TREE-WALKING AUTOMATA DO NOT RECOGNIZE ALL REGULAR LANGUAGES

MIKOŁAJ BOJAŃCZYK[†§] AND THOMAS COLCOMBET[‡§]

Abstract. Tree-walking automata are a natural sequential model for recognizing tree languages. It is well known that every tree language recognized by a tree-walking automaton is regular. We show that the converse does not hold.

1. Introduction. A tree-walking automaton is a natural type of finite automaton for trees. At every moment in a run, a tree-walking automaton is located in one node of the tree. In one step, the automaton moves to a neighboring node and changes its state according to the transition relation. The step depends on the current state of the automaton and some local information: the label of the current node, whether or not it is the root, a leaf, a left child or a right child. The tree is accepted if one of the accepting states is reached. For instance, a tree-walking automaton can check if all nodes of the tree have the same label by doing a depth-first search.

Even though tree-walking automata were introduced in the early seventies by Aho and Ullman [1], not much is known about this model.

This situation is different from the "usual" tree automata – branching tree automata – which are a well-understood object. In particular top-down, bottom-up and two-way nondeterministic branching tree automata recognize the same class of tree languages. The tree languages of this class are called *regular*, the name being so chosen because the class enjoys many nice properties of regular word languages. A comprehensive introduction to the standard theory of tree automata can be found in [4].

As tree-walking automata are a particular case of two-way branching automata, tree-walking automata recognize regular tree languages. Closure under union and intersection is also simple. Until recently, however, other fundamental questions pertaining to tree-walking automata remained unanswered:

1. Is every regular tree language recognized by a tree-walking automaton?
2. Can tree-walking automata be determinized?
3. Are tree-walking automata closed under complementation?

Much research has been dedicated to tree-walking automata. There are nondefinability results for weakened models of tree-walking automata [9, 11, 2], as well as definability results for strengthened models of tree-walking automata [9, 5, 7]. A line of research is dedicated to logical characterizations of tree-walking automata [11] and their pebble extensions [6]. There has also been some research on tree-walking automata with an output tape — which define tree-to-word transductions — [1] and to expressiveness issues concerning this model [8].

Question 2 has been answered negatively in [3]. Question 3 is still open, the only known result being closure under complementation of deterministic tree-walking automata [10]. The contribution of this paper is to give a negative answer to the first question.

[†]Work undertaken in Liafa (Paris, France) and Warsaw University (Poland).

[‡]Work undertaken in Warsaw University (Poland) and Irisa-CNRS (Rennes, France).

[§]Both authors were partially supported by the EU Research training network "*Games and Automata for Synthesis and Validation*".

2. Preliminaries and the separating tree language. In this section we define the basic concepts and state our main result.

2.1. Basic definitions. The trees in this paper are finite, binary trees labeled by a given finite alphabet Σ. A Σ-*tree* is a partial mapping $t : \{0,1\}^* \to \Sigma$ of finite, nonempty and prefix-closed domain $dom(t)$. Elements of this domain are called *nodes* of the tree; the *label* of a node u is the value $t(u)$. Additionally we assume that if $v0$ is a node of the tree, then so is $v1$, and vice versa. Nodes are partially ordered by the prefix relation; when a node x is prefix of a node y, we say that x is *above* y, or y is *below* x. The least node ε is called the *root*, maximal nodes are called *leaves*. The nodes are also ordered lexicographically; we say that x is *to the left* (resp. *right*) of y if x and y are incomparable by the prefix relation, and x is lexicographically before y (resp. after y). Given a node u, the *subtree of t rooted in u* — we simply say the *subtree of u* when the tree t is clear from the context — is the Σ-tree t' of domain $\{v : uv \in dom(t)\}$ defined for all nodes v of t' by $t'(v) = t(uv)$. The *depth of a node u* is $|u| + 1$, where $|u|$ is the length of u as a word. The *depth of a tree* is the maximal depth of its nodes. A *balanced tree* is one where all leaves are at the same depth.

A set of trees over a given alphabet is called a *tree language*. A *regular* tree language is a tree language recognized by a bottom-up branching tree automaton. We assume the reader to be familiar with branching automata; see [4] for further reading. We denote by REG the class of regular tree languages.

We now define (nondeterministic) tree-walking automata. The *type* of a node says whether the node is a leaf and whether it is the root. There are four possible types; we denote the set of types by Types.

DEFINITION 1. *A* tree-walking automaton *is a tuple $\mathcal{A} = (Q, \Sigma, I, F, \delta)$, where Q is a finite set of* states*, $I, F \subseteq Q$ are respectively the sets of* initial *and* accepting *states, and δ is the* transition relation *of the form*

$$\delta \subseteq (Q \times \text{Types} \times \Sigma \times \{\varepsilon, 0, 1\})^2 \ .$$

A *configuration* is a pair of a state and a node. The automaton can go in one step from a configuration (q_1, v_1) to a configuration (q_2, v_2) if δ contains a transition

$$(q_1, t_1, a_1, d_1, q_2, t_2, a_2, d_2)$$

such that the type and label of v_i are t_i, a_i and there is a node u such that $v_i = u \cdot d_i$ for $i = 1, 2$. A *run* is a nonempty sequence of configurations $c_1, \ldots, c_n$ in which every two consecutive configurations are consistent with the transition relation. We say that such a run is *from c_1 to c_n*; if both configurations c_1 and c_n are located at the same node u, then the run is called a *loop in u*. A run is *accepting* if it starts and ends in the root of the tree, the first state is initial and the last state is accepting. The automaton *accepts* a Σ-tree if it has an accepting run in that tree. The set of Σ-trees accepted is called the tree language *recognized* by the automaton. We use TWA to denote the class of tree languages recognized by some tree-walking automaton.

The reader may be surprised by our definition of tree-walking automata. In other texts, the transition relation is of the form

$$\delta \subseteq Q \times \{\text{root, left child, right child}\} \times \{\text{leaf, nonleaf}\} \times \Sigma \times Q \times \{\uparrow, 0, 1, \epsilon\}$$

In this definition – which can easily be shown equivalent to the one we use – the second coordinate extends Types by saying if a node is a left child / right child, while the $\{\uparrow, 0, 1, \epsilon\}$ causes the automaton to move to the parent, left child, or right child of the current node (or not move at all). The point of using our slightly more verbose definition is that it allows to easily define "reversed" automata, which visit the tree in a chronologically (or spatially) opposite manner. We will comment on these reversed automata in the next section.

We would like to point out here that testing whether a node is a left or right child is an essential feature of a tree-walking automaton. Indeed, Kamimura and Slutzki show in [9] that tree-walking automata that do not have access to this information cannot even test if all nodes in a tree have the same label.

Symmetry principles. As pointed out before, our definition of tree-walking automata — in particular in their nondeterministic form — easily adapts itself to symmetry arguments, which are very convenient in the proofs. There are two symmetries: *time symmetry* and *space symmetry*. Their formal definition is in terms of transformations of a tree-walking automaton; namely each automaton has a time-reversed and a space-reversed variant. The *time-reversed automaton of* $\mathcal{A}$, denoted $\mathcal{A}^{-T}$, is obtained from $\mathcal{A}$ be replacing each transition

$$(q_1, t_1, a_1, d_1, q_2, t_2, a_2, d_2) \in \delta$$

with the transition

$$(q_2, t_2, a_2, d_2, q_1, t_1, a_1, d_1) \ .$$

On the other hand, the *space-reversed automaton of* $\mathcal{A}$, denoted $\mathcal{A}^{-S}$, is obtained by replacing each transition

$$(q_1, t_1, a_1, d_1, q_2, t_2, a_2, d_2) \in \delta$$

with the transition

$$(q_1, t_1, a_1, s(d_1), q_2, t_2, a_2, s(d_2)) \ ,$$

where $s(\varepsilon) = \varepsilon$, $s(0) = 1$ and $s(1) = 0$. One can easily see that

$$(\mathcal{A}^{-S})^{-T} = (\mathcal{A}^{-T})^{-S} \ .$$

We extend the space symmetry $s : \{\varepsilon, 0, 1\} \to \{\varepsilon, 0, 1\}$ mapping to nodes ($s : \{0,1\}^* \to \{0,1\}^*$) and trees in the natural manner. The following obvious fact encapsulates the properties of the reversed automata:

FACT 2.1. *Let* $(q_1, v_1), \ldots, (q_n, v_n)$ *be a run of* $\mathcal{A}$ *in a tree* t.
- $(q_n, v_n), \ldots, (q_1, v_1)$ *is a run of* $\mathcal{A}^{-T}$ *in* t.
- $(q_1, s(v_1)), \ldots, (q_n, s(v_n))$ *is a run of* $\mathcal{A}^{-S}$ *in* $s(t)$.

We say that an automaton is *isomorphic* to another if there exists a one-to-one mapping f from the states of the first one to the states of the second such that there exists a transition

$$(q_1, t_1, a_1, d_1, q_2, t_2, a_2, d_2)$$

in the first automaton iff there exists a transition

$$(f(q_1), t_1, a_1, d_1, f(q_2), t_2, a_2, d_2)$$

in the second automaton. In particular, the isomorphism notion does not involve the initial nor the accepting states of the automata.

We say an automaton is *self-symmetric* if it is isomorphic to its time-reversed and also to its space-reversed automaton. By adding (unreachable) states to an automaton, any tree-walking automaton can be made self-symmetric. In the sequel, self-symmetric automata will be very convenient in reducing the number of cases to be analyzed. Assume for instance, that we have proved a statement of the form: "for every two states p, q, any run from p in the root to q in a leaf can be transformed into one without loops". If the automaton for which we proved this statement was self-symmetric, we would also get the following statement for free: "for every two states p, q, any run from q in a leaf to p in the root can be transformed into one without loops". This is because the time-reversal of a run from the root to a leaf is a run from a leaf to the root, and being loop-free is invariant under time-reversals.

2.2. The Separating Language. As mentioned in the introduction, it is well known that all tree languages recognized by tree-walking automata are regular:

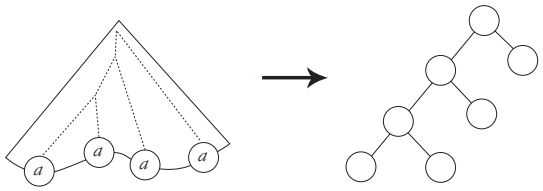
$$\mathrm{TWA} \subseteq \mathrm{REG} \ . \tag{2.1}$$

It has been long open whether this inclusion is strict. Engelfriet, Hoogeboom and Van Best conjectured that this is indeed the case [7]. The subject of this paper is to establish this conjecture.

In this section we describe a tree language L that is regular but not accepted by any tree-walking automaton, and therefore witnesses the strictness of the inclusion (2.1).

We consider trees with two possible labels: **a** and **b**. Moreover, **a** is only allowed in the leaves. We sometimes refer to the symbol **b** as the *blank symbol*. Trees containing only the blank symbol are called *blank trees*. In a blank tree, only the set of nodes is important.

Let t be a nonblank tree with **a** occurring only in the leaves. We will define now the *branching structure* $bs(t)$ of t, which is a blank tree (since only the nodes of $bs(t)$, and not their labels, are relevant). We say a node u is a *branching node* of t if it is either an **a**-labeled leaf of t, or if both left and right successors of u have **a**-labeled leaves in their subtrees. We define the *branching structure* of t as the (unique) blank tree $bs(t)$ such that there is a bijection between the branching nodes of t and the nodes of $bs(t)$ that preserves the prefix relation and the lexicographic ordering of nodes. The following drawing illustrates this definition on an example:



Let K be the set of blank trees where all leaves are at even depth. The separating tree language L mentioned at the beginning of this section is $bs^{-1}(K)$, i.e., the set

of trees whose branching structures have all leaves at even depth. We now state the main result of this paper:

THEOREM 2. *The tree language L is regular, but is not recognized by any tree-walking automaton.*

Showing that L is regular is not difficult: it is recognized by a bottom-up deterministic automaton with three states recognizing respectively the set of blank trees, the set of trees whose branching structure has only branches of even length, and the set of trees whose branching structure has only branches of odd length. For completeness, a fourth "error" state can be added.

Fact 2.2 below shows that a stronger result holds:

FACT 2.2. *The tree language L is definable in first-order logic with the prefix relation and the left and right successor relations.*

Proof. First note that there is a first-order logic formula with one free variable, which is true in exactly the branching nodes of a tree. Therefore, the nodes of the branching structure $bs(t)$ can be interpreted as the branching nodes of t. (Note that for a node interpreted by v, its left successor is interpreted as "first branching node lexicographically after or at the left successor of v", similarly for the right successor.) It follows that if K is a property of branching structures defined by a first-order formula φ, then $bs^{-1}(K)$ is also defined by a first-order formula. The first-order formula for $bs^{-1}(K)$ is obtained from φ by restricting quantification to branching nodes, and replacing the left/right successors by their above-mentioned interpretations (the prefix relation is not changed).

Therefore, the statement of the fact will follow if we establish that the language K is definable in first-order logic. This latter result is Lemma 5.1.8 in [12]. For completeness, we provide a proof for it below.

The main idea is that first-order logic can express whether a leaf in $(01)^*(\varepsilon + 0)$ is at even depth or not. We will refer to such a leaf as the *middle leaf* of the tree. A first-order formula can detect the middle leaf by checking that each node above it is either the leaf itself, the father of the leaf, the right child of a left child, the left child of a right child, the left child of the root, or the root itself. The *middle parity* of a tree is defined to be the parity of the depth of the middle leaf; it is definable in first-order logic since the middle leaf is at even depth if and only if it is a left child. The *middle parity* of a node is defined to be the middle parity of the subtree rooted at this node.

Let M be the set of trees whose middle parity is even, and for which all children of any internal node have the same middle parity. We claim that $K = M$. According to the previous remarks, this implies that K is definable in first-order logic.

The inclusion $K \subseteq M$ is obvious. For the other direction, let t be a tree outside K. If all leaves in t have the same depth parity, then the middle parity is odd and $t \notin M$. Otherwise, consider a node in t of maximal depth whose subtree has leaves of both even and odd depth. But then by maximality, the middle parities of this node's children must be different and $t \notin M$. □

The hard part in the proof of Theorem 2 remains: we need to show that the tree language L is not recognized by any tree-walking automaton. The remainder of this paper is devoted to proving this result.

In the next section, we define *patterns*; these are the same as used in [3]. A pattern is a particular type of tree with distinguished nodes, called ports. As in [3],

we consider three particular patterns (the *basic patterns*) that confuse a tree-walking automaton. Then, in Section 4, to every blank tree t we associate a tree made out of basic patterns (its *pattern expansion*) whose branching structure – where ports are considered as **a**-leaves – is t and is confusing for the automaton. We then study throughout Sections 5 and 6 the possible runs of the automaton in expansions, and their images in the original tree t. This study results in a precise understanding of the behavior of the automaton in expansions: it can only perform a fixed number of simple behaviors – such as left-to-right depth-first search, or move back to the root, or nondeterministic search of a leaf, etc – and can only (nondeterministically) switch between these behaviors a bounded number of times. In Section 7, we use this knowledge about the tree-walking automaton for proving that it cannot recognize L. In particular, we show that a simple local transformation, called the *rotation*, applied to a sufficiently big tree cannot be detected by the automaton, while it transforms a tree in the language into one which is not.

A more detailed overview of the proof is found in Section 4.2, after the concepts of pattern and pattern expansion have been introduced.

We fix for the remainder of the paper a tree-walking automaton

$$\mathcal{A} = (Q, \{\mathbf{a}, \mathbf{b}\}, I, F, \delta) \ .$$

Eventually, we will show that $\mathcal{A}$ cannot recognize the tree language L. As noted in the section on symmetry, we assume without loss of generality that the automaton $\mathcal{A}$ is self-symmetric. We also assume without loss of generality that the automaton has at least two states.

3. Patterns. In this section we define patterns, develop a pumping argument for them, and then study its consequences for the automaton.

Patterns are fragments of trees with holes (called ports) in them. There are two types of ports: leaf ports, which are in the leaves, and the root, which is also called a port. Patterns can be assembled by gluing the root port of one pattern to a leaf port of another pattern. A tree-walking automaton naturally induces an equivalence relation on such patterns: two patterns (with the same number of ports) are equivalent if in any context, the automaton cannot detect the difference when one pattern is replaced by another. This equivalence relation (technically, a slightly finer one, which speaks of states of the automaton) is the key notion in the study of patterns.

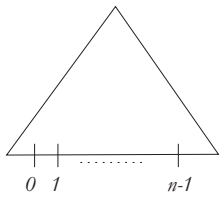


FIG. 3.1. *A pattern of arity n*

3.1. Patterns and pattern equivalence. A *pattern* is an $\{\mathbf{a}, \mathbf{b}, *\}$-tree where the labels **a** and $*$ are only found in the leaves. For technical reasons we require that a pattern has at least two nodes and that all $*$-labeled leaves are left children. A *blank pattern* is any pattern with no **a**-labeled leaf. The i-th $*$-labeled leaf (numbered from left to right, starting from 0) is called the i-th *port*. We number the ports from 0 to

be consistent with the usual tree terminology, where a left successor is denoted by 0 and a right successor by 1. *Port* ε stands for the root. The number of leaf ports is called the *arity* of the pattern. In particular, patterns of arity 0 are $\{\mathbf{a}, \mathbf{b}\}$-trees. See Fig. 3.1 for an illustration. Given an n-ary pattern Δ and n patterns $\Delta_0, \ldots, \Delta_{n-1}$, the *composition* $\Delta[\Delta_0, \ldots, \Delta_{n-1}]$ is the pattern obtained from Δ by simultaneously substituting each pattern Δ_i for the i-th port. We also allow some Δ_i's to be $*$. In this case, nothing is changed for the corresponding ports. We write $\Delta[\Delta_i/i]$ in the particular case where all Δ_j's but Δ_i are $*$; i.e., a single substitution is performed at port i. 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.

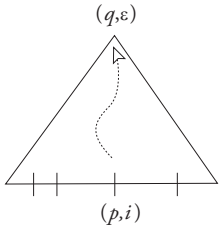


FIG. 3.2. *A pattern* Δ *with* (p, i, q, ε) *in* δ_Δ

A run in a pattern is defined just as a run in a tree, except that the ports (both root and leaf) are treated as being a nonleaf left child with the blank label. The latter assumption is for technical reasons; it will allow us to compose runs in larger patterns from runs in smaller ones. Moreover, we require that a run in a pattern visits ports at most twice: a port may occur only in the first and last configurations. In the following definition, illustrated by Fig. 3.2, we show how to describe a transition relation corresponding to a pattern for the automaton.

DEFINITION 3. *The* automaton's transition relation over an n-ary pattern Δ*,*

$$\delta_\Delta \subseteq (Q \times \{\varepsilon, 0, \ldots, n-1\})^2 \ ,$$

contains (p, i, q, j) *if in* Δ *there is a run from state* p *in port* i *to state* q *in port* j*.*

From the point of view of the automaton, the relation δ_Δ sums up all important properties of a pattern and we consider two patterns *equivalent* if they induce the same δ relation, i.e., patterns Δ and Δ' are equivalent if $\delta_\Delta = \delta_{\Delta'}$. This equivalence relation is a congruence with respect to composition of patterns, thanks to the technical assumptions. The essence of this equivalence is that if one replaces a sub-pattern by an equivalent one, the automaton is unable to see the difference. Here, we only consider contexts that are consistent with our technical assumptions: the root of the pattern corresponds to a left child, and the nodes plugged into the leaf ports are not leaves and have the blank label.

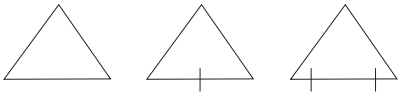


FIG. 3.3. *The patterns* Δ_0*,* Δ_1 *and* Δ_2

The following lemma was shown in [3], Lemma 9:

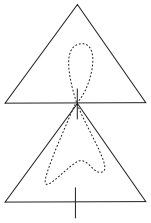


FIG. 3.4. *An inner loop*

LEMMA 3.1. *There are blank patterns* $\Delta_0, \Delta_1, \Delta_2$ *– of respective arities* 0, 1 *and* 2 *– such that any pattern in* $\mathcal{C}(\{\Delta_0, \Delta_1, \Delta_2\})$ *of arity* $i = 0, 1, 2$ *is equivalent to* Δ_i.

Those patterns will be used a lot in the constructions below. To keep the drawings uncluttered, we omit to specify the names Δ_0, Δ_1 and Δ_2 as this information can be reconstructed from the number of leaves; see Fig. 3.3.

