relation by keeping in $\Delta^{\mathsf{n}}(s, a)$ only the elements that are minimal with respect to component-wise set inclusion. This pruning entails that in the special case of words, the automaton $\mathcal{C}$ is forward-deterministic: it is the classical subset construction. Hence this construction is deterministic for word automata, although it is nondeterministic for tree automata. We present the main ideas for a correctness proof in Section 4.

In our example for $L_3$ (Figure 1), the tiles representing $\Delta^{\mathsf{n}}(\{q_1\}, f)$ are:

$$f^{\{q_1\}}(\{q_2\}, \{q_2\}) \qquad f^{\{q_1\}}(Q_3, \varnothing) \qquad f^{\{q_1\}}(\varnothing, Q_3).$$

**Theorem 2.** *$L(\mathcal{C}) = \overline{L}(\mathcal{A})$.*

## 4. Correctness

The correctness argument for Theorem 2 relies on a fundamental correspondence between acceptance of a tree $T = \langle t, \mu \rangle$ by an automaton $\mathcal{A}$ and winning in a two-player *acceptance game* $G_{\mathcal{A},T}$ associated with $\mathcal{A}$ and $T$. This correspondence is the core of the (modern) proof of Rabin's theorem, showing that tree automata on *infinite* trees are closed under complementation [8, 9, 10]. We rephrase the argument in the simpler case of finite trees, which highlights the role of games and determinacy in complementation of tree automata and may be useful for pedagogical purpose.

The players in the acceptance game $G_{\mathcal{A},T}$ are called Automaton and Pathfinder. Initially, the game starts in a node $(\varepsilon, q_\varepsilon)$ where $q_\varepsilon \in Q_\varepsilon$ is chosen by Automaton. The game proceeds in rounds where each round starts in a node $(p, q)$ denoting a position $p \in t$ of the input tree labeled by a state $q \in Q$ of the automaton, and is played as follows:

1. Automaton chooses a transition $(q_1, \dots, q_n) \in \Delta(q, a)$ where $a = \mu(p)$; if no such transition exists (i.e., $\Delta(q, a) = \varnothing$), then Automaton loses (and Pathfinder wins);
2. then Pathfinder chooses a child $1 \le i \le n$ and the game proceeds to the next round in node $(p \cdot i, q_i)$; if $n = 0$ (i.e., the letter $a$ has arity 0 and the position $p$ is a leaf), then no such $i$ exists and Pathfinder loses (and Automaton wins);

Given an initial node $(\varepsilon, q_\varepsilon)$, a *strategy* for Automaton is a map $f_A : t \times Q \to Q^*$ such that $f_A(p, q) \in \Delta(q, \mu(p))$ for all $p \in t$ and $q \in Q$. Playing according to $f_A$ means that in every round, given the node $(p, q)$ at the beginning of the round, Automaton chooses the transition $f_A(p, q)$. The strategy $f_A$ is winning if the acceptance game starting in node $(\varepsilon, q_\varepsilon)$ ends up in a leaf $(p, q)$ where $\Delta(q, \mu(p)) = \{\varepsilon\}$ (thus Pathfinder loses) when playing according to $f_A$, regardless of the choices of Pathfinder.

It is straightforward to see that there exists a winning strategy for Automaton with initial node $(\varepsilon, q_\varepsilon)$ if and only if there exists an initialized run of $\mathcal{A}$ on $T$: $(i)$ from a run $\langle t, r\rangle$ with $r(\varepsilon) = q_\varepsilon$ we can construct a winning strategy $f_A$ for Automaton defined by $f_A(p, q) = (r(p \cdot 1), \dots, r(p \cdot n))$. Note that when playing according to $f_A$, in every round of the game the node $(p, q)$ at the beginning of the round is consistent with the run (that is, $r(p) = q$), regardless of the choices of Pathfinder; $(ii)$ from a winning strategy $f_A$, define the initialized run $\langle t, r\rangle$ inductively by $r(\varepsilon) = q_\varepsilon$ and, for all $p \in t$ and $1 \leq i \leq n$, by $r(p \cdot i) = q_i$ where $f_A(p, r(p)) = (q_1, \dots, q_n)$.

In order to complement $\mathcal{A}$, we need to construct an automaton $\mathcal{C}$ that accepts a tree $T$ if it can verify the non-existence of a winning strategy for Automaton in the acceptance game $G_{\mathcal{A},T}$, from every initial state $q_\varepsilon \in Q_\varepsilon$. For every fixed $T = \langle t, \mu\rangle$, the game $G_{\mathcal{A},T}$ is of finite duration, and therefore a simple argument shows that the game is determined, that is the non-existence of a winning strategy for Automaton is equivalent to the existence of a winning strategy for Pathfinder, where a strategy for Pathfinder is a map $f_P : t \times Q^* \to \mathbb{N}$ such that $f_P(p, q_1, \dots, q_n) = i$ for some $1 \leq i \leq n$ and the definition of winning is analogous to Automaton strategies.

