Tree Automata over Infinite Alphabets

Michael Kaminski and Tony Tan

Department of Computer Science, Technion – Israel Institute of Technology,
Haifa 32000, Israel
{kaminski,tantony}@cs.technion.ac.il

Dedicated to Boris (Boaz) Trakhtenbrot
on the occasion of his 85th birthday.

Abstract. A number of models of computation on trees labeled with symbols from an *infinite* alphabet is considered. We study closure and decision properties of each of the models and compare their computation power.

1 Introduction

In recent years a new application area for regular word and tree languages has evolved during one of the most important developments in World Wide Web (WWW) – the emergence of the *Extensible Markup Language* (XML). For many purposes, XML documents are modeled as labeled finite trees, where the *finite* set of labels corresponds to the set of element names allowed in the document. Thus, concepts from regular word and tree languages became important in XML research; see [1,8,11,9,12,14,18].

However, this abstraction ignores an important aspect of XML – the presence of *attributes* attached to the leaves of trees. Since attributes may assume values from an *infinite* set, modeling XML documents by trees over a finite alphabet is not adequate in any scenario. Therefore, a more natural way to model XML documents in this setting is to allow, besides the finite set of element labels, an *infinite* set of possible data values. Consequently, there is a need to extend the notion of regular and tree languages in such a way that, on one hand, as many as possible settings involving attributes can be captured and (most of) the desirable properties of the language class are retained on the other.

In this paper we extend *finite-memory automata*; see [2,5,6], to *tree* automata over *infinite alphabets.*

It should be noted, however, that there is a different model of computation over infinite alphabets, called *pebble automata over infinite alphabets*; see [8,10,13]. They are "orthogonal" to finite-memory automata. Also, the *tree walking* automata with pebbles ([8]) naturally extend to infinite alphabets for type-checking purposes. In our opinion, all these automata are inappropriate for modeling XML. This is because the tree languages they accept lack basic decision properties, though, unlike the language considered in this paper, they are closed under *complement.* However, for practical purposes it is very common to ask whether an XML scheme admits even one document. Since the emptiness problem for pebble automata is undecidable, using these automata for modeling XML documents is, at least, arguable.

In our extension of finite-memory automata to trees *each* head scanning a symbol at the tree node is equipped with a finite number of registers. This number is the same for all heads of the automaton. There are two ways to scan an input tree: *top-down* from the root to the leaves; see [15], and *bottom-up* from the leaves to the root; see [4,17].[1] When moving top-down from a parent node to its children, the automaton *splits* the head and the corresponding set of registers; and when moving bottom-up from the child nodes to their parent node, the automaton replaces the two[2] heads by one and *merges* the corresponding set of registers by forgetting some of their contents. Whereas the definition of a top-down finite-memory tree automata is very natural: the split results in two heads each carrying a set of registers with the same contents, the definition of the bottom-up one is less obvious, because it requires an automaton to merge two sets of registers whose contents may be quite different and even disjoint.

In addition, there are (at least) two possibilities of updating the content of the automaton registers. One possibility is to replace the content of one of the registers with the currently scanned *new* input symbol, as was done in [5,6], and the other is to replace the content of one of the registers with any *new* symbol from the infinite input alphabet, as was done in [2]. That is, in the latter case, the automaton does not necessarily have to arrive to the symbol in order to store it in its registers. Such ability will be referred to as a *nondeterministic reassignment*.

The above two distinctions result in four models of finite-memory tree automata which we consider in our paper. It appears that both top-down and bottom-up models with deterministic reassignment (the first possibility) are not strong enough for modeling XML. The former cannot even accept the tree language consisting of all trees having two different leaves labeled with the same symbol; see Example 2,[3] whereas the latter cannot accept a very simple tree language consisting of all trees whose root label differs from the labels of all other nodes. In fact, unlike in the case of a finite alphabet, the computation powers of these two models are incomparable.

In contrast with the above, the computation models with nondeterministic reassignment, which are stronger than those with the deterministic one, seem to be appropriate for modeling XML. We show that top-down and bottom-up tree automata with nondeterministic reassignment have the same computation power (which indicates that the definition is robust) and are proper extensions

[1] Recall that in Computer Science trees grow top-down.

[2] This paper deals with binary trees, only. Since finite branching trees can be encoded by binary trees in a standard manner, our computation models naturally generalize to unranked trees.

[3] For some purposes, an XML document might require certain integrity constraints. For example, it might require that the value at a certain position of a document also occurs (or differs from the value) at some other position.

of the deterministic reassignment models. In particular, the tree languages from the above two examples are accepted by tree automata with nondeterministic reassignment.

Also, we establish some closure and decision properties of tree languages accepted by tree automata with deterministic and nondeterministic reassignment and show how these tree languages are related to *context-free languages over infinite alphabets* introduced in [2]. This relationship is of importance, because the latter are tightly connected to *Document Type Definitions* which define XML documents.

The paper is organized as follows. In the next section we give a rough review of XML concepts. Section 3 contains the notation used throughout this paper. In Sect. 4 and 5 we introduce the tree automata with deterministic and nondeterministic reassignment, respectively. Section 6 deals with decision problems related to our models of computation. In Sect. 7 we establish a relationship between the tree languages defined in this paper and the context-free languages introduced in [2]. Finally, Appendices A and B contain the proof of equivalence of top-down and bottom-up tree automata with nondeterministic reassignment and Appendix C summarizes the closure properties of tree languages accepted by our tree automata.

2 Basic Notions of XML

In this section we briefly sketch the core idea of XML and its connection to tree languages. This is done via an example below. Readers interested in the details are referred to [16] or [19].

Consider the following XML document that displays information about a factory.

```
<factory name = "super">
    <section name = "productions">
        <product id = 011>
            <No.> 1 </No.>
            <label> notebook </label>
        </product>
        <product id = 294>
            <No.> 2 </No.>
            <label> pencil </label>
        </product>
    </section>
    <section name = "advertisement">
        <product id = 011>
            <No.> 1 </No.>
            <label> notebook </label>
        </product>
    </section>
</factory>
```

Like in the case of HTML, the building blocks of XML are elements which are delimited by the start- and end-tags. A start-tag of a `product`-element, for example, is `<product>` and the end-tag is `</product>`. Everything in between `<product>` and `</product>` constitutes a `product`-element.

An element can be nested into another one. For example, the element `<No.> 1 </No.>` is a subelement of the outer `product`-element. Elements may also have attributes. These are name value pairs separated by the equality sign. For example, `<product id = 011>` indicates that the value of the `id` attribute of that particular `product`-element is 011.

An XML document can be viewed as a tree in a natural way. Fig. 1 below shows the above XML document in form of a tree.

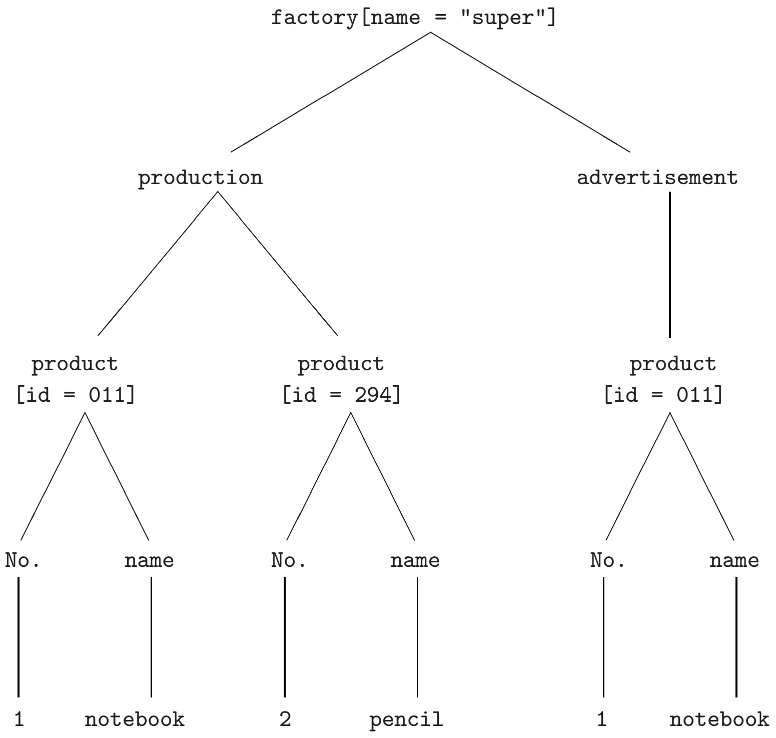


Fig. 1. A tree representation of an XML document

XML documents can be defined by *Document Type Definitions* (DTDs). These are, basically, *extended context-free grammars* (cf. context-free grammars over infinite alphabets in Sect. 7), i.e., context-free grammars with regular expressions at the right-hand sides. For example, the DTD in Fig. 2 defines the scheme of an XML document for a "factory."

This DTD specifies that the element `factory` is the outermost and consists of the `production`- and the `advertisement`-elements. Each of them, in turn,

```
<!DOCTYPE factory [
        <!ELEMENT factory (production, advertisement)>
        <!ELEMENT production (product)*>
        <!ELEMENT advertisement (product)*>
        <!ELEMENT product (No.,name) | ϵ >
        <!ELEMENT No. (#PCDATA)>
        <!ELEMENT name (#PCDATA)>
        <!ATTLIST factory name #PCDATA>
        <!ATTLIST product id #PCDATA>
]>
```

Fig. 2. The DTD scheme for the XML document in Fig. 1

consists of a finite number of `product`-elements. A `product`-element can either consist of nothing (i.e., the empty word ϵ), or of the `No.`- and the `name`-elements. The `No.`- and `name`-elements do not consist of any other elements, but a word, as the term #PCDATA indicates. The DTD also indicates that the `factory`- and the `product`-elements have attributes attached to it: `name` and `id`, respectively. These attributes take the data value `#PCDATA`, that is a word.

Furthermore, in addition to conforming to the DTD, an XML document may require some constraints imposed on the data values of the attributes. For example, the DTD `factory` may require that the advertised product is produced by the factory itself. This constraint is defined by the following formula in which the binary predicate *Parent*(x, y) is, naturally, read as "x is the parent of y" and y is the advertised product.

$$\exists x(\mathit{Parent}(x,y) \wedge x.\mathit{label} = \texttt{advertisement} \wedge y.\mathit{label} = \texttt{product}) \rightarrow$$

$$\exists u \exists v(\mathit{Parent}(u,v) \wedge u.\mathit{label} = \texttt{production} \wedge v.\mathit{label} = \texttt{product} \wedge v.\mathit{id} = y.\mathit{id})$$

So, any `product`-element that appears under the `advertisement`-element must also appear under the `production`-element. As `product`-element is identified by its `id`-attribute, the number of possible `product`-elements is unbound.

3 Notation

We fix an infinite alphabet Σ not containing # that is reserved to denote an empty register. For a word $\boldsymbol{w} = w_1 w_2 \cdots w_r$ over $\Sigma \cup \{\#\}$, we define the *content* of $\boldsymbol{w}$, denoted $[\boldsymbol{w}]$, by $[\boldsymbol{w}] = \{w_j \neq \# : j = 1, 2, \ldots, r\}$. That is, $[\boldsymbol{w}]$ consists of all symbols of Σ which appear in $\boldsymbol{w}$.

A word $w_1 \cdots w_r$ such that $w_i = w_j$ and $i \neq j$ imply $w_i = \#$ is called an *assignment*. That is, an assignment is a word over $\Sigma \cup \{\#\}$ where each symbol from Σ appears at most one time. Assignments represent the contents of the registers of an automaton: the symbol in the ith register is w_i. If $w_i = \#$, then the ith register is empty. The set of all assignments of length r is denoted by $\Sigma^{r \neq}$.

A set of words $T \subseteq \{0,1\}^*$ is called a (binary) *tree* if it satisfies the following two conditions.

- For each $n \in T$ and each prefix n' of n, $n' \in T$. That is, T is *prefix closed.*
- If $n \in T$, then either both or none of $\{n0, n1\}$ are in T.

We write $n_1 \preccurlyeq n_2$, if n_1 is a prefix of n_2.

Elements of a tree are called *nodes*. The *root* of the tree is the empty word ϵ. A node n of a tree T is called a *leaf*, if neither of $\{n0, n1\}$ belongs to T. If n is not a leaf, then nodes $n0$ and $n1$ are called the *left* and *right children* of n, respectively, and n is called the *parent* of both $n0$ and $n1$. A node n_1 is an ancestor of n_2, or n_2 is a descendant of n_1, if $n_1 \preccurlyeq n_2$.

The hight of a node n in a tree T is the length of n. The root ϵ is of hight 0. The hight of a tree T is the hight of the longest node of T.

A *Σ-tree* is a map $\boldsymbol{\sigma}$ from a tree into Σ. The set of all Σ-trees is denoted by $\mathcal{T}(\Sigma)$. We denote by $dom(\boldsymbol{\sigma}) = T$, the domain of $\boldsymbol{\sigma} : T \to \Sigma$.

Finally, let $\boldsymbol{\sigma} : T \to \Sigma$ be a Σ-tree and let $n_1, n_2, \ldots, n_m$ be the list of all leaf nodes of T in the lexicographical order. The Σ-word $\boldsymbol{\sigma}(n_1)\boldsymbol{\sigma}(n_2)\cdots\boldsymbol{\sigma}(n_m)$ is called the *frontier* of $\boldsymbol{\sigma}$ and is denoted by $\ell(\boldsymbol{\sigma})$.

4 Tree Automata with Deterministic Reassignment

In this section we define the tree version of finite memory automata (FMA) introduce in [6], both the top-down and bottom-up cases. We start with the top-down case, which is a straightforward extension of the ordinary word FMA.