Note that the lemma may fail for $i = 3$ when nondeterministic automata are involved, see [3]. The patterns Δ_0, Δ_1 and Δ_2 are the key to our proof. In a sense, their construction encapsulates all of the pumping arguments that we will do with respect to the automaton $\mathcal{A}$. For instance, the pattern Δ_1 is equivalent to a composition of any number of copies of Δ_1 patterns. In particular, if the automaton can go from the leaf port of Δ_1 to the root port, then there must be a state that is used twice along the way. We write $\mathcal{C}_\mathcal{A}$ to denote the set $\mathcal{C}(\{\Delta_0, \Delta_1, \Delta_2\})$; from now on almost all patterns considered will be taken from $\mathcal{C}_\mathcal{A}$.

3.2. Inner loops. Although simply defined, the relation δ_Δ is rather cumbersome to work with. The automaton may do some redundant moves, such as going one step down, and then one step up, without any apparent purpose (a phenomenon called oscillation in [3]). It is convenient to eliminate this obfuscating phenomenon. This is the purpose of the inner loop relation introduced in the next definition.

First however, we state Fact 3.2 which is a consequence of Lemma 3.1. In this statement and elsewhere, by the expression *plugging the* Δ_1 *pattern into some/any port of a pattern* Δ, we refer to one of the patterns $\Delta_1[\Delta], \Delta[\Delta_1/1], \ldots, \Delta[\Delta_1/n]$, where n is the arity of the pattern Δ. Similarly, the pattern obtained by *plugging* Δ_1 *into all ports of a pattern* Δ represents the pattern $\Delta_1[\Delta[\Delta_1, \ldots, \Delta_1]]$.

FACT 3.2. *Plugging the* Δ_1 *pattern into some port of a pattern in* $\mathcal{C}_\mathcal{A}$ *yields an equivalent pattern.*

Proof. Induction on the structure of the pattern, using Lemma 3.1 in the basis of the induction. □

Consider now a composition of two patterns $\Delta[\Delta'/i]$, and the junction of these patterns, i.e., the node v that corresponds to port i in Δ. By the fact above, we may well assume that v is on the junction of two Δ_1 patterns: one plugged into the leaf port i of Δ and one plugged into the root port of Δ'. In particular, any loop that can be done on the junction of two Δ_1 patterns can be replicated in $\Delta[\Delta'/i]$. Hence the importance of such loops; we call them "inner loops" in the following definition:

DEFINITION 4. *The* inner loop relation *over states is the least transitive and reflexive relation* $\to_\varepsilon$ *over states such that* $p \to_\varepsilon q$ *holds whenever* $(p, \varepsilon, q, \varepsilon)$ *or* $(p, 0, q, 0)$ *belongs to* δ_{Δ_1}.

The following lemma formalizes the comments preceding the introduction of the inner loop relation. It shows how $\rightarrow_\varepsilon$ describes all possible loops on interfaces between patterns from $\mathcal{C}_\mathcal{A}$:

LEMMA 3.3. *Let $\Delta, \Delta' \in \mathcal{C}_\mathcal{A}$ be patterns of nonzero arity and let v be the junction node corresponding to the leaf port i of Δ and the root of Δ'. There is an inner loop $p \rightarrow_\varepsilon q$ if and only if there is a run from (p, v) to (q, v) in $\Delta[\Delta'/i]$.*

Before proceeding with the proof, we would like to comment on the relevance of this lemma. Recall that by our definition of runs within patterns, the loop from (p, v) to (q, v) is not allowed to visit any of the ports. Therefore, the relation $\rightarrow_\varepsilon$ tells us what are the possible loops that can be done on the interface of two patterns without visiting any ports. In particular, the possible types of such loops do not depend on the two patterns Δ and Δ', as long as they are from $\mathcal{C}_\mathcal{A}$.

Another important consequence of this lemma is that it gives us a sort of normal form of runs through patterns in $\mathcal{C}_\mathcal{A}$. Any loop on a junction between patterns can be replaced by the $\rightarrow_\varepsilon$ relation, therefore a run through a pattern in $\mathcal{C}_\mathcal{A}$ can be seen as going directly from the source to the target, with all the loops being represented by the $\rightarrow_\varepsilon$ relation.

Proof. Assume that $p \rightarrow_\varepsilon q$. By definition of $\rightarrow_\varepsilon$, there is a run from (p, w) to (q, w) in $\Delta_1[\Delta_1]$ where w is at the junction of the two Δ_1 patterns. But this run can be reused within $\Delta[\Delta'/i]$, since by Fact 3.2 we may assume without loss of generality that both Δ and Δ' have Δ_1 plugged into all their ports.

Reciprocally, assume that there is a run from (p, v) to (q, v) in the pattern $\Delta[\Delta'/i]$. This run can be reused in the same pattern where a Δ_0 has been substituted for all ports but for some leaf port – say port 0 – of Δ'. By Lemma 3.1, this new pattern is equivalent to the composition of two Δ_1 patterns. It then follows by definition that $p \rightarrow_\varepsilon q$ holds. □

DEFINITION 5. *For a pattern Δ, the relation γ_Δ is the set of tuples (p, i, q, j) such that $p \rightarrow_\varepsilon p'$ and $q' \rightarrow_\varepsilon q$ for some p', q' satisfying $(p', i, q', j) \in \delta_\Delta$.*

Observe that a consequence of the definition above is that if $p \rightarrow_\varepsilon q$, then (p, i, q, i) belongs to γ_Δ for all ports i of Δ.

The γ relation has nicer closure properties than δ; hence from now on we will be using it – and not the δ relation – to describe runs in patterns. For instance, γ satisfies the following "swallowing" property:

$$(p, \varepsilon, q, 0), (q, \varepsilon, r, 0), (r, 0, s, \varepsilon) \in \gamma_{\Delta_1} \qquad \text{implies} \qquad (p, \varepsilon, s, 0) \in \gamma_{\Delta_1} \ .$$

This is because $(q, \varepsilon, r, 0), (r, 0, s, \varepsilon) \in \gamma_{\Delta_1}$ implies $q \rightarrow_\varepsilon s$. Another useful property of the γ relation — resulting from the equivalence of Δ_1 and $\Delta_1[\Delta_1]$ — is as follows:

$$(p, \varepsilon, q, 0) \in \gamma_{\Delta_1} \qquad \text{iff} \qquad (p, \varepsilon, r, 0), (r, \varepsilon, q, 0) \in \gamma_{\Delta_1} \text{ for some } r \ .$$

Note that the left-to-right implication fails for the δ relation, since the state r may require some loops on the junction between the Δ_1 patterns before the run reaches $(q, 0)$.

Obviously, if two patterns are equivalent, then they have the same γ relations. Let us remark also that a form of the converse also holds: if two patterns in $\mathcal{C}_\mathcal{A}$ have the same γ relations, then they are equivalent. However this fact is of no use in the remainder of the proof, and we need not establish it.

4. Pattern expansions and the proof strategy. In this section we introduce pattern expansions and then give an overview of our proof strategy.

4.1. Pattern expansions. The *pattern pre-expansion* of a blank tree t is the pattern obtained by replacing every inner node of t with the pattern Δ_2 and replacing every leaf with a port $*$. Thus, the pattern pre-expansion has as many leaf ports as t has leaves.

The *pattern expansion* Δ_t of t is the pattern obtained by plugging Δ_1 into all ports of the pattern pre-expansion (see Fig. 4.1). Note that the expansion and the pre-expansion are equivalent as patterns. With every node v of t we associate a node $[v]$ in the pattern Δ_t in the natural way (see Fig. 4.1); this node does not depend on t. A *junction node* in a pattern expansion is any node of the form $[v]$; it is called a *junction leaf* when v is a leaf of t. Note that a junction leaf is not a leaf in the pattern Δ_t, since it has Δ_1 as its subtree. The Δ_1 patterns plugged in the pattern expansion are used so that every junction node is on the interface of two patterns of nonzero arity in $\mathcal{C}_\mathcal{A}$. In particular, junction nodes are a suitable place for using Lemma 3.3.

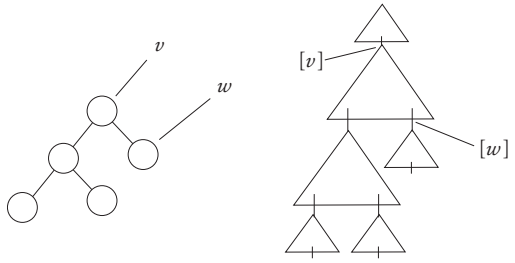


FIG. 4.1. *A blank tree t and its pattern expansion Δ_t*

We denote by $\Delta_\mathbf{a}$ a fixed pattern of arity 0 equal to $\Delta_1[\Delta'_\mathbf{a}]$ where $\Delta'_\mathbf{a}$ is some zero arity pattern containing exactly one $\mathbf{a}$-labeled leaf. The particular form of $\Delta'_\mathbf{a}$ is not important, but we can fix it to be a two-leaf tree with $\mathbf{a}$ in the left leaf, and $\mathbf{b}$ in the other nodes. The only two important points concerning $\Delta_\mathbf{a}$ is that first it contains a single leaf labeled by $\mathbf{a}$, and second, that $\Delta_\mathbf{a}$ is equivalent to $\Delta_1[\Delta_\mathbf{a}]$. This last point is obtained by remarking that $\Delta_\mathbf{a}$ equals $\Delta_1[\Delta'_\mathbf{a}]$ which is equivalent to $\Delta_1[\Delta_1[\Delta'_\mathbf{a}]]$, itself equal to $\Delta_1[\Delta_\mathbf{a}]$.

Given a blank tree t, the tree $\Delta_t^\mathbf{a}$ is obtained by plugging $\Delta_\mathbf{a}$ into all leaf ports of Δ_t, i.e., $\Delta_t^\mathbf{a} = \Delta_t[\Delta_\mathbf{a}, \ldots, \Delta_\mathbf{a}]$. Clearly the branching structure of $\Delta_t^\mathbf{a}$ is t. If the tree-walking automaton were to accept the tree language L, it would have to accept every tree $\Delta_t^\mathbf{a}$ for $t \in K$ and reject every tree $\Delta_t^\mathbf{a}$ for $t \notin K$. We will eventually show that this is impossible, due to the way tree-walking automata get lost in pattern expansions.

In order to avoid confusion we remark here that $\Delta_t^\mathbf{a}$ is treated as a tree, and not a pattern of zero arity. Therefore, a run over $\Delta_t^\mathbf{a}$ is allowed to visit the root several times, as opposed to runs over patterns of zero arity.

A *junction configuration* is defined to be a configuration of the form $(q, [v])$ for some node $v \in \{0,1\}^*$. We will write such a configuration $[q, v]$. If v is a node of a blank tree t, then $[q, v]$ can be interpreted as a configuration in either the pattern Δ_t

or the tree $\Delta_t^{\mathbf{a}}$. In either case, $[q, v]$ is a configuration whose node is a junction node. Moreover, if v is a leaf of t (i.e., $[v]$ is a junction leaf), the junction configuration is also called a *leaf configuration* (this, of course, is relative to the tree t). We use square brackets for junction configurations; these describe configurations in the branching structure t. On the other hand, normal configurations are written with round brackets; these describe configurations in the pattern expansion Δ_t or in the tree $\Delta_t^{\mathbf{a}}$.

The following lemma shows that the γ_{Δ_2} relation describes the way our fixed tree-walking automaton can move between neighboring junction nodes in pattern expansions:

LEMMA 4.1. *Let t be a blank tree and v a node of t. The following statements are equivalent for any states p and q:*

1. *In the pattern expansion Δ_t there is a run from $[p, v]$ to $[q, v]$.*
2. *$p \to_\varepsilon q$.*
3. *In the pattern expansion Δ_t there is a run from $[p, v]$ to $[q, v]$ that does not visit any other junction node $[w]$, $w \neq v$.*

Proof. From *1* to *2*. By cutting the pattern expansion Δ_t at the junction node $[v]$, we write it $\Delta[\Delta'/i]$ for some two patterns Δ and Δ' in $\mathcal{C}_A$ of nonzero arity, and i the number of $[v]$ as a leaf port of Δ. Applying Lemma 3.3 to it yields $p \to_\varepsilon q$.

From *2* to *3*. Let Δ and Δ' be either Δ_1 or Δ_2, and i be a leaf port of Δ. Let v' be the port i of Δ. By using $p \to_\varepsilon q$ together with Lemma 3.3 to $\Delta[\Delta'/i]$, we obtain that there is a run from (v', p) to (v', q) which does not visit the ports of $\Delta[\Delta'/i]$. By definition of the pattern expansion Δ_t, the junction node $[v]$ appears either as port 0 of a Δ_1 pattern or as port 0 or 1 of a Δ_2 pattern. Similarly it is also the root port of a Δ_1 or a Δ_2 pattern. Hence $[v]$ can be identified with node v' in a pattern $\Delta[\Delta'/i]$ above. We can transfer the run we had witnessed on $\Delta[\Delta'/i]$ to Δ_t, obtaining a run from $[p, v]$ to $[q, v]$ that does not visit any other junction node $[w]$, $w \neq v$.

From *3* to *1*. Straightforward. □

LEMMA 4.2. *Let t be a blank tree and $v \cdot a, v \cdot b$ nodes of t, with $v \in \{0, 1\}^*$ and $a, b \in \{\varepsilon, 0, 1\}$. The following statements are equivalent for any states p and q:*

1. *In the pattern expansion Δ_t, there is a run from $[p, v \cdot a]$ to $[q, v \cdot b]$.*
2. *$(p, a, q, b) \in \gamma_{\Delta_2}$.*
3. *In the pattern expansion Δ_t, there is a run from $[p, v \cdot a]$ to $[q, v \cdot b]$ that does not visit any other junction node $[w]$, $w \notin \{v \cdot a, v \cdot b\}$.*

Proof. Let us treat first the case $a = b$. In this case $p \to_\varepsilon q$ iff $(p, a, q, b) \in \gamma_{\Delta_2}$ (recall the observation below Definition 5). Hence the equivalence between items *1*, *2* and *3* is a direct translation of Lemma 4.1. We assume below that $a \neq b$, and set $c \in \{\varepsilon, 0, 1\}$ to be different from a and b.

From *1* to *2*. Assume there is a run in Δ_t from configuration $[p, v \cdot a]$ to configuration $[q, v \cdot b]$. Let p' be the last state assumed by the run while visiting the junction node $[v \cdot a]$, and let q' be the first state assumed at the junction node $[v \cdot b]$ after crossing $[v \cdot a]$ for the last time. If the corresponding subrun from $[p', v \cdot a]$ to $[q', v \cdot b]$ does not visit $[v \cdot c]$, then (p', a, q', b) belongs to δ_{Δ_2}. Otherwise it visits $[v \cdot c]$, and let p'' and q'' be the first and last state, respectively, assumed by that subrun at $[v \cdot c]$. Then $p'' \to_\varepsilon q''$ by Lemma 4.1. Also, by construction, (p', a, p'', c) and (q'', c, q', b) are in δ_{Δ_2}. This shows (by plugging Δ_1 into port c of Δ_2) that (p', a, q', b) belongs to δ_{Δ_2}. Moreover, by Lemma 4.1, we have $p \to_\varepsilon p'$ and $q' \to_\varepsilon q$. By definition of γ_{Δ_2}, $(p, a, q, b) \in \gamma_{\Delta_2}$ follows.

From *2* to *3*. Since $(p, a, q, b) \in \gamma_{\Delta_2}$ there exist two states p' and q' such that:

$$p \to_\varepsilon p', \quad (p', a, q', b) \in \delta_{\Delta_2}, \quad \text{and} \quad q' \to_\varepsilon q \ .$$

By Lemma 4.1, these three properties provide a run in the pattern expansion Δ_t that successively passes through the configurations:

$$[p, v{\cdot}a], \ [p', v{\cdot}a], \ [q', v{\cdot}b], \ [q, v{\cdot}b]$$

without visiting any junction node other than $[v{\cdot}a]$ and $[v{\cdot}b]$.

From *3* to *1*. Straightforward. □

The above lemma shows that runs of the automaton between neighboring junction nodes in pattern expansions can be assumed to have a very particular form. Take for instance a blank tree t and two nodes v and w, with v above w. If there is a run in Δ_t that goes from $[v]$ to $[w]$ then, by Lemma 4.2, there is a run that does this by going directly from v to w using the shortest path. This means performing only a series of "steps" of the form $(p, \varepsilon, q, 0), (p, \varepsilon, q, 1) \in \gamma_{\Delta_2}$. A similar characterization holds when v and w are incomparable: the automaton first goes directly from $[v]$ in the up direction, then does one of the steps $(p, 0, q, 1), (p, 1, q, 0) \in \gamma_{\Delta_2}$ (a "go to the sibling" move) and then goes directly downward to $[w]$. This principle is formalized in Lemma 6.1.

4.2. The proof strategy. We are now ready to overview the proof strategy. Recall that our aim is to find trees $s \in L$ and $s' \notin L$ such that any accepting run in s can be transferred to s'. In fact, the trees s, s' will be respectively of the form $\Delta_t^{\mathbf{a}}$ and $\Delta_{t'}^{\mathbf{a}}$ for some blank trees $t \in K$ and $t' \notin K$. Therefore, we need to develop a good understanding of runs within trees of the form $\Delta_t^{\mathbf{a}}$.

The remainder of this paper is divided into three sections, which correspond to ever larger parts of a run over a tree of the form $\Delta_t^{\mathbf{a}}$. Such a run can be analyzed on three scales.

The greatest scale is analyzed in Section 7. Fix a tree $\Delta_t^{\mathbf{a}}$. In this greatest scale, we will be most interested in runs that connect leaf configurations to one another, without passing through the root of the tree. (This is because without loss of generality, we may assume the root is visited at most $|Q|$ times.) Consider such a run that goes from one leaf configuration $[p, v]$ to another leaf configuration $[q, w]$. Within such a run, we can isolate all the intermediate leaf configurations:

$$[p, v] = [r_1, u_1], \ldots, [r_n, u_n] = [q, w] \ .$$

Since no leaf configurations are visited in the meantime, a run from $[r_i, u_i]$ to $[r_{i+1}, u_{i+1}]$ corresponds to either: (a) a loop in the root of the pattern $\Delta_1[\Delta_{\mathbf{a}}]$; or (b) a run from one junction leaf to another in the pattern Δ_t. The case (a) can be treated as a sort of ε-transition for leaf configurations. The interesting case is (b).

In Section 6, we treat the runs of type (b), which correspond to the intermediate scale. These runs are in $\Delta_t^{\mathbf{a}}$, but since no $\mathbf{a}$-labeled leaf is visited during those runs, they are also runs in the pattern expansion Δ_t. We first show that whether or not there is a run of type (b) from $[r_i, u_i]$ to $[r_{i+1}, u_{i+1}]$ does not depend on the tree t but only on the nodes u_i and u_{i+1}. This allows us to consider the notation $[p, v] \to [q, w]$, meaning: there is a run from $[p, v]$ to $[q, w]$ in some (equivalently, every) pattern expansion Δ_t for which v, w are nodes of t. A type of run that realizes $[p, v] \to [q, w]$ is called a *move*; a classification of the possible types of move is the subject of Section 6.

As a preparation, in Section 5, we consider the smallest scale: the pattern Δ_2. By Lemma 4.2, any move within a pattern expansion can be decomposed into a certain number of traversals of the pattern Δ_2. Hence the need for an investigation of the relation γ_{Δ_2}.

Before proceeding, we describe in general terms what are the results of these investigations in the Sections 5, 6 and 7.

The main result of Section 5 is Proposition 5.10, which gives a characterization of the possible ways the automaton can go from the leaf port to the root port of Δ_1. Generally speaking, this characterization says that the automaton either gets completely lost, or it must do some sort of depth-first search. Even though stated in terms of the Δ_1 pattern, these results can also be applied to the Δ_2 pattern. Indeed, by Lemma 3.1, any run from a leaf port to the root port in Δ_2 can also be transfered to Δ_1, by simply plugging the unused leaf port with Δ_0.

The main result of Section 6 is Proposition 6.10. This proposition roughly says that there are only eleven types of interesting moves between junction leaves in a pattern expansion. The interesting moves — called elementary moves — are moves such as: "go to the next junction leaf to the left"; or "go to any junction leaf to the left". Proposition 6.10 states that if a move is not elementary, then it contains a "shift", a phenomenon of inherent confusion for the automaton.

Finally, in Section 7, we show that tree-walking automata cannot detect a properly placed rotation, which concludes the proof. Given a blank tree T and a node x, the *rotation of T with the pivot x* is the tree T' defined as follows: we move the subtrees of $x{\cdot}00$, $x{\cdot}01$ and $x{\cdot}1$ to the new positions $x{\cdot}0$, $x{\cdot}10$ and $x{\cdot}11$ (see Fig. 7.1). Clearly doing a rotation in a tree with all leaves at even depth creates a leaf at odd depth. We will show, however, that given a very large balanced blank tree T, one can find a pivot x such that $\Delta^{\mathbf{a}}_{T'}$ cannot be distinguished from $\Delta^{\mathbf{a}}_{T}$ by our fixed tree-walking automaton $\mathcal{A}$.

