Tree-Walking Automata Cannot Be Determinized

Mikołaj Bojańczyk* and Thomas Colcombet**

Uniwersytet Warszawski, Wydział MIM, Banacha 2, Warszawa, Poland

Abstract. Tree-walking automata are a natural sequential model for recognizing tree languages. It is shown that deterministic tree-walking automata are weaker than nondeterministic tree-walking automata.

Introduction

A tree-walking automaton (TWA) is a natural type of finite automaton working over trees. The automaton is a finite memory device which walks around a tree, choosing what move to make according to its current state and some information about its current position in the tree. After a certain amount of walking the automaton can choose to accept the tree. Even though TWA were introduced in the early seventies by Aho and Ullman [AU71], very little is known about this model.

This situation is different from the "usual" tree automata – branching tree automata – which are a very well understood object. Both top-down and bottom-up nondeterministic branching tree automata recognize the same class of languages. Languages of this class are called *regular*, the name being so chosen because it enjoys many nice properties of the class of regular word languages. The deterministic variants of branching tree automata are similarly well understood – deterministic bottom-up automata also recognize all regular tree languages, while deterministic top-down automata recognize a strict subclass of the class of regular languages.

It is a classical result that every language recognized by a TWA is regular. However most other fundamental questions pertaining to tree-walking automata remain unanswered:

1. Is every regular language recognized by a TWA?
2. Can TWA be determinized?
3. Is the class of languages recognized by TWA closed under complementation?

It is believed that the answers to all three questions above are negative. There has been much related research, which can be roughly grouped in two categories: nondefinability results for weakened models of tree-walking automata [NS00,Boj03] and definability results for strengthened models of tree-walking automata [KS81,EH99,EHvB99]. The three questions stated above, however, have remained open.

In this paper we answer question 2: we prove that there exists a language which is recognized by a tree-walking automaton, but by no deterministic one.

* Supported by Polish KBN grant No. 4 T11C 042 25.
** Supported by the European Community Research Training Network GAMES.

1 Tree walking automata, patterns and the general idea

In this section we define tree-walking automata, specify our separating language and prove it is recognized by a nondeterministic tree-walking automaton.

Preliminaries

For two integers i and j, we denote by $[i, j]$ the set $\{n : i \leq n \leq j\}$. The trees we deal with in this paper are finite, binary trees labeled by a given finite alphabet Σ. Formally, a *Σ-tree* t is a mapping from $N_t \subseteq \{1, 2\}^*$ to Σ, where N_t is a finite, non-empty, prefix-closed set such that for any $v \in N_t$, $v1 \in N_t$ iff $v2 \in N_t$. Elements of the set N_t are called *nodes* of the tree. We use the set $\text{Types} = \{r, 1, 2\} \times \{l, f\}$ to encode the possible *types* of a node: the first component has the value r for the root, 1 for a left son and 2 for a right one; the second component is l for a leaf or else f for fathers. For $v \in N_t$, let $type_t(v) \in \text{Types}$ denote the type of this node. A *direction* is an element in $[0, 2]$, where informally 0 stands for 'up', 1 for 'down-left' and 2 for 'down-right'. Let

$$d : N_t \times N_t \to [0, 2] \cup \{\perp\}$$

be the function assigning: i to pairs of the form $(v, v{\cdot}i)$, for $i \in \{1, 2\}$; 0 to pairs of the form $(v{\cdot}i, v)$, for $i \in \{1, 2\}$; and $\perp$ otherwise.

Definition 1 A *tree-walking automaton* over Σ-trees is a tuple $\mathcal{A} = (Q, q_I, F, \delta)$, where Q is a finite set of *states*, $q_I \in Q$ is the *initial state*, $F \subseteq Q$ is the set of *accepting states* and δ is the *transition relation* of the form

$$\delta \subseteq Q \times \text{Types} \times \Sigma \times Q \times [0, 2].$$

A *run* of $\mathcal{A}$ over a Σ-tree t is a sequence $(v_0, q_0) \ldots (v_n, q_n)$ satisfying

$$(q_i, type_t(v_i), t(v_i), q_{i+1}, d(v_i, v_{i+1})) \in \delta \quad \text{for all } i \in [0, n-1].$$

A run is *accepting*, if $v_0 = v_n = \varepsilon$, $q_0 = q_I$ and $q_n \in F$. The automaton $\mathcal{A}$ accepts a tree if it has an accepting run over it. A set of Σ-trees L is *recognized* by $\mathcal{A}$ if $\mathcal{A}$ accepts exactly the trees in L. Finally, we say that a tree-walking automaton is *deterministic* if δ is a function from $Q \times \text{Types} \times \Sigma$ to $Q \times [0, 2]$. We use TWA to denote the class of tree languages which are recognized by some TWA and DTWA for languages recognized by some deterministic TWA.