4.1 Top-Down Finite-Memory Automata with Deterministic Reassignment

Definition 1. (Cf. [6, Definition 1].) *A* top-down finite-memory automaton with deterministic reassignment *($\downarrow$-FMA) is a system $A = \langle S, s_0, \boldsymbol{u}, \rho, \mu, F \rangle$, where*

- *S is finite set of* states*.*
- *$s_0 \in S$ is the* initial *state.*
- *$\boldsymbol{u} = u_1 u_2 \cdots u_r \in \Sigma^{r \neq}$ is the* initial *assignment to the r registers of A.*
- *$\rho : S \to \{1, 2, \ldots, r\}$ is a function from S into $\{1, 2, \ldots, r\}$ called the* deterministic reassignment*. The intuitive meaning of ρ is as follows. If A is in a state p and the input symbol appears in no register, then the automaton reassigns the $\rho(p)$th register with the input symbol.*
- *$\mu \subseteq S \times \{1, 2, \ldots, r\} \times S^2$ is the* transition *relation, whose elements are called* transitions *and are also written in the form*

$$(p, i) \to (p_0, p_1),$$

where $p, p_0, p_1 \in S$ *and* $i \in \{1, 2, \ldots, r\}$. *Intuitively, in a* Σ*-tree* $\boldsymbol{\sigma}$ *the transition* $(p, i) \to (p_0, p_1)$ *"applies" at a node* n *if* n *is labeled with the state* p *and the content of the* i*th register is* $\boldsymbol{\sigma}(n)$. *Subsequently, the children* $n0$ *and* $n1$ *of* n *are labeled with the states* p_0 *and* p_1, *respectively.*
– $F \subseteq S \times \{1, 2, \ldots, r\}$ *is the set of* final relations.

Like in the case of FMA, an actual state of A is a state of S together with the content of all registers. That is, A has infinitely many states which are pairs $(p, \boldsymbol{w})$, where $p \in S$ and $\boldsymbol{w} \in \Sigma^{r\neq}$ is the content of the registers of A. These are called *configurations* of A. The set of all configurations of A is denoted by S^c. The pair $(s_0, \boldsymbol{u})$, denoted s_0^c, is called the *initial* configuration.

The transition relation μ induces the following relation μ^c on $S^c \times \Sigma \times (S^c \times S^c)$, whose elements are written in the form

$$(p, \boldsymbol{w}), \sigma \to (p_0, \boldsymbol{w}_0), (p_1, \boldsymbol{w}_1),$$

where $p, p_0, p_1 \in S$ and $\boldsymbol{w}, \boldsymbol{w}_0, \boldsymbol{w}_1 \in \Sigma^{r\neq}$.

Let $\boldsymbol{w} = w_1 w_2 \cdots w_r$, $\boldsymbol{w}_0 = w_{0,1} w_{0,2} \cdots w_{0,r}$, and $\boldsymbol{w}_1 = w_{1,1} w_{1,2} \cdots w_{1,r}$. Then $(p, \boldsymbol{w}), \sigma \to (p_0, \boldsymbol{w}_0), (p_1, \boldsymbol{w}_1)$ belongs to μ^c if and only if the following conditions are satisfied.

– If $\sigma = w_i \in [\boldsymbol{w}]$, then $\boldsymbol{w}_0 = \boldsymbol{w}_1 = \boldsymbol{w}$ and $(p, i, (p_0, p_1)) \in \mu$.
– If $\sigma \notin [\boldsymbol{w}]$, then $w_{0,\rho(p)} = w_{1,\rho(p)} = \sigma$, $w_{0,i} = w_{1,i} = w_i$ for each $i \neq \rho(p)$, and $(p, \rho(p), (p_0, p_1)) \in \mu$.

The set of final relations F defines the set of *final* "Σ-relations" F^c. A pair $((p, \boldsymbol{w}), \sigma) \in F^c$ if the following holds.

– If $\sigma = w_i$, then $(p, i) \in F$; and
– if $\sigma \notin [w_1 w_2 \cdots w_r]$, then $(p, \rho(p)) \in F$.

A *run* of A on a Σ-tree $\boldsymbol{\sigma} : T \to \Sigma$ is a mapping $R : T \to S^c$ such that

– $R(\epsilon) = s_0^c$ (recall that $s_0^c = (s_0, \boldsymbol{u})$ is the initial configuration of A) and
– for each non-leaf node $n \in T$, $(R(n), \boldsymbol{\sigma}(n)) \to (R(n0), R(n1)) \in \mu^c$.

We say that A *accepts* a Σ-tree $\boldsymbol{\sigma} : T \to \Sigma$, if there exists a run R of A on $\boldsymbol{\sigma}$, called an *accepting* run, such that for each leaf n of T, $(R(n), \boldsymbol{\sigma}(n)) \in F^c$. The set of all trees accepted by A is denoted by $L(A)$.

Example 1. Let $A_\epsilon = \langle \{s_0, p\}, s_0, \#\#, \rho, \mu, \{(s_0, 1), (p, 2)\} \rangle$, where

– $\rho(s_0) = 1$, $\rho(p) = 2$, and
– $\mu = \{(s_0, 1, (p, p)), (p, 2, (p, p))\}$.

Then $L(A_\epsilon) = L_\epsilon$, where

$$L_\epsilon = \{\boldsymbol{\sigma} : T \to \Sigma : \text{ for each } n \in T \setminus \{\epsilon\},\ \boldsymbol{\sigma}(n) \neq \boldsymbol{\sigma}(\epsilon)\}.$$

For example, an accepting run $R : T \to \{s_0, p\} \times \Sigma^{2\neq}$ of A_ϵ on a Σ-tree $\boldsymbol{\sigma} : T \to \Sigma$ such that for each $n \in T \setminus \{\epsilon\}$, $\boldsymbol{\sigma}(n) \neq \boldsymbol{\sigma}(\epsilon)$ is defined by

- $R(\epsilon) = (s_0, \#\#)$,
- $R(0) = (p, \boldsymbol{\sigma}(\epsilon)\#)$,
- $R(1) = (p, \boldsymbol{\sigma}(\epsilon)\#)$, and
- for $n \neq \epsilon$ and $i = 0, 1$, $R(ni) = (p, \boldsymbol{\sigma}(\epsilon)\boldsymbol{\sigma}(n))$.

Example 2. In this example we show that the tree language

$$L_2 = \left\{ \boldsymbol{\sigma} : T \to \Sigma : \begin{array}{l} \text{there exist two different leaves } n', n'' \in T \\ \text{such that } \boldsymbol{\sigma}(n') = \boldsymbol{\sigma}(n'') \end{array} \right\}$$

is not accepted by a top-down finite-memory automaton.

Indeed, assume to the contrary there exists a top-down tree finite-memory automaton $A = \langle S, s_0, \boldsymbol{u}, \rho, \mu, F \rangle$ that accepts L_2. In particular, A accepts the Σ-tree $\boldsymbol{\sigma} : \{\epsilon, 0, 1\} \to \Sigma$, where $\boldsymbol{\sigma}(\epsilon) \neq \boldsymbol{\sigma}(0)$ and $\boldsymbol{\sigma}(0) = \sigma(1) \notin [\boldsymbol{u}]$. Let R be an accepting run of A on $\boldsymbol{\sigma}$ and $R(0) = (q, \boldsymbol{w})$, Then, $\boldsymbol{\sigma}(0) \notin [\boldsymbol{w}]$, implying $(q, \rho(q)) \in F$.

Consider the Σ-tree $\boldsymbol{\sigma}' : \{\epsilon, 0, 1\} \to \Sigma$, where $\boldsymbol{\sigma}'(\epsilon) = \boldsymbol{\sigma}(\epsilon)$, $\boldsymbol{\sigma}'(0) \neq \boldsymbol{\sigma}(0)$, $\boldsymbol{\sigma}'(0) \notin [\boldsymbol{u}] \cup \{\boldsymbol{\sigma}(\epsilon)\}$, and $\boldsymbol{\sigma}'(1) = \boldsymbol{\sigma}(1)$. Then R is also a run of A on $\boldsymbol{\sigma}'$. Since $\boldsymbol{\sigma}'(\epsilon) = \boldsymbol{\sigma}(\epsilon)$, $R(0) = (q, \boldsymbol{w})$. In addition, $\boldsymbol{\sigma}'(0) \notin [\boldsymbol{w}]$ and $(q, \rho(q)) \in F$ imply $\boldsymbol{\sigma}' \in L(A)$. However, this contradicts $L(A) = L_2$.

4.2 Bottom-Up Finite-Memory Automata with Deterministic Reassignment

To define bottom-up automata we need to choose r symbols (to fill r registers at the parent node) out of, possibly, $2r$ symbols stored in the registers at the child nodes. Such a choice is based on the notion of a *type* defined below.

Definition 2. *An r-*type *is a subset t of $\{1, 2, \ldots, r\} \times \{1, 2, \ldots, r\}$ such that for all $(i_0, i_1), (j_0, j_1) \in t$,*

$$(i_0, i_1) \neq (j_0, j_1) \textit{ implies both } i_0 \neq i_1 \textit{ and } j_0 \neq j_1.$$

The set of all r-types is denoted by $\mathcal{T}_r$.

For two assignments $\boldsymbol{w}_0 = w_{0,1}w_{0,2} \cdots w_{0,r}$ and $\boldsymbol{w}_1 = w_{1,1}w_{1,2} \cdots w_{1,r}$ we define the type $t(\boldsymbol{w}_0, \boldsymbol{w}_1)$ by

$$t(\boldsymbol{w}_0, \boldsymbol{w}_1) = \{(i', i'') : w_{0,i'} = w_{1,i''}\}.[4]$$

A function $f : \{1, \ldots, r\} \to \{0, 1\} \times \{1, \ldots, r\}$ is said to be a valid selector *for an r-type t if for all $1 \leq i < j \leq r$,*

$$f(i) = (0, i') \textit{ and } f(j) = (1, j') \textit{ imply } (i', j') \notin t.$$

Intuitively, f is used to select r registers out of $2r$ registers for the assignment at the parent node. If $f(i) = (0, i')$, then the content of the ith register at the parent node comes from the i'th register of the left child. Similarly, if $f(i) = (1, i')$, then

[4] That is, the elements of $t(\boldsymbol{w}_0, \boldsymbol{w}_1)$ indicate the registers with the same content. It follows from the defintion of an assignment that the type $t(\boldsymbol{w}_0, \boldsymbol{w}_1)$ is well defined.

the content of the ith register at the parent node comes from the i'th register of the right child. By definition, if f is a valid selector for an r-type t, then the resulting assignment does not have two registers with the same content.

A triple of assignments $(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w})$ is an instance of (t, f), where $t \in \mathcal{T}_r$ and f is a valid selector for t if

– $t = t(\boldsymbol{u}, \boldsymbol{v})$, and
– $\boldsymbol{w} = w_1 \cdots w_r$ is defined by

$$w_i = \begin{cases} u_j \text{ if } f(i) = (0, j) \\ v_j \text{ if } f(i) = (1, j) \end{cases},$$

where $\boldsymbol{u} = u_1 \cdots u_r$ and $\boldsymbol{v} = v_1 \cdots v_r$.

Intuitively, the assignment $\boldsymbol{w}$ is the result of merging the assignments $\boldsymbol{u}$ and $\boldsymbol{v}$ according to the the function f. Since f is a valid selector for t, w does not have two registers with the same content.

Definition 3. *A* bottom-up finite-memory automaton with deterministic reassignment *($\uparrow$-FMA) is a system $A = \langle S, s_0, \boldsymbol{u}, \rho, \tau, \mu, F \rangle$, where*

– *S is a finite set of* states*.*
– *$s_0 \in S$ is the* initial *state.*
– *$\boldsymbol{u} = u_1 u_2 \cdots u_r \in \Sigma^{r\neq}$ is the* initial *assignment to the r registers of A.*
– *$\rho : S \to \{1, 2, \ldots, r\}$ is the* deterministic reassignment*.*
– *$\mu \subseteq (S \times \{1, 2, \ldots, r\})^2 \times S$ is the* transition *relation, whose elements are* transitions *and are also written in the form*

$$(p_0, k_0), (p_1, k_1) \to p,$$

where $p_0, p_1, p \in S$ and $k_0, k_1 \in \{1, 2, \ldots, r\}$. The intuitive meaning of transition $(p_0, k_0), (p_1, k_1) \to p$ is as follows. In a Σ-tree $\boldsymbol{\sigma}$ it "applies" at nodes $n0$ and $n1$ labeled states p_0 and p_1, respectively, if the content of the k_0th register at node $n0$ is $\boldsymbol{\sigma}(n0)$ and the content of the k_1th register at node $n1$ is $\boldsymbol{\sigma}(n1)$. Then the label of n is p.
– *τ is a* merging relation *whose elements are of the form*

$$((p_0, k_0), (p_1, k_1), t, f),$$

where $p_0, p_1 \in S$ and $k_0, k_1 \in \{1, 2, \ldots, r\}$, $t \in \mathcal{T}_r$, and f is a valid selector for t. The intuitive meaning of an element $((p_0, k_0), (p_1, k_1), t, f)$ of τ is as follows. In a Σ-tree $\boldsymbol{\sigma}$ it "applies" at nodes $n0$ and $n1$ labeled p_0 and p_1, respectively, if the content of the k_0th register at node $n0$ is $\boldsymbol{\sigma}(n0)$ and the content of the k_1th register at node $n1$ is $\boldsymbol{\sigma}(n1)$ and the type of the assignments at $n0$ and $n1$ is t. Then the assignments are merged according to f.
– *$F \subseteq S \times \{1, 2, \ldots, r\}$ is the set of* final relations*.*

The transition relation μ induces the following relation μ^c on $(S^c \times \Sigma)^2 \times S^c$ whose elements are also written in the form

$$(p_0, \boldsymbol{w}_0, \sigma_0), (p_1, \boldsymbol{w}_1, \sigma_1) \to (p, \boldsymbol{w}),$$

where $p, p_0, p_1 \in S$ and $\boldsymbol{w}, \boldsymbol{w}_0, \boldsymbol{w}_1 \in \Sigma^{r\neq}$.