A simple argument for determinacy is given by backward induction: we mark the nodes $(p, q)$ as either winning or losing (for Automaton), and for non-leaf nodes we store an optimal choice (for Automaton if the node is winning, for Pathfinder if the node is losing) as follows. We start at the leaf nodes $(p, q)$ (where $p$ is a leaf in $t$, and thus $a = \mu(p)$ is of arity 0) and mark them as winning if $\Delta(q, a) \neq \varnothing$, as losing otherwise. Inductively, a node $(p, q)$ is marked as winning if there exists $(q_1, \dots, q_n) \in \Delta(q, \mu(p))$ such that all nodes $(p \cdot i, q_i)$ are winning, and define $f_A(p, q) = (q_1, \dots, q_n)$; the node $(p, q)$ is marked as losing otherwise, thus for all $(q_1, \dots, q_n) \in \Delta(q, \mu(p))$ there exists $1 \leq i \leq n$ such that the node $(p \cdot i, q_i)$ is losing and we define $f_P(p, q_1, \dots, q_n) = i$. It is immediate that $f_A$ is a winning strategy for Automaton from winning nodes, and $f_P$ is a winning strategy for Pathfinder from losing nodes.

Hence we need to construct an automaton $\mathcal{C}$ that verifies the existence of a winning strategy for Pathfinder (from every initial state), in order to ensure that the input tree is not accepted by $\mathcal{A}$. A strategy for Pathfinder is a map $f_P : t \to (Q^* \times \mathbb{N})$, which can be viewed as a labeling of the tree $t$ that the automaton $\mathcal{C}$ will guess along with processing the input tree.

The guess is correct (and yields a run of $\mathcal{C}$) if along every branch there is a position $p$ labeled by a state $q$ such that $\Delta(q, \mu(p)) = \varnothing$. As the state $q$ labeling a position is chosen by Automaton, the guess must be verified for all possible choices of Automaton. We store in the state space of the automaton $\mathcal{C}$ the set $s$ of all states that may label a position $p$ (initially $s = Q_\varepsilon$ for $p = \varepsilon$), and we check that for all $q \in s$, and for all transitions $(q_1, \dots, q_n) \in \Delta(q, \mu(p))$, there exists a direction $1 \leq i \leq n$ such that Pathfinder wins the acceptance game from the subtree rooted at the $i$-th child of $p$, in state $q_i$. Gathering in a set $s_i$ the states chosen by Pathfinder along direction $i$ (across all such transitions), we obtain the transition relation of $\mathcal{C}$ as defined in Condition (1). The correctness of the construction then follows from determinacy of the acceptance game, and the correspondence between the runs of $\mathcal{C}$ on input trees $T$ and winning strategies for Pathfinder from the nodes $(\varepsilon, q_\varepsilon)$ with $q_\varepsilon \in Q_\varepsilon$ in the acceptance games $G_{\mathcal{A},T}$.

## 5. Concluding remarks

The constructions of Section 3 can be generalized to alternating tree automata, which are straightforwardly closed under complementation by dualizing the transition relation, to obtain a nondeterministic tree automaton that accepts the same language as a given alternating tree automaton. This is known for the construction of Section 3.1, showing that bottom-up deterministic tree automata have the same expressive power as alternating tree automata [3, Section 7.4]. Analogously, the idea of the construction in Section 3.2 can be generalized to alternating tree automata as follows.

In alternating automata, the transition relation over a letter $a \in \Sigma$ is a mapping $\varphi_a : Q \to B^+(Q \times \{1, \dots, n\})$ where $n = \text{arity}(a)$ and $B^+(S)$ is the set of positive Boolean formulas $\varphi$ over the set $S$ (viewed as a set of variables), constructed from the elements of $S \cup \{\mathsf{true}, \mathsf{false}\}$ using the connectives $\vee$ and $\wedge$. For $P \subseteq S$, we write $P \models \varphi$ if the formula $\varphi$ is satisfied under the assignment that sets the variables in $P$ to $\mathsf{true}$ and the variables in $S \setminus P$ to $\mathsf{false}$. For example, the transition relation $\Delta_a$ of a nondeterministic automaton corresponds to $\varphi_a(q) = \bigvee_{(q,q_1,\dots,q_n) \in \Delta_a} \bigwedge_{1 \leq i \leq n} (q_i, i)$. Note that for $n = 0$ we have $\varphi_a(q) = \mathsf{true}$ if $\Delta(q, a) = \{\varepsilon\}$, and $\varphi_a(q) = \mathsf{false}$ otherwise (i.e., if $\Delta(q, a) = \varnothing$). We further assume without loss of generality that alternating automata have a single initial state $q_\varepsilon \in Q$ [3, Lemma 7.3.1].