We would like to point out here that reading the type of a node is an essential feature of tree-walking automata. Indeed, Kamimura and Slutzki showed in [KS81] that TWA which do not have access to this information cannot recognize all regular languages, being incapable of even searching a tree in a systematic manner by doing a depth-first search, for instance.

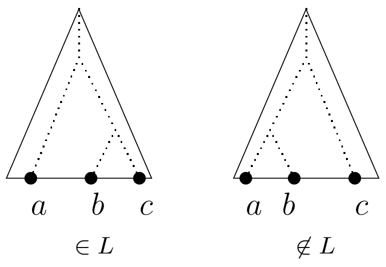

Fig. 1. The two kinds of well-formed trees

The separating language L

In this section we specify our separating language L, which we will eventually prove to witness the inequality DTWA $\subsetneq$ TWA. Also in this section, we present a nondeterministic TWA which recognizes L. A proof that no DTWA can recognize L is more involved and will be spread across the subsequent sections.

The language L involves a very simple kind of trees, which we call *well-formed* trees: $\{B, a, b, c\}$-trees which have all nodes labeled by the *blank symbol* B but for three leaves: one leaf having label a; one leaf having label b and one leaf having label c. Let us simply call a (resp. b, c) the only node labeled by a (resp. by b, by c). Furthermore, in a well-formed tree we require the nodes a, b and c to be lexicographically ordered. One can check that the set of well-formed trees belongs to DTWA.

There are two possible kinds of well-formed trees: ones where the deepest common ancestor of a and b is above c; and ones where the deepest common ancestor of b and c is above a. The language L is the set of well-formed trees of the first kind. This definition is illustrated in Figure 1.

We now proceed to describe a nondeterministic tree-walking automaton which recognizes the language L.

Lemma 1. *There is a nondeterministic TWA which recognizes L.*

Proof. We will only give here an informal description of the automaton. This automaton first checks deterministically that the tree is well-formed, then reaches somehow the node labeled by c. From this node, it goes toward the root and at some point v decides (using nondeterminism) to perform a depth-first search from left to right. It then accepts the tree if in this search the first non-blank node encountered is a b, i.e. the left-most non-blank leaf below v is labeled by b.

One can verify that there exists an accepting run of this automaton if and only if the tree belongs to L. Indeed, when the tree belongs to L the automaton chooses v to be the deepest common ancestor of b and c. On the other hand, if a tree is well-formed but does not belong to L, there is no v ancestor of c such that the left-most non-blank leaf below v is labeled by b (this leaf is either a or c), and thus the automaton does not accept the tree.

2 Patterns

In this section, we introduce the key technical concept of patterns and outline how they can be used to show that L is outside DTWA. From now on we assume that a deterministic tree-walking automaton

$$\mathcal{A} = (Q, q_I, F, \delta)$$

is fixed; our aim is to prove that $\mathcal{A}$ does not recognize the language L.

Patterns and pattern equivalence

A *pattern* Δ is a $\{B, *\}$-tree where the symbol $*$ labels only leaves which are left sons. The i-th $*$-labeled leaf (numbered from left to right) is called the i-th-*port*. *Port* 0 stands for the root. The largest port number is called the *arity* of the pattern, and we use Pat^n to denote the set of n-ary patterns.

Given $\Delta \in \mathrm{Pat}^n$, and n patterns $\Delta_1, \ldots, \Delta_n$, the *composition* $\Delta[\Delta_1, \ldots, \Delta_n]$ is obtained from Δ by simultaneously substituting each Δ_i for the i-th port. We may use $*$ instead of some substituted patterns in a composition, the intended meaning being that the corresponding ports remain untouched. When all Δ_i's are $*$ but for D_k we simply write $\Delta[\Delta_k/k]$. If furthermore Δ is a unary pattern, we write $\Delta \cdot \Delta'$ instead of $\Delta[\Delta'/1]$. Given a set P of patterns, we denote by $\mathcal{C}(P)$ the least set of patterns which contains P and is closed under composition.