Let $\boldsymbol{w} = w_1 w_2 \cdots w_r$, $\boldsymbol{w}_0 = w_{0,1} w_{0,2} \cdots w_{0,r}$, and $\boldsymbol{w}_1 = w_{1,1} w_{1,2} \cdots w_{1,r}$. Then $(p_0, \boldsymbol{w}_0, \sigma_0), (p_1, \boldsymbol{w}_1, \sigma_1) \to (p, \boldsymbol{w})$ belongs to μ^c if and only if there exist $\boldsymbol{v}_0, \boldsymbol{v}_1 \in \Sigma^{r\neq}$, $\boldsymbol{v}_0 = v_{0,1} v_{0,2} \cdots v_{0,r}$ and $\boldsymbol{v}_1 = v_{1,1} v_{1,2} \cdots v_{1,r}$, $(p_0, k_0), (p_1, k_1) \to p \in \mu$, and $((p_0, k_0), (p_1, k_1), t, f) \in \tau$ such that the following conditions are satisfied.

– If $\sigma_0 \in [\boldsymbol{w}_0]$, then $\boldsymbol{v}_0 = \boldsymbol{w}_0$. Otherwise,

$$v_{0,i} = \begin{cases} w_{0,i} & \text{if } i \neq \rho(p_0) \\ \sigma_0 & \text{if } i = \rho(p_0) \end{cases}.$$

– Similarly, if $\sigma_1 \in [\boldsymbol{w}_1]$, then $\boldsymbol{v}_1 = \boldsymbol{w}_1$. Otherwise,

$$v_{1,i} = \begin{cases} w_{1,i} & \text{if } i \neq \rho(p_1) \\ \sigma_1 & \text{if } i = \rho(p_1) \end{cases}.$$

– $v_{0,k_0} = \sigma_0$ and $v_{1,k_1} = \sigma_1$.[5]
– $t = t(\boldsymbol{v}_0, \boldsymbol{v}_1)$.
– $(\boldsymbol{v}_0, \boldsymbol{v}_1, \boldsymbol{w})$ is an instance of (t, f).

A *run* of A on a Σ-tree $\boldsymbol{\sigma} : T \to \Sigma$ is a mapping $R : T \to S^c$ such that

– for each leaf n of T, $R(n) = s_0^c$ (recall that $s_0^c = (s_0, \boldsymbol{u})$ is the initial configuration of A), and
– for each non-leaf node $n \in T$, $(R(n0), \boldsymbol{\sigma}(n0)), (R(n1), \boldsymbol{\sigma}(n1)) \to R(n) \in \mu^c$.

The set of final relations F defines the following set *final* "Σ-relations" F^c. A pair $((p, \boldsymbol{w}), \sigma)$ is in F^c if the following holds.

– If $\sigma = w_i$, then $(p, i) \in F$; and
– if $\sigma \notin [w_1 w_2 \cdots w_r]$, then $(p, \rho(p)) \in F$.

We say that A *accepts* a Σ-tree $\boldsymbol{\sigma} : T \to \Sigma$ if there exists a run R of A on $\boldsymbol{\sigma}$, called an *accepting* run, such that $(R(\epsilon), \boldsymbol{\sigma}(\epsilon)) \in F^c$. The set of all trees accepted by A is denoted by $L(A)$.

Example 3. (Cf. Example 1.) The set of Σ-trees L_ϵ from Example 1 is not accepted by bottom-up finite-memory automata.

Indeed, assume to the contrary L_ϵ is accepted by an r-register bottom-up automaton A with the initial assignment $\boldsymbol{u}$. Consider a tree $\boldsymbol{\sigma} \in L_\epsilon$, where $\boldsymbol{\sigma}(\epsilon) \notin [\boldsymbol{u}]$ and $\boldsymbol{\sigma}(n_1) \neq \boldsymbol{\sigma}(n_2)$, whenever $n_1 \neq n_2$. During the course of computation,

[5] Note that if $\sigma_0 \notin [\boldsymbol{w}_0]$ (respectively, $\sigma_1 \notin [\boldsymbol{w}_1]$), then $k_0 = \rho(p_0)$ (respectively, $k_1 = \rho(p_1)$).

till the last step, $\boldsymbol{\sigma}(\epsilon)$ does not appear in the registers of A. When A reaches the root, it reassigns one of the registers with $\boldsymbol{\sigma}(\epsilon)$ and verifies whether one of the final Σ-relations holds.

Assume, in addition, that the range of $\boldsymbol{\sigma}$ contains more than r different symbols. Therefore, when A reaches the root of the tree, there is a node n such that $\boldsymbol{\sigma}(n)$ is no longer in the registers of A. If we replace $\boldsymbol{\sigma}(\epsilon)$ with $\boldsymbol{\sigma}(n)$, A would still accept the obtained Σ-tree, in contradiction with $L(A) = L_\epsilon$.

Example 4. (Cf. Example 2.) The set of Σ-trees L_2 from Example 2 is accepted by a bottom-up finite-memory automaton that operates as follows. It "guesses" two different nodes, remembers their labels, and carries them up to the node where the paths from the two guessed nodes meet. At this meeting node the automaton checks whether the labels at the two guessed nodes are the same.

5 Tree Automata with Nondeterministic Reassignment

In this section we introduce the notion of tree automata with *nondeterministic reassignment* – both for the top-down and the bottom-up cases. Unlike the deterministic reassignment automata which may only reassign one of their registers with the *current* input symbol, these automata are allowed to reassign a number of registers with arbitrary symbols from Σ. That is, the reassignment function $\rho : S \to 2^{\{1,\dots,r\}}$ mapping the states from S into the power set $2^{\{1,\dots,r\}}$.

5.1 Top-Down Finite-Memory Automata with Nondeterministic Reassignment

Definition 4 below is the top-down tree counterpart of the *infinite-alphabet pushdown automata* introduced in [2] and the *look-ahead finite-memory automata* introduced in [20].

Definition 4. *A* top-down finite-memory automaton with nondeterministic reassignment *($\downarrow$-NR-FMA) is a system $A = \langle S, s_0, \boldsymbol{u}, \rho, \mu, F\rangle$, where all components of A, except ρ, are as Definition 1. The* nondeterministic reassignment ρ *is a function from S into $2^{\{1,2,\dots,r\}}$. The intuitive meaning of ρ is as follows. In state p the automaton may reassign the registers whose indices belong to $\rho(p)$ with* any *pairwise different symbols of Σ. Of course, these symbols must differ from those in the registers whose indices do not belong to $\rho(p)$.*

The transition relation μ^c on $S^c \times \Sigma \times (S^c \times S^c)$ is defined similarly to that of top-down finite-memory automata. Let $p, p_0, p_1 \in S$ and $\boldsymbol{w}, \boldsymbol{w}_0, \boldsymbol{w}_1 \in \Sigma^{r\neq}$. Then $(p, \boldsymbol{w}), \sigma \to (p_0, \boldsymbol{w}_0), (p_1, \boldsymbol{w}_1)$ belongs to μ^c if and only if the following conditions are satisfied. Let $\boldsymbol{w} = w_1 w_2 \cdots w_r$, $\boldsymbol{w}_0 = w_{0,1} w_{0,2} \cdots w_{0,r}$, and $\boldsymbol{w}_1 = w_{1,1} w_{1,2} \cdots w_{1,r}$. Then

- $\boldsymbol{w}_0 = \boldsymbol{w}_1$,
- for all $i \notin \rho(p)$, $w_{0,i}\ (= w_{1,i}) = w_i$,

– for some $k = 1, 2, \ldots, r$, $w_{0,k} = \sigma \,(= w_{1,k})$, and
– $(p, k) \to (p_0, p_1) \in \mu$.

To extend F onto $S^c \times \Sigma$ we need one more bit of notation. For two assignments $\boldsymbol{v}, \boldsymbol{w} \in \Sigma^{r\neq}$, $\boldsymbol{v} = v_1 \cdots v_r$ and $\boldsymbol{w} = w_1 \cdots w_r$, and $S \subseteq \{1, \ldots, r\}$, we write $\boldsymbol{v} =_S \boldsymbol{w}$ if for all $i \notin S$, $v_i = w_i$. That is, $\boldsymbol{v}$ and $\boldsymbol{w}$ are equal "modulo" the symbols in the positions in S.

The set of *final* "Σ-relations" F^c is defined as follows. A pair $((p, \boldsymbol{w}), \sigma) \in F^c$ if for some $\boldsymbol{w}' \in \Sigma^{r\neq}$, $w' = w'_1 w'_2 \cdots w'_r$, such that $\boldsymbol{w} =_{\rho(p)} \boldsymbol{w}'$, $\sigma = w'_i$, and $(p, i) \in F$.

Now a run and acceptance of for $\downarrow$-NR-FMAs are defined exactly as for $\downarrow$-FMAs.

Example 5. (Cf. Example 2.) The set of Σ-trees L_2 from Example 2 is accepted by a $\downarrow$-NR-FMA that operates as follows. In the root of the input it "guesses" the symbol that appears at two different nodes and then (nondeterministically) verifies that the guess is correct.

Proposition 1. *If a set of Σ-trees is accepted by a $\downarrow$-FMA, then it is also accepted by a $\downarrow$-NR-FMA.*

Proof. Given a $\downarrow$-FMA $A = \langle S, s_0, \boldsymbol{u}, \rho, \mu, F\rangle$, consider the following two $\downarrow$-NR-FMAs $A^- = \langle S', s_0^-, \boldsymbol{u}, \rho', \mu', F'\rangle$ and $A^+ = \langle S', s_0^+, \boldsymbol{u}, \rho', \mu', F'\rangle$, where

– $S' = \bigcup\limits_{s \in S} \{s^-, s^+\}$;[6]
– $\rho'(s^-) = \emptyset$ and $\rho'(s^+) = \{\rho(s)\}$, $p \in S$;
– μ' is the union of
  - $\{(s^-, i, s_0^{\mp}, s_1^{\mp}) : (s, i, s_0, s_1) \in \mu\}$ and
  - $\{(s^+, \rho(s), s_0^{\mp}, s_1^{\mp}) : (s, \rho(s), s_0, s_1) \in \mu\}$;

  and
– $F' = \bigcup\limits_{s \in F} \{s^-, s^+\}$.

It can be easily seen that $L(A) = L(A) \cup L(A^+)$. Indeed, both automata are allowed to make a nondeterministic reassignment (according to ρ) only in states of the form s^+, but they have to use it immediately. Consequently, the reassigned symbol must be the current input. Thus, actually, both of them behave like A, except, possibly, the first move at the root of the input Σ-tree.

Since $\downarrow$-NR-FMA languages are closed under union; see Appendix C, the proposition follows. □

5.2 Bottom-Up Finite-Memory Automata with Nondeterministic Reassignment

Definition 5 below is the "bottom-up" counterpart of Definition 4.

[6] That is, S' consists of two copies of S.

Definition 5. *A* bottom-up finite-memory automaton with nondeterministic reassignment *($\uparrow$-NR-FMA) is a system $A = \langle S, s_0, \boldsymbol{u}, \rho, \tau, \mu, F \rangle$, where all components of A, but ρ are as Definition 1 and the* nondeterministic reassignment *ρ is a function from S into $2^{\{1,2,\dots,r\}}$.*

The relation μ^c on $S^c \times \Sigma \times (S^c \times S^c)$ is defined similarly to that of bottom-up finite-memory automata. The only difference is that each head may reassign nondeterministically a set of their registers before merging. Namely, for $p, p_0, p_1 \in S$ and $\boldsymbol{w}, \boldsymbol{w}_0, \boldsymbol{w}_1 \in \Sigma^{r\neq}$, $(p_0, \boldsymbol{w}_0, \sigma_0), (p_1, \boldsymbol{w}_1, \sigma_1) \to (p, \boldsymbol{w})$ belongs to μ^c if and only if the following holds.

Let $\boldsymbol{w} = w_1 w_2 \cdots w_r$, $\boldsymbol{w}_0 = w_{0,1} w_{0,2} \cdots w_{0,r}$, and $\boldsymbol{w}_1 = w_{1,1} w_{1,2} \cdots w_{1,r}$. Then there exist assignments $\boldsymbol{v}_0 = v_{0,1} \cdots v_{0,r}$ and $\boldsymbol{v}_1 = v_{1,1} \cdots v_{1,r}$, a transition $(p_0, k_0), (p_1, k_1) \to p \in \mu$, and a merging relation $((p_0, k_0), (p_1, k_1), t, f) \in \tau$ such that

- $v_{0,i} = w_{0,i}$ for all $i \notin \rho(p_0)$;
- $v_{1,i} = w_{1,i}$ for all $i \notin \rho(p_1)$;
- $v_{0,k_0} = \sigma_0$ and $v_{1,k_1} = \sigma_1$;
- $t = t(\boldsymbol{v}_0, \boldsymbol{v}_1)$; and
- $(\boldsymbol{v}_0, \boldsymbol{v}_1, \boldsymbol{w})$ is an instance of (t, f).

That is, for the assignment $\boldsymbol{w}$ at the parent node, f selects r out of $2r$ values of the "reassigned"assignments at the child nodes.

The set of final relations F defines the set of *final* "Σ-relations" F^c. A pair $((p, \boldsymbol{w}), \sigma) \in F^c$ if for some $\boldsymbol{w}' \in \Sigma^{r\neq}$, $w' = w'_1 w'_2 \cdots w'_r$, such that $\boldsymbol{w} =_{\rho(p)} \boldsymbol{w}'$ $\sigma = w'_i$, and $(p, i) \in F$.

Example 6. (Cf. Example 3.) The set of Σ-trees L_ϵ from Example 1 is accepted by a $\uparrow$-NR-FMA that operates as follows. In each leaf of the input the automaton "guesses" the symbol that appears at the root. Then, going down, it verifies that the input symbols are different from those at the leaves and that the "guessed" symbols are the same. Finally, when arriving to the root the automaton verifies that the guess is correct, i.e., the guessed symbol is one that appears at the root.