5. The pattern Δ_2. In this section we investigate the γ relations of the patterns Δ_0, Δ_1 and Δ_2. The main result, Proposition 5.10, uncovers by a case distinction the possible ways the tree-walking automaton can cross the pattern Δ_1. This is important for our analysis of pattern expansions, since by Lemma 4.2 every path through a pattern expansion corresponds to a sequence of traversals of Δ_2 patterns.

From now on, instead of the γ_{Δ_0}, $\gamma_{\Delta_{\mathbf{a}}}$, γ_{Δ_1} and γ_{Δ_2} relations, we will be using the more graphical notation depicted in Fig. 5.1. Note that the $p \curvearrowleft q$ notation may be somewhat misleading: we *start* with state p in port 1 and *end* in state q in port 0. The left state is chronologically before the right one, although the movement is in the left direction.

Due to the equivalences in Lemma 3.1, the relations $\gamma_{\Delta_1}, \gamma_{\Delta_2}$ satisfy properties such as the following, which we call swallowing rules:

$$(p,0,q,1) \in \gamma_{\Delta_2}\ , (q,\varepsilon,r,0) \in \gamma_{\Delta_1} \qquad \text{implies} \qquad (p,0,r,1) \in \gamma_{\Delta_2}\ .$$

Using our graphical notation, this can be rewritten into the first property among the following ones:

$$\begin{array}{lll} p \curvearrowright q \downarrow r & \text{implies} & p \curvearrowright r \\ p \uparrow q \curvearrowleft r & \text{implies} & p \curvearrowleft r \\ p \curvearrowleft q \downarrow r & \text{implies} & p \curvearrowleft r \\ p \uparrow q \curvearrowright r & \text{implies} & p \curvearrowright r\ . \end{array} \tag{5.1}$$

$$
\begin{array}{rlll}
p \circlearrowleft q & \text{if} & (p,\varepsilon,q,\varepsilon) \in \gamma_{\Delta_0} & \qquad p \nwarrow q \quad \text{if} \quad (p,1,q,\varepsilon) \in \gamma_{\Delta_2} \\
 & & & \qquad p \nearrow q \quad \text{if} \quad (p,0,q,\varepsilon) \in \gamma_{\Delta_2} \\
p \circlearrowleft_{\mathbf{a}} q & \text{if} & (p,\varepsilon,q,\varepsilon) \in \gamma_{\Delta_{\mathbf{a}}} & \qquad p \searrow q \quad \text{if} \quad (p,\varepsilon,q,1) \in \gamma_{\Delta_2} \\
 & & & \qquad p \swarrow q \quad \text{if} \quad (p,\varepsilon,q,0) \in \gamma_{\Delta_2} \\
p \uparrow q & \text{if} & (p,0,q,\varepsilon) \in \gamma_{\Delta_1} & \qquad p \curvearrowleft q \quad \text{if} \quad (p,1,q,0) \in \gamma_{\Delta_2} \\
p \downarrow q & \text{if} & (p,\varepsilon,q,0) \in \gamma_{\Delta_1} & \qquad p \curvearrowright q \quad \text{if} \quad (p,0,q,1) \in \gamma_{\Delta_2}
\end{array}
$$

$$
\begin{array}{ll}
p \downarrow\!\!\!\nwarrow q & \text{if } p \nwarrow q \text{ and not } p \nearrow q \\
p \nwarrow\!\!\!\uparrow q & \text{if } p \nearrow q \text{ and not } p \nwarrow q \\
p \swarrow\!\!\!\downarrow q & \text{if } p \swarrow q \text{ and not } p \searrow q \\
p \downarrow\!\!\!\searrow q & \text{if } p \searrow q \text{ and not } p \swarrow q
\end{array}
\qquad
\begin{array}{l}
p \updownarrow q \text{ if } p \uparrow r \nearrow r \nwarrow r \uparrow q \text{ for some } r \\
p \downarrow\!\!\updownarrow q \text{ if } p \downarrow r \searrow r \swarrow r \downarrow q \text{ for some } r
\end{array}
$$

FIG. 5.1. *Graphical notation for* $\gamma_{\Delta_0}, \gamma_{\Delta_{\mathbf{a}}}, \gamma_{\Delta_1}, \gamma_{\Delta_2}$

We will now illustrate how time symmetry can be used to show the second implication; the third and fourth are then obtained using space symmetry.

Let then p, q and r be such that $p \uparrow q \curvearrowleft r$ holds. As the reader may recall, our tree-walking automaton is self-symmetric; i.e., there is an isomorphism $i : Q \to Q$, which maps the automaton onto its time-reversed variant. Let $i(p)$, $i(q)$ and $i(r)$ be the time-reversed counterparts of p, q and r. By Fact 2.1, every run from p to q can be reversed to obtain a run from $i(q)$ to $i(p)$; likewise for q and r. If we reverse a run witnessing $p \uparrow q$, we obtain a run witnessing $i(q) \downarrow i(p)$. In the same way, there is a run witnessing $i(r) \curvearrowright i(q)$. In particular, we have

$$i(r) \curvearrowright i(q) \downarrow i(p) \ .$$

Now we can apply the already shown first implication in (5.1), to obtain $i(r) \curvearrowright i(p)$. Since the isomorphism i is its own inverse, we obtain the desired $p \curvearrowleft r$.

In a similar way, using space symmetry, we can derive the two last statements of (5.1) from the two first one. Later on, we will be using this type of reasoning a lot, omitting the details of the argumentation.

The following lemma shows that the $\circlearrowleft$ and $\circlearrowleft_{\mathbf{a}}$ notation is not misleading in suggesting a loop:

LEMMA 5.1. *The relations* $\circlearrowleft$ *and* $\circlearrowleft_{\mathbf{a}}$ *are transitive.*

Proof. We only do the proof for $\circlearrowleft$. We first claim that $p \circlearrowleft q$ holds if and only if either $p \to_\varepsilon q$ holds, or $p \downarrow p' \circlearrowleft^* q' \uparrow q$ holds for some states p', q' (where $\circlearrowleft^*$ is the transitive closure of $\circlearrowleft$).

The left to right implication of the claim is shown as follows. If $p \circlearrowleft q$ holds, then there exist p'', q'' such that

$$p \to_\varepsilon p'' \ , \quad (p'', \varepsilon, q'', \varepsilon) \in \delta_{\Delta_0} \quad \text{and} \quad q'' \to_\varepsilon q \ .$$

. Let us analyze the run corresponding to $(p'', \varepsilon, q'', \varepsilon) \in \delta_{\Delta_0}$ in the pattern $\Delta_1[\Delta_0]$ (which is equivalent to Δ_0), the junction node being v. If this run does not visit v, then we have $p'' \to_\varepsilon q''$, and consequently $p \to_\varepsilon q$. Otherwise, there exist states p'

and q' such that $(p'', \varepsilon, p', 0)$ and $(q', 0, q'', \varepsilon)$ belong to δ_{Δ_1}, and there is a path from configuration (v, p') to configuration (v, q') in the pattern $\Delta_1[\Delta_0]$. From this path we deduce $p' \circlearrowleft^* q'$. Hence $p \downarrow p' \circlearrowleft^* q' \uparrow q$.

The right to left implication of the claim is shown as follows. If $p \to_\varepsilon q$, we obviously have $p \circlearrowleft q$. Otherwise, assume that there exist states p', q' such that $p \downarrow p' \circlearrowleft^* q' \uparrow q$ holds. This means there are states p'', p''', q''', q'' such that:

$$\begin{array}{lll} p \to_\varepsilon p'', & (p'', \varepsilon, p''', 0) \in \delta_{\Delta_1}, & p''' \to_\varepsilon p', \\ q' \to_\varepsilon q''', & (q''', 0, q'', \varepsilon) \in \delta_{\Delta_1}, & \text{and} \quad q'' \to_\varepsilon q\ . \end{array}$$

From $p''' \to_\varepsilon p' \circlearrowleft^* q' \to_\varepsilon q'''$ we obtain $p''' \circlearrowleft^* q'''$. Together with $(p'', \varepsilon, p''', 0) \in \delta_{\Delta_1}$ and $(q''', 0, q'', \varepsilon) \in \delta_{\Delta_1}$ we obtain a run from state p'' in the root to q'' in the root in $\Delta_1[\Delta_0]$. Hence $(p'', \varepsilon, q'', \varepsilon)$ belongs to δ_{Δ_0}. Together with $p \to_\varepsilon p''$ and $q'' \to_\varepsilon q$ we obtain $p \circlearrowleft q$. This concludes the proof of the claim.

Let us now show the transitivity of $\circlearrowleft$. Assume $p \circlearrowleft q \circlearrowleft r$. If either $p \to_\varepsilon q$ or $q \to_\varepsilon r$ holds, then we have $p \circlearrowleft r$ by definition of $\circlearrowleft$. Otherwise, according to the claim above, there exist states p', q', q'', r' such that:

$$p \downarrow p' \circlearrowleft^* q' \uparrow q \downarrow q'' \circlearrowleft^* r' \uparrow r\ .$$

Since $q' \uparrow q \downarrow q''$ implies $q' \to_\varepsilon q''$, we obtain $p' \circlearrowleft^* r'$ by transitivity of $\circlearrowleft^*$. Using the other direction from the claim, we get $p \circlearrowleft r$.

For $\circlearrowleft_{\mathbf{a}}$, the proof is the identical. The only property required from the pattern $\Delta_{\mathbf{a}}$ is that it is equivalent to $\Delta_1[\Delta_{\mathbf{a}}]$; this fact was observed above while defining the pattern $\Delta_{\mathbf{a}}$. □

5.1. Depth-first search. In this section we define the key concept of depth-first search (DFS). The main result, Lemma 5.5, states that $p \downarrow p$ can only be realized using a DFS (similarly for $\uparrow$, $\downarrow$ and $\downarrow$).

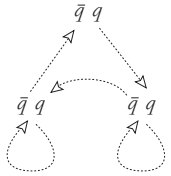


FIG. 5.2. *A right-to-left DFS*

DEFINITION 6. *A state pair* $(q, \bar{q})$ *is a* right-to-left DFS *(see Figure 5.2) if*

$$q \searrow q\ , \qquad q \circlearrowleft \bar{q}\ , \qquad \bar{q} \curvearrowleft q\ , \quad \textit{and} \quad \bar{q} \nearrow \bar{q}\ .$$

The pair is a left-to-right DFS *if*

$$q \swarrow q\ , \qquad q \circlearrowleft \bar{q}\ , \qquad \bar{q} \curvearrowright q\ , \quad \textit{and} \quad \bar{q} \nwarrow \bar{q}\ .$$

Throughout the paper, we will try to keep the convention that if two states $\bar{q}$ and q appear simultaneously, then $\bar{q}$ is a state that is going up in the tree and q is a state that is going down in the tree.

We now illustrate the way a left-to-right DFS allows $\mathcal{A}$ to walk through a pattern expansion. Consider a pattern expansion Δ_t and a left-to-right DFS $(q, \bar{q})$. Using $q \swarrow q$ repeatedly, the automaton can go from q in $[\varepsilon]$ to q in the leftmost junction leaf (all this reasoning is done using Lemma 4.2). If v and w are successive leaves of t, then the automaton can go from $[\bar{q}, v]$ to $[q, w]$. This is done by using a sequence of steps $\bar{q} \nwarrow \bar{q}$, then doing a step of the form $\bar{q} \curvearrowright q$, and then a sequence of $q \swarrow q$ steps. Finally, using $\bar{q} \nwarrow \bar{q}$, the automaton can go from $\bar{q}$ in the rightmost junction leaf to $\bar{q}$ in $[\varepsilon]$. Moreover, if we plug Δ_0 in every leaf port of Δ_t, then $q \circlearrowleft \bar{q}$ together with the above observations can be used to obtain a left-to-right depth-first search of all the junction nodes in the pattern $\Delta_t[\Delta_0, \ldots, \Delta_0]$.

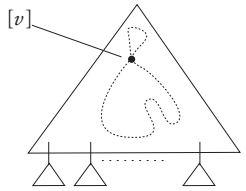
In Lemma 5.4, we will also show the converse: without doing a DFS, the automaton cannot systematically visit all junction nodes in a pattern of the form $\Delta_t[\Delta_0, \ldots, \Delta_0]$. First, however, we provide some preparatory results. In the following lemma, we say that a run *omits* a node if it never crosses it.

LEMMA 5.2. *Let t be a blank tree, with nodes v, w. Let ρ be a run in $\Delta_t[\Delta_0, \ldots, \Delta_0]$ from configuration $[p, v]$ to configuration $[q, w]$ for some states p, q. Then;*

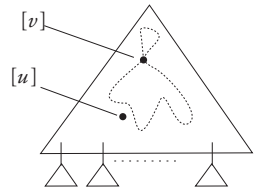
1. *If $v = w$ then $p \circlearrowleft q$.*
2. *Assume $v = w$ and let u be a node strictly below v. If ρ omits $[u]$, then $p \to_\varepsilon q$.*
3. *Assume v is above w and let u be a node strictly below w. If ρ omits $[u]$, then $p \downarrow q$.*
4. *Assume v is above w, let u be a node strictly below w, and let u' be a node to the right of w and strictly below v. If ρ omits $[u]$ and $[u']$ then $p \swarrow q$.*
5. *Assume v is to the left of w, and let u, u' be nodes respectively strictly below v, w. If ρ omits $[u]$ and $[u']$ then $p \curvearrowright q$.*

Proof. We would like to clarify that $\Delta_t[\Delta_0, \ldots, \Delta_0]$ is treated here as a pattern of arity zero, and not a tree. Therefore, by definition of runs in patterns, ρ never visits the root port. Below, we treat successively the five cases. Each explanation is followed by a drawing illustrating the situation.

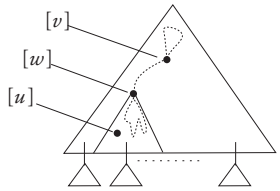
1. By transitivity of $\circlearrowleft$ (Lemma 5.1), it suffices to consider runs ρ where v is only visited in the first and last configuration. If the run never goes below $[v]$, in this case, by putting a port in the node $[v]$, the run can be replicated in a pattern equivalent to Δ_1, yielding $(p, 0, q, 0) \in \delta_{\Delta_1}$ and hence $p \to_\varepsilon q$. Otherwise, the run only visits nodes below $[v]$, and is therefore a root-to-root run in a pattern equivalent to Δ_0.



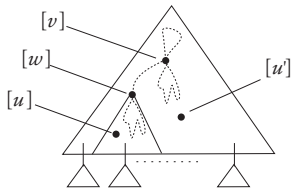
2. Again, by transitivity of the $\to_\varepsilon$ relation, it suffices to consider the case where v is only visited in the first and last configuration. There are two cases. Either the run never goes below $[v]$, and we can reuse the argument of item 1. Otherwise the run only visits nodes below $[v]$, but not the node $[u]$. Therefore, if we put a port into node $[u]$, ρ becomes a root-to-root loop in a pattern equivalent to Δ_1, witnessing $(p, \varepsilon, q, \varepsilon) \in \delta_{\Delta_1}$; consequently $p \to_\varepsilon q$.



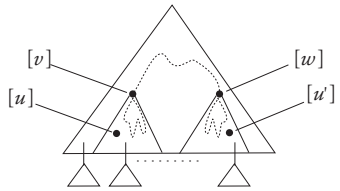
3. First we show, that without loss of generality we may assume that ρ visits $[v]$, $[w]$ only in the first and last configurations. Indeed, if we take the longest prefix of ρ that is a loop in $[v]$, this prefix satisfies the assumptions of item 2, and can therefore be replaced by an inner loop $\rightarrow_\varepsilon$. The same can then be done for the suffix, removing additional visits to $[w]$. Once ρ is assumed to visit $[v]$ and $[w]$ only once, it is easily seen to witness $(p, \varepsilon, q, 0) \in \delta_{\Delta_1}$ and hence $p \downarrow q$: if we put the root port in $[v]$ and a leaf port in $[w]$, we get a pattern equivalent to Δ_1.



4. As in the previous point, we may assume that ρ visits $[v]$, $[w]$ only in the first and last configurations (we use the assumption on the node $[u]$ being omitted and below both $[v]$ and $[w]$). If we put one leaf port (the left port) in $[w]$, one leaf port (the right port) in $[u']$, and the root port in node $[v]$, we get a pattern equivalent to Δ_2. Then the run ρ witnesses $(p, \varepsilon, q, 0) \in \delta_{\Delta_2}$, and hence $p \swarrow q$.



5. Again, we assume that ρ visits $[v]$, $[w]$ only in the first and last configurations (this time, we need to use both $[u]$ and $[u']$). We put the left port in node $[v]$, the right port in node $[w]$ (the root port stays unchanged).



□

We need now a simple combinatorial result concerning labeling of trees.

LEMMA 5.3. *Let Σ be an alphabet and consider a balanced Σ-tree t of depth at least $|\Sigma|+1$. There exist three nodes w, w_0, w_1 with the same label, such that w_0 is to the left of w_1 and w is above both w_0 and w_1.*

Proof. Induction on $|\Sigma|$. The base case of $|\Sigma| = 1$ is obvious. Otherwise let a be the root label of t. If both the subtrees of nodes 0 and 1 contain a's, we are done. Otherwise one of these is a $\Sigma \setminus \{a\}$-tree and the induction hypothesis can be applied to it. □

The following lemma is the first important characterization of runs on patterns. It says that there are only two types of root-to-root runs over the pattern Δ_0: either a run that does not visit anything, only does an inner loop $\rightarrow_\varepsilon$; or a systematic DFS traversal.

LEMMA 5.4. *Let $q, \bar{q}$ be such that $q \circlearrowleft \bar{q}$. Then either $q \rightarrow_\varepsilon \bar{q}$, or there is a (left-to-right or right-to-left) DFS $(r, \bar{r})$ such that $q \downarrow r$ and $\bar{r} \uparrow \bar{q}$.*

Proof. Let t be the blank balanced tree of depth $|Q|^2 + 2$. Let Γ be the pattern obtained from Δ_t by substituting Δ_0 for all leaf ports, i.e., $\Gamma = \Delta_t[\Delta_0, \ldots, \Delta_0]$. By definition of expansions, the pattern Γ can be rewritten as $\Delta_1[\Gamma']$ in which, by Lemma 3.1, the pattern Γ' is equivalent to Δ_0.

By $q \circlearrowleft \bar{q}$, there exists a run in Γ from $[q, \varepsilon]$ to $[\bar{q}, \varepsilon]$. First we show that we can furthermore enforce the following property (*) of ρ: every subrun starting and ending at the same junction node $[v]$ for v a nonleaf node of t, only visits junction nodes below $[v]$. This is proved by induction on the number of junction nodes in the run. Indeed, take a minimal loop in some junction node $[v]$ that visits junction nodes not below $[v]$. By minimality, the loop never visits nodes below $[v]$. Hence, by Lemma 4.1, this loop can be replaced by another of same initial and final configurations which does not visit any junction node other than $[v]$.

Let then ρ be a run from $[q, \varepsilon]$ to $[\bar{q}, \varepsilon]$ that satisfies property (*).

If ρ does not visit some junction node, then by Lemma 5.2, $q \rightarrow_\varepsilon \bar{q}$ holds.

Otherwise, given a node v of t, let *first*(v) be the state in which the junction node $[v]$ is visited for the first time in the run ρ. Similarly we define *last*(v). By Lemma 5.3, there are three nonleaf nodes w, w_0, w_1 of t such that *first* and *last* coincide on them, w is above both w_0 and w_1, and w_0 is to the left of w_1 (see Figure 5.3).

Consider first the case where $[w_0]$ is visited before $[w_1]$. Let r be *first*(w) and $\bar{r}$ be *last*(w). By Lemma 5.2, $r \circlearrowleft \bar{r}$. From property (*) we derive that the run cannot visit $[w_0]$ then $[w_1]$ and then again $[w_0]$. Hence we can apply Lemma 5.2 to configurations $[\bar{r}, w_0]$ and $[r, w_1]$ and obtain $\bar{r} \curvearrowright r$. Also, from the definition of *first* and *last* and Lemma 5.2, we obtain $r \swarrow r$ and $\bar{r} \nwarrow \bar{r}$. Overall $(r, \bar{r})$ is a left-to-right DFS and by Lemma 5.2, $q \downarrow r$ and $\bar{r} \uparrow \bar{q}$.

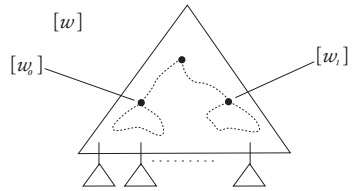


FIG. 5.3. *The nodes* $[w]$, $[w_0]$ *and* $[w_1]$

If $[w_1]$ is visited before $[w_0]$, a similar argument gives a right-to-left DFS. □

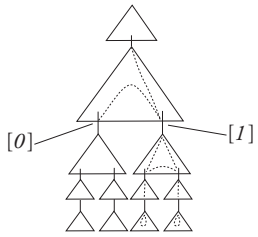
We now proceed to another one of our DFS characterizations: the relation $\curlywedge$ can only be realized by doing a DFS.