Definition 2 The *automaton's transition relation over an n-ary pattern Δ*,

$$\delta_\Delta \subseteq Q \times [0, n] \times Q \times [0, n] \ ,$$

contains a tuple (q, i, r, j) if it is possible for $\mathcal{A}$ to go from state q in port i to state r in port j in Δ. This run is assumed not to visit any port along the way but for the initial and final configurations, in which the ports are treated as having type $(1, f)$ (i.e. non-leaf left sons). In particular the port 0 is not seen as the root and non null ports are not seen as leaves (to make composition work).

From the point of view of the automaton $\mathcal{A}$, the relation δ_Δ sums up all important properties of a pattern and we consider two patterns equivalent if they induce the same relation. More precisely, for two patterns Δ and Δ' of the same arity n, we write

$$\Delta \simeq \Delta' \quad \text{iff} \quad \delta_\Delta = \delta_{\Delta'} \ .$$

The essence of this equivalence is that if one replaces a sub-pattern by an equivalent one, the automaton $\mathcal{A}$ is unable to see the difference. The following lemma shows that $\simeq$ acts as a congruence with respect to pattern composition:

Lemma 2. *For $\Delta_1 \simeq \Delta_1'$ of arity n, $\Delta_2 \simeq \Delta_2'$ and $i \in [1, n]$,*

$$\Delta_1[\Delta_2/i] \simeq \Delta_1'[\Delta_2'/i]$$

A consequence of the determinism of $\mathcal{A}$ is that for any pattern Δ of arity n, the relation δ_Δ is a partial function from $Q \times [0,n]$ to $Q \times [0,n]$ (it may be partial even if the original transition function is not since the automaton can be trapped in a loop). For this reason, we use from now and on a functional notation for δ relations.

Outline of the proof

In order to prove that $\mathcal{A}$ cannot recognize L, we will produce three patterns: a nullary pattern Δ_0, a unary pattern Δ_1 and a binary pattern Δ_2. We then prove that compositions of these patterns satisfy several desirable properties. In particular, we ultimately show that for deterministic automata the following equivalence holds:

$$\Delta_2[*, \Delta_2] \simeq \Delta_2[\Delta_2, *] \ . \tag{1}$$

Having this equivalence, proving that $\mathcal{A}$ does not recognize L becomes a simple matter. Consider a context where a B-labeled tree is plugged for the port 0, and three trees with one a, b and c respectively are plugged into the ports $1, 2, 3$. If we plug the left pattern from (1) into this context, we obtain a tree in L, and if we plug the right pattern, we obtain a tree outside L. However, since the patterns are equivalent, the automaton L cannot distinguish the two resulting trees and will either accept both or reject both, hence $\mathcal{A}$ does not recognize L.

Since the deterministic automaton $\mathcal{A}$ was chosen arbitrarily, it follows that $L \notin$ DTWA. Together with Lemma 1, we obtain this paper's contribution:

Theorem 1. *The class DTWA is strictly included in the class TWA.*

What remains to be done is to construct the patterns Δ_0, Δ_1 and Δ_2, which we do in Section 3; and then study properties of those patterns using the determinism of $\mathcal{A}$, which we do in Section 4. The culmination of this study is Corollary 4, from which the key equivalence (1) follows.

3 Basic patterns

In this section, we define the patterns Δ_0, Δ_1 and Δ_2 and prove a basic property related to their composition (Lemma 4 and 5). Before we do this, we need to first a simple result concerning finite semigroups.

In order to define the patterns Δ_0, Δ_1, Δ_2 we need to state first a classical result concerning semigroups. Let us recall that a *semigroup* is a set together with an associative binary operation, which we write multiplicatively here.

Lemma 3. *For every finite semigroup S and any $u, v \in S$, there exist $u', v' \in S$ such that the elements $U = u{\cdot}u'$ and $V = v{\cdot}v'$ satisfy the following equations:*

$$U = U{\cdot}U = U{\cdot}V \qquad \textit{and} \qquad V = V{\cdot}U = V{\cdot}V \ .$$

Let us now describe the construction of the patterns Δ_0, Δ_1, Δ_2 and prove Lemma 4. The insightful reader will notice that the determinism of $\mathcal{A}$ is not used in this part of the proof.