Proposition 2. *If a set of Σ-trees is accepted by a $\uparrow$-FMA, then it is also accepted by a $\uparrow$-NR-FMA.*

The proof of Proposition 2 is similar to that of Proposition 1 and is omitted.

5.3 The Main Result

We conclude this section with the main result of our paper stating that top-down and bottom-up finite-memory automata with nondeterministic reassignment have the same computation power.

Theorem 1. *A set of Σ-trees is accepted by a $\downarrow$-NR-FMA if and only if it is accepted by a $\uparrow$-NR-FMA. Moreover, the conversions of a $\downarrow$-NR-FMA to its equivalent $\uparrow$-NR-FMA and vice versa are effective.*

The proof of Theorem 1 is long and technical. It is presented in the appendices in the end of this paper.

Corollary 1. *Both $\downarrow$-FMA and $\uparrow$-FMA can be simulated by either of $\downarrow$-NR-FMA or $\uparrow$-NR-FMA.*

Note that by Examples 1, 2, 3, and 4, the inclusions provided by Corollary 1 are proper.

6 Decision Properties

In this section we show that the membership and emptiness problems for $\downarrow$-NR-FMAs are decidable. Thus, by Theorem 1 and Propositions 1 and 2, these problems are decidable for all other models of automata introduced in this paper. We also show that the universality and, consequently, the inclusion problems are undecidable for all models of automata introduced in this paper.

Propositions 3 and 4 below deal with decidability of the membership and emptiness problems. The former asks whether a given $\downarrow$-NR-FMA accepts a given Σ-tree, and the latter asks whether the language of a given $\downarrow$-NR-FMA is empty.

Proposition 3. *The membership problem for $\downarrow$-NR-FMAs is decidable.*

Proof. Let $A = \langle S, s_0, \boldsymbol{u}, \rho, \mu, F\rangle$ be a $\downarrow$-NR-FMA and let $\boldsymbol{\sigma} : T \to \Sigma$ be a Σ-tree. We contend that $\boldsymbol{\sigma} \in L(A)$ if and only if there is an accepting run of A on $\boldsymbol{\sigma}$ in which the assignment at each node belongs to $\Sigma_0^{r\neq}$, where

$$\Sigma_0 = \boldsymbol{\sigma}(T) \cup [\boldsymbol{u}] \cup \{\#\} \cup \{\theta_1, \theta_2, \ldots, \theta_r\}, \quad \theta_i \notin \boldsymbol{\sigma}(T),\ i = 1, 2, \ldots, r.$$

The "if" direction is immediate, and for the proof of the "only if" direction we just replace the symbols which appear in an accepting run of A on $\boldsymbol{\sigma}$, but do not belong to $\boldsymbol{\sigma}(T) \cup [\boldsymbol{u}] \cup \{\#\}$ with appropriate elements of $\{\theta_1, \theta_2, \ldots, \theta_r\}$.[7]

Therefore, given an input Σ-tree $\boldsymbol{\sigma} : T \to \Sigma$, we may restrict ourselves to the configurations of A from $S \times \Sigma_0^{r\neq}$, which brings us to an ordinary *finite alphabet* tree automaton. Since the membership problem for the latter is decidable, the proposition follows. □

Proposition 4. *The emptiness problem for $\downarrow$-NR-FMAs is decidable.*

The proof of Proposition 4 is based on Lemma 1 below.

Lemma 1. *Let $A = \langle S, s_0, \boldsymbol{u}, \rho, \mu, F\rangle$ be an r-register $\downarrow$-NR-FMA such that $L(A) \neq \emptyset$ and let $\Sigma_r = \{\theta_1, \theta_2, \ldots, \theta_r\}$ be an r-element subset of Σ that includes $[\boldsymbol{u}]$. Then there is a Σ-tree $\boldsymbol{\sigma} : T \to \Sigma_r$ in $L(A)$.*

Proof. Let $\boldsymbol{\sigma} : T \to \Sigma \in L(A)$ and let $R : T \to S^c$ be an accepting run of A on $\boldsymbol{\sigma}$. To construct a Σ-tree $\boldsymbol{\sigma} : T \to \Sigma_r$ in $L(A)$ we need the function $I : T \to \{1, 2, \ldots, r\}$ defined below.

[7] Obviously, such symbols can be introduced by reassignment, only.

– For a non-leaf node $n \in T$, if $R(n0) = (p, w_1 w_2 \cdots w_r)$ and $\boldsymbol{\sigma}(n) = w_i$, then $I(n) = i$.
– For a leaf node $n \in T$, if $R(n) = (p, w_1 w_2 \cdots w_r)$ and $\boldsymbol{w}' \in \Sigma^{r\neq}$, $w' = w'_1 w'_2 \cdots w'_r$, is such that $\boldsymbol{w} =_{\rho(p)} \boldsymbol{w}'$, $\sigma = w'_i$, and $(p, i) \in F$, then $I(n) = i$.

Let $\boldsymbol{u} = u_1 \cdots u_r$. We may assume that for each $i = 1, 2, \ldots, r$, $u_i \neq \#$ implies $u_i = \theta_i$. Then a Σ-tree $\boldsymbol{\sigma}_r : T \to \Sigma_r$ satisfying the lemma is defined by $\boldsymbol{\sigma}_r(n) = \theta_{I(n)}$, $n \in T$. This Σ-tree is accepted by the run of A whose state components are the same as of R and that *always* reassigns the i register with Θ_i, $i = 1, 2, \ldots, r$. □

Proof. (of Proposition 4) Let $A = \langle S, s_0, \boldsymbol{u}, \rho, \mu, F \rangle$ be an r-register $\downarrow$-NR-FMA and let $\Sigma_r = \{\theta_1, \theta_2, \ldots, \theta_r\}$ be an r-element subset of Σ that includes $[\boldsymbol{u}]$. It follows from Lemma 1 that $L(A) \neq \emptyset$ if and only if there is a Σ-tree $\boldsymbol{\sigma} : T \to \Sigma_r$ in $L(A)$.

Since on the inputs $\boldsymbol{\sigma} : T \to \Sigma_r$ the configurations of A belong to $S \times \Sigma_r^{r\neq}$, the emptiness of A is reduced to the emptiness of an ordinary *finite alphabet* tree automaton that is is decidable. □

It was shown that in [13] the universality problem of finite-memory automata is undecidable.[8] Consequently, this problem is also undecidable for all above models of tree automata.

Proposition 5. *The universality problem for $\downarrow$-FMAs and $\uparrow$-FMAs is undecidable.*

Corollary 2. *The inclusion problem for $\downarrow$-FMAs and $\uparrow$-FMAs is undecidable.*

7 Context-Free Languages over Infinite Alphabets and Their Relationship with Tree Automata

In this section we recall the definition of *quasi context-free languages* from [2] and show how they are related to the tree languages introduced in this paper.

In short, a *quasi context-free grammar* is a context-free grammar, where each variable carries the same number r of of registers. The terminal alphabet of a grammar G is the set $\{1, 2, \ldots, r\}$. Let V be the set of variables of G. The productions of G are of the form

$$(A, k) \to \alpha_1 \alpha_2 \cdots \alpha_n,$$

where $1 \leq k \leq r$ and $\alpha_i \in V \cup \{1, \ldots, r\}$, $i = 1, 2, \ldots, n$. The above production allows us

– to replace the content of the kth register carried by A with any symbol of Σ that differs from the symbols stored in the other registers,[9] and

[8] That is, is undecidable whether a given finite-memory automaton accepts Σ^*.
[9] Actually, automata with nondeterministic reassignment were motivated by [2].

– to replace A with the word $\beta_1 \cdots \beta_n$, where β_i is the content of the jth register, if $\alpha_i = j$, and is α_i, if α_i is a variable, $i = 1, 2, \ldots, n$.

The language generated by G consists of all words in Σ^* obtained by repeatedly applying the productions of G, starting with the *start* variable. It is called a *quasi context-free* language, cf. DTDs in Sect. 2. The precise definition of quasi context-free grammars and languages is as follows.

Definition 6. ([2, Definition 1]) *An* infinite-alphabet context-free grammar *is a system $G = \langle V, \boldsymbol{u}, R, S \rangle$, where*

– *V is a finite set of* variables *disjoint with Σ;*
– *$\boldsymbol{u} = u_1 u_2 \cdots u_r \in \Sigma^{r\neq}$ is the* initial *assignment;*
– *$R \subseteq (V \times \{1, 2, \ldots, r\}) \times (V \cup \{1, 2, \ldots, r\})^*$ is a set of* productions*, whose elements are written in the form $(A, i, \boldsymbol{a})$ as $(A, i) \to \boldsymbol{a}$, where $A \in V$, $i = 1, 2, \ldots, r$, and $\boldsymbol{a} \in (V \cup \{1, 2, \ldots, r\})^*$; and*
– *$S \in V$ is the* start *variable.*

For $A \in V$, $\boldsymbol{w} = w_1 w_2 \cdots w_r \in \Sigma^{r\neq}$, and $\boldsymbol{X} = X_1 X_2 \cdots X_n \in (\Sigma \cup (V \times \Sigma^{r\neq}))^*$, we write $(A, \boldsymbol{w}) \Rightarrow \boldsymbol{X}$ if there exist a production $(A, i) \to \boldsymbol{a} \in R$, $\boldsymbol{a} = a_1 a_2 \cdots a_n \in (V \cup \{1, 2, \ldots, r\})^*$, and a symbol $\sigma \notin [\boldsymbol{w}] \setminus \{w_i\}$ such that the condition below is satisfied.

Let $\boldsymbol{w}' \in \Sigma^{r\neq}$ be obtained from $\boldsymbol{w}$ by replacing w_i with σ. Then, for $j = 1, 2, \ldots, n$ the following holds.

– If $a_j = k$ for some $k = 1, 2, \ldots, r$, then $X_j = w'_k$.
– If $a_j = B$ for some $B \in V$, then $X_j = (B, \boldsymbol{w}')$.

For two words $\boldsymbol{X}$ and $\boldsymbol{Y}$ over $\Sigma \cup (V \times \Sigma^{r\neq})$, we write $\boldsymbol{X} \Rightarrow \boldsymbol{Y}$ if there exist words $\boldsymbol{X}_1$, $\boldsymbol{X}_2$, and $\boldsymbol{X}_3$ over $\Sigma \cup (V \times \Sigma^{r\neq})$ and $(A, \boldsymbol{w}) \in V \times \Sigma^{r\neq}$, such that $\boldsymbol{X} = \boldsymbol{X}_1 (A, \boldsymbol{w}) \boldsymbol{X}_2$, $\boldsymbol{Y} = \boldsymbol{X}_1 \boldsymbol{X}_3 \boldsymbol{X}_2$, and $(A, \boldsymbol{w}) \Rightarrow \boldsymbol{X}_3$.

As usual, the reflexive and transitive closure of $\Rightarrow$ is denoted by $\Rightarrow^*$. The language $L(G)$ generated by G is defined by $L(G) = \{\boldsymbol{\sigma} \in \Sigma^* : (S, \boldsymbol{u}) \Rightarrow^* \boldsymbol{\sigma}\}$ and is referred to as a *quasi-context-free* language.

Example 7. Let G be a 1-register grammar with the set of variables $V = \{S\}$, the *initial assignment* $\#$, and the following two production.

$$(S, 1) \to 1S1 \mid \epsilon.$$

Then $L(G) = \{\boldsymbol{\sigma}\boldsymbol{\sigma}^R \mid \boldsymbol{\sigma} \in \Sigma^*\}$.[10] For example, the word $\sigma_1\sigma_2\sigma_3\sigma_3\sigma_2\sigma_1$ is derived as follows.

$$\begin{aligned}(S, \#) &\Rightarrow \sigma_1 (S, \sigma_1) \sigma_1 \Rightarrow \sigma_1 \sigma_2 (S, \sigma_2) \sigma_2 \sigma_1 \\ &\Rightarrow \sigma_1 \sigma_2 \sigma_3 (S, \sigma_3) \sigma_3 \sigma_2 \sigma_1 \Rightarrow \sigma_1 \sigma_2 \sigma_3 \sigma_3 \sigma_2 \sigma_1.\end{aligned}$$

[10] As usual, $\boldsymbol{\sigma}^R$ is the *reversal* of $\boldsymbol{\sigma}$.

We end this paper with the theorem below that relates quasi context-free languages to the tree languages introduced in this paper. Recall that for a Σ-tree $\boldsymbol{\sigma} : T \to \Sigma$, the *frontier* $\ell(\boldsymbol{\sigma})$ of $\boldsymbol{\sigma}$ is the word $\boldsymbol{\sigma}(n_1)\boldsymbol{\sigma}(n_2)\cdots\boldsymbol{\sigma}(n_m)$, where $n_1, n_2, \ldots, n_m$ is the list of all leaf nodes of T in the lexicographical order. Below the set frontiers of all elements of a set of Σ-trees L is denoted by $\ell(L)$: $\ell(L) = \{\ell(\boldsymbol{\sigma}) : \boldsymbol{\sigma} \in L\}$.

Theorem 2. *Let L be a tree language accepted by a top-down (or bottom-up) finite-memory automaton A with a deterministic (or non-deterministic) reassignment. Then $\ell(L)$ is quasi-context-free language.*

Conversely, for every quasi-context-free language L, there exists a top-down (or bottom-up) finite-memory automata A with deterministic (or non-deterministic) reassignment such that $\ell(L(A)) = L$.

We omit the proof that is quite straightforward. For example, any Σ- tree $\boldsymbol{\sigma}$ accepted by a $\downarrow$-FMA A, after an appropriate modification, can be thought of as a derivation tree of the word $\ell(\boldsymbol{\sigma})$. Conversely, given a quasi context-free grammar G, we may assume that all derivation trees of the words in $L(G)$ are binary.[11] Therefore, the set of productions of G can be thought of as the set of transition of a $\downarrow$-FMA A, implying that $\ell(L(A))$ is exactly the language generated by G.