LEMMA 5.5. *If* $\bar{q} \curlywedge \bar{q}$, *then* $(q, \bar{q})$ *is a left-to-right DFS for some state* q.

Proof. Unraveling the definition of $\bar{q} \curlywedge \bar{q}$, we have that $\bar{q} \nwarrow \bar{q}$ holds but $\bar{q} \nearrow \bar{q}$ does not. Let t be the blank balanced binary tree of depth 3 (i.e., with four leaves). Let Γ be the pattern $\Delta_t[\Delta_0, \Delta_0, \Delta_0, \Delta_0]$. Since $\bar{q} \nwarrow \bar{q}$ implies $\bar{q} \uparrow \bar{q}$, and the pattern

$$\Delta_2[*, \Delta_2[\Delta_1[\Delta_0], \Delta_1[\Delta_0]]]$$

is equivalent to Δ_1, there is a run in this pattern from $[\bar{q}, 0]$ to $[\bar{q}, \varepsilon]$:



This run has to visit the junction node [1] since otherwise Lemma 5.2 would give $\bar{q} \nearrow \bar{q}$; a contradiction. Let p be the first state assumed by this run at the junction node [1], and let $\bar{p}$ be the last.

By Lemma 5.2, we have $\bar{q} \curvearrowright p, p \circlearrowleft \bar{p}$ and $\bar{p} \nwarrow \bar{q}$. We cannot have $p \rightarrow_\varepsilon \bar{p}$, since we would otherwise get $\bar{q} \nearrow \bar{q}$. By Lemma 5.4, we obtain states $r, \bar{r}$ such that $(r, \bar{r})$ is a DFS and $p \downarrow r, \bar{r} \uparrow \bar{p}$. By swallowing, we obtain $\bar{q} \curvearrowright r, \bar{r} \nwarrow \bar{q}$ (and we can forget about states p and $\bar{p}$).

Two cases have to be considered depending on the orientation of the DFS $(r, \bar{r})$.

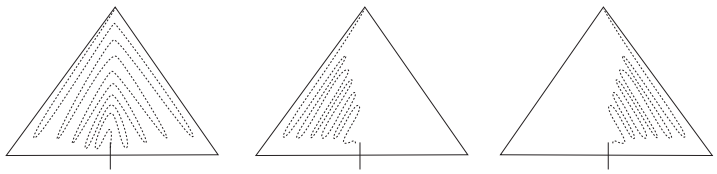
- Assume first that $(r, \bar{r})$ is a left-to-right DFS. We have $r \downarrow r$ and $\bar{r} \uparrow \bar{q}$. Thus, $r \circlearrowleft \bar{q}$ follows from $r \downarrow r \circlearrowleft \bar{r} \uparrow \bar{q}$. We obtain that $(r, \bar{q})$ is a left-to-right DFS as well.
- Otherwise, $(r, \bar{r})$ is a right-to-left DFS. By $r \searrow r \circlearrowleft \bar{r} \nwarrow \bar{q}$, we get $r \rightarrow_\varepsilon \bar{q}$. But then $\bar{q} \curvearrowright r \rightarrow_\varepsilon \bar{q} \nwarrow \bar{q}$ gives $\bar{q} \nearrow \bar{q}$; a contradiction.

□

5.2. Subtree omission. This section is devoted to showing the following proposition.

PROPOSITION 5.6. *For all* p *and* q, $p \uparrow q$ *if and only if* $p \nwarrow q$ *or* $p \nearrow q$.

The right to left implication is obvious; the remainder of this section is devoted to showing the left to right implication. The intuitive idea is illustrated in the picture below: whenever there is a run as on the left, there is also an equivalent run as in the middle or in the right:



We first show the following intermediate result.

LEMMA 5.7. *If $\bar{q} \uparrow \bar{q}$, then*

- *$\bar{q} \nearrow \bar{q}$ or $\bar{q} \nwarrow \bar{q}$; or*
- *there is a right-to-left DFS $(r, \bar{r})$ such that $\bar{q} \curvearrowright r$ and $\bar{r} \uparrow \bar{q}$.*

Proof. As in the proof of Lemma 5.5, we obtain that $\bar{q} \nearrow \bar{q}$ holds or there are two states $r, \bar{r}$ such that $(r, \bar{r})$ is a DFS, and both $\bar{q} \curvearrowright r$ and $\bar{r} \uparrow \bar{q}$ hold. The first case as well as the second when the DFS is right-to-left are conclusions of the lemma.

In the remaining case, $(r, \bar{r})$ is a left-to-right DFS. But then by $\bar{q} \curvearrowright r \circlearrowleft \bar{r} \nwarrow \bar{r}$ we obtain $\bar{q} \uparrow \bar{r}$. Combining this with $\bar{r} \nwarrow \bar{r} \uparrow \bar{q}$, we obtain the desired $\bar{q} \nwarrow \bar{q}$. □

A variant symmetric to the one above can be obtained, where $(r, \bar{r})$ is a left-to-right DFS and $\bar{q} \curvearrowleft r$ and $\bar{r} \uparrow \bar{q}$ hold.

LEMMA 5.8. *If $p \uparrow q$ then $p \uparrow r \uparrow r \uparrow q$ for some state r.*

Proof. This results from a pumping argument. Since $\Delta_1[\Delta_1]$ is equivalent to Δ_1, we can expand the Δ_1 pattern into the composition of n times itself, for any n. This means that there are states $r_1, \ldots, r_n$ such that $p = r_1 \uparrow \ldots \uparrow r_n = q$.

If n is large enough, some state r is repeated, and the result follows by transitivity of $\uparrow$. □

We will now prove Proposition 5.6. Since we have the implication

$$p \uparrow r \nwarrow r \uparrow q \qquad \text{implies} \qquad p \nwarrow q \ ,$$

and its symmetric counterpart for $\nearrow$, Lemma 5.8 allows us to restrict attention to the case where $p = q$. That is, we need to show that $p \uparrow p$ implies $p \nwarrow p$ or $p \nearrow p$.

If in either Lemma 5.7 or its symmetric variant the first case holds, we are done. Otherwise there are states q, $\bar{q}$, r and $\bar{r}$ such that:

$$\begin{array}{llllll} p \curvearrowright q, & q \searrow q, & q \circlearrowleft \bar{q}, & \bar{q} \curvearrowleft q, & \bar{q} \nearrow \bar{q}, & \bar{q} \uparrow p, \\ p \curvearrowleft r, & r \swarrow r, & r \circlearrowleft \bar{r}, & \bar{r} \curvearrowright r, & \bar{r} \nwarrow \bar{r}, & \bar{r} \uparrow p \ . \end{array}$$

By $p \curvearrowleft r \circlearrowleft \bar{r} \curvearrowright r$ we get $p \rightarrow_\varepsilon r$. Together with $\bar{q} \uparrow p$ and $r \downarrow r$ this gives $\bar{q} \rightarrow_\varepsilon r$. Together with $q \circlearrowleft \bar{q}$ and $r \circlearrowleft \bar{r}$, this yields $q \circlearrowleft \bar{r}$ by Lemma 5.1. Then $p \curvearrowright q \circlearrowleft \bar{r} \nwarrow \bar{r}$ shows $p \uparrow \bar{r}$. Finally, we combine this with $\bar{r} \nwarrow \bar{r} \uparrow p$ and obtain $p \nwarrow p$.

5.3. A characterization of moves over Δ_1. In this section, we present a classification of the possible ways the automaton can go in Δ_1 from the leaf port to the root port. This is the main result of Section 5. Before we proceed with Proposition 5.10, we show a certain "denseness" property of the relations $\nearrow$, $\llcorner$ and $\uparrow$:

LEMMA 5.9. *For any states p, q and $R = \llcorner, \nearrow, \uparrow$:*

$$p \; R \; q \qquad \textit{implies} \qquad p \; R \; r \; R \; r \; R \; q \textit{ for some state } r \; .$$

Proof. The case of $\uparrow$ follows straight from the definition. We only do $\llcorner$, the other case is done symmetrically. If $p \llcorner q$, then $p \nearrow q$, and thus also $p \uparrow q$. By Lemma 5.8, there must be some state r such that $p \uparrow r \uparrow r \uparrow q$. By Proposition 5.6, we must have at least one of $r \nearrow r$, $r \nwarrow r$. But we cannot have $r \nwarrow r$, since this would yield $p \nwarrow q$ and contradict $p \llcorner q$; hence $r \llcorner r$. For similar reasons $p \llcorner r$ and $r \llcorner q$ must also hold. □

Note that the converse implication may fail. This is because $p \llcorner q$ requires $p \nwarrow q$ to fail, while there may be some other state s satisfying $p \uparrow s \nwarrow s \uparrow q$.

PROPOSITION 5.10. *If $p \uparrow q$, then:*

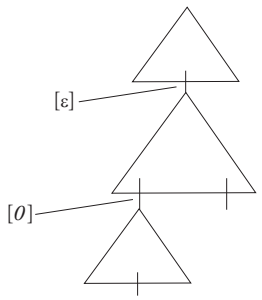
1. *$p \uparrow q$; or*
2. *$p \nearrow q$; or*
3. *$p \llcorner q$; or*
4. *for some states r_1, r_2:*
   - *(a) $p \uparrow r_1 \nearrow r_1 \uparrow q$ and $p \uparrow r_2 \llcorner r_2 \uparrow q$; or*
   - *(b) $p \uparrow r_1 \nearrow r_1 \nearrow r_2 \nearrow r_2 \uparrow q$; or*
   - *(c) $p \uparrow r_1 \llcorner r_1 \nwarrow r_2 \llcorner r_2 \uparrow q$.*

Proof. If neither case 2 nor 3 holds, then by Proposition 5.6 we must have both $p \nearrow q$ and $p \nwarrow q$. Let $R^{\uparrow}(p,q)$ be the set of states r such that $p \uparrow r \uparrow r \uparrow q$ holds; this set is nonempty by Lemma 5.8. Let $R^{\nearrow}(p,q) \subseteq R^{\uparrow}(p,q)$ be the set of those states r in $R^{\uparrow}(p,q)$ such that $r \nearrow r$. Similarly we define $R^{\nwarrow}(p,q)$. Note that by Proposition 5.6,

$$R^{\uparrow}(p,q) = R^{\nearrow}(p,q) \cup R^{\nwarrow}(p,q) \; .$$

Now a case analysis proves the lemma:

- If $R^{\nearrow}(p,q) \cap R^{\nwarrow}(p,q)$ is nonempty, then item 1 holds;
- $R^{\nearrow}(p,q)$ is empty. By $p \nearrow q$, in the pattern $\Delta_1[\Delta_2[\Delta_1, *]]$, there is a run from state p' in port 0 to state q' in port ε that does not visit port 1, where $p \rightarrow_{\varepsilon} p'$ and $q' \rightarrow_{\varepsilon} q$. This run uses two states q_1, q_2 at the intermediate junction nodes $[\varepsilon]$ and $[0]$. These states satisfy $p \uparrow q_1 \nearrow q_2 \uparrow q$:



The set $R^{\nearrow}(p, q_1)$ is empty, since it is included in $R^{\nearrow}(p, q)$. Furthermore, since $p \uparrow q_1$ holds, the set $R^{\uparrow}(p, q_1)$ must be nonempty and consequently $R^{\nwarrow}(p, q_1)$ is nonempty by Proposition 5.6. Therefore we can choose r_1 in $R^{\nwarrow}(p, q_1)$ which is not in $R^{\nearrow}(p, q_1)$. This means

$$p \uparrow r_1 \not\downarrow r_1 \uparrow q_1 \ .$$

Similarly, there is some state r_2 such that

$$q_2 \uparrow r_2 \not\downarrow r_2 \uparrow q$$

Since $q_1 \nearrow q_2$, we have $r_1 \nearrow r_2$, and therefore item (b) holds;

- $R^{\nwarrow}(p, q)$ is empty. By a reasoning as above, we have item (c);
- Finally, if $R^{\nearrow}(p, q) \cap R^{\nwarrow}(p, q)$ is empty, yet both $R^{\nearrow}(p, q)$ and $R^{\nwarrow}(p, q)$ are nonempty then clearly item (a) holds.

□

The point of characterizing $\uparrow$ and $\downarrow$ is that these are the most basic types of move the automaton can make in a pattern expansion. Indeed, by Lemma 4.2, in order to move from one junction node to another, the automaton needs to traverse the Δ_2 pattern. Since the pattern Δ_2 can be seen as having Δ_1 plugged in each of its ports, each such traversal must employ one of the moves $\uparrow$ or $\downarrow$. But then we can use Proposition 5.10 in order to uncover other possible moves of the automaton.

When put together, Proposition 5.10, Lemma 5.9 and Lemma 5.5 give us some idea of how a tree-walking automaton can move upward within a pattern expansion: it may either get completely lost (by allowing a move from a node to any node above it, case 1 in Proposition 5.10), allow a depth-first search in some fixed direction and nothing else (cases 2 and 3), or, finally, do some depth-first searches coupled with moves in opposing directions (case 4).

6. Moves. In the previous section, we analyzed the way an automaton can move through single instances of the basic patterns Δ_0, Δ_1 and Δ_2. In this section, we consider runs through larger objects built as compositions of Δ_1 and Δ_2 patterns, i.e., pattern expansions. We are especially interested in the way the tree-walking automaton can go from one junction leaf of such an expansion to another. Recall Lemma 4.1, which said that any loop in a junction node can be replaced by the $\rightarrow_\varepsilon$ relation, and hence swallowed by the γ relations. This means, that any run between two junction nodes in a pattern expansion can be assumed to be a nonlooping sequence of steps consistent with the γ relation. In a tree, a nonlooping path is the shortest possible path.

Note that all patterns considered in this section and the previous ones use only the blank symbol. The **a** label will only be introduced in the final section, Section 7. From this perspective, the sections leading up to Section 7 can be seen as an analysis of runs that never see the **a** label.

6.1. Pattern paths and moves. Before proceeding with a classification of possible moves, we introduce a more convenient syntax for describing runs between junction nodes within a pattern expansion. Essentially, by Lemma 4.2, such a run can be decomposed as a sequence of moves taken from γ_{Δ_2}. Moreover, by closure properties of γ_{Δ_2}, the runs can be assumed to have a certain normal form.

A *pattern path* (path for short) is a word over the alphabet $\{\varepsilon, 0, 1\} \times \{\varepsilon, 0, 1\}$. A pattern path can be used to go from one junction node to another in a pattern

expansion in the following manner. An empty pattern path can stay in the same junction node, while the pattern path $\pi \cdot (a, b)$ can go from $[v]$ to $[u \cdot b]$ if its prefix π can go from $[v]$ to $[u \cdot a]$. We write $v \to_\pi w$ when π can go from $[v]$ to $[w]$.

The $\to_\pi$ relation can also be annotated with states of the automaton. Given states p, q, and a pattern path $\pi = (a_1, b_1) \cdots (a_n, b_n)$, we write $p \to_\pi q$ if there are states $p = r_1, \ldots, r_{n+1} = q$ such that for all $i = 1, \ldots, n$, the tuple (r_i, a_i, r_{i+1}, b_i) belongs to γ_{Δ_2}. In the special case of $\pi = \varepsilon$, we require $p \to_\varepsilon q$. Given two states p, q and two nodes v, w, we write $[p, v] \to_\pi [q, w]$ if both $p \to_\pi q$ and $v \to_\pi w$ hold.

A pattern path is called *normalized* if it is the shortest path between two junction nodes. For having more understandable normalized paths, we use the following abbreviations: $\swarrow = (\varepsilon, 0)$, $\searrow = (\varepsilon, 1)$, $\nearrow = (0, \varepsilon)$, $\nwarrow = (1, \varepsilon)$, $\curvearrowright = (0, 1)$, and $\curvearrowleft = (1, 0)$. Let us define the following languages:

$$\begin{aligned} \mathrm{Up} &= (\nearrow + \nwarrow)^+, & \mathrm{Down} &= (\swarrow + \searrow)^+, \\ \mathrm{Left} &= (\mathrm{Up} + \varepsilon) \curvearrowleft (\mathrm{Down} + \varepsilon), & \mathrm{Right} &= (\mathrm{Up} + \varepsilon) \curvearrowright (\mathrm{Down} + \varepsilon), \\ \mathrm{Side} &= \varepsilon + \mathrm{Left} + \mathrm{Right}\ . \end{aligned}$$

The sets $\mathrm{Up}, \mathrm{Down}, \mathrm{Left}, \mathrm{Right}, \mathrm{Side}$ are called respectively the set of *upward*, *downward*, *left*, *right* and *sideway* paths. A pattern path is normalized iff it belongs to $\mathrm{Up} + \mathrm{Down} + \mathrm{Side}$. Given nodes of a tree v and w, $\pi(v, w)$ denotes the only normalized path such that $v \to_{\pi(v,w)} w$. As expected, for nodes v and w, $\pi(v, w) \in \mathrm{Up}$ iff w is strictly above v; $\pi(v, w) \in \mathrm{Down}$ iff w is strictly below v; $\pi(v, w) \in \mathrm{Left}$ iff w is to the left of v and; $\pi(v, w) \in \mathrm{Right}$ iff w is to the right of v. A set of normalized pattern paths is called a *move*; and we write vMw if $\pi(v, w) \in M$.

We will now show some results about the possible paths that the automaton can use when going from one node to another, these were mentioned after Lemma 4.2. We begin with the following lemma, which shows how paths correspond to runs of the automaton, at least as far as junction nodes are concerned:

LEMMA 6.1. *The following are equivalent for nodes v, w in a blank tree t.*

1. *There is a run in Δ_t from $[p, v]$ to $[q, w]$.*
2. *$[p, v] \to_{\pi(v,w)} [q, w]$ holds.*
3. *There is a run in Δ_t from $[p, v]$ to $[q, w]$ which visits only junction nodes $[u]$ such that $v \to_\pi u$ for some prefix π of $\pi(v, w)$.*

Proof. This is a generalization of Lemma 4.2, and the same proof works: any loop appearing in a run can be contracted using the $\to_\varepsilon$ relation. □

Since the normalized path connecting v and w does not depend on the tree t but only on the nodes v, w, we obtain the following:

COROLLARY 6.2. *Let v, w be nodes of a blank tree t. Whether or not there is a run from $[p, v]$ to $[q, w]$ in Δ_t depends only on $\pi(v, w)$ and not on t.*

The above corollary justifies the notation $[p, v] \to [q, w]$, where no particular path or tree is mentioned; it is equivalent to $p \to_{\pi(v,w)} q$. We will often be using this notation from now on.

DEFINITION 7. *For states p and q, we define $U(p, q)$ to be the set of upward paths π such that $p \to_\pi q$. Similarly, we define $D(p, q)$, $L(p, q)$, $R(p, q)$ and $S(p, q)$ for downward, left, right and sideways paths respectively.*

In particular, a direct consequence of this definition is that for two distinct nodes v and w and states p and q, $[p, v] \to [q, w]$ if and only if

$$\pi(v, w) \in U(p, q) \cup D(p, q) \cup S(p, q)\ .$$

The following lemma will be used several times; it transfers some properties of γ_{Δ_2} to equalities on the sets U, D, L, R. Its meaning is natural, but the statement as well as the proof are slightly clouded by the case of the normalized path ε.

LEMMA 6.3. *The move $R(p,q)$ is the union of $(U(p,p')+\varepsilon) \curvearrowright (D(q',q)+\varepsilon)$ for states p', q' satisfying $p \uparrow p' \curvearrowright q' \downarrow q$. A similar statement holds for L.*

Proof. We only do the case of R. We begin with the right-to-left inclusion. Let p', q' be states such that $p \uparrow p' \curvearrowright q' \downarrow q$, and let

$$\pi \in (U(p,p')+\varepsilon) \curvearrowright (D(q',q)+\varepsilon) \ .$$

We can write π as $\pi_1 \curvearrowright \pi_2$ with $\pi_1 \in U(p,p')+\varepsilon$ and $\pi_2 \in D(q',q)+\varepsilon$).

The first case is when both π_1, π_2 are empty and therefore $\pi = \curvearrowright$. By assumption, we have $p \uparrow p' \curvearrowright q' \downarrow q$ and hence $p \curvearrowright q$, by swallowing. Consequently $p \rightarrow_\pi q$ and $\pi \in R(p,q)$. If neither of π_1, π_2 is empty, then $\pi_1 \in U(p,p')$ and $\pi_2 \in D(q',q)$, which gives the desired result. The remaining cases where only one of π_1 or π_2 is empty are treated by combining the two first cases.