Let us denote by B_k the full binary tree of depth k. As the pattern equivalence relation $\simeq$ is of finite index, there exists m, n such that $m + 1 < n$ and $B_m \simeq B_n$. Let Δ_0 be B_m. In the tree B_n, the tree Δ_0 appears at least twice as a subtree rooted in a left son, thus there exists a binary pattern Δ_X such that $\Delta_X[\Delta_0, \Delta_0] = B_n$. Consider now the following two unary patterns:

$$\Delta_u = \Delta_X[*, \Delta_0] \quad \text{and} \quad \Delta_v = \Delta_X[\Delta_0, *] \ .$$

Let S be the semigroup whose elements are patterns in $\mathcal{C}(\{\Delta_u, \Delta_v\})$ and where the multiplication operation is the composition of unary patterns. Since S is a finite semigroup (modulo $\simeq$), there exist, by Lemma 3, unary patterns $\Delta_{u'}$ and $\Delta_{v'}$ in $\mathcal{C}(\{\Delta_u, \Delta_v\})$ such that the two patterns $\Delta_U = \Delta_u \cdot \Delta_{u'}$ and $\Delta_V = \Delta_v \cdot \Delta_{v'}$ satisfy the following equivalences:

$$\Delta_U \simeq \Delta_U \cdot \Delta_U \simeq \Delta_U \cdot \Delta_V \quad \text{and} \quad \Delta_V \simeq \Delta_V \cdot \Delta_U \simeq \Delta_V \cdot \Delta_V \ .$$

Let us define now Δ_1 to be Δ_U and Δ_2 to be $\Delta_1 \cdot \Delta_X[\Delta_{u'} \cdot \Delta_1, \Delta_{v'} \cdot \Delta_1]$. Finally, let $\mathcal{C}_\mathcal{A}$ stand for the set $\mathcal{C}(\{\Delta_0, \Delta_1, \Delta_2\})$ and let $\mathcal{C}^n_\mathcal{A}$ stand for $\mathcal{C}_\mathcal{A} \cap \mathrm{Pat}^n$. The following lemma shows that, from the point of view of the automaton $\mathcal{A}$, all patterns of a given small arity in $\mathcal{C}_\mathcal{A}$ look the same:

Lemma 4. *For all $k \in [0, 2]$ and all $\Delta \in \mathcal{C}^k_\mathcal{A}$, $\Delta \simeq \Delta_k$.*

Proof. Let us establish first the three following equivalences:

- $\Delta_1 \cdot \Delta_0 \simeq \Delta_0$. It is enough to prove by a simple induction that for all patterns Δ in $\mathcal{C}(\{\Delta_u, \Delta_v\})$, $\Delta \cdot \Delta_0 \simeq \Delta_0$.
- $\Delta_1 \cdot \Delta \simeq \Delta \simeq \Delta[\Delta_1/i]$ for all n-ary patterns $\Delta \in \mathcal{C}_\mathcal{A}$ and $i \in [1, n]$. This follows from the equivalence

$$\Delta_1 = \Delta_U \simeq \Delta_U \cdot \Delta_U = \Delta_1 \cdot \Delta_1$$

  and the definition of Δ_2, where the pattern Δ_1 appears next to every port.
- $\Delta_2[\Delta_0, *] \simeq \Delta_2[*, \Delta_0] \simeq \Delta_1$. By symmetry, we only prove one equivalence:

$$\begin{aligned} \Delta_2[\Delta_0, *] &= \Delta_1 \cdot \Delta_X[\Delta_{u'} \cdot \Delta_1 \cdot \Delta_0, \Delta_{v'} \cdot \Delta_1] \\ &\simeq \Delta_1 \cdot \Delta_X[\Delta_0, \Delta_{v'} \cdot \Delta_1] \simeq \Delta_U \cdot \Delta_V \cdot \Delta_U \simeq \Delta_U = \Delta_1 \ . \end{aligned}$$

Note now that every pattern in $\mathcal{C}_\mathcal{A}$ of arity in $[0, 2]$ is either one of $\Delta_0, \Delta_1, \Delta_2$ or is a composition in which only patterns of arity no bigger than 2 are involved. The lemma is then established by using an induction driven by this decomposition, where each step corresponds to one of the equivalences above.