Acknowledgment

This research was supported by the Jewish communities of Germany research fund, by the Technion vice-president fund for the promotion of research at the Technion, and by a grant from the Software Technology Laboratory (STL) in the Department of Computer Science, the Technion. In addition, the work of the second author was supported by the Raphael and Miriam Mishan Fellowship.

References

1. Bex, G.J., Maneth, S., Neven, F.: A formal model for an expressive fragment of XSLT. Information and System 27(1), 21–39 (2002)
2. Cheng, E.Y.C., Kaminski, M.: Context-free languages over infinite alphabets. Acta Informatica 35, 245–267 (1998)
3. Comon, H., et al.: Tree Automata Techniques and Applications (2005), `http://www.grappa.univ-lille3.fr/tata/`
4. Doner, J.E.: Tree acceptors and some of their applications. Journal of Computer and System Sciences 4, 406–451 (1970)
5. Kaminski, M., Francez, N.: Finite-memory automata. In: Proceedings of the 31th Annual IEEE Symposium on Foundations of Computer Science, pp. 683–688. IEEE Computer Society Press, Los Alamitos (1990)

[11] It is well known that any tree can be converted to a binary tree that preserves the order of the leaves.

6. Kaminski, M., Francez, N.: Finite-memory automata. Theoretical Computer Science 138, 329–363 (1994)
7. Kaminski, M., Tan, T.: Regular expressions for languages over infinite alphabets. Fundamenta Informaticae 69, 301–318 (2006)
8. Milo, T., Suciu, D., Vianu, V.: Type checking for XML transformers. Journal of Computer and System Sciences 66, 66–97 (2003)
9. Neven, F., Schwentick, T.: Expressive and efficient pattern languages for tree-structured data. In: Proceedings of the Nineteenth International Symposium on Principles of Database Systems, pp. 145–156. ACM Press, New York (2000)
10. Neven, F., Schwentick, T., Vianu, V.: Towards regular languages over infinite alphabets. In: Sgall, J., Pultr, A., Kolman, P. (eds.) MFCS 2001. LNCS, vol. 2136, pp. 560–572. Springer, Heidelberg (2001)
11. Neven, F.: Automata, logic and XML. In: Bradfield, J.C. (ed.) CSL 2002 and EACSL 2002. LNCS, vol. 2471, pp. 2–26. Springer, Heidelberg (2002)
12. Neven, F., Schwentick, T.: Query automata on finite trees. Theoretical Computer Science 275, 633–674 (2002)
13. Neven, F., Schwentick, T., Vianu, V.: Finite state machines for strings over infinite alphabets. ACM Transactions on Computational Logic 5, 403–435 (2004)
14. Papakonstantinou, Y., Vianu, V.: DTD inference for views of XML data. In: Proceedings of the Twentieth International Symposium on Principles of Database Systems, pp. 35–46. ACM Press, New York (2001)
15. Rabin, M.: Decidability of second order theories and automata on infinite trees. Transactions of the American Mathematical Society 141, 1–35 (1969)
16. Ray, E.: Learning XML. O'Reilly & Associates, Inc, Sebastopol (2001)
17. Thatcher, J., Wright, J.: Generalized finite automata theory. Mathematical System Theory 2, 57–81 (1968)
18. Vianu, V.: A web odyssey: from Codd to XML. In: Proceedings of the 20th International Symposium on Principles of Database Systems, pp. 1–15. ACM Press, New York (2001)
19. XML Core Working Group: Extensible Markup Language (XML). World Wide Web Consortium, `http://www.w3.org/XML/`
20. Zeitlin, D.: Look-ahead finite-memory automata. Master's thesis, Department of Computer Science, Technion - Israel Institute of Technology (2006)

A Proof of the "only if" Part of Theorem 1

For an r-register $\downarrow$-NR-FMA $A = \langle S, s_0, \boldsymbol{u}, \rho, \mu, F\rangle$ we construct an r-register $\uparrow$-NR-FMA $\widetilde{A} = \langle \widetilde{S}, \widetilde{s}_0, \widetilde{\boldsymbol{u}}, \widetilde{\rho}, \widetilde{\tau}, \widetilde{\mu}, \widetilde{F}\rangle$ such that $L(A) = L(\widetilde{A})$.

Similarly to the proof of [7, Lemma 5.1] it can be shown that, without loss of generality, the following assumptions hold.

- $\boldsymbol{u} = \#^{r-m}\theta_1 \cdots \theta_m$, where $\theta_1, \ldots, \theta_m \in \Sigma$, and
- only the first $r - m$ registers of A can be reassigned, i.e, the range of ρ is a subset of $\{1, 2, \ldots, r - m\}$.

We precede the formal description of $\widetilde{A}$ with a general intuitive explanation. One would expect the construction to be just the transition reversing, i.e., a transition $(p, k) \to (p_0, p_1)$ of A to become a "transition" $((p_0, p_1), k) \to p$ of $\widetilde{A}$.

This is indeed almost so, but with the following modification. Since transitions of a $\uparrow$-NR-FMA merge two heads and depend on two input symbols from Σ, we combine two transitions of A into one "reversed" transition of $\widetilde{A}$. That is, two transitions $(p_0, k_0) \rightarrow (p_{00}, p_{01})$ and $(p_1, k_1) \rightarrow (p_{10}, p_{11})$ of A are combined into one transition $((p_{00}, p_{01}), k_0), ((p_{10}, p_{11}), k_1) \rightarrow (p_0, p_1)$ of $\widetilde{A}$ and "moved" one level up, as illustrated in Fig. 3 and 4 below. Fig. 3 shows an application of two top-down transitions at two nodes (sharing the same parent node) labeled σ_0 and σ_1. Fig. 4 shows their reversal bottom-up transition applied at the same two nodes, but in the converse direction.

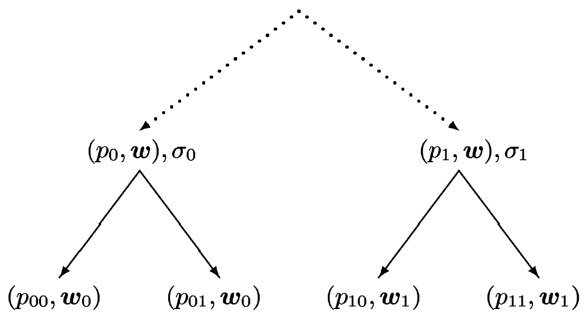

$(p_0, \boldsymbol{w}), \sigma_0$
$(p_1, \boldsymbol{w}), \sigma_1$
$(p_{00}, \boldsymbol{w}_0)$
$(p_{01}, \boldsymbol{w}_0)$
$(p_{10}, \boldsymbol{w}_1)$
$(p_{11}, \boldsymbol{w}_1)$

Fig. 3. An application of two transitions of A

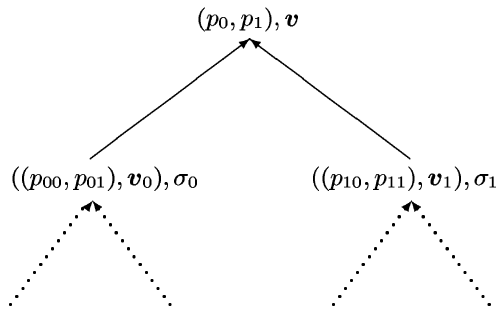

$(p_0, p_1), \boldsymbol{v}$
$((p_{00}, p_{01}), \boldsymbol{v}_0), \sigma_0$
$((p_{10}, p_{11}), \boldsymbol{v}_1), \sigma_1$

Fig. 4. Reversing and combining two transitions of A

Note that assignments $\boldsymbol{w}_0$ and $\boldsymbol{w}_1$ equal to $\boldsymbol{w}$ "modulo" the symbols in the positions belonging to $\rho(p_0)$ and $\rho(p_1)$, respectively; and except for the symbols at the positions in $\rho(p_0) \cap \rho(p_1)$, $\boldsymbol{w}$ can be recovered from $\boldsymbol{w}_0$ and $\boldsymbol{w}_1$. The symbols at the positions in $\rho(p_0) \cap \rho(p_1)$ can be "guessed" by a nondeterministic reassignment of the "reversal" automaton $\widetilde{A}$. Thus, for $p_0, p_1 \in S$, we let $\widetilde{\rho}((p_0, p_1))$ be $\rho(p_0) \cap \rho(p_1)$. In fact, the reversed transitions in $\widetilde{A}$ yield the assignments $\boldsymbol{v}, \boldsymbol{v}_0, \boldsymbol{v}_1$ such that $\boldsymbol{v} =_{\rho(p_0) \cap \rho(p_1)} \boldsymbol{w}$, $\boldsymbol{v}_0 =_{\rho(p_{00}) \cap \rho(p_{01})} \boldsymbol{w}_0$ and $\boldsymbol{v}_1 =_{\rho(p_{10}) \cap \rho(p_{11})} \boldsymbol{w}_1$.

To define the transition relation $\widetilde{\mu}$ and the merging relation $\widetilde{\tau}$ we need the following definition and the corresponding auxiliary result.

Let t be an r-type and let f be a valid selector for t. The pair (t,f) is called an *inverse structure associated with a pair of states* (p_0,p_1) if the following conditions are satisfied.

– $(i,i) \in t$ for all $i \in \{1,\dots,r\} - (\rho(p_0) \cup \rho(p_1))$.

– $f(i) = \begin{cases} (0,i) \text{ or } (1,i) & \text{for } i \in \{1,\dots,r\} \setminus (\rho(p_0) \cup \rho(p_1)) \\ (1,i) & \text{for } i \in \rho(p_0) \setminus \rho(p_1) \\ (0,i) & \text{for } i \in \rho(p_1) \setminus \rho(p_0) \end{cases}$

Note that the value of f on the elements of $\rho(p_0) \cap \rho(p_1)$ is arbitrary. The set of all inverse structures associated with a pair of states (p_0,p_1) will be denoted $I(p_0,p_1)$.

Lemma 2. *Let (t,f) be an inverse structure associated with a pair of states (p_0,p_1) and let $\boldsymbol{v}_0,\boldsymbol{v}_1,\boldsymbol{v},\boldsymbol{w} \in \Sigma^{r\neq}$ be such that $\boldsymbol{v}_0 =_{\rho(p_0)} \boldsymbol{w}$ and $\boldsymbol{v}_1 =_{\rho(p_1)} \boldsymbol{w}$. Then $(\boldsymbol{v}_0,\boldsymbol{v}_1,\boldsymbol{v})$ is an instance of (t,f) if and only if $\boldsymbol{v} =_{\rho(p_0)\cap\rho(p_1)} \boldsymbol{w}$.*

Moreover, if $\boldsymbol{v} =_{\rho(p_0)\cap\rho(p_1)} \boldsymbol{w}$ and $(\boldsymbol{v}_0,\boldsymbol{v}_1,\boldsymbol{v})$ is an instance of (t,f), then $\boldsymbol{v}_0 =_{\rho(p_0)} \boldsymbol{w}$ and $\boldsymbol{v}_1 =_{\rho(p_1)} \boldsymbol{w}$.

Proof. Let $\boldsymbol{v}_0 = v_{0,1}\cdots v_{0,r}$, $\boldsymbol{v}_1 = v_{1,1}\cdots v_{1,r}$, $\boldsymbol{v} = v_1\cdots v_r$, and $\boldsymbol{w} = w_1\cdots w_r$. It immediately follows from the definition that $(\boldsymbol{v}_0,\boldsymbol{v}_1,\boldsymbol{v})$ is an instance of (t,f) if and only if

$$v_i = \begin{cases} v_{0,i} = w_i \text{ if } i \notin \rho(p_1) \\ v_{1,i} = w_i \text{ if } i \notin \rho(p_0) \end{cases}, \quad i = 1,\dots,r,$$

which is equivalent to $\boldsymbol{v} =_{\rho(p_0)\cap\rho(p_1)} \boldsymbol{w}$.

Now we prove the second part. Let $\boldsymbol{v} =_{\rho(p_0)\cap\rho(p_1)} \boldsymbol{w}$. In particular, for all $i \notin \rho(p_0)$, $w_i = v_i$. If $(\boldsymbol{v}_0,\boldsymbol{v}_1,\boldsymbol{v})$ is an instance of (t,f), by the definition of (t,f), $v_i = v_{0,i}$. Therefore, $\boldsymbol{w} =_{\rho(p_0)} \boldsymbol{v}_0$. In a similar way we can show that $\boldsymbol{w} =_{\rho(p_1)} \boldsymbol{v}_1$. □

Now we are ready to define the desired ↑-NR-FMA $\widetilde{A} = \langle \widetilde{S}, \widetilde{s}_0, \widetilde{\boldsymbol{u}}, \widetilde{\rho}, \widetilde{\tau}, \widetilde{\mu}, \widetilde{F} \rangle$.