We now treat the left-to-right inclusion. Let π be in $R(p,q)$. By definition of Right, π can be written as $\pi_1 \curvearrowright \pi_2$ with $\pi_1 \in \mathrm{Up}+\varepsilon$ and $\pi_2 \in \mathrm{Down}+\varepsilon$. As above, we first consider the case when π_1, π_2 are both empty. In this case, we have $p \curvearrowright q$. But then, by looking at the path from port 0 to port 1 in the pattern $\Delta_2[\Delta_1, \Delta_1]$ which is equivalent to Δ_2, we can find states p', q' such that $p \uparrow p' \curvearrowright q' \downarrow q$, which completes the proof.

Assume now that both π_1, π_2 are nonempty. Let p', q' be the states such that $p \rightarrow_{\pi_1} p' \curvearrowright q' \rightarrow_{\pi_2} q$. We need to show that $p \uparrow p'$ and $q' \downarrow q$. But this follows from transitivity of $\downarrow, \uparrow$ and the inclusions $\swarrow, \searrow \subseteq \downarrow$ and $\nwarrow, \nearrow \subseteq \uparrow$. The remaining cases where only one of π_1 or π_2 is empty are treated by combining the two first cases. □

Furthermore, there exists a strong link between the set of upward moves (and by time symmetry downward moves) and the behaviors of the automaton exhibited in the previous section; this is the subject of the next lemma.

LEMMA 6.4. *If $p \uparrow q$ then $U(p,q) = \mathrm{Up}$. The analogous results hold for $\downarrow$, $\upharpoonright$, $\upharpoonleft$, $\downharpoonleft$ and $\downharpoonright$, the corresponding moves being respectively* Down, $\nwarrow^+$, $\nearrow^+$, $\swarrow^+$ *and* $\searrow^+$.

Proof. We treat the case $\upharpoonright$. If $p \upharpoonright q$, then by Lemma 5.9, there is a state r such that $p \upharpoonright r \upharpoonright r \upharpoonright q$. This shows $\nwarrow^+ \subseteq U(p,q)$. The opposite inclusion must also hold, since otherwise we would get $p \nearrow q$. □

The next lemma gives other required facts about $U(p,q)$ and $S(p,q)$.

LEMMA 6.5. *If $p \uparrow q$ then $\nwarrow^+ \subseteq U(p,q)$ or $\nearrow^+ \subseteq U(p,q)$. If $p \nearrow q$, then $\nwarrow^* \nearrow \subseteq U(p,q)$ or $\nearrow^+ \subseteq U(p,q)$. If $p \upharpoonleft p' \curvearrowright q$ then $\varepsilon \in S(p,q)$.*

Proof. First statement: for some r, $p \uparrow r \nwarrow r \uparrow q$ or $p \uparrow r \nearrow r \uparrow q$ by Lemma 5.8 and Proposition 5.6. Second statement: for some p', $p \uparrow p' \nearrow q$ must hold; then use the first statement. Third statement: by Lemmas 5.9 and 5.5, there exists $r, \bar{r}$ such that $p \uparrow \bar{r} \curvearrowleft r \downarrow r \circlearrowleft \bar{r} \uparrow p' \curvearrowright q$. Hence $p \rightarrow_\varepsilon q$, i.e., $\varepsilon \in S(p,q)$. □

As hinted by Proposition 5.10 and Lemmas 6.3 and 6.4, there are not so many ways that a sideway move can be done. The following eleven moves will play a special role below.

DEFINITION 8. *An* elementary move *is any one of the eleven moves in Figure 6.1.*

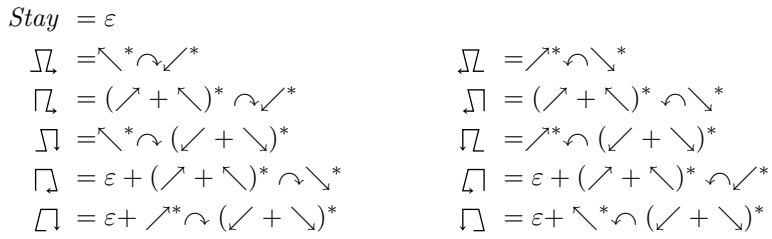


FIG. 6.1. *Elementary moves*

6.2. Move Offsets. This section is devoted to moves that only depend on the number of junction leaves between the source and destination, i.e., moves that do not really depend on the structure of the tree.

We number the leaves of a tree t from left to right, starting from 0. Formally, given a blank tree t and a leaf v of t, we denote by $\#_t(v)$ the number of leaves in the tree t that are lexicographically before v. For v and w leaves of t, we denote by $\#_t(v, w)$ the offset from v to w within t, i.e., the difference $\#_t(w) - \#_t(v)$. This number is positive when v is to the left of w. If v or w is not a leaf of t, then $\#_t(v, w)$ is not defined.

DEFINITION 9. *A* move offset *of two states p, q is an integer i such that for every tree t and leaves v and w of t, $\#_t(v, w) = i$ implies $[p, v] \to [q, w]$. We write* $\mathrm{moff}(p, q)$ *for the set of move offsets of p, q. We say that p, q admit a* shift *if* $\mathrm{moff}(p, q)$ *contains two successive integers from* $\{-2, -1, 0, 1, 2\}$.

The next lemma shows how move offsets can be described in terms of paths.

LEMMA 6.6. *For every states p, q,*

$$
\begin{aligned}
0 \in \mathrm{moff}(p,q) \quad & \textit{iff} \quad \varepsilon \in S(p,q)\ , \\
1 \in \mathrm{moff}(p,q) \quad & \textit{iff} \quad \nwarrow^* \curvearrowright \swarrow^* \subseteq S(p,q)\ , \\
2 \in \mathrm{moff}(p,q) \quad & \textit{iff} \quad \nwarrow^* \nearrow \nwarrow^* \curvearrowright \swarrow^* + \nwarrow^* \curvearrowright \swarrow^* \searrow \swarrow^* \subseteq S(p,q)\ , \\
-1 \in \mathrm{moff}(p,q) \quad & \textit{iff} \quad \nearrow^* \curvearrowleft \searrow^* \subseteq S(p,q)\ , \\
-2 \in \mathrm{moff}(p,q) \quad & \textit{iff} \quad \nearrow^* \nwarrow \nearrow^* \curvearrowleft \searrow^* + \nearrow^* \curvearrowleft \searrow^* \swarrow \searrow^* \subseteq S(p,q)\ .
\end{aligned}
$$

Proof. The case of 0 follows straight from the definition: if $\#_t(v, w) = 0$, then $v = w$ and therefore $\pi(v, w) = \varepsilon$ (and vice versa). The remaining cases follow by listing the paths that can connect nodes separated by $1, 2, -1, -2$ leaves respectively. □

In particular, directly from the definition of elementary moves, we deduce the following corollary.

COROLLARY 6.7. *Every elementary move has an offset among $-1, 0, 1$.*

A typical example of a move offset of 1 is the depth-first search.

LEMMA 6.8. *If $(p, \bar{p})$ is a left-to-right DFS then 1 is a move offset of $\bar{p}, p$. If $(p, \bar{p})$ is a right-to-left DFS then -1 is a move offset of $\bar{p}, p$.*

Proof. If $(p, \bar{p})$ is a left-to-right DFS then $\bar{p} \nwarrow \bar{p} \curvearrowright p \swarrow p$ holds. Thus the move $S(\bar{p}, p)$ contains $\nwarrow^* \curvearrowright \swarrow^*$ and the move offset 1 follows by Lemma 6.6. The right-to-left case is the same. □

The following lemma gathers a number of sufficient conditions for move offsets.

LEMMA 6.9. *For all states $\bar{p}$, $\bar{r}$ and q:*
(i) If $\bar{p} \downarrow \bar{p} \curvearrowright q$ then 0 is a move offset of $\bar{p}, q$.
(ii) If $\bar{p} \downarrow \bar{p} \curvearrowright q$ then 1 is a move offset of $\bar{p}, q$.
(iii) If $\bar{p} \downarrow \bar{p} \nearrow \bar{r} \downarrow \bar{r} \curvearrowright q$ then both 1 and 2 are move offsets of $\bar{p}$, q.
(iv) If $\bar{p} \downarrow \bar{p} \nwarrow \bar{r} \downarrow \bar{r} \curvearrowright q$ then both -1 and 0 are move offsets of $\bar{p}$, q.
(v) If $\bar{p} \downarrow \bar{p} \curvearrowright q \downarrow q$ then both -1 and 0 are move offsets of $\bar{p}$, q.

Proof.

(i) By Lemma 5.5, there is a state p such that $(p, \bar{p})$ is a right-to-left DFS. This gives, among others, $\bar{p} \curvearrowleft p$ and $p \circlearrowleft \bar{p}$. Together with the assumption that $\bar{p} \curvearrowright q$, we obtain $\bar{p} \rightarrow_\varepsilon q$, and hence 0 is a move offset of $\bar{p}, q$ by Lemma 6.6.

(ii) By Lemma 5.5, there is a state p such that $(p, \bar{p})$ is a left-to-right DFS. By Lemma 6.8, $\bar{p}, p$ has offset 1. From $p \swarrow p \circlearrowleft \bar{p} \curvearrowright q$, we have $p \downarrow q$. Using Lemma 6.6, it follows by swallowing that $\bar{p}, q$ also has offset 1.

(iii) The move offset 1 follows from the previous case, since $\bar{p} \downarrow \bar{p} \curvearrowright q$. By Lemma 6.6, the move offset 2 will follow once we show that $S(\bar{p}, q)$ contains both

$$\nwarrow^* \nearrow \nwarrow^* \curvearrowright \swarrow^* \quad \text{and} \quad \nwarrow^* \curvearrowright \swarrow^* \searrow \swarrow^* \ .$$

By Lemma 5.5, there are states p, r such that $(p, \bar{p})$ and $(r, \bar{r})$ are left-to-right DFS's. By $r \swarrow r \circlearrowleft \bar{r} \curvearrowright q$, we obtain $r \downarrow q$. Then by

$$\bar{p} \nwarrow \bar{p} \nearrow \bar{r} \nwarrow \bar{r} \curvearrowright r \swarrow r \downarrow q$$

we obtain that $S(\bar{p}, q)$ contains $\nwarrow^* \nearrow \nwarrow^* \curvearrowright \swarrow^*$.
It remains to show that $S(\bar{p}, q)$ contains $\nwarrow^* \curvearrowright \swarrow^* \searrow \swarrow^*$. This will follow once we prove $p \searrow r$, by

$$\bar{p} \nwarrow \bar{p} \curvearrowright p \swarrow p \searrow r \swarrow r \downarrow q \ .$$

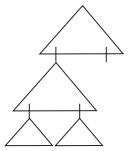
It thus remains to show $p \searrow r$. By $p \swarrow p \circlearrowleft \bar{p} \nearrow \bar{r}$, we get $p \rightarrow_\varepsilon \bar{r}$. Finally, if we consider the path in $\Delta_2[\Delta_1, *]$ witnessed by $p \swarrow p \rightarrow_\varepsilon \bar{r} \curvearrowright r$, we get the desired $p \searrow r$.

(iv) By item (i), there is a move offset of 0.
By Lemma 5.5, there are states p, r such that both $(p, \bar{p})$ and $(r, \bar{r})$ are right-to-left DFS's. Using the space-symmetric variant of the proof of $p \searrow r$ in item (iii), we obtain $p \swarrow r$. Now, by $p \swarrow r \circlearrowleft \bar{r} \curvearrowright q$, we have $p \downarrow q$. Consequently $\bar{p} \downarrow \bar{p} \curvearrowleft p \downarrow q$; and applying the space-symmetric version of (ii) we obtain an offset of -1 for $\bar{p}, p$.

(v) The offset 0 follows from item (i). By Lemma 5.5, there are states $p, \bar{q}$ such that both $(p, \bar{p})$ and $(q, \bar{q})$ are right-to-left DFS's.
Let us show that $\bar{p} \uparrow \bar{q}$. For this, we trace the following run

$$\bar{p} \curvearrowleft p \searrow p \circlearrowleft \bar{p} \curvearrowleft p \circlearrowleft \bar{p} \curvearrowright q \circlearrowleft \bar{q} \curvearrowleft q \circlearrowleft \bar{q} \nearrow \bar{q} \nearrow \bar{q}$$

that goes from state $\bar{p}$ in port 0 to state $\bar{q}$ in port ε of the following pattern, which is equivalent to Δ_1:



Now by $\bar{p} \uparrow \bar{q} \nearrow \bar{q} \curvearrowleft q \searrow q$, we obtain that $\nearrow^* \curvearrowleft \searrow^* \subseteq S(\bar{p}, q)$. By Lemma 6.6, we obtain offset -1.

□

6.3. Classification of moves. In this section we state and prove Proposition 6.10, which says that if $S(p,q)$ is nonempty then it is either a union of some of the eleven elementary moves, or there is a shift. This separation of cases is at the core of the argumentation of Section 7.

PROPOSITION 6.10. *If $S(p,q)$ is nonempty then p, q have a move offset in $\{-1, 0, 1\}$. Furthermore, either p, q admits a shift or $S(p,q)$ is a union of elementary moves.*

Proof. By Lemma 6.3, the move $S(p,q)$ is a finite union of sets of the form $\{\varepsilon\}$, which is an elementary move, or

$$(U(p,p') + \varepsilon) \curvearrowright (D(q',q) + \varepsilon) \qquad \text{where } p', q' \text{ satisfy } p \uparrow p' \curvearrowright q' \downarrow q \qquad \text{or}$$
$$(U(p,p') + \varepsilon) \curvearrowleft (D(q',q) + \varepsilon) \qquad \text{where } p', q' \text{ satisfy } p \uparrow p' \curvearrowleft q' \downarrow q\ .$$

Let $M \subseteq S(p,q)$ be a move of one of those forms, we prove that either $S(p,q)$ contains a shift, or there is an elementary move E such that $M \subseteq E \subseteq S(p,q)$. If this holds for all such moves M, we directly obtain the statement of the proposition, using Corollary 6.7 for the offset of elementary moves.

By symmetry, we only consider the case $M = (U(p,p') + \varepsilon) \curvearrowright (D(q',q) + \varepsilon)$. We now apply Proposition 5.10 to $p \uparrow p'$. This proposition distinguishes six cases, namely $\uparrow$, $\downharpoonleft$, $\downharpoonright$, (a), (b) and (c). Similarly, we can do the time-reversed reasoning for $q' \downarrow q$ and also consider six cases.

If we have (b) or (c) for $p \uparrow p'$, then by items (iii) and (iv) of Lemma 6.9 respectively, we get a shift. In case of (a), we get a shift by using items (i) and (ii). By the time-space-reverse variant of Lemma 6.9, the same happens if (a), (b) or (c) holds for $q' \downarrow q$.

Only the other cases remain. There are nine possibilities:

- if $p \downharpoonleft p'$ and $q' \downharpoonright q$, we get $M = \sqcup\!\!\downharpoonright$ by Lemma 6.4. Similarly, if $p \downharpoonleft p'$ and $q' \downarrow q$, we get $M = \sqcup\!\!\downarrow$, and when $p \uparrow p'$ and $q' \downharpoonright q$, we have $M = \sqcap\!\!\downharpoonright$.
- if $p \downharpoonright p'$ and $q' \downarrow q$, we have $M \subseteq \sqcap\!\!\downarrow$ by Lemma 6.4. Furthermore, using Lemmas 6.4 and 6.5, we get $\sqcap\!\!\downarrow \subseteq S(p,q)$. Similarly, if $p \uparrow p'$ and $q' \downharpoonleft q$, we obtain $M \subseteq \sqcap\!\!\downharpoonleft \subseteq S(p,q)$.
- if $p \downharpoonleft p'$ and $q' \downharpoonleft q$ then p, q admit a shift. Indeed, 1 is a move offset of p, q: by Lemma 5.9, $p \uparrow p'' \downharpoonleft p'' \uparrow p'$, and item (ii) of Lemma 6.9 gives offset 1. Offset 0 is also obtained by Lemma 5.9 and the time-space-reverse variant of item (i) (i.e., if $\bar{p} \curvearrowright q \downharpoonleft q$ then $\bar{p}, q$ have offset 0). Similarly for $p \downharpoonright p'$ and $q' \downharpoonright q$.

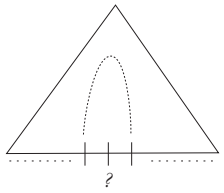
- if $p \lfloor p'$ and $q' \rfloor q$ then p, q admit a shift by item (v) of Lemma 6.9.
- if $p \uparrow p'$ and $q' \downarrow q$ then p, q clearly admit a shift; 1 and 2 are move offsets.

□

6.4. Right-skipping moves. The results of Section 6.4 concern right-skipping moves. A move is called *right-skipping* if it contains an element of

$$\mathrm{Right} \setminus \textstyle\coprod_{\rightarrow} .$$

A right-skipping move can go the right in a pattern while omitting (skipping) the junction leaf immediately to the right:



We say that states p, q are *right-skipping* if the move $R(p, q)$ is right-skipping. A *left-skipping* move is defined in the same fashion.

LEMMA 6.11. *Let p, q be states with a maximal move offset k. Let u, v be leaves of a tree t, with $\#_t(u, v) > k$. If $[p, u] \to [q, v]$, then p, q are right-skipping.*

Proof. Note that if the offset k can be arbitrarily large—i.e. there is no maximal offset k—then p, q are right-skipping straight from the definition, thanks to any move offset $k \geq 2$. By the first clause of Proposition 6.10, $k \geq -1$, and so $\#_t(u, v)$ is at least 0. Furthermore, it cannot be 0, since otherwise we would have $p \to_\varepsilon q$, and therefore $0 \in \mathrm{moff}(p, q)$; this would give $k \geq 0$, a contradiction with $\#_t(u, v) > k$.

If $\#_t(u, v)$ is at least 2, then p, q are right-skipping by definition; the remaining case is 1. The path $\pi(u, v)$ witnesses $p \curvearrowright q$, and consequently the existence of p', q' such that $p \uparrow p' \curvearrowright q' \downarrow q$. By Proposition 5.6, either $p \nwarrow p'$ or $p \nearrow p'$ must hold. If $p \nearrow p'$, then $\nearrow\!\!\curvearrowright \in R(p, q)$ is a witness for p, q being right-skipping. Otherwise, $p \rfloor p' \curvearrowright q$. By Lemma 5.9 and item (ii) of Lemma 6.9, p, q has offset 1, which gives $k \geq 1$; a contradiction with $\#_t(u, v) > k$. □

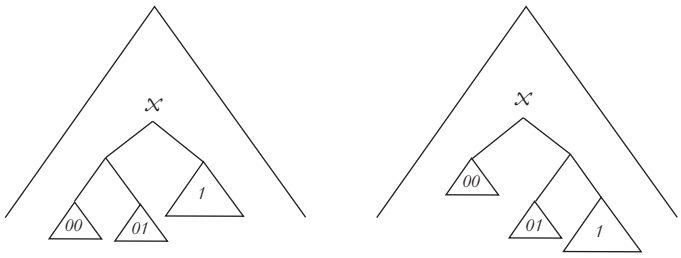


FIG. 7.1. *Rotating at node x*

7. The Rotation. We have now gathered enough information about runs of the automaton that never see the label **a**. In this section we consider runs that do

see $\mathbf{a}$, and conclude the proof of Theorem 2. We will show that the tree-walking automaton $\mathcal{A}$ cannot detect a well-placed rotation in a large balanced tree.

We proceed as follows. We start with a blank balanced binary tree T of large even depth. Clearly all leaves of T are at even depth, and therefore $\mathcal{A}$ must accept the tree $\Delta^{\mathbf{a}}_T$. We then choose a pivot node x in T and perform a rotation at that node. Rotation is the operation depicted in Fig. 7.1; it moves the subtrees rooted in $x00, x01$ and $x1$ to the new positions $x0, x10$ and $x11$. One can easily see that the resulting tree T' has some leaves at odd depth and hence $\mathcal{A}$ should not accept $\Delta^{\mathbf{a}}_{T'}$. We will, however, show that for a carefully chosen pivot, $\mathcal{A}$ does accept this tree, thereby completing the proof of Theorem 2 :

PROPOSITION 7.1. *The tree $\Delta^{\mathbf{a}}_{T'}$ is accepted by $\mathcal{A}$.*

First we describe how to properly choose the height of the tree T and the pivot in it. The remainder of the paper is then devoted to showing Proposition 7.1.

7.1. The pivot. The goal of this section is to construct a tree T, an accepting run of $\mathcal{A}$ over $\Delta^{\mathbf{a}}_T$, and a node x of T (the pivot), such that the properties of Definition 10 are satisfied. Essentially, these properties say that: the tree is balanced and large and the path leading to the pivot contains a zigzag. Furthermore, some undesirable parts of the accepting run do not use nodes below the pivot. These properties will be used in the remainder of the paper in order to prove that doing a rotation in the pivot x on the tree T gives a tree T' such that $\Delta^{\mathbf{a}}_{T'}$ is accepted by $\mathcal{A}$. We begin by defining the "undesirable" parts of the run. After that, we state Definition 10, and then we show that the tree and pivot can indeed be found, in Lemma 7.3.