It is standard to define the runs of alternating tree automata [3, Definition 7.2.4], and the accepted language. Complementation is straightforward by exchanging the connectives $\vee$ and $\wedge$, as well as $\mathsf{true}$ and $\mathsf{false}$ in the transition function (note that the initial state remains unchanged). A nondeterministic tree automaton $\mathcal{D} = \langle S, S_\varepsilon, (\Delta^{\mathsf{n}}_a)_{a\in\Sigma}\rangle$ that accepts the same language as a given alternating tree automaton $\mathcal{A} = \langle Q, q_\varepsilon, (\varphi_a)_{a\in\Sigma}\rangle$ can be constructed in a top-down fashion by setting $S = 2^Q$ and $S_\varepsilon = \{\{q_\varepsilon\}\}$ as in the automaton $\mathcal{C}$ of Section 3.2, and the transition relation is defined, for all $a \in \Sigma$ and for all $s, s_1, \ldots, s_n \in S$ (where $n = \text{arity}(a)$), by:

$$(s_1, \ldots, s_n) \in \Delta^{\mathsf{n}}(s, a)$$
$$\text{iff}$$
$$\{(q, i) \mid q \in s_i, 1 \leq i \leq n\} \models \bigwedge_{q\in s} \varphi_a(q). \tag{2}$$

In particular, when $\mathcal{A}$ is nondeterminisitic the condition (2) is equivalent to condition (1) in Section 3.2. As before, we further prune the sets $\Delta^{\mathsf{n}}(s, a)$ to keep only their minimal elements.

Given $s \subseteq Q$, it is easy to show by induction on the height of input trees that the set $L(\mathcal{D}_s)$ of all trees accepted by $\mathcal{D}_s$ is equal to the set $\bigcap_{q\in s} L(\mathcal{A}_q)$ of all trees accepted by all automata $\mathcal{A}_q$ for $q \in s$, and to conclude that $L(\mathcal{A}) = L(\mathcal{D})$ since the initial state $s_\varepsilon = \{q_\varepsilon\}$ in $\mathcal{D}$ is a singleton.

Finally, note that if the automaton $\mathcal{A}$ is top-down deterministic, then in the constructed automaton $\mathcal{C}$ (or $\mathcal{D}$) after pruning of the non-minimal elements, the reachable states are singletons and the number of transitions is at most $N$ times the number of transitions in $\mathcal{A}$ where $N$ is the largest arity, thus the construction is linear (for a fixed alphabet) in that case.

**Acknowledgment**. The author is grateful to anonymous reviewers for insightful feedback and suggestions, specially regarding the generalization of our construction to alternating tree automata.

## References

[1] J. W. Thatcher, J. B. Wright, Generalized finite automata theory with an application to a decision problem of second-order logic, Math. Syst. Theory 2 (1) (1968) 57–81.

[2] J. Doner, Tree acceptors and some of their applications, J. Comput. Syst. Sci. 4 (5) (1970) 406–451.

[3] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, C. Löding, S. Tison, M. Tommasi, Tree Automata Techniques and Applications, 2008.

[4] C. Löding, W. Thomas, Automata on finite trees, in: Handbook of Automata Theory, European Mathematical Society, 2021, pp. 235–264.

[5] W. Martens, F. Neven, T. Schwentick, Deterministic top-down tree automata: past, present, and future, in: Logic and Automata: History and Perspectives [in Honor of Wolfgang Thomas], Vol. 2 of Texts in Logic and Games, Amsterdam University Press, 2008, pp. 505–530.

[6] S. Maneth, H. Seidl, Checking in polynomial time whether or not a regular tree language is deterministic top-down, Inf. Process. Lett. 184 (2024) 106449.

[7] E. Börger, E. Grädel, Y. Gurevich, The Classical Decision Problem, Perspectives in Mathematical Logic, Springer, 1997.

[8] W. Thomas, Languages, automata, and logic, in: Handbook of Formal Languages, Vol. 3, Beyond Words, Springer, 1997, Ch. 7, pp. 389–455.

[9] J. R. Büchi, Using determinancy of games to eliminate quantifiers, in: Proc. of FCT: Fundamentals of Computation Theory, LNCS 56, Springer, 1977, pp. 367–378.

[10] Y. Gurevich, L. Harrington, Trees, automata, and games, in: Proc. of STOC: Symposium on Theory of Computing, ACM, 1982, pp. 60–65.

[11] W. J. Sakoda, M. Sipser, Nondeterminism and the size of two way finite automata, in: Proc. of STOC: Symposium on Theory of Computing, ACM, 1978, pp. 275–286.

[12] J.-C. Birget, Partial orders on words, minimal elements of regular languages and state complexity, Theor. Comput. Sci. 119 (2) (1993) 267–291.