– $\widetilde{S} = S \times S \cup \{\widetilde{s}_0\}$, where $\widetilde{s}_0$ is a new state.
– The initial state of $\widetilde{A}$ is $\widetilde{s}_0$.
– $\widetilde{\boldsymbol{u}} = \#^{r-m}\theta_1\cdots\theta_m$.
– $\widetilde{\rho}(\widetilde{s}_0) = \{1,\dots,r-m\}$, and
  $\widetilde{\rho}((p_0,p_1)) = \rho(p_0) \cap \rho(p_1)$, for all $(p_0,p_1) \in \widetilde{S}$.
– $\widetilde{\mu} = \widetilde{\mu}_1 \cup \widetilde{\mu}_2 \cup \widetilde{\mu}_3 \cup \widetilde{\mu}_4$, where
  - $\widetilde{\mu}_1 = \{(\widetilde{s}_0,k_0),(\widetilde{s}_0,k_1) \to (p_0,p_1) : (p_0,k_0),(p_1,k_1) \in F\}$;
  - $\widetilde{\mu}_2 = \{(\widetilde{s}_0,k_0),((p_{10},p_{11}),k_1) \to (p_0,p_1) :$
    $(p_0,k_0) \in F \text{ and } (p_1,k_1) \to (p_{10},p_{11}) \in \mu\}$;
  - $\widetilde{\mu}_3 = \{((p_{00},p_{01}),k_0),(\widetilde{s}_0,k_1) \to (p_0,p_1) :$
    $(p_0,k_0) \to (p_{00},p_{01}) \in \mu \text{ and } (p_1,k_1) \in F\}$;
  - $\widetilde{\mu}_4 = \{((p_{00},p_{01}),k_0),((p_{10},p_{11}),k_1) \to (p_0,p_1) :$
    $(p_0,k_0) \to (p_{00},p_{01}),(p_1,k_1) \to (p_{10},p_{11}) \in \mu\}$.
– $\widetilde{\tau} = \widetilde{\tau}_1 \cup \widetilde{\tau}_2 \cup \widetilde{\tau}_3 \cup \widetilde{\tau}_4$, where

- $\widetilde{\tau}_1 = \{((\widetilde{s}_0, k_0), (\widetilde{s}_0, k_1), t, f) :$ for some $p_0, p_1 \in S$, $(p_0, k_0), (p_1, k_1) \in F$ and $(t, f) \in I(p_0, p_1)\}$;[12]
- $\widetilde{\tau}_2 = \{((\widetilde{s}_0, k_0), ((p_{1,0}, p_{1,1}), k_1), t, f) :$ for some $p_0, p_1 \in S$, $(p_0, k_0) \in F$ and $(p_1, k_1) \to (p_{1,0}, p_{1,1}) \in \mu$ and $(t, f) \in I(p_0, p_1)\}$;
- $\widetilde{\tau}_3 = \{(((p_{0,0}, p_{0,1}), k_0), (\widetilde{s}_0, k_1), t, f) :$ for some $p_0, p_1 \in S$, $(p_0, k_0) \to (p_{0,0}, p_{0,1}) \in \mu$ and $(p_1, k_1) \in F$ and $(t, f) \in I(p_0, p_1)\}$;
- $\widetilde{\tau}_4 = \{(((p_{00}, p_{01}), k_0), ((p_{10}, p_{11}), k_1), t, f) :$ for some $p_0, p_1 \in S$, $(p_0, k_0) \to (p_{00}, p_{01}), (p_1, k_1) \to (p_{10}, p_{11}) \in \mu$ and $(t, f) \in I(p_0, p_1)\}$.

– $\widetilde{F} = \{(q_0, q_1), k) : (s_0, k) \to (q_0, q_1) \in \mu\}$.

The proof of the equality $L(\widetilde{A}) = L(A)$ is based on Lemma 3 below. Roughly speaking, Lemma 3 is the formal description of the construction in Fig. 3 and 4. It shows how an accepting run of A can be "reversed" into an accepting run of $\widetilde{A}$, or more precisely, it shows how transitions from μ^c are converted into transitions from $\widetilde{\mu}^c$, and vice versa.

Lemma 3 consists of four parts corresponding to the type of nodes on which transitions take place. Its part (i) shows how an accepting run of A can be "reversed" into an accepting run of $\widetilde{A}$ at the leaf nodes and vice versa. Part (ii) shows how an accepting run of A can be "reversed" into an accepting run of $\widetilde{A}$ when one of the two sibling nodes is a leaf and the other is an interior node. Part (iii) of the lemma settles the case of the interior nodes. Finally, part (iv) of Lemma 3 deals with the case of the root node ϵ.

Lemma 3

(i) (See Fig. 5 and 6.) *If* $((p_0, \boldsymbol{w}), \sigma_0), ((p_1, \boldsymbol{w}), \sigma_1) \in F^c$, *then there is an assignment* $\boldsymbol{v} =_{\widetilde{\rho}(p_0,p_1)} \boldsymbol{w}$ *such that*

$$((\widetilde{s}_0, \widetilde{\boldsymbol{u}}), \sigma_0), ((\widetilde{s}_0, \widetilde{\boldsymbol{u}}), \sigma_1) \to ((p_0, p_1), \boldsymbol{v}) \in \widetilde{\mu}^c.$$

Conversely, if

$$((\widetilde{s}_0, \widetilde{\boldsymbol{u}}), \sigma_0), ((\widetilde{s}_0, \widetilde{\boldsymbol{u}}), \sigma_1) \to ((p_0, p_1), \boldsymbol{v}) \in \widetilde{\mu}^c.$$

and $\boldsymbol{w} =_{\widetilde{\rho}(p_0,p_1)} \boldsymbol{v}$, *then* $((p_0, \boldsymbol{w}), \sigma_0), ((p_1, \boldsymbol{w}), \sigma_1) \in F^c$.

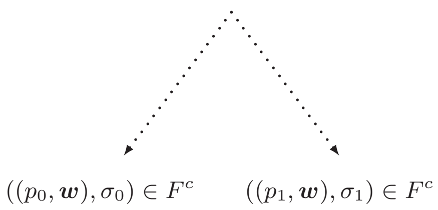
(ii) (a) (See Fig. 7 and 8.) *If* $((p_0, \boldsymbol{w}), \sigma_0) \in F^c$,

$$((p_1, \boldsymbol{w}), \sigma_1) \to (p_{10}, \boldsymbol{w}_1), (p_{11}, \boldsymbol{w}_1) \in \mu^c$$

and $\boldsymbol{v}_1 =_{\widetilde{\rho}(p_{10},p_{11})} \boldsymbol{w}_1$, *then there is an assignment* $\boldsymbol{v} =_{\widetilde{\rho}(p_0,p_1)} \boldsymbol{w}$ *such that*

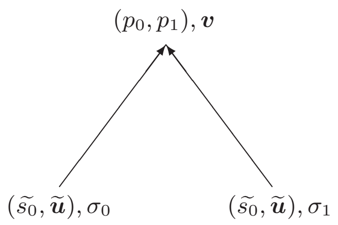
$$((\widetilde{s}_0, \widetilde{\boldsymbol{u}}), \sigma_0), (((p_{10}, p_{11}), \boldsymbol{v}_1), \sigma_1) \to ((p_0, p_1), \boldsymbol{v}) \in \widetilde{\mu}^c.$$

[12] Recall that $I(p_0, p_1)$ denotes the set of all inverse structures associated with the pair of states (p_0, p_1).



$((p_0, \boldsymbol{w}), \sigma_0) \in F^c$
$((p_1, \boldsymbol{w}), \sigma_1) \in F^c$

Fig. 5. Application of two final relations of A



$(p_0, p_1), \boldsymbol{v}$
$(\widetilde{s}_0, \widetilde{\boldsymbol{u}}), \sigma_0$
$(\widetilde{s}_0, \widetilde{\boldsymbol{u}}), \sigma_1$

Fig. 6. Reversing and combining two final relations of A

Conversely, if

$$((\widetilde{s}_0, \widetilde{\boldsymbol{u}}), \sigma_0), (((p_{10}, p_{11}), \boldsymbol{v}_1), \sigma_1) \to ((p_0, p_1), \boldsymbol{v}) \in \widetilde{\mu}^c$$

and $\boldsymbol{w} =_{\widetilde{\rho}(p_0,p_1)} \boldsymbol{v}$*, then* $((p_0, \boldsymbol{w}), \sigma_0) \in F^c$ *and there is an assignment* $\boldsymbol{w}_1 =_{\widetilde{\rho}(p_{10},p_{11})} \boldsymbol{v}_1$ *such that*

$$((p_1, \boldsymbol{w}), \sigma_1) \to (p_{10}, \boldsymbol{w}_1), (p_{11}, \boldsymbol{w}_1) \in \mu^c.$$

(b) *If*

$$((p_0, \boldsymbol{w}), \sigma_0) \to (p_{00}, \boldsymbol{w}_0), (p_{01}, \boldsymbol{w}_0) \in \mu^c,$$

$((p_1, \boldsymbol{w}), \sigma_1) \in F^c$ *and* $\boldsymbol{v}_0 =_{\widetilde{\rho}(p_{00},p_{01})} \boldsymbol{w}_0$*, then there is an assignment* $\boldsymbol{v} =_{\widetilde{\rho}(p_0,p_1)} \boldsymbol{w}$ *such that*

$$(((p_{00}, p_{01}), \boldsymbol{v}_0), \sigma_0), ((\widetilde{s}_0, \widetilde{\boldsymbol{u}}), \sigma_1) \to ((p_0, p_1), \boldsymbol{v}) \in \widetilde{\mu}^c.$$

Conversely, if

$$(((p_{00}, p_{01}), \boldsymbol{v}_0), \sigma_0), ((\widetilde{s}_0, \widetilde{\boldsymbol{u}}), \sigma_1) \to ((p_0, p_1), \boldsymbol{v}) \in \widetilde{\mu}^c$$

and $\boldsymbol{w} =_{\widetilde{\rho}(p_0,p_1)} \boldsymbol{v}$*, then there is an assignment* $\boldsymbol{w}_0 =_{\widetilde{\rho}(p_{00},p_{01})} \boldsymbol{v}_0$ *such that*

$$((p_0, \boldsymbol{w}), \sigma_0) \to (p_{00}, \boldsymbol{w}_0), (p_{01}, \boldsymbol{w}_0) \in \mu^c$$

and $((p_1, \boldsymbol{w}), \sigma_1) \in F^c$*.*

(*iii*) (See Fig. 3 and 4.) *If*

$$((p_0, \boldsymbol{w}), \sigma_0) \to (p_{00}, \boldsymbol{w}_0), (p_{01}, \boldsymbol{w}_0) \in \mu^c,$$

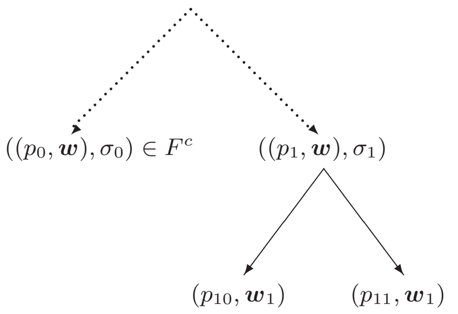
$$((p_1, \boldsymbol{w}), \sigma_1) \to (p_{10}, \boldsymbol{w}_1), (p_{11}, \boldsymbol{w}_1) \in \mu^c,$$

$\boldsymbol{v}_0 =_{\widetilde{\rho}(p_{00},p_{01})} \boldsymbol{w}_0$ *and* $\boldsymbol{v}_1 =_{\widetilde{\rho}(p_{10},p_{11})} \boldsymbol{w}_1$*, then there is an assignment* $\boldsymbol{v} =_{\widetilde{\rho}(p_0,p_1)} \boldsymbol{w}$ *such that*

$$(((p_{00}, p_{01}), \boldsymbol{v}_0), \sigma_0), (((p_{10}, p_{11}), \boldsymbol{v}_1), \sigma_1) \to ((p_0, p_1), \boldsymbol{v}) \in \widetilde{\mu}^c.$$

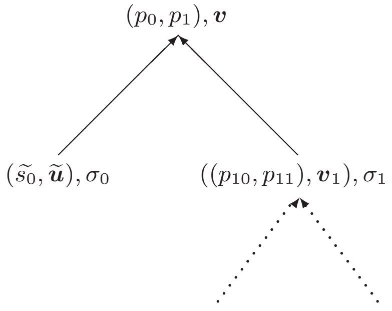
Conversely, if

$$(((p_{00}, p_{01}), \boldsymbol{v}_0), \sigma_0), (((p_{10}, p_{11}), \boldsymbol{v}_1), \sigma_1) \to ((p_0, p_1), \boldsymbol{v}) \in \widetilde{\mu}^c$$



$((p_0, \boldsymbol{w}), \sigma_0) \in F^c$
$((p_1, \boldsymbol{w}), \sigma_1)$
$(p_{10}, \boldsymbol{w}_1)$
$(p_{11}, \boldsymbol{w}_1)$

Fig. 7. An application of a final relation and a transitions of A



$(p_0, p_1), \boldsymbol{v}$
$(\widetilde{s_0}, \widetilde{\boldsymbol{u}}), \sigma_0$
$((p_{10}, p_{11}), \boldsymbol{v}_1), \sigma_1$

Fig. 8. Reversing a final relation and a transition of A into a transition of $\widetilde{A}$

and $\boldsymbol{w} =_{\widetilde{\rho}(p_0,p_1)} \boldsymbol{v}$*, then there are assignments* $\boldsymbol{w}_0 =_{\widetilde{\rho}(p_{00},p_{01})} \boldsymbol{v}_0$ *and* $\boldsymbol{w}_1 =_{\widetilde{\rho}(p_{10},p_{11})} \boldsymbol{v}_1$ *such that*

$$((p_0, \boldsymbol{w}), \sigma_0) \to (p_{00}, \boldsymbol{w}_0), (p_{01}, \boldsymbol{w}_0) \in \mu^c$$

and

$$((p_1, \boldsymbol{w}), \sigma_1) \to (p_{10}, \boldsymbol{w}_1), (p_{11}, \boldsymbol{w}_1) \in \mu^c.$$

(*iv*) (See Fig. 9 and 10.) *If*

$$((s_0, \boldsymbol{u}), \sigma) \to (p_0, \boldsymbol{w}), (p_1, \boldsymbol{w}) \in \mu^c$$

and $\boldsymbol{v} =_{\widetilde{\rho}(p_0,p_1)} \boldsymbol{w}$*, then* $(((p_0, p_1), \boldsymbol{v}), \sigma) \in \widetilde{F}^c$*.*
Conversely, if $(((p_0, p_1), \boldsymbol{v}), \sigma) \in \widetilde{F}^c$*, then there is an assignment* $\boldsymbol{w} =_{\widetilde{\rho}(p_0,p_1)} \boldsymbol{v}$ *such that*

$$((s_0, \boldsymbol{u}), \sigma) \to (p_0, \boldsymbol{w}), (p_1, \boldsymbol{w}) \in \mu^c.$$

We postpone the proof of the lemma to the end of this appendix and prove the "only if" part of Theorem 1 first.