Let t be a blank tree. Recall the definitions of junction and leaf configurations from Section 4.2. By distinguishing all the configurations whose node is either a junction leaf, or the root of $\Delta^{\mathbf{a}}_t$, every accepting run in $\Delta^{\mathbf{a}}_t$ can be decomposed into a sequence of the following form:

$$(q_1, v_1), \ldots, (q_n, v_n) \ , \tag{7.1}$$

where each v_i $(i = 1, \ldots, n)$ is either the root ε or $[u_i]$ for some leaf u_i of t, and in between two such configurations, no junction leaf nor the root is visited. In this case the run linking (q_i, v_i) to (q_{i+1}, v_{i+1}) is either

(a) a run visiting at least once the root of $\Delta^{\mathbf{a}}_t$ (and no $\mathbf{a}$-labeled leaf of $\Delta^{\mathbf{a}}_t$);
(b) a loop inside the $\Delta_1[\Delta_{\mathbf{a}}]$ subtree rooted in v_i, which is equivalent to $\Delta_{\mathbf{a}}$ (hence $q_i \circlearrowleft_{\mathbf{a}} q_{i+1}$); or
(c) a run from a junction leaf to another junction leaf in the pattern Δ_t which does not visit the root, (hence by Lemma 6.1, $[q_i, u_i] \to [q_{i+1}, u_{i+1}]$ holds, where $v_i = [u_i]$ and $v_{i+1} = [u_{i+1}]$).

Such a sequence is called a *rooted leaf run in* $\Delta^{\mathbf{a}}_t$. An *unrooted leaf run* is one that never uses a step of the form (a). By shortcutting each part of the run starting and ending in the root with the same state, we can safely assume that every rooted leaf run uses at most $2|Q|$ steps of the form (a); hence the greater part of a rooted leaf run is unrooted. Since an unrooted leaf run uses only leaf configurations, it can be written as $[q_1, u_1], \ldots, [q_n, u_n]$.

For junction configurations $[p, v]$ and $[q, w]$, we write $[p, v] \Rightarrow_t [q, w]$ if in the tree $\Delta^{\mathbf{a}}_t$ there is a run from $[p, v]$ to $[q, w]$ that does not visit the root. Note that when v, w are leaves, this means that there is an unrooted leaf run in $\Delta^{\mathbf{a}}_t$ from $[p, v]$ to $[q, w]$. As opposed to the relation $[p, v] \to [q, w]$, this run may depend on the tree t and not only on the nodes v and w.

We say that one state q is *leaf reachable* from another p if they can be connected by an unrooted leaf run in some tree, i.e., $[p, v] \Rightarrow_t [q, w]$ for some tree t and leaves v, w. Equivalently, q is leaf reachable from p if there exist $p = p_1, \ldots, p_n = q$ such that for every $i = 1, \ldots, n-1$, either $S(p_i, p_{i+1})$ is nonempty (which corresponds to case (c)) or $p_i \circlearrowleft_{\mathbf{a}} p_{i+1}$ holds (corresponding to case (b); recall the definition of $\circlearrowleft_{\mathbf{a}}$ from Figure 5.1). (The right-to-left part of this equivalence is a consequence of the existence of a move offset in $\{-1, 0, 1\}$ shown in Proposition 6.10 and is not a priori obvious). A *component* of the automaton is a maximal set of pairwise leaf reachable states; in other words, it is a strongly connected component of the directed graph with Q as nodes and an edge from p to q iff $S(p, q)$ is nonempty or $p \circlearrowleft_{\mathbf{a}} q$.

Let us consider a rooted leaf run ρ as in (7.1), which witnesses the acceptance of $\Delta_t^{\mathbf{a}}$ by $\mathcal{A}$. The main point in choosing the pivot is to restrict our attention to fragments of ρ that are unrooted leaf runs and only use states from one component. We say the run *changes components* below a node y of t if it contains two successive leaf configurations (q_i, v_i), (q_{i+1}, v_{i+1}) such that q_i and q_{i+1} are in different components and at least one of v_i, v_{i+1} is below $[y]$. The run is *rooted below* y if there is some i such that $v_i = \varepsilon$, and either v_{i-1} or v_{i+1} exists and is below $[y]$.

Definition 10. *We define the following properties for a node x in a blank balanced tree t with respect to a rooted leaf run of $\mathcal{A}$ in $\Delta_t^{\mathbf{a}}$,*

1. *the subtrees rooted in x and the children of x are $\log_2(|Q|)$-fractal (see below);*
2. *the subtree of x has depth larger than $4 + |Q| + 2\log_2(|Q|)$;*
3. *the node x is below the node 01010101;*
4. *the run does not change components below x;*
5. *the run is not rooted below x.*

Note that since the tree t is balanced, the number of leaves below a node v only depends on its depth. Let then v be a node of t, whose depth $|v|+1$ is at most $|x|+2$. From condition 2, it follows that the number of leaves in the subtree of v exceeds all the following thresholds (the constants D, E defined below will be used in the subsequent proofs):

$$|Q|, \quad D = |Q|(|Q| + 1), \quad E = 2D + 3|Q| \ . \tag{7.2}$$

We will refer to the above property later on in the paper.

We will now proceed to show that such a run and a node (called the pivot) can be found (Lemma 7.3) as long as t is a sufficiently large balanced tree of even depth. Before we do so however, we need to define what a fractal tree is.

Within a tree t, we distinguish five *characteristic types* of nodes: 1) the root, 2) the leftmost leaf, 3) the rightmost leaf, 4) the remaining leaves, and — for the sake of completeness — 5) the remaining nodes. We say a tree t' *simulates* a tree t, if for every two junction configurations in t that satisfy $[p, v] \Rightarrow_t [q, w]$, one can find two configurations satisfying $[p, v'] \Rightarrow_{t'} [q, w']$ such that v and v' have the same characteristic type, as well as w and w'. Given a natural number m, a tree is called *m-fractal* if it contains a proper subtree that simulates it and has depth larger than m.

Lemma 7.2. *For every natural number m, all balanced binary trees of sufficiently large depth are m-fractal.*

Proof. For a tree t and states p, q, we can calculate the characteristic types of nodes v, w satisfying $[p, v] \Rightarrow_t [q, w]$. This information is sufficient to see whether one tree simulates another. Moreover, it can be calculated by a deterministic bottom-up finite tree automaton. In case of a balanced binary tree, this information is a

(regular) property of its depth and is thus ultimately periodic. This means that there exist constants N and k such that every balanced blank tree of depth $n \geq N$ is simulated by the balanced blank tree of depth $n - k$. Hence every balanced tree of depth at least $\max\{m, N\}$ is m-fractal. □

LEMMA 7.3. *There exist a blank balanced binary tree T, an accepting run ρ of $\mathcal{A}$ in $\Delta_T^{\mathbf{a}}$ and a pivot x such that x satisfies properties 1 to 5 of Definition 10 with respect to ρ.*

Proof. Let N be more than the minimum depth of balanced tree obtained from Lemma 7.2, and also larger than the depth $|Q| \log_2(|Q|)$ from condition 2 in Definition 10. Let K be above $\log_2(4|Q|) + 1$. Finally, let T be a balanced blank tree of even depth larger than $N + K + 8$. The tree $\Delta_T^{\mathbf{a}}$ is accepted by $\mathcal{A}$ by hypothesis on $\mathcal{A}$.

Consider now an accepting run, and its representation as in (7.1),

$$(q_1, v_1), \ldots, (q_n, v_n),$$

where only the root and junction leaf configurations are displayed. We assume that in this run, the root of $\Delta_T^{\mathbf{a}}$ is seen at most $|Q|$ times. If this is not the case, some state appears twice in the root, hence a configuration is seen twice in the run; one can then shortcut the corresponding part of the run.

Let V_{root} be all those junction leaves v_i such that at least one of v_{i-1}, v_{i+1} is the root ε. One can easily see that the run is rooted below a node y if and only if $[y]$ is above some node in V_{root}. Likewise, let V_{cc} be all those junction leaves v_i such that the state q_i is in a different component than either q_{i-1} or q_{i+1} (or both). Again, the run changes component below a node y if and only if $[y]$ is above some node in V_{cc}. Combining the above, a node x satisfies properties 4 and 5 from Definition 10 if and only if x is not above some node in one of $V_{\text{root}}, V_{\text{cc}}$. By assumption on the run not visiting the root more than $|Q|$ times, the sets $V_{\text{root}}, V_{\text{cc}}$ have together at most $4|Q|$ nodes.

Let us count the number of nodes at depth $K + 8$ in T that are below 01010101. There are 2^{K-1} such nodes; i.e., more than $4|Q|$ by construction of K. In particular, there is a node x at depth $K + 8$ that satisfies conditions 3, 4 and 5. Furthermore, since the whole tree has depth at least $N + K + 8$, the subtree of x has depth at least N, and therefore x satisfies conditions 1 and 2. □

Detection of the rotation. From now till the end of the paper, the tree T, the run ρ and the pivot x are fixed according to Lemma 7.3. Let T' be the tree obtained from T by doing a rotation in the pivot x; this tree clearly contains leaves at odd depth. Our objective is to show that $\Delta_{T'}^{\mathbf{a}}$ is accepted by $\mathcal{A}$. For this, we have to show that the run can in some sense be replicated after the rotation. We will use properties 4 and 5 from Definition 10 to show that only unrooted leaf runs that do not change components need be considered.

We say that a component $\Gamma \subseteq Q$ of the automaton $\mathcal{A}$ *cannot detect the rotation* if for every two leaf configurations $[p, v]$, $[q, w]$ with p, q in Γ and v, w not below the pivot,

$$[p, v] \Rightarrow_T [q, w] \qquad \text{implies} \qquad [p, v] \Rightarrow_{T'} [q, w] \ . \tag{7.3}$$

Observe that it makes sense to speak about the configurations $[p, v]$ and $[q, w]$ in the tree T' since the leaves v and w are not below the pivot, and hence are not affected by the rotation.

Consequence for Proposition 7.1. Let us show that, under the assumption that no component can detect the rotation, we have Proposition 7.1.

Consider the run ρ obtained from Lemma 7.3, and its representation as in (7.1):

$$(q_1, v_1), \ldots, (q_n, v_n) \ .$$

Recall how we classified each of the runs linking (q_i, v_i) to (q_{i+1}, v_{i+1}), for $i = 1, \ldots, n-1$, according to their form, (a), (b) or (c). Define also u_i to be the node of T to be such that $v_i = [u_i]$, when possible, for $i = 1, \ldots, n$.

We claim the following: for $i < j$ in $\{1, \ldots, n\}$ such that both v_i and v_j are not below $[x]$, there is a run of the automaton $\mathcal{A}$ from (q_i, v_i) to (q_j, v_j) in $\Delta^{\mathbf{a}}_{T'}$. Since neither $v_1 = \varepsilon$ nor $v_n = \varepsilon$ is below $[x]$, our claim yields a run in $\Delta^{\mathbf{a}}_{T'}$ that goes from (q_1, ε) to (q_n, ε). Since the original run from (q_1, ε) to (q_n, ε) in $\Delta^{\mathbf{a}}_{T}$ was accepting, then q_1 is initial and q_n is final, and therefore the tree $\Delta^{\mathbf{a}}_{T'}$ is accepted by the automaton.

The proof of the claim is by induction on $j - i$. The induction step is obvious, and follows by concatenating runs. The nontrivial case is the base case of the induction, when for every k with $i < k < j$, the node v_k is below the pivot x. We now prove the claim for this case.

Consider first the case when $i+1 = j$. We now look at the form of the subrun that goes from (q_i, v_i) to (q_{i+1}, v_{i+1}). If this subrun is of the form (b), it can be directly replicated on $\Delta^{\mathbf{a}}_{T'}$ without change. If the subrun from (q_i, v_i) to (q_{i+1}, v_{i+1}) is of the form (c), then we use Corollary 6.2, which shows that a similar run can be used in $\Delta^{\mathbf{a}}_{T'}$ from configuration (q_i, v_i) to (q_{i+1}, v_{i+1}). The last remaining case is when the subrun from (q_i, v_i) to (q_{i+1}, v_{i+1}) is of the form (a), and in particular either $v_i = \varepsilon$, or $v_j = \varepsilon$, or both hold. If $v_i = \varepsilon$ but $v_j \neq \varepsilon$, we decompose the run from (q_i, ε) to (q_j, v_j) into a run from (q_i, ε) to $[q', \varepsilon]$, which does not visit more than once the nodes ε and $[\varepsilon]$, followed by a run from $[q', \varepsilon]$ to $[q_j, u_j]$, which does not visit ε nor a junction leaf other than $[u_j]$. This second piece of run can be reused in $\Delta^{\mathbf{a}}_{T'}$ according to Corollary 6.2. Hence there is a run in $\Delta^{\mathbf{a}}_{T'}$ from (q_i, v_i) to (q_j, v_j). The case $v_i \neq \varepsilon$ and $v_j = \varepsilon$ is obtained by time-symmetry. Finally, when $v_i = v_j = \varepsilon$, either the run does not visit $[\varepsilon]$, and can directly be reused in $\Delta^{\mathbf{a}}_{T'}$, or it visits $[\varepsilon]$. In this latter case, let q' be the first, and q'' be the last state assumed by the run when visiting $[\varepsilon]$. The piece of run between $[q', \varepsilon]$ to $[q'', \varepsilon]$ does not visit any junction leaf, and consequently by Lemma 4.1, there is a similar run which does not visit any junction node other than $[\varepsilon]$. This run can be reused in $\Delta^{\mathbf{a}}_{T'}$ to obtain a run from (q_i, v_i) to (q_j, v_j).

Otherwise, we have $i + 1 < j$. For all k with $i < k < j$, the node v_k is below $[x]$, which together with condition 5 of Definition 10 gives that no subruns of form (a) happen between (q_i, v_i) and (q_j, v_j). Consequently, all the nodes v_k are junction leaves, and the run from (q_i, v_i) to (q_j, v_j) is an unrooted leaf run:

$$[q_i, u_i] \Rightarrow_T [q_j, u_j] \ .$$

Furthermore, by condition 4 of Definition 10, the states q_i and q_j belong to the same component Γ of the automaton. Also, by hypothesis on i and j, neither u_i nor u_j is below x. It follows that we can apply our assumption that no component can detect the rotation, and get a run corresponding to:

$$[q_i, u_i] \Rightarrow_{T'} [q_j, u_j] \ .$$

This completes the proof of the claim, and consequently of Proposition 7.1. What remains to be done is to establish that no component of the automaton can detect the rotation. The remainder of this paper is dedicated to establishing it.

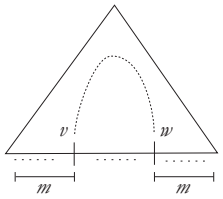


FIG. 7.2. *An m-run offset*

Using Proposition 6.10, we divide all components into two categories: components with a shift, i.e., containing two states which admit a shift; and components without a shift. Proposition 7.1 is then proved in the two following Sections 7.2 and 7.3 for each of the two categories separately.

7.2. Components with a shift cannot detect the rotation. In this section we fix a component Γ with a shift and prove that it cannot detect the rotation. In order to do this, we extend the definition of move offsets to *run offsets*, where more than one move can be used. The idea is that the shift in the component can be exploited to allow the automaton to move around the tree in an almost arbitrary fashion, independently of the structure of T.

A run offset between state p and state q is defined similarly to a move offset:

DEFINITION 11. *Given a natural number $m \geq 0$ called the* safety margin*, an* m-run offset *of states p, q is an integer i such that $[p, v] \Rightarrow_t [q, w]$ holds for every two leaves v, w of a tree t where $\#_t(v, w) = i$ and v,w have both at least m leaves to their left and right. We write $\mathrm{roff}_m(p, q)$ for the set of m-run offsets of p, q.*

We remark here, slightly ahead of time, that a consequence of x being below 01010101 is that all nodes below the pivot, and even some nodes not below the pivot, have at least m leaves to their left and right, for fairly large values of m. This means that those nodes are suitable for using run offsets.

The differences between move offsets and run offsets are that: 1) we replace $\rightarrow$ with $\Rightarrow_t$ (which depends on t and can read the label $\mathbf{a}$); and 2) the leaves v, w must have a "safety margin" of at least m leaves to their left and to their right, see Figure 7.2.

We now list some basic properties of roff, which hold for any given states p, q, r. First, $\mathrm{moff}(p, q)$ is included in $\mathrm{roff}_0(p, q)$. Furthermore, if $p \circlearrowleft_{\mathbf{a}} q$ holds, then 0 belongs to $\mathrm{roff}_0(p, q)$. Also, if both $i, j \leq 0$ or both $i, j \geq 0$, then

$$i \in \mathrm{roff}_m(p, q),\ j \in \mathrm{roff}_m(q, r) \qquad \Rightarrow \qquad i + j \in \mathrm{roff}_m(p, r)\ .$$

Finally, if $ij \leq 0$ (i.e., if i, j are of opposite sign), then

$$i \in \mathrm{roff}_m(p, q),\ j \in \mathrm{roff}_m(q, r) \qquad \Rightarrow \qquad i + j \in \mathrm{roff}_{m+\min(|i|,|j|)}(p, r)\ .$$

In particular, if $j \in \{-1, 0, 1\}$ then $i + j \in \mathrm{roff}_{m+1}(p, r)$.

A consequence of these properties, together with Proposition 6.10 is:

LEMMA 7.4. *For every component Γ and states p, q in Γ, $\mathrm{roff}_{|Q|-1}(p, q)$ is nonempty, and contains a value k with $|k| < |Q|$.*

Proof. Since p and q are in the same component, there exists a sequence of states $p = r_1, \ldots, r_n = q$ such that for all $i = 1, \ldots, n-1$, either $r_i \circlearrowleft_{\mathbf{a}} r_{i+1}$ holds, or the

move $S(r_i, r_{i+1})$ is nonempty. Without loss of generality, we assume that no state is seen twice in this sequence, and therefore $n \leq |Q|$.

By induction on $i = 1, \ldots, n$, we will show that $\mathrm{roff}_{i-1}(r_1, r_i)$ contains a value k with $|k| < n$. For $i = n \leq |Q|$, the statement of the lemma follows. For $i = 1$, we naturally have 0 in $\mathrm{roff}_0(r_1, r_1)$. By induction hypothesis, the set $\mathrm{roff}_{i-2}(r_1, r_{i-1})$ contains a value k with $|k| < i - 1$. If $r_{i-1} \circlearrowleft_{\mathbf{a}} r_i$ holds, then $\mathrm{roff}_{i-2}(r_1, r_i)$ also contains k, and therefore so does $\mathrm{roff}_{i-1}(r_1, r_i)$. On the other hand, if $S(r_i, r_{i+1})$ is nonempty, then $\mathrm{moff}(r_{i-1}, r_i)$ contains an offset $j \in \{-1, 0, 1\}$ by Property 6.10. In particular, the offset j belongs to $\mathrm{roff}_0(r_{i-1}, r_i)$. Using the properties described above, we obtain that $k + j$ belongs to $\mathrm{roff}_{i-1}(r_1, r_i)$. This concludes the induction step, since $|k + j| < i$.
□

For a safety margin m and a natural number d called the *threshold*, a pair of states p, q is called an (m, d)-*right-teleport* if $\mathrm{roff}_m(p, q)$ contains all integers no smaller than d. An (m, d)-*left-teleport* is when $\mathrm{roff}_m(p, q)$ contains all the integers smaller than $-d$. An m-*full-teleport* corresponds to $\mathrm{roff}_m(p, q)$ containing all integers. Remark that if p, q is an (m, d)-right-teleport and q, r is an (m, d)-left-teleport, then p, r is an $(m + d)$-full-teleport.

Recall the constant $D = |Q|(|Q| + 1)$ defined after Definition 10.

LEMMA 7.5. *If a component Γ contains a shift, either all pairs of states in Γ are $(2|Q|, D)$-right-teleports, or all are $(2|Q|, D)$-left-teleports.*

Proof. Let q_1, q_2 be states in the component Γ that admit a shift, i.e. have two consecutive move offsets $i, i + 1 \in \{-2, -1, 0, 1, 2\}$. By adding $i, i + 1$ to the offset obtained by applying Lemma 7.4 to the pair q_2, q_1, we infer that the cycle q_1, q_1 admits two consecutive $|Q|$-run offsets k and $k + 1$ with $|k|, |k + 1| \leq n + 1 \leq |Q| + 1$.