As an application of Lemma 4, we conclude this section by a description of runs that start and end in the same port of a pattern:

Lemma 5. *For $n \geq 1$, all patterns $\Delta \in \mathcal{C}^n_{\mathcal{A}}$, all states q, r and all $i \in [1,n]$:*

$$\begin{aligned} \delta_\Delta(q,0) = (r,0) \quad &\textit{iff} \quad \delta_{\Delta_1}(q,0) = (r,0) \ , \\ \delta_\Delta(q,i) = (r,i) \quad &\textit{iff} \quad \delta_{\Delta_1}(q,1) = (r,1) \ . \end{aligned}$$

Proof. The right to left implications follow from the fact that for all $\Delta \in \mathcal{C}^n_{\mathcal{A}}$ and all $i \in [1,n]$, the following equivalence holds (Lemma 4):

$$\Delta \simeq \Delta_1 \cdot \Delta \simeq \Delta[\Delta_1/i] \ .$$

The left to right implications follow from the fact that for all $\Delta \in \mathcal{C}^n_{\mathcal{A}}$ and all $i \in [1,n]$, the following equivalence holds (Lemma 4):

$$\Delta_1 \simeq \Delta[\overbrace{\Delta_0, \ldots, \Delta_0}^{i-1 \text{ times}}, *, \overbrace{\Delta_0, \ldots, \Delta_0}^{n-i \text{ times}}] \ .$$

4 Swinging removal

From now on, we will be using the fact that the automaton $\mathcal{A}$ is deterministic. We start by hiding the case when the automaton "swings", i.e. enters and exits a pattern of nonzero arity by the same port. Technically, we replace the δ functions with a higher-level construct γ, which can be considered as equivalent to δ, but furthermore has several desirable extra properties (Lemmas 7, 9 and 10).

Consider a unary pattern of the form $\Delta_1 \cdot \Delta_1$, with the nodes $v < w$ corresponding to the 1-ports of the two component Δ_1 patterns. For a state q, consider the unique maximal run of $\mathcal{A}$ which starts in (q,v) and visits neither the root nor w. If this run is finite, we call the state r in which node v is last visited the ***ε-successor*** $s_\varepsilon(q)$ of q, else $s_\varepsilon(q)$ is undefined.

We say q is an ***upward*** state if it can appear *after* $\mathcal{A}$ has traversed a pattern Δ_1 in the up direction, i. e. for some state r, $\delta_{\Delta_1}(r,1) = (q,0)$ holds. Similarly we define a ***downward*** state by swapping the role of ports 0 and 1. We use Q_U and Q_D to denote the sets of upward and downward states respectively.

We now introduce a new type of function which we will use instead of the δ functions. This replacement aims at eliminating the hassle involved with swinging. For $\Delta \in \mathcal{C}^n_{\mathcal{A}}$, the partial function

$$\gamma_\Delta : Q_D \times \{0\} \cup Q_U \times [1,n] \to Q_U \times \{0\} \cup Q_D \times [1,n]$$

is defined as $\gamma_\Delta(q,i) = \delta_\Delta(s_\varepsilon(q),i)$. From now on, we simplify slightly the notations by using γ_0, γ_1 and γ_2 for respectively γ_{Δ_0}, γ_{Δ_1} and γ_{Δ_2}.

We remark here, somewhat ahead of time, that the function γ_Δ is turns out to be completely defined for $\Delta \in \mathcal{C}_{\mathcal{A}}$. This because – thanks to the choice of the function's domain – we can be sure that the automaton does not loop. The intuitive reason for this is that an upward (or downward) state has already "survived" the traversal of a Δ_1 pattern, and it will not loop without good reason. The formal proofs are in Lemma 6 for patterns of nonzero arity and in Lemma 9 for the pattern Δ_0.