Proof. (of the "only if" part of Theorem 1.) We prove that $L(A) = L(\widetilde{A})$ by showing how to convert an accepting run of A on a Σ-tree $\boldsymbol{\sigma} : T \to \Sigma$, into an accepting run of $\widetilde{A}$ on $\boldsymbol{\sigma}$, and vice versa.

For an accepting run $R : T \to S^c$, $R(n) = (p_n, \boldsymbol{w}_n)$, $n \in T$, of A on $\boldsymbol{\sigma}$ we construct an accepting run $\widetilde{R} : T \to S^c$ of $\widetilde{A}$ on $\boldsymbol{\sigma}$ bottom-up, i.e., from the leaves to the root, by induction, as follows.

By definition, for a leaf node $n \in T$, $\widetilde{R}(n) = \widetilde{s_0}^c = (\widetilde{s_0}, \widetilde{\boldsymbol{u}})$, and for an interior node $n \in T$, $\widetilde{R}(n) = ((p_{n0}, p_{n1}), \boldsymbol{v}_n)$, where the assignment $\boldsymbol{v}_n =_{\widetilde{\rho}(p_{n0},p_{n1})} \boldsymbol{w}_{n0}(= \boldsymbol{w}_{n1})$ is defined as follows.[13]

[13] Recall that by definition of μ^c, $\boldsymbol{w}_{n0} = \boldsymbol{w}_{n1}$.

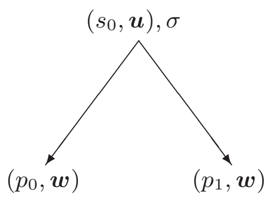


Fig. 9. An application of a transition of A at the root node

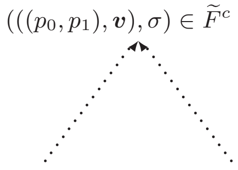

$(((p_0, p_1), \boldsymbol{v}), \sigma) \in \widetilde{F}^c$

Fig. 10. Reversing a transition of A at the root node into a final relation of $\widetilde{A}$

- If both children of n are leaf nodes, then $\boldsymbol{v}_n$ is provided by part (i) of Lemma 3.
- If one child of n is a leaf node and the other is an interior node, then $\boldsymbol{v}_n$ is provided by part (ii) of Lemma 3.
- Finally, if both children of n are interior nodes, then $\boldsymbol{v}_n$ is provided by part (iii) of Lemma 3.

Now, by part (iv) of Lemma 3, $\widetilde{R}$ is an accepting run of $\widetilde{A}$ on $\boldsymbol{\sigma}$.

The converse direction can shown in a similar manner. That is, an accepting run R of A on $\boldsymbol{\sigma}$ is constructed from an accepting run $\widetilde{R}$ of $\widetilde{A}$ by applying the converse direction of Lemma 3. □

It remains to prove Lemma 3.

Proof. (of Lemma 3) We will prove only part (iii) of the lemma. The proofs of the other parts are very similar.

Let

$$(p_0, \boldsymbol{w}), \sigma_0 \to (p_{00}, \boldsymbol{w}_0), (p_{01}, \boldsymbol{w}_0) \in \mu^c$$

and

$$(p_1, \boldsymbol{w}), \sigma_1 \to (p_{10}, \boldsymbol{w}_1), (p_{11}, \boldsymbol{w}_1) \in \mu^c,$$

$\boldsymbol{w} = w_1 \cdots w_r$, $\boldsymbol{w}_0 = w_{0,1} \cdots w_{0,r}$ and $\boldsymbol{w}_1 = w_{1,1} \cdots w_{1,r}$. That is,

- $(p_0, k_0) \to (p_{00}, p_{01}) \in \mu$ and $w_{0,k_0} = \sigma_0$;
- $(p_1, k_1) \to (p_{10}, p_{11}) \in \mu$ and $w_{1,k_1} = \sigma_1$.

Then, by the definition of $\widetilde{\mu}_4$,

$$((p_{00}, p_{01}), k_0), ((p_{10}, p_{11}), k_1) \to (p_0, p_1) \in \widetilde{\mu}_4$$

and

$$(((p_{00}, p_{01}), k_0), ((p_{10}, p_{11}), k_1), t, f) \in \widetilde{\tau}_4,$$

where (t, f) is an inverse structure of (p_0, p_1); see the definition of $\widetilde{\mu}$ and $\widetilde{\tau}$.

Let $\boldsymbol{v}_0 =_{\widetilde{\rho}(p_{00},p_{01})} \boldsymbol{w}_0$, $\boldsymbol{v}_1 =_{\widetilde{\rho}(p_{10},p_{11})} \boldsymbol{w}_1$, and let an assignment $\boldsymbol{v}$ be such that $(\boldsymbol{w}_0, \boldsymbol{w}_1, \boldsymbol{v})$ is an instance of (t, f). Then

$$(((p_{00}, p_{01}), \boldsymbol{v}_0), \sigma_0), (((p_{10}, p_{11}), \boldsymbol{v}_1), \sigma_1) \to ((p_0, p_1), \boldsymbol{v}) \in \widetilde{\mu}^c.$$

Since $\boldsymbol{w}_0 =_{\rho(p_0)} \boldsymbol{w}$ and $\boldsymbol{w}_1 =_{\rho(p_1)} \boldsymbol{w}$, by the first part of Lemma 2, $\boldsymbol{v} =_{\widetilde{\rho}(p_0,p_1)} \boldsymbol{w}$.[14]

[14] Recall that $\widetilde{\rho}(p_0, p_1) = \rho(p_0) \cap \rho(p_1)$.

For the proof of the converse part of the lemma, let

$$(((p_{00},p_{01}),\boldsymbol{v}_0),\sigma_0),(((p_{10},p_{11}),\boldsymbol{v}_1),\sigma_1) \to ((p_0,p_1),\boldsymbol{v}) \in \widetilde{\mu}^c.$$

That is, there exist

- $\boldsymbol{v}'_0 \in \Sigma^{r\neq}$, $\boldsymbol{v}'_0 = v'_{0,1}v'_{0,2}\cdots v'_{0,r}$, such that $\boldsymbol{v}'_0 =_{\widetilde{\rho}(p_{00},p_{01})} \boldsymbol{v}_0$;
- $\boldsymbol{v}'_1 \in \Sigma^{r\neq}$, $\boldsymbol{v}'_1 = v'_{1,1}v'_{1,2}\cdots v'_{1,r}$, such that $\boldsymbol{v}'_1 =_{\widetilde{\rho}(p_{10},p_{11})} \boldsymbol{v}_1$;
- $((p_{00},p_{01}),k_0),((p_{10},p_{11}),k_1) \to (p_0,p_1) \in \widetilde{\mu}$, where $v'_{0,k_0} = \sigma_0$ and $v'_{1,k_1} = \sigma_1$; and
- $((p_{00},p_{01}),k_0,(p_{10},p_{11}),k_1,t,f) \in \widetilde{\tau}$, where $(\boldsymbol{v}'_0,\boldsymbol{v}'_1,\boldsymbol{v})$ is an instance of (t,f).

By the definition of $\widetilde{\mu}$, both transitions $(p_0,k_0) \to (p_{00},p_{01})$ and $(p_1,k_1) \to (p_{10},p_{11})$ are in μ; and (t,f) is an inverse structure of (p_0,p_1).

Let $\boldsymbol{w}$ be an assignment such that $\boldsymbol{w} =_{\widetilde{\rho}(p_0,p_1)} \boldsymbol{v}$. Since $\boldsymbol{w} =_{\widetilde{\rho}(p_0,p_1)} \boldsymbol{v}$ and $(\boldsymbol{v}'_0,\boldsymbol{v}'_1,\boldsymbol{v})$ is an instance of (t,f), by the second part of Lemma 2, $\boldsymbol{w} =_{\rho(p_0)} \boldsymbol{v}'_0$ and $\boldsymbol{w} =_{\rho(p_1)} \boldsymbol{v}'_1$. Therefore, we can put $\boldsymbol{w}_0 = \boldsymbol{v}'_0$ and $\boldsymbol{w}_1 = \boldsymbol{v}'_1$, implying

$$(p_0,\boldsymbol{w}),\sigma_0 \to (p_{00},\boldsymbol{w}_0),(p_{01},\boldsymbol{w}_0) \in \mu^c$$

and

$$(p_1,\boldsymbol{w}),\sigma_1 \to (p_{10},\boldsymbol{w}_1),(p_{11},\boldsymbol{w}_1) \in \mu^c.$$

Since $\boldsymbol{v}'_0 =_{\widetilde{\rho}(p_{00},p_{01})} \boldsymbol{v}_0$ and $\boldsymbol{v}'_1 =_{\widetilde{\rho}(p_{10},p_{11})} \boldsymbol{v}_1$, the converse part of the lemma follows. □

B Proof of the "if" Part of Theorem 1

For an r-register $\uparrow$-NR-FMA $A = \langle S, s_0, \boldsymbol{u}, \rho, \tau, \mu, F\rangle$ we construct a $2r$-register $\downarrow$-NR-FMA $\widetilde{A} = \langle \widetilde{S}, \widetilde{s}_0, \widetilde{\boldsymbol{u}}, \widetilde{\rho}, \widetilde{\mu}, \widetilde{F}\rangle$ such that $L(A) = L(\widetilde{A})$.

Like in the previous proof, we assume that

- $\boldsymbol{u} = \#^{r-m}\theta_1\cdots\theta_m$, where $\theta_1,\ldots,\theta_m \in \Sigma$, and
- only the first $r-m$ registers of A can be reassigned, i.e, the range of ρ is a subset of $\{1,2,\ldots,r-m\}$.

We precede the formal description of $\widetilde{A}$ with a general intuitive explanation. One would expect the construction to resemble the reversing the classical automata, i.e., a transition $(p_0,k_0),(p_1,k_1) \to p$ of A to become the "transition" $(p,k_0,k_1) \to (p_0,p_1)$ of $\widetilde{A}$. This is indeed almost so, but with the following modification. Dually to the construction in Appendix A, reversing of a bottom-up transition $(p_0,k_0),(p_1,k_1) \to p$ results in two top-down transitions $((p,0),k_0) \to (p_0,0),(p_0,1)$ and $((p,1),k_1) \to (p_1,0),(p_1,1)$ at the lower level. The state components 0 and 1 indicate the child nodes of the parent node, where these transitions are applied: 0 indicates the left child and 1 indicates the right one.

One half of the $2r$ registers of $\widetilde{A}$, the *main* registers, is intended to contain the corresponding assignment of A at the parent node, while the other half is intended to "recover" the symbols forgotten in the merging. To identify the (r out of $2r$) main registers, the states of $\widetilde{A}$ are equipped with a pointer function

$$\pi : \{1, 2, \ldots, r\} \to \{1, 2, \ldots, 2r\},$$

where the value $\pi(i)$ is the index of the register of $\widetilde{A}$ containing the symbol stored in ith register of A. These pointers are also used to mimic the merging relation. That is, the pointers at the child nodes and the pointers at the parent node are defined in such a way that mimics the type and the valid selector used to merge the assignments at the child nodes.

More precisely,

$$\widetilde{S} = S \times \{0, 1\} \times \mathit{\Pi}_r^2 \times \mathcal{T}_r \times \mathcal{F}_r \cup \{\widetilde{s}_0\},$$

where $\widetilde{s}_0$ is a *new* state (the initial state of $\widetilde{A}$), $\mathit{\Pi}_r$ is the set of all injective functions from $\{1, \ldots, r\}$ into $\{1, \ldots, 2r\}$, and $\mathcal{F}_r$ is the set of functions from $\{1, \ldots, r\}$ into $\{0, 1\} \times \{1, \ldots, r\}$. A bottom-up transition $(p_0, k_0), (p_1, k_1) \to p$ and a corresponding "merging attribute" $((p_0, k_0), (p_1, k_1), t, f)$ are simulated by two top-down transitions

$$((p, 0, \pi, \pi_0, t, f), \pi_0(k_0)) \to (p_0, 0, \pi_0, \pi_0', t_0, f_0), (p_0, 1, \pi_0, \pi_0'', t_0, f_0)$$

and

$$((p, 1, \pi, \pi_1, t, f), \pi_1(k_1)) \to (p_1, 0, \pi_1, \pi_1', t_1, f_1), (p_1, 1, \pi_1, \pi_1'', t_1, f_1),$$

where

1. $\pi_0(i) = \pi_1(j)$ implies $(i, j) \in t$, and
2. $\pi(j) = \begin{cases} \pi_0(i) \text{ if } f(j) = (0, i) \\ \pi_1(i) \text{ if } f(j) = (1, i) \end{cases}$

A triple of pointers (π_0, π_1, π) satisfying the above conditions 1 and 2 is said to *comply with* the pair (t, f).

The pointers π_0 and π_1 in the states $(p, 0, \pi, \pi_0, t, f)$ and $(p, 1, \pi, \pi_1, t, f)$, respectively, point at the assignments of A at the corresponding child nodes, and the pointer π points at the assignment A at the parent node.