Without loss of generality, let us assume that $0 \leq k \leq n$ and $k' = k + 1$. Let

$$m \geq |Q|(|Q| - 1) \geq k(k - 1) \ .$$

We will show that m belongs to $\mathrm{roff}_{|Q|}(q_1, q_1)$. Indeed, the number m can be written as $\alpha k + \beta$ with $\beta \in \{0, \ldots, k - 1\}$. In particular,

$$m = (\alpha - \beta)k + \beta(k + 1) \ .$$

Let us remark that since $m \geq k(k - 1)$, then $\alpha \geq k - 1$, and consequently $\alpha - \beta \geq 0$. Hence by using $\alpha - \beta$ times the run offset k and β times the run offset $k' = k + 1$, one obtains

$$m \in \mathrm{roff}_{|Q|}(q_1, q_1) \ .$$

This proves that q_1, q_1 is a $(|Q|, |Q|(|Q| - 1))$-right-teleport.

Once one pair of states q_1, q_1 is a right-teleport, the same can be shown for all other state pairs in the component Γ. Indeed, we will show that any two states p, q in Γ are a $(2|Q|, |Q|(|Q|+1))$-right-teleport, which completes the proof of the lemma. We need to show that the automaton can go from $[p, v]$ to $[q, w]$. First, using Lemma 7.4, the automaton can go from $[p, v]$ to a configuration of the form $[q_1, u_1]$, where u_1 is a leaf separated from v by at most $|Q|$ leaves (note that u_1 may be to the left or to the right of v). In the same way, there is a leaf u_2, separated from w by at most $|Q|$ leaves, such that the automaton can go from $[q_1, u_2]$ to $[q, w]$. If the leaves v, w

have safety margins of $2|Q|$, then the leaves u_1, u_2 have safety margins of at least $|Q|$. Furthermore, if w was at least

$$|Q|(|Q|-1) + 2|Q| = |Q|(|Q|+1) = D$$

leaves to the right of v, then u_2 is at least $|Q|(|Q|-1)$ leaves to the right of u_1. Therefore the $(|Q|, |Q|(|Q|-1))$-right-teleport q_1, q_1 can be used to go from $[q_1, u_1]$ to $[q_1, u_2]$. This completes the proof that p, q is a $(2|Q|, D)$-right-teleport.

The left-teleport is obtained in the case when k is negative. □

For the remainder of this section (Section 7.2), we assume that the second case in the above lemma holds, i.e., all pairs of states are $(2|Q|, D)$-left-teleports. The case of right-teleports is symmetric. We now proceed to show that the component Γ cannot detect the rotation, i.e., that the implication (7.3) holds for any two leaves v, w not below the pivot, and any two states p, q of the component Γ.

Consider all the leaf configurations of the unrooted leaf run from $[p, v]$ to $[q, w]$:

$$[p, v] = [r_0, u_0] \Rightarrow_T [r_1, u_1] \Rightarrow_T \cdots \Rightarrow_T [r_n, u_n] \Rightarrow_T [r_{n+1}, u_{n+1}] = [q, w] \tag{7.4}$$

Without loss of generality we can assume that all the leaves $u_1, \ldots, u_n$ are below the pivot; the other parts of the run can be easily replicated in T' (as explained at the end of Section 7.1). Note that since Γ is a component, all the states $r_0, \ldots, r_{n+1}$ belong to Γ.

We will do a case analysis. We say the leaf run from $[p, v]$ to $[q, w]$ satisfies property (*) if for some $0 \leq i < j \leq n+1$, the leaf u_j is at least $|Q|$ leaves to the right of u_i, i.e., $\#_T(u_i, u_j) \geq |Q|$. We do the proof first for leaf runs that do not satisfy this property, and then for those that do.

7.2.1. Leaf runs not satisfying (*). Recall that we assume that Γ is a component such that every pair of states in it is a left-teleport. We make a case distinction depending on the relative position of v and w with respect to the pivot.

If v is to the left of the pivot and w is to its right, then all the leaves below the pivot separate v from w. By (7.2), there are more than $|Q|$ of these leaves, which contradicts our assumption on property (*) failing.

Consider now the case when v is to the right of the pivot and w is to the left. If v has more than $2|Q|$ leaves to its right and w has more than $2|Q|$ leaves to its left, we can use the $(2|Q|, D)$-left-teleport and go from $[p, v]$ to $[q, w]$ independently of the rotation, since there are at least D leaves below the pivot x, thanks to (7.2). Let us assume now that v has less than $2|Q|$ leaves to its right and w has less than $2|Q|$ leaves to its left. By (7.2), node 11 has more than $2|Q|$ leaves in its subtree. In particular, if there are less than $2|Q|$ leaves to the right of v, then v must be below 11. Since u_1 is below the pivot x, hence below 01, the path going from $[p, v]$ to $[r_1, u_1]$ is of the form $\pi \nwarrow \curvearrowleft \searrow \pi' \in S(p, r_1)$. It follows that $\pi \curvearrowleft \pi'$ also belongs to $S(p, r_1)$, by swallowing. Hence, the automaton can go from $[p, v]$ to $[r_1, v']$ for some v' below 10. In the same way, if w has less than $2|Q|$ leaves to its left, one shows that there exists a leaf w' below 001 such that the automaton can go in one move from $[r_n, w']$ to $[q, w]$. We are in the situation where v' has more than $2|Q|$ leaves to its right, w' has more than $2|Q|$ leaves to its left, and there are more than D leaves between w' and v'. Now we can use the $(2|Q|, D)$-left-teleport to go from $[r_1, v']$ to $[r_n, w']$. Furthermore, since v', w' are not below the pivot, and the teleport is used to jump over the leaves below the pivot, the resulting run cannot detect the rotation. The other cases are a combination of the two previous ones.

Next we consider the case for which both v and w are to the right of the pivot. Since condition (*) is not satisfied, all leaves $u_1, \ldots, u_n$ are at most $|Q|$ leaves away from w. In particular all these leaves are below $x1$ since the subtree rooted in $x1$ contains at least $|Q|$ leaves by (7.2). Let f be the unique bijection between the leaves

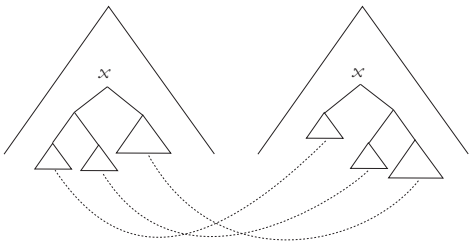


FIG. 7.3. *The bijection f*

of T and the leaves of T' that preserves the leaf numbering (i.e., $\#_T(v) = \#_{T'}(f(v))$, see Fig. 7.3). Note that f is the identity function on leaves not below the pivot; in particular $f(v) = v$ and $f(w) = w$. We claim that the unrooted leaf run from $[p, v]$ to $[q, w]$ can be replicated in the tree T' as follows:

$$[p, v] \Rightarrow_{T'} [r_1, f(u_1)] \Rightarrow_{T'} \cdots \Rightarrow_{T'} [r_n, f(u_n)] \Rightarrow_{T'} [q, w] \ .$$

Since all leaves $u_1, \ldots, u_n$ are located below $x1$, the path connecting $f(u_i)$ to $f(u_{i+1})$ is identical to the one connecting u_i to u_{i+1}, for $i = 1, \ldots, n-1$. It follows that the run from $[r_1, f(u_1)]$ to $[r_n, f(u_n)]$ is valid in T'. Only the first and last steps remain to be considered. We only do the first one, the other being time-symmetric. Consider the first step in the leaf run, when the automaton goes from $[p, v]$ to $[r_1, u_1]$. Since v is to the right of the pivot and u_1 is below the pivot, the path from v to u_1 is of the form

$$\pi \ \curvearrowleft \ \pi_1\pi_2 \qquad \text{with} \quad \pi \in \{\nwarrow, \nearrow\}^* \quad \text{and } \pi_1, \pi_2 \in \{\swarrow, \searrow\}^* \ .$$

Here π_1 is chosen so that $\pi \curvearrowleft \pi_1$ leads from v to the pivot, while π_2 leads from the pivot to u_1. Since u_1 is at a distance at most $|Q|$ from v, it is separated by at most $|Q|$ leaves from the rightmost leaf below the pivot. Therefore, the only left turns $\swarrow$ in π_2 happen in its last $\log_2(|Q|)$ letters. It follows from condition 2 of Definition 10, that π_2 has a prefix $\searrow^k$ with $k \geq |Q|$. Hence some state r has to be used twice in the prefix. Since $\searrow$ is transitive, we deduce that

$$\pi \ \curvearrowleft \pi_1 \ \searrow \ \pi_2$$

also belongs to the move $S(p, r_1)$. But this path is the one linking v to $f(u_1)$ in T', proving $[p, v] \Rightarrow_{T'} [r_1, f(u_1)]$.

The last remaining case is when both v and w are to the left of the pivot. This time we show that from a run of the form $\pi \curvearrowright \pi_1 \swarrow \pi_2$, one can deduce a run of the form $\pi \curvearrowright \pi_1\pi_2$, which can be used after the rotation. This case is in fact simpler since Definition 10 need not be invoked, but rather swallowing is used.

7.2.2. Leaf runs satisfying (*). In this case we will show that the component Γ contains a right-skipping move (or a right-teleport). After combining this with the left-teleports from our assumption, we will show that any two leaf configurations with

states from Γ and nodes below the pivot can be reached one from the other, which implies that Γ cannot detect the rotation. For this, we will use the assumption that the pivot is below 01010101.

Recall the constant $E = 2D + 3|Q|$ defined after Definition 10.

LEMMA 7.6. *Either all pairs of states in Γ are E-full-teleports, or some states r, r' in Γ are right-skipping.*

Proof. Recall that we are analyzing a run as in (7.4). For $i = 0, \ldots, n$, let k_i be the offset $\#_T(u_i, u_{i+1})$. If some k_i exceeds 1, the corresponding move $R(r_i, r_{i+1})$ is by definition right-skipping and we are done. Furthermore, if some k_i exceeds the maximal move offset of r_i, r_{i+1} (if that exists), then the corresponding move is right-skipping by Lemma 6.11. We assume neither case happens.

Since (*) is satisfied, there are $i < j$ such that $\#_T(u_i, u_j) \geq |Q|$. We will inspect the run from u_i to u_j and find in it a state used twice, the first configuration involved being to the left of the second one. Since $k_i, \ldots, k_{j-1} \leq 1$, we can assume that $\#_T(u_i, u_j)$ is exactly $|Q|$. Furthermore, if we choose i, j so that $j - i$ is minimal, all the leaves $u_{i+1}, \ldots, u_{j-1}$ are to the right of u_i and to the left of u_j. Since, without loss of generality, we can assume that no leaf is visited more than $|Q|$ times, and using the fact that the automaton has at least 2 states, we obtain $j - i < |Q|^2$.

Claim. Let $k = 1, \ldots, |Q|$. If $g < h$ are such that the sequence $r_g, \ldots, r_h$ contains at most k distinct states, and furthermore $\#_T(u_g, u_h) \geq k$, then there are $g' < h'$ in $\{g, \ldots, h\}$ such that $r_{g'} = r_{h'}$ and $\#_T(u_{g'}, u_{h'}) \geq 1$.

Proof. The proof is by induction on k. For $k = 1$ the statement is obvious. Consider now $k > 1$. We take a longest suffix $u_m, \ldots, u_h$ of $u_g, \ldots, u_h$ where the leaf u_g is not visited anymore (in particular, all leaves $u_m, \ldots, u_h$ are to the right of u_g). By assumption on all moves going at most one position to the right, the leaf u_m must be the leaf immediately to the right of u_g, and therefore $\#_T(u_m, u_h) \geq k - 1$. If the states $r_m, \ldots, r_h$ do not contain the state r_g, then we apply the induction hypothesis. Otherwise, the position among $m, \ldots, h$ where state r_g is used gives us the desired loop, since all nodes $u_m, \ldots, u_h$ are to the right of u_g. □

Using this claim with $g = i$ and $h = j$, together with $j - i < |Q|^2$, we find two indices $i' < j'$ such that $r_{i'} = r_{j'} = r$, $j' - i' < |Q|^2$ and $\#_T(u_{i'}, u_{j'}) \geq 1$. We will use this to show that there is also a right-teleport in the component. When combined with the left-teleport from our assumption on Γ, we will get a full-teleport.

By assumption on the values k_l, each pair r_l, r_{l+1} has some move offset $m_l \geq k_l$. (We use here the convention that there is a move offset $m_l = 0 \geq k_l$ when the run from $[r_l, u_l]$ to $[r_{l+1}, u_{l+1}]$ is of type (b), i.e. when $r_l \circlearrowleft_{\mathbf{a}} r_{l+1}$ holds.) By Proposition 6.10 each nonempty move has a move offset in $\{-1, 0, 1\}$, and we have $m_l \geq -1$. Furthermore, if $m_l > 1$ then r_l, r_{l+1} would be right-skipping, hence we have $-1 \leq m_l \leq 1$. Therefore,

$$1 \leq \#_T(u_{i'}, u_{j'}) = k_{i'} + \cdots + k_{j'-1} \leq m_{i'} + \cdots + m_{j'-1} < |Q|^2 \ .$$

(The last inequality is due to $j' - i' < |Q|^2$, and $m_l \leq 1$.) However, since the sum

$$m = m_{i'} + \cdots + m_{j'-1}$$

is composed only of move offsets, it must belong to $\mathrm{roff}_{|Q|^2}(r, r)$, and therefore also to $\mathrm{roff}_D(r, r)$, since

$$D = |Q|(|Q| + 1) > |Q|^2 \ .$$

We will now show that $\mathrm{roff}_{2D+2|Q|}(r,r)$ contains all integers, and therefore r,r is a $(2D+2|Q|)$-full-teleport. Indeed, let v and w be two leaves in a tree t, both with safety margin at least $2D+2|Q|$. We will show that the automaton can go from $[r,v]$ to $[r,w]$. Using $m \in \mathrm{roff}_D(r,r)$ (recall that $m \geq 1$), starting from $[r,v]$, we successively move to the right by steps of m leaves. We stop as soon when have reached a configuration $[r,u]$ with u located at least D leaves to the right of w, possibly stopping immediately. This leaf u is to the right of v, and hence has $2|Q|$ leaves to its left. Observe also that the leaf u is numbered at most

$$\max(\#_t(v), \#_t(w) + D + m - 1) \ .$$

If $u = v$, then u has $2|Q|$ leaves to its right. Otherwise if u has number at most $\#_t(w) + D + m - 1$, and then u has at least

$$2D + 2|Q| - (D + m - 1)$$

leaves to its right. Since $m < D$, this value is larger than $2|Q|$. We can therefore use the $(2|Q|, D)$-left-teleport – that all state pairs in Γ have by assumption – to go from $[r,u]$ to $[r,w]$.

Once one state pair r,r has been shown to be a full-teleport, the same can be argued for the other state pairs in Γ. This is done in the same way as in the last part of the proof of Lemma 7.5. As in that lemma, the safety margin must be increased by $|Q|$, hence the value $E = 2D + |Q|$ in the statement of the lemma. □

LEMMA 7.7. *For v, w leaves below the pivot in T', and p, q in Γ, $[p,v] \Rightarrow_{T'} [q,w]$.*

Before we proceed with the proof, we show how this implies that Γ cannot detect the rotation. Indeed, consider the leaf run

$$[p,v] \Rightarrow_T [r_1,u_1] \Rightarrow_T \cdots \Rightarrow_T [r_n,u_n] \Rightarrow_T [q,w]$$

that goes from $[p,v]$ to $[q,w]$. As mentioned before, we assume that all $u_1, \ldots, u_n$ are below the pivot, and hence only the first and last moves cross the pivot. Thanks to the yet unproved Lemma 7.7, it suffices to connect $[p,v]$ with some leaf configuration in T' below the pivot and also connect some leaf configuration in T' below the pivot with $[q,w]$. We will therefore show that there are leaves u'_1 and u'_n in T' below the pivot such that:

$$[p,v] \Rightarrow_{T'} [r_1,u'_1] \qquad \text{and} \qquad [r_n,u'_n] \Rightarrow_{T'} [q,w] \ .$$

Consider first the path from v to u_1 in T. This path first goes to the pivot, arriving there in some state s, and then moves down to the configuration $[r_1,u_1]$. This implies $s \downarrow r_1$. Hence, by the time-symmetric variant of Lemma 6.5, one of the sets $\swarrow^+$ or $\searrow^+$ is included in $D(s,r_1)$. In the first case we pick u'_1 to be the leftmost leaf below the pivot in T', while in the second case we take the rightmost one. The symmetric reasoning also works for u'_n.

Only the proof of Lemma 7.7 remains. Let then v, w be leaves below the pivot, and let p, q belong to Γ. We need to show that $[p,v] \Rightarrow_{T'} [q,w]$ holds.

By Lemma 7.6, either all pairs of states r, r' in Γ are E-full-teleports, or there exists a pair of states r, r' in Γ that is right-skipping. In the first case, there are more

than E leaves to the left and to the right of the pivot by (7.2), and hence to the left and to the right of both v and w. The E-full-teleport p, q is usable, and the statement of the lemma follows.

We now treat the case when some state pair r, r' in Γ is right-skipping. By assumption, all state pairs in the component Γ are $(2|Q|, D)$-left-teleports. Our strategy is to combine the left-teleports with the right-skipping move r, r'. First, we use the left-teleport to go from $[p, v]$ to $[r, u]$, with u being a specially chosen leaf to the left of the pivot. We then use the right-skipping move and the properties of u in order to move to $[r', u']$, with u' being a specially chosen leaf to the right of the pivot. Finally, we use the left-teleport to reach the configuration $[q, w]$. This process is illustrated in Fig. 7.4.

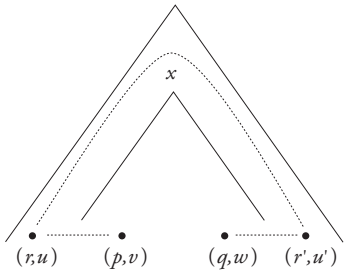


FIG. 7.4. *The leaf run witnessing* $[p, v] \Rightarrow_{T'} [q, w]$

We need to find leaves u and u' such that the above strategy works. This is the goal of the claim below. The first property in the statement allows to perform the right-skip while the last three allow us to use the left-teleport.

Claim. There exist leaves u, u' in T' such that:

- the path between u and u' belongs to $S(r, r')$ (r, r' taken from Lemma 7.6);
- there are at least D leaves between u and any node below the pivot;
- there are at least D leaves between any node below the pivot and u'; and
- there are at least $2|Q|$ leaves to the left and right of both u, u'.

Proof. The states r, r' are right-skipping by assumption. Hence, by definition of a right-skipping move and Lemma 6.3, one can find states s, s' with

$$r \uparrow s \curvearrowright s' \downarrow r' ,$$

such that either $U(r, s) \backslash \nwarrow^+$ or $D(s', r') \backslash \swarrow^+$ is nonempty. Consider first the case when $U(r, s) \backslash \nwarrow^+$ is nonempty. This means that $r \nearrow s$ holds. Hence, by Lemma 6.5, $U(r, s)$ contains either $\nwarrow^* \nearrow$ or $\nearrow^+$. By time-symmetry, in the case when $D(s', r') \backslash \swarrow^+$ is nonempty, $D(s', r')$ contains either $\searrow^+$ or $\searrow \swarrow^*$.

Altogether, we obtain that $R(r, r')$ contains at least one of the following five moves (using swallowing, we have simplified $\nwarrow^* \nearrow$ into $\nwarrow^*$, and $\searrow \swarrow^*$ into $\swarrow^*$):

$$\nearrow^* \curvearrowright \swarrow^*, \quad \nwarrow^* \curvearrowright \searrow^*, \quad \nearrow^* \curvearrowright \searrow^*, \quad \nwarrow^* \nearrow \curvearrowright \swarrow^*, \quad \text{or} \quad \nwarrow^* \curvearrowright \searrow \swarrow^* .$$

We treat each of these cases separately. For $\nearrow^* \curvearrowright \swarrow^*$, take u to be the leftmost leaf below 01 and u' to be the leftmost leaf below 011 (see Fig. 7.5). Clearly the path from u to u' belongs to $\nearrow^* \curvearrowright \swarrow^*$, hence the first property of the statement holds. By changing the leaves u and u', we obtain similarly the other cases: for $\nwarrow^* \curvearrowright \searrow^*$, take u to be the rightmost leaf below 0100 and u' to be the rightmost leaf below 010;

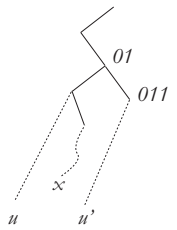


FIG. 7.5. *The move from u to u'*

for $\nearrow^*\curvearrowright\searrow^*$, take u to be the leftmost leaf below 01 and u' to be the rightmost leaf below 010; for the case $\nwarrow^*\nearrow\curvearrowright\swarrow^*$, take u to be the rightmost leaf below 0100 and u' to be the leftmost leaf below 011; for $\nwarrow^*\curvearrowright\searrow\swarrow^*$, take u to be the rightmost leaf below 0100 and u' to be the leftmost leaf below 01011. In each case, the path from u to u' belongs to $S(r, r')$ and therefore the first item of the statement is satisfied.