Lemma 6. *For any $q \in Q_D$, $\gamma_1(q,0) = (q,1)$. For any $q \in Q_U$, $\gamma_1(q,1) = (q,0)$.*

Proof. Let $q \in Q_D$. By definition, there is some r such that $\delta_{\Delta_1}(r,0) = (q,1)$. Consider the pattern $\Delta_1 \cdot \Delta_1$, with v labeling the interface between the two Δ_1 patterns and a run on this pattern which starts in $(r,0)$. The first time the node v is passed, state q is assumed, since $\delta_{\Delta_1}(r,0) = (q,1)$. The last time v is passed state $s_\varepsilon(q)$ is assumed, by definition of s_ε. Since $\Delta_1 \simeq \Delta_1 \cdot \Delta_1$, the run of the automaton starting with state r in port 0 of $\Delta_1 \cdot \Delta_1$ must end with state q in port 1, the last traversal of the lower Δ_1 pattern going downward from $s_\varepsilon(q)$ in v to q in port 1. Hence $\gamma_1(q,0) = \delta_{\Delta_1}(s_\varepsilon(q),0) = (q,1)$. The proof for $q \in Q_U$ is obtained by swapping the roles of ports 0 and 1.

The following lemma, which follows straight from the definition of the γ function, shows why γ allows us to ignore swinging:

Lemma 7. *For any $\Delta \in \mathcal{C}_\mathcal{A}$ of arity $n \geq 1$, states q, r and any port $i \in [0,n]$, if $\gamma_\Delta(q,i) = (r,j)$ then $i \neq j$.*

Proof. We only do the case where q is a downward state and hence $i = 0$. Let $q' = s_\varepsilon(q)$. Assume that the lemma is not true, hence $j = 0$. This means that $\delta_\Delta(q',0) = (r,0)$. By Lemma 5, $\delta_{\Delta_1}(q',0) = (r,0)$ and hence $\gamma_1(q,0) = (r,0)$, a contradiction with Lemma 6.

We start with a simple description of the behaviour of downward states:

Lemma 8. *For $q \in Q_D$, either $\gamma_2(q,0) = (q,1)$ or $\gamma_2(q,0) = (q,2)$ holds.*

Proof. Let $\gamma_2(q,0) = (r,i)$. By Lemma 7 , $i \in \{1,2\}$. Let us assume, without lessening of generality, that $i = 1$. Since $\Delta_1 \simeq \Delta_2[*,\Delta_0]$, we obtain that $\gamma_1(q,0) = (r,1)$. Hence it must be that $r = q$, since $\gamma_1(q,0) = (q,1)$ holds by Lemma 6.

Lemma 9. *For all $\Delta \in \mathcal{C}_\mathcal{A}$, the partial function γ_Δ is completely defined.*

Proof. Consider a pattern Δ of nonzero arity n and assume that $\gamma_\Delta(q,i)$ is undefined for some $i \in [0,n]$. Let us consider first the case when $i \neq 0$. If we plug all the ports of Δ but i with a Δ_0 pattern, we get a pattern equivalent to Δ_1. But this would imply that $\gamma_1(q,i)$ is undefined, a contradiction with Lemma 6. The case of $i = 0$ is proved analogously.

For the case of $\Delta = \Delta_0$, let us assume for a moment that when entering from $(q,0)$, the automaton gets lost in the pattern. But then, since $\Delta_1 \simeq \Delta_2[*,\Delta_0] \simeq \Delta_2[\Delta_0,*]$ and by Lemma 8, $\mathcal{A}$ would also get lost in Δ_1 when entering from $(q,0)$, a contradiction with Lemma 6.

The following lemma shows that to establish the equivalence of two patterns, it is enough to study the γ functions.

Lemma 10. *For $n \geq 0$ and all $\Delta, \Delta' \in \mathcal{C}^n_\mathcal{A}$, $\Delta \simeq \Delta'$ if and only if $\gamma_\Delta = \gamma_{\Delta'}$.*

Proof. The left to right implication is straightforward and, in fact, true for any pattern. The right to left implication is more involved. It is known for arities up to two from Lemma 4. Let us consider two patterns Δ and Δ' of arity at least one such that $\gamma_\Delta = \gamma_{\Delta'}$, and a state q. Three cases have to be studied.

- If $\delta_{\Delta_1}(q, 0)$ is undefined then so is $\delta_\Delta(q, 0)$, since $\Delta_1 \cdot \Delta \simeq \Delta$. The same goes for $\delta_{\Delta'}(q, 0)$.
- If $\delta_{\Delta_1}(q, 0) = (r, 0)$ for some r, by Lemma 5, $\delta_\Delta(q, 0) = \delta_{\Delta'}(q, 0) = (r, 0)$.
- Otherwise $\delta_{\Delta_1}(q, 0) = (r, 1)$ for some r. As Δ_1 is equivalent to Δ in which all nonzero ports but one are replaced by Δ_0, we obtain that $\delta_\Delta(q, 0)$ is defined. Let (r', i) be this value. According to Lemma 5, $i \neq 0$. Let us consider now the run of the automaton in pattern $\Delta_1 \cdot \Delta \simeq \Delta$. It crosses first the junction point with state $\delta_{\Delta_1}(q, 0) = (r, 1)$, then after some walk, reaches the same node with state $s_\varepsilon(r)$. Finally it crosses the Δ pattern and reaches port i in state r', which means $\delta_\Delta(r, 0) = (r', i)$. We obtain that

  $$\delta_\Delta(q, 0) = \delta_\Delta(s_\varepsilon(r), 0) = \gamma_\Delta(r, 0) \ .$$

  Similarly $\delta_{\Delta'}(q, 0) = \gamma_{\Delta'}(r, 0)$, and hence $\delta_\Delta(q, 0) = \delta_{\Delta'}(q, 0)$.

The same method can be applied for ports in $[1, n]$.

From now on, the γ function is used as if it was the original δ function, in particular with respect to composition.

5 Generic behaviours

In this last part we show that, essentially, $\mathcal{A}$ can only do depth-first searches over patterns in $\mathcal{C}_\mathcal{A}$. Using this characterization, we prove the main technical lemma of this paper, Lemma 12. This characterization is obtained by analyzing the γ functions. Due to the domain of γ we need to consider two cases: downward states in port 0 and upward states in the other ports.

A good understanding of the behaviour of downward states in patterns from $\mathcal{C}_\mathcal{A}$ comes from Lemma 6: if a downward state q starts in port 0 of a pattern $\Delta \in \mathcal{C}_\mathcal{A}$, it will emerge in state q either in the leftmost or the rightmost nonzero port.

The description of the behaviour of upward states is more involved. When starting in a nonzero port, an upward state may go in the direction of the root, but it may also try to visit a neighboring nonzero port (something that does not appear in the Δ_1 pattern used to define upward states). The following definition, along with Lemma 11, gives a classification of the possible behaviours of upward states:

Definition 3 We say that a pair of states $(q, r) \in Q_D \times Q_U$ has *right to left depth-first search behaviour* if

$$\gamma_2(q, 0) = (q, 2), \quad \gamma_0(q, 0) = (r, 0), \quad \gamma_2(r, 2) = (q, 1), \quad \text{and } \gamma_2(r, 1) = (r, 0).$$

A *left to right depth-first search behaviour* is defined symmetrically by swapping the roles of ports 1 and 2. An upward state r has *ascending behaviour* if

$$\gamma_2(r,1) = \gamma_2(r,2) = (r,0) \ .$$

The following lemma shows that Definition 3 is exhaustive:

Lemma 11. *For any upward state r, either r has ascending behaviour, or there exists a downward state q such that the pair (q,r) has either right to left or left to right depth-first search behaviour.*

Proof. Let r be an upward state. We first show that either $\gamma_2(r,1) = (r,0)$ or $\gamma_2(r,2) = (r,0)$. Let us suppose that $\gamma_2(r,1) \neq (r,0)$ must hold. By Lemma 7 we have $\gamma_2(r,1) = (q,2)$ for some downward state q. Let r' be the downward state such that $\gamma_0(q,0) = (r',0)$. Since $\Delta_1 \simeq \Delta_2[*,\Delta_0]$ and $\gamma_1(r,1) = (r,0)$ (Lemma 6), we obtain that $\gamma_2(r',2) = (r,0)$, and hence $\gamma_1(r',1) = (r,0)$. Since $\gamma_1(r',1) = (r',0)$, we obtain $r = r'$ and $\gamma_2(r,2) = (r,0)$. Thus, either $\gamma_2(r,1) = (r,0)$ or $\gamma_2(r,2) = (r,0)$.