For the second item of the statement, note that in all five cases above, the leaf u is located below 0100. Using condition 3 of Definition 10, we also know that the pivot is below the node 01010101. Hence, all the leaves below 010100 are to the right of u and to the left of the pivot. Furthermore, by (7.2), there are more than D such leaves. The third item is similar: in all five cases, u' is below 011 or 01011. Hence, the leaves below 0101011 are to the right of the pivot and to the left of u'. And there are more than D of them.

The fourth point is obtained by the same kind of arguments. The leaves below 00 are all to the left of u and the leaves below 1 are all to the right of u'. And in each case there are more than $2|Q|$ of them. This completes the proof of the claim. □

This completes the proof that no component with a shift can detect the rotation.

7.3. Components without a shift cannot detect the rotation. In this section we consider a component Γ without shifts. This is the second and last case to be considered in the proof of Proposition 7.1. According to Proposition 6.10, every nonempty move $S(p, q)$ with p, q in Γ is a union of the elementary moves

$$\textit{Stay},\ \sqcup\!\!\rightarrow,\ \sqcup\!\!\downarrow,\ \sqcap\!\!\rightarrow,\ \sqcup\!\!\sqcap,\ \sqcup\!\!\downarrow,\ \sqcap\!\!\rightarrow,\ \sqcup\!\!\sqcap,\ \sqcup\!\!\sqcap \text{ and } \sqcap\!\!\downarrow$$

(see Fig. 6.1). Our strategy is as follows. First, we distinguish some elementary moves, called "adjacency moves". Then we show that all other moves can be simulated using adjacency moves. Finally, we show that a component where all moves are adjacency moves cannot detect the rotation.

Two paths in Right are called *right adjacency similar* if one can be obtained from the other by replacing one fragment in $\sqcup\!\!\rightarrow$ by another one. More formally, two paths are right adjacency similar if they can be decomposed as

$$y \nwarrow^k \curvearrowright \swarrow^l z \quad \text{and} \quad y \nwarrow^m \curvearrowright \swarrow^n z \quad \text{where } k, l, m, n \in \mathbb{N},\ y \in \mathrm{Up} + \varepsilon,\ z \in \mathrm{Down} + \varepsilon\ .$$

Left adjacency similarity is defined in the same way by replacing $\nwarrow$, $\curvearrowright$ and $\swarrow$ by $\nearrow$, $\curvearrowleft$ and $\searrow$ respectively. Two paths are *adjacency similar* if they are either left or right adjacency similar.

DEFINITION 12. *An* adjacency move *is an elementary move closed under adjacency similarity.*

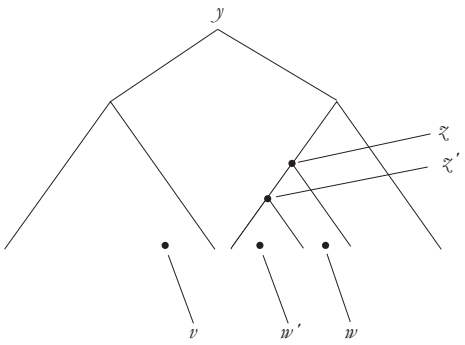


FIG. 7.6. *The nodes y, v, z, z', w and w'*

The following simple fact is given without further proof:

FACT 7.8. *Stay, $\sqcup\!\!\downarrow$, $\downarrow\!\!\sqcup$, $\sqcap\!\!\rightarrow$, $\leftarrow\!\!\sqcap$, $\sqcap\!\!\downarrow$, and $\downarrow\!\!\sqcap$ are adjacency moves.*

Adjacency moves are going to be used in conjunction with fractality (see condition 1 of Definition 10). The following lemma presents a typical example of such an argument (fractality is not explicitly mentioned, but the lemma refers to characteristic types, and by consequence can be used with fractality).

LEMMA 7.9. *Fix a blank tree t, a node y and two nodes z, z' on the leftmost branch below $y1$. Furthermore, let w be a leaf in the subtree of z and let w' be a leaf in the subtree of z'; both with the same characteristic types within the subtrees of z and z', respectively (see Figure 7.6). Then, for any given leaf v below $y0$ and any adjacency move M,*

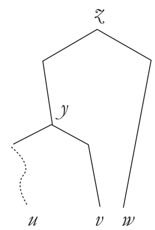
$$\textit{if} \quad vMw \quad \textit{then} \quad vMw' \ .$$

Proof. Since M is an adjacency move, and hence also an elementary move, it is of the form $U \curvearrowright D$. Assume now that vMw holds. This implies $\pi(y1, w) \in D$. It is sufficient to prove that $\pi(y1, w') \in D$ also holds. Since M is an adjacency move, D is of the form either $\swarrow^*$ or $(\swarrow + \searrow)^*$. If D is $\swarrow^*$, this means that w is the leftmost leaf below $y1$, and consequently also the leftmost leaf below z. Since, w' has the same characteristic type (wrt. z') as w (wrt. z), this means that w' is the leftmost leaf below z'. By consequence $z = z'$, and we have $\pi(y1, w') \in D$. Otherwise D is of the form $(\swarrow + \searrow)^*$. Since w' is below $y1$, this implies $\pi(y1, w') \in D$. □

We will now eliminate the moves $\sqcap\!\!\downarrow$, $\leftarrow\!\!\sqcup$, $\sqcap\!\!\rightarrow$ and $\downarrow\!\!\sqcap$, which are not adjacency moves. This is done by simulating them by a sequence of adjacency moves. The following lemma treats the case of $\sqcap\!\!\downarrow$, which is simulated by $\sqcap\!\!\rightarrow \downarrow\!\!\sqcup$. The other cases are symmetric.

LEMMA 7.10. *Let t be a tree and u, v be two leaves of t. If $u \sqcap\!\!\downarrow v$ and v is not the rightmost leaf of t, then there exists a leaf w such that $u \sqcap\!\!\rightarrow w \downarrow\!\!\sqcup v$. If there exists a leaf w such that $u \sqcap\!\!\rightarrow w \downarrow\!\!\sqcup v$, then $u \sqcap\!\!\downarrow v$.*

Proof. We only prove here the first implication; both implications can be seen in the following picture:



Let w be the next leaf in t after v. Since v is not the rightmost leaf, w exists. By the choice of w, we have $w \,\lrcorner\!\sqcap\!\llcorner\, v$. Let us show $u \sqcap\!\!\lrcorner\, w$.

Let y be the deepest node above both u and v. Since $u \sqcap\!\!\downarrow v$ holds, v is the rightmost leaf below y. Let z be the deepest node above both y and w. Since w is the next leaf after v – which is the rightmost leaf below y – the leaf w is the leftmost one below $z1$. But then we can use $\sqcap\!\!\lrcorner$ to go from u to w (the appropriate path doing a $\curvearrowright$ from $z0$ to $z1$). $\square$

In the remainder of the proof, we only use the fact that for each $p, q \in \Gamma$, the move $S(p, q)$ is a union of elementary moves. We then use Lemma 7.10 in the following manner. We enrich the automaton by allowing (after a nondeterministic choice) each move $\sqcap\!\!\downarrow$ to be possibly replaced by the sequence of $\sqcap\!\!\lrcorner$ followed by $\lrcorner\!\sqcap\!\llcorner$, likewise for the other moves $\llcorner\!\sqcap$, $\downarrow\!\!\sqcap$ and $\sqcap\!\!\searrow$, by using time-, space-, and time-space-symmetric variants of this operation. (Note that this transformation requires the use of extra states.) According to the second implication of Lemma 7.10, the resulting automaton is equivalent to $\mathcal{A}$. Furthermore, any unrooted leaf run of the original automaton that only uses states from Γ can be transformed – using the first implication of Lemma 7.10 – into an unrooted leaf run of the modified automaton where all moves happening below the pivot (i.e., the source and target leaves of the move are below the pivot) are adjacency moves. For this reason, from now, we assume that all moves happening below the pivot are adjacency moves.

We proceed to show that Γ cannot detect the rotation. We have to show that

$$[p, v] \Rightarrow_T [q, w] \qquad \text{implies} \qquad [p, v] \Rightarrow_{T'} [q, w]$$

for any states $p, q \in \Gamma$ and nodes v, w not below the pivot. As before (in Section 7.2), since T and T' are equal over nodes not below the pivot, it suffices to establish the lemma for unrooted leaf runs where all positions but the initial and final one are below the pivot. In other words, the first move of the unrooted leaf run is used to enter the subtree of the pivot, the last move is used to exit it, and in between all moves are below the pivot. In this run all moves but the initial and final one are used between leaf configurations below the pivot. Hence, according to the comment above, we can assume that all the moves used in this unrooted leaf run are adjacency moves, possibly except the first and last one.

Recall from Section 7.2.1 the bijection f that assigns to every leaf in T a leaf in T' and preserves the numbering. Let V_1, V_2 and V_3 be the sets of leaves of T respectively below $x00$, $x01$ and $x1$. Let W_1, W_2, W_3 be the sets of leaves of T' respectively below $x0, x10$ and $x11$ (see Fig. 7.7 for an illustration). By definition of rotation, $f(V_i) = W_i$ for $i = 1, 2, 3$. We say that two leaves $v \in V_i$ and $w \in V_j$ are *neighbors* if $|i - j| \leq 1$. If v, w are neighbors, then the path linking v to w and the path linking $f(v)$ to $f(w)$ are adjacency similar. In particular, whenever the automaton can go from v to w in one adjacency move, then it can do this also from $f(v)$ to $f(w)$. Therefore, a leaf run that only does moves between neighbor nodes is mapped by f onto a valid leaf run in the tree T'.

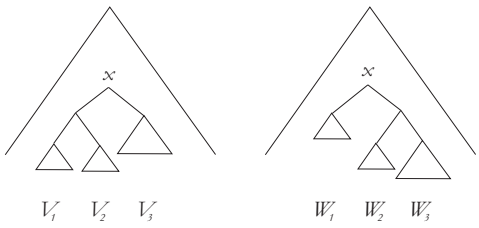


FIG. 7.7. *The trees T and T'*

This remark is the key to our proof. The idea is that we will transform the run from $[p, v]$ to $[q, w]$ into one where all moves below the pivot are between neighbor leaves. Let the leaf run corresponding to $[p, v] \Rightarrow_T [q, w]$ be

$$[p, v] = [r_0, u_0], [r_1, u_1], \ldots, [r_n, u_n], [r_{n+1}, u_{n+1}] = [q, w] \ .$$

We will transform it into the following leaf run in T':

$$[p, v] = [r_0, u_0], [r_1, f(u_1)], \ldots, [r_n, f(u_n)], [r_{n+1}, u_{n+1}] = [q, w] \ .$$

For this construction to work – i.e., for this sequence to be a valid leaf run over T' – it is sufficient to verify that: the step from $[r_0, u_0]$ to $[r_1, u_1]$ can be transformed into a step from $[r_0, u_0]$ to $[r_1, f(u_1)]$ (similarly for the last step); and furthermore that the following property (*) holds: for every $1 \leq i < n$, the leaves u_i and u_{i+1} are neighbors. We first establish the first property; then we will show that every run can be transformed into one where (*) holds. (Property (*) may not hold for the original run.)

First step of the leaf run. The move from $[r_0, u_0]$ to $[r_1, u_1]$ is an elementary move, though perhaps not an adjacency move. Hence, there are three ways of entering the subtree of the pivot: by going to the leftmost leaf below the pivot (using one of $\sqcup\!\!\rightarrow$, $\sqcap\!\!\rightarrow$, $\llcorner\!\!\rightarrow$), to the rightmost one (using one of $\sqcup\!\!\leftarrow$, $\sqcup\!\!\downarrow$, $\sqcap\!\!\downarrow$) or anywhere (using one of $\sqcup\!\!\downarrow$, $\sqcap\!\!\leftarrow$, $\llcorner\!\!\downarrow$, $\sqcap\!\!\downarrow$). In each of those cases, using the same argument as in Lemma 7.9, we show that the same move goes from $[r_0, u_0]$ to $[r_1, f(u_1)]$. The proof for the last step of the leaf run is the same.

Proof of (*). According to the remark above, there are three ways to enter the subtree of the pivot. By time symmetry, there are also three ways to exit this subtree. This results in nine possibilities.

Case leftmost-leftmost. Consider first the case where the automaton enters in the leftmost node of V_1 and leaves by the same node. This means that $u_1 = u_n$ is the leftmost leaf below x.

Since the subtree of the pivot is fractal, it contains a proper subtree that simulates it. Since all subtrees of T are complete binary trees, we may as well assume that there is a node u on the leftmost branch below $x0$ whose subtree simulates the subtree of x. Since the leftmost node is one of the characteristic types (from the definition of fractality), the leaf run that went from the leftmost node below the pivot back to this leftmost node can be assumed to visit only nodes below u (see Fig. 7.8). Such a leaf run satisfies property (*), since it never leaves $V_1 \cup V_2$.

All other cases are solved using the same argument, except for two: when the automaton enters in the leftmost leaf below the pivot and leaves in the rightmost one,

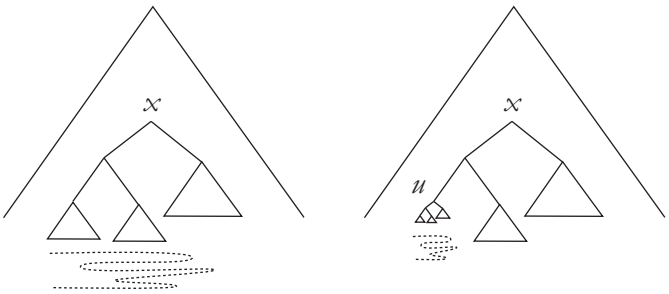


FIG. 7.8. *Moving the leaf run to* V_1

and when the automaton enters in the rightmost leaf below the pivot and leaves in the leftmost one. The first of these is treated in the next item, the other follows by time symmetry.

Case leftmost-rightmost. This time u_1 is the leftmost leaf of V_1 and u_n the rightmost leaf of V_3. We are going to construct a similar leaf run satisfying (*).

As an intermediate step, we first construct a similar leaf run satisfying the a different property (#): once a position in V_3 is encountered during the run, no position in V_1 is visited anymore. Let us prove that we can transform the unrooted leaf run into one that satisfies (#). Let $1 < i \leq j < n$ be positions in the run that witness a violation of (#): the leaves u_{i-1} and u_{j+1} belong to V_3, the leaves $u_i, \ldots, u_j$ belong to $V_1 \cup V_2$, and at least one of $u_i, \ldots, u_j$ belongs to V_1. We will replace this violating subrun with one that does not visit V_1. By iterating this operation, we get a run where property (#) is satisfied.

By condition 1 of Definition 10, we can find on the rightmost branch below $x01$ a node u such that the subtree of u simulates the subtree of $x0$. Hence, one can find leaves u'_i, u'_j below u, such that the characteristic type of u_i (resp. u_j) in the subtree of $x0$ is the same as the characteristic type of u'_i (resp. u'_j) in the subtree of u, and there is a run from $[r_i, u'_i]$ to $[r_j, u'_j]$ that only uses leaves below u. By choice of u, all these leaves are in V_2. Therefore, in order to remove the violation of (#) witnessed by $i < j$, it remains to connect $[r_{i-1}, u_{i-1}]$ with $[r_i, u'_i]$, and $[r_j, u'_j]$ with $[r_{j+1}, u_{j+1}]$. This is a direct application of Lemma 7.9.

Thanks to the above argument, we may assume that the leaf run satisfies property (#). If it already has property (*), then the problem is over. Otherwise there is some moment in the leaf run where two consecutive leaf configurations are not neighboring. Since after visiting V_3 we never come back to V_1, this can happen at most once, where the source configuration $[r_i, u_i]$ is in V_1 and the target configuration $[r_{i+1}, u_{i+1}]$ is in V_3. Moreover, the only way to go from a position in V_1 to a position in V_3 via an adjacency move is by using $\sqcap\!\!\downarrow$. In particular, u_{i+1} is the leftmost leaf in V_3. However, if we want to use the move $\sqcap\!\!\downarrow$ from $[r_i, u_i]$ and satisfy property (*), the only place we can go to is the leftmost leaf of V_2.

In order to complete the proof, we will construct a new leaf run that satisfies property (*) and goes from state r_{i+1} in the leftmost leaf of V_2 to state r_i in some leaf u of V_2. Then we can reuse the move $r_i \sqcap\!\!\downarrow r_{i+1}$ to go from u to u_{i+1}, since $\sqcap\!\!\downarrow$ does not care about the position of the leaf u within $V_1 \cup V_2$.

The leaf run that goes from r_{i+1} in the leftmost leaf of V_2 – call this leaf v — to r_i in some leaf u of V_2 is constructed in two stages. In the first stage, we show that there is some leaf u' in V_2 such that the configuration $[r_n, u']$ can be reached from $[r_{i+1}, v]$.

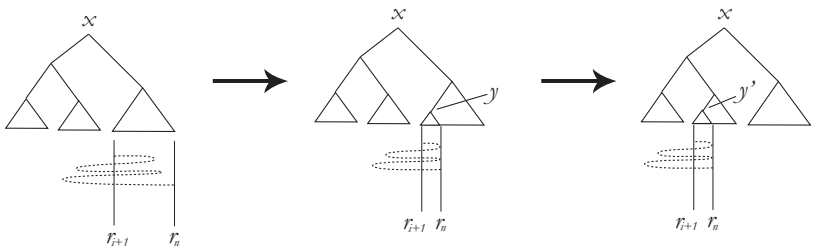


FIG. 7.9. *Transforming the run from r_{i+1} to r_n*

Furthermore, the leaf u' has at least $|Q|$ leaves to the left and to the right that belong to V_2. In the second stage, we use this latter assumption to go from $[r_n, u']$ to state r_i without leaving V_2. This is done simply by using Corollary 6.7. The node where we arrive is going to be u.

Therefore, in order to complete the proof of Proposition 7.1 – and therefore also the proof of Theorem 2 – it remains to find some leaf u' in V_2 such that the configuration $[r_n, u']$ can be reached from $[r_{i+1}, v]$, and u' has $|Q|$ leaves to the left and right inside V_2. This process is illustrated in Figure 7.9.

Recall our assumption that the nodes $u_{i+1}, \ldots, u_n$ are all in $V_2 \cup V_3$. By condition 1, there is a node y on the leftmost branch below $x10$ whose subtree simulates the subtree of $x1$ and has more than $|Q|$ leaves. Using the same proof as for property (#), we can transform the unrooted leaf run from $[r_{i+1}, u_{i+1}]$ to $[r_n, u_n]$ into an unrooted leaf run from $[r_{i+1}, u_{i+1}]$ to $[r_n, u'_n]$, where only leaves from V_2 or below y are used, and where u'_n is the rightmost leaf below y. (This run is illustrated in the middle picture of Figure 7.9.) Let y' be the node on the leftmost branch below $x01$ such that the subtree of y' is the same as the subtree of y. By using the properties of adjacency moves, one can shift – by $|V_2|$ leaves to the left – the run that goes from $[r_{i+1}, u_{i+1}]$ to $[r_n, u'_n]$ into a run that goes from $[r_{i+1}, v]$ to $[r_n, u']$ where u' is the rightmost leaf below y', which concludes the proof. Note that in V_2, there are more than $|Q|$ leaves to the left and to the right of u', by construction of y.

REFERENCES

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

[2] M. Bojańczyk. 1-bounded TWA cannot be determinized. In *Foundations of Software Technology and Theoretical Computer Science*, volume 2914 of *Lecture Notes in Computer Science*, pages 62–73. Springer, 2003.

[3] M. Bojańczyk and T. Colcombet. Tree-walking automata cannot be determinized. *Theoretical Computer Science*, 350(2-3):164–173, 2006.

[4] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available on: `http://www.grappa.univ-lille3.fr/tata`, 1997. release October, 1rst 2002.

[5] 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.

[6] J. Engelfriet and H. J. Hoogeboom. Automata with nested pebbles capture first-order logic with transitive closure. Technical report, Leiden Institute of Advanced Computer Science, 2005.

[7] J. Engelfriet, H. J. Hoogeboom, and J. P. Van Best. Trips on trees. *Acta Cybernetica*, 14(1):51–64, 1999.

[8] J. Engelfriet, G. Rozenberg, and G. Slutzki. Tree transducers, l systems and two-way machines. *Journal of Computer and System Sciences*, 20:150–202, 1980.

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

[10] A. Muscholl, L. Ségoufin, and M. Samuelides. Complementing deterministic tree-walking automata. *Information Processing Letters*, 2005.

[11] F. Neven and T. Schwentick. On the power of tree-walking automata. *Information and Computation*, 183(1):86–103, 2003.

[12] A. Potthoff. *Logische Klassifizierung regulärer Baumsprachen*. PhD thesis, Institut für Informatik und Praktische Mathematik, Universität Kiel, 1994.