If both cases hold, then r has ascending behaviour. Otherwise exactly one case holds, without lessening of generality let us assume it is:

$$\gamma_2(r,2) = (r,0), \quad \text{and} \quad \gamma_2(r,1) \neq (r,0). \tag{2}$$

As in the reasoning above, let q be the state such that

$$\gamma_2(r,1) = (q,2) \ . \tag{3}$$

We claim that (q,r) has left to right depth-first search behaviour. As we have seen before,

$$\gamma_0(q,0) = (r,0) \ . \tag{4}$$

Since we already have the equations (2), (3) and (4), to establish that the pair (q,r) has left to right depth-first search behaviour we only need to prove the equation (5).

Since q is a downward state, then by Lemma 8 the value of $\gamma_2(q,0)$ must be either $(q,1)$ or $(q,2)$. But the second case cannot hold, since together with equations (2) and (4) this would give us $\gamma_{\Delta_2[*,\Delta_0]}(q,0) = (r,0)$, a contradiction with Lemma 7. This means that

$$\gamma_2(q,0) = (q,1) \ . \tag{5}$$

and, hence, (q,r) has left to right depth-first search behaviour. The right to left depth-first search behaviour case is obtained by a symmetric argument when the roles of ports 1 and 2 are exchanged in the assumptions (2).

Now that we know exactly how the automaton behaves for upward and downward states, we obtain the following as a simple consequence of Lemmas 8 and 11, none of whose described behaviours make it possible to distinguish patterns of the same arity:

Lemma 12. *For all n and all $\Delta, \Delta' \in \mathcal{C}^n_{\mathcal{A}}$, the functions γ_Δ and $\gamma_{\Delta'}$ are equal.*

Proof. The statement of the lemma follows from the following characterization of moves over an arbitrary pattern $\Delta \in \mathcal{C}_{\mathcal{A}}$ of arity $n \geq 1$:

- For any downward state q, by Lemma 8, two cases may happen:
  - If $\gamma_1(q,0) = (q,1)$ then $\gamma_\Delta(q,0) = (q,1)$.
  - If $\gamma_1(q,0) = (q,2)$ then $\gamma_\Delta(q,0) = (q,n)$.
- If q is an upward state then, by Lemma 11, three cases may happen:
  - If the state q has ascending behaviour, $\gamma_\Delta(q,i) = (q,0)$ for all $i \in [1,n]$.
  - If for some downward state r, the pair (q,r) has right to left depth-firt-search behaviour, then $\gamma_\Delta(q,i) = (r,i+1)$ for all $i \in [1,n-1]$ and $\gamma_\Delta(q,n) = (q,0)$.
  - If for some downward state r, the pair (q,r) has right to left depth-firt-search behaviour, then $\gamma_\Delta(q,0) = (q,0)$ and $\gamma_\Delta(q,i) = (r,i-1)$ for all $i \in [2,n]$.

The above lemma, together with Lemma 10, gives us the required:

Corollary 4 The equivalence $\Delta_2[\Delta_2,*] \simeq \Delta_2[*,\Delta_2]$ holds.

Acknowledgment

We would like to thank C. Löding and A. Meyer for reading previous versions of this paper.

References

[AU71] A. V. Aho and J. D. Ullman. Translations on a context-free grammar. *Information and Control*, 19(5):439–475, dec 1971.

[Boj03] M. Bojańczyk. 1-bounded TWA cannot be determinized. In *FSTTCS 2003: Foundations of Software Technology and Theoretical Computer Science, 23rd Conference, Mumbai, India, December 15-17, 2003, Proceedings*, volume 2914 of *Lecture Notes in Computer Science*, pages 62,73. Springer, 2003.

[EH99] J. Engelfriet and H. J. Hoogeboom. Tree-walking pebble automata. In G. Paum J. Karhumaki, H. Maurer and G. Rozenberg, editors, *Jewels are forever, contributions to Theoretical Computer Science in honor of Arto Salomaa*, pages 72–83. Springer-Verlag, 1999.

[EHvB99] J. Engelfriet, H. J. Hoogeboom, and J.-P. van Best. Trips on trees. *Acta Cybernetica*, 14:51–64, 1999.

[KS81] T. Kamimura and G. Slutzki. Parallel two-way automata on directed ordered acyclic graphs. *Information and Control*, 49(1):10–51, 1981.

[NS00] F. Neven and T. Schwentick. On the power of tree-walking automata. In *Automata, Languages and Programming, 27th International Colloquium, ICALP 2000*, volume 1853 of *LNCS*, 2000.