The registers whose indices lie outside of the ranges of π_0 and π_1 are intended to contain the forgotten symbols which are recovered by a non-deterministic reassignment. That is, we define

$$\widetilde{\rho}(p, 0, \pi, \pi_0, t, f) = \{\pi_0(i) : f(j) = (0, i) \text{ and } j \in \rho(s)\} \cup \Big(\{1, \ldots, 2r\} \setminus \mathrm{Range}(\pi_0)\Big)$$

and

$$\widetilde{\rho}(p,1,\pi,\pi_1,t,f) = \\ \{\pi_1(i) : f(j) = (1,i) \text{ and } j \in \rho(s)\} \cup \Big(\{1,\dots,2r\} \setminus \mathrm{Range}(\pi_1)\Big).$$

The above description of $\widetilde{A}$ is illustrated in Fig. 11 and 12 below. In Fig. 11 $\boldsymbol{w}$, $\boldsymbol{w}_0$, and $\boldsymbol{w}_1$ are the assignments at the nodes labeled with the states p, p_0, and p_1, respectively,[15] in a run of A. Fig. 12 shows the reversing of the transition in Fig. 11. The intended meaning of the states in Fig. 12 (that depicts the corresponding run of $\widetilde{A}$) is that $\pi_0(\boldsymbol{v}_0) =_{\rho(p_0)} \boldsymbol{w}_0$, $\pi_1(\boldsymbol{v}_1) =_{\rho(p_1)} \boldsymbol{w}_1$, $\pi(\boldsymbol{v}) =_{\rho(p)} \boldsymbol{w}$,[16] and t and f are the type and the valid selector applied in the merging transition of A that results in $\boldsymbol{w}$.

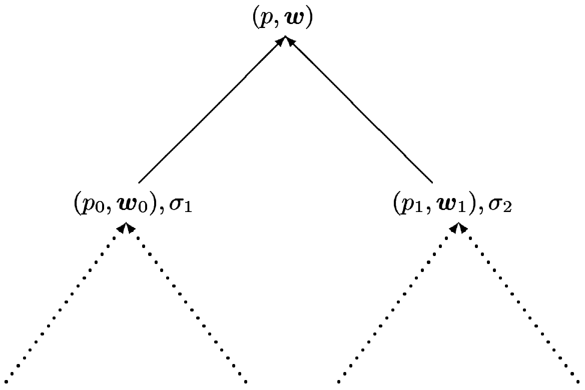

$(p, \boldsymbol{w})$
$(p_0, \boldsymbol{w}_0), \sigma_1$
$(p_1, \boldsymbol{w}_1), \sigma_2$

Fig. 11. An application of a bottom-up transition of A

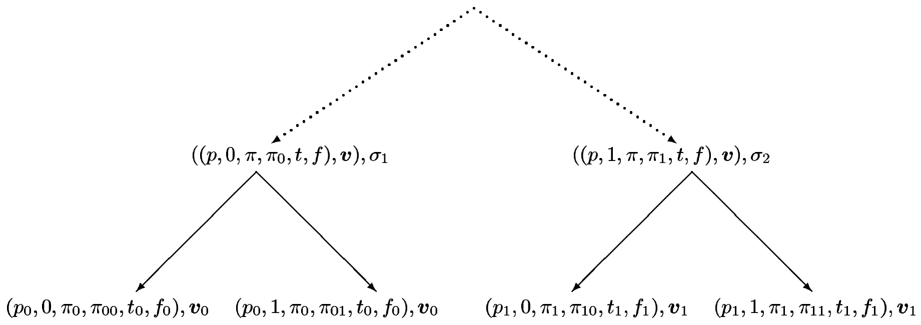

$((p,0,\pi,\pi_0,t,f),\boldsymbol{v}),\sigma_1$
$((p,1,\pi,\pi_1,t,f),\boldsymbol{v}),\sigma_2$
$(p_0,0,\pi_0,\pi_{00},t_0,f_0),\boldsymbol{v}_0$
$(p_0,1,\pi_0,\pi_{01},t_0,f_0),\boldsymbol{v}_0$
$(p_1,0,\pi_1,\pi_{10},t_1,f_1),\boldsymbol{v}_1$
$(p_1,1,\pi_1,\pi_{11},t_1,f_1),\boldsymbol{v}_1$

Fig. 12. Reversing and "splitting" the transition of A in Fig. 11

We proceed with the formal description of $\widetilde{A} = \langle \widetilde{S}, \widetilde{s}_0, \widetilde{\boldsymbol{u}}, \widetilde{\rho}, \widetilde{\mu}, \widetilde{F} \rangle$ (that has $2r$ registers). Even though some of the components of $\widetilde{A}$ have been define earlier, we list them one more time fore the sake of continuity.

[15] In particular, $\boldsymbol{w}$ results in merging $\boldsymbol{w}_0$ and $\boldsymbol{w}_1$ after reassignment.

[16] $\pi(w_1 \cdots w_r)$ denotes $w_{\pi(1)} \cdots w_{\pi(r)}$, etc..

- $\widetilde{S} = S \times \{0,1\} \times \Pi_r^2 \times \mathcal{T}_r \times \mathcal{F}_r \cup \{\widetilde{s}_0\}$, where $\widetilde{s}_0$ is a new state.
- $\widetilde{s}_0$ is the initial state.
- $\widetilde{\boldsymbol{u}} = \#^{r-m}\theta_1 \cdots \theta_m \#^r$.
- The reassignment $\widetilde{\rho}$ is defined as follows.

$$\widetilde{\rho}(\widetilde{s}_0) = \{1, \ldots, r-m\} \cup \{r+1, \ldots, 2r\},$$

  and for each $p \in S$,
  $\widetilde{\rho}(p, 0, \pi, \pi_0, t, f) =$

$$\{\pi_0(i) : f(j) = (0, i) \text{ and } j \in \rho(p)\} \cup \Big(\{1, \ldots, 2r\} \setminus \mathrm{Range}(\pi_0)\Big)$$

  and
  $\widetilde{\rho}(p, 1, \pi, \pi_1, t, f) =$

$$\{\pi_1(i) : f(j) = (1, i) \text{ and } j \in \rho(p)\} \cup \Big(\{1, \ldots, 2r\} \setminus \mathrm{Range}(\pi_1)\Big).$$

- The transition relation $\widetilde{\mu}$ consists of the following transitions.
  - For each $(p, k) \in F$ and all π_0, π_1 such that (π_0, π_1, π_{id})[17] complies with (t, f) it contains

$$(\widetilde{s}_0, k) \to (p, 0, \pi_{id}, \pi_0, t, f), (p, 1, \pi_{id}, \pi_1, t, f);$$

    and
  - for each $(p_0, k_0), (p_1, k_1) \to p \in \mu$, each $((p_0, k_0), (p_1, k_1), t, f) \in \tau$, and all $\pi, \pi_0, \pi_1, \pi_{00}, \pi_{01}, \pi_{10}, \pi_{11} \in \Pi_r$, $t_0, t_1 \in \mathcal{T}_r$, and $f_0, f_1 \in \mathcal{F}_r$ such that (π_0, π_1, π) complies with (t, f), $(\pi_{00}, \pi_{01}, \pi_0)$ complies with (t_0, f_0), and $(\pi_{10}, \pi_{11}, \pi_1)$ complies with (t_1, f_1), it contains both

$$(p, 0, \pi, \pi_0, t, f), \pi_0(k_0) \to (p_0, 0, \pi_0, \pi_{00}, t_0, f_0), (p_1, 1, \pi_0, \pi_{01}, t_0, f_0),$$

    and

$$(p, 1, \pi, \pi_1, t, f), \pi_1(k_1) \to (p_1, 0, \pi_1, \pi_{10}, t_1, f_1), (p_1, 1, \pi_1, \pi_{11}, t_1, f_1).$$

- Finally, $\widetilde{F}$ is defined as follows. For each $(s_0, k_0), (s_0, k_1) \to p \in \mu$ (that starts from the initial state s_0) each $((s_0, k_0), (s_0, k_1), t, f) \in \tau$, and all $\pi, \pi_0, \pi_1 \in \Pi_r$ such that (π_0, π_1, π) complies with (t, f), it contains

$$((p, 0, \pi, \pi_0, t, f), \pi_0(k_0))$$

  and

$$((p, 1, \pi, \pi_1, t, f), \pi_1(k_1)).$$

The proof of the equality $L(\widetilde{A}) = L(A)$ is based on Lemma 4 below that is a formalization of Fig. 11 and 12. It shows how an accepting run of A can be "reversed" into an accepting run of $\widetilde{A}$, or more precisely, it shows how transitions from μ^c are converted into transitions from $\widetilde{\mu}^c$, and vice versa.

[17] Here π_{id} denotes the identity function on $\{1, 2, \ldots, r\}$. That is, $\pi_{id}(i) = i$, for all $i = 1, \ldots, r$.

Lemma 4 consists of four parts corresponding to the type of nodes on which transitions take place. Its part (*i*) deals with the case of the root node ϵ. Part (*ii*) of the lemma settles the case of the interior nodes. Part (*iii*) shows how an accepting run of A can be "reversed" into an accepting run of $\widetilde{A}$ when one of the two sibling nodes is a leaf and the other is an interior node. Finally, part (*iv*) of Lemma 3 shows how an accepting run of A can be "reversed" into an accepting run of $\widetilde{A}$ at the leaf nodes and vice versa.

Lemma 4

(*i*) (See Fig. 13 and 14.) *For all pointers* π_0, π_1*, all types* t*, all valid selectors* f *for* t *such that* (π_0, π_1, π_{id}) *comply with* (t, f)*, and all final relations* $((p, \boldsymbol{w}), \sigma) \in F^c$*, there is an assignment* $\boldsymbol{v}$ *such that* $\pi_{id}(\boldsymbol{v}) =_{\rho(p)} \boldsymbol{w}$ *and*

$$((\widetilde{s}_0, \widetilde{\boldsymbol{u}}), \sigma) \to ((p, 0, \pi_{id}, \pi_0, t, f), \boldsymbol{v}), ((p, 1, \pi_{id}, \pi_1, t, f), \boldsymbol{v}) \in \widetilde{\mu}^c.$$

Conversely, for all

$$((\widetilde{s}_0, \widetilde{\boldsymbol{u}}), \sigma) \to ((p, 0, \pi_{id}, \pi_0, t, f), \boldsymbol{v}), ((p, 1, \pi_{id}, \pi_1, t, f), \boldsymbol{v}) \in \widetilde{\mu}^c$$

and all assignments $\boldsymbol{w}$ *such that* $\boldsymbol{w} =_{\rho(p)} \pi_{id}(\boldsymbol{v})$*,* $((p, \boldsymbol{w}), \sigma) \in F^c$*.*

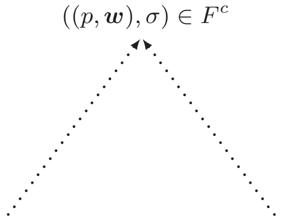


Fig. 13. An application of a final relation of A at the root node

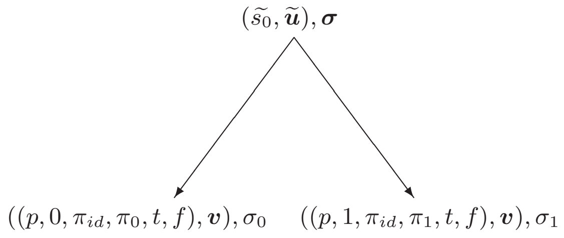

$(\widetilde{s}_0, \widetilde{\boldsymbol{u}}), \boldsymbol{\sigma}$
$((p, 0, \pi_{id}, \pi_0, t, f), \boldsymbol{v}), \sigma_0$
$((p, 1, \pi_{id}, \pi_1, t, f), \boldsymbol{v}), \sigma_1$

Fig. 14. Reversing the final relation of A at the root node in Fig. 13

(*ii*) (See Fig. 11 and 12.) *For all*

$$(p_0, \boldsymbol{w}_0), \sigma_0, (p_1, \boldsymbol{w}_1), \sigma_1 \to (p, \boldsymbol{w}) \in \mu^c,$$

all types t *and all valid selectors* f *for* t *which yield* $\boldsymbol{w}$ *in the above transition, all assignments* $\boldsymbol{v}$ *such that* $\pi(\boldsymbol{v}) =_{\rho(p)} \boldsymbol{w}$*, all pointers* $\pi, \pi_0, \pi_1,$ $\pi_{00}, \pi_{01}, \pi_{10}, \pi_{11}$*, and all types* t_0 *and* t_1 *and valid selectors* f_0 *and* f_1 *for* t_0 *and* t_1*, respectively, such that* (π_0, π_1, π) *comply with* (t, f)*,* $(\pi_{00}, \pi_{01}, \pi_0)$ *comply with* (t_0, f_0)*, and* $(\pi_{10}, \pi_{11}, \pi_1)$ *comply with* (t_1, f_1)*, there are assignments* $\boldsymbol{v}_0$ *and* $\boldsymbol{v}_1$ *such that* $\pi_0(\boldsymbol{v}_0) =_{\rho(p_0)} \boldsymbol{w}_0$*,* $\pi_1(\boldsymbol{v}_1) =_{\rho(p_1)} \boldsymbol{w}_1$*, and both transitions*

$(((p,0,\pi,\pi_0,t,f),\boldsymbol{v}),\sigma_0) \to$
$$((p_0,0,\pi_0,\pi_{00},t_0,f_0),\boldsymbol{v}_0,(p_0,1,\pi_0,\pi_{01},t_0,f_0),\boldsymbol{v}_0)$$

and

$(((p,1,\pi,\pi_1,t,f),\boldsymbol{v}),\sigma_1) \to$
$$((p_1,0,\pi_1,\pi_{10},t_1,f_1),\boldsymbol{v}_1,(p_1,1,\pi_1,\pi_{11},t_1,f_1),\boldsymbol{v}_1)$$

are in $\widetilde{\mu}^c$.

Conversely, for all transitions
$(((p,0,\pi,\pi_0,t,f),\boldsymbol{v}),\sigma_0) \to$
$$((p_0,0,\pi_0,\pi_{00},t_0,f_0),\boldsymbol{v}_0,(p_0,1,\pi_0,\pi_{01},t_0,f_0),\boldsymbol{v}_0)$$

and

$(((p,1,\pi,\pi_1,t,f),\boldsymbol{v}),\sigma_1) \to$
$$((p_1,0,\pi_1,\pi_{10},t_1,f_1),\boldsymbol{v}_1,(p_1,1,\pi_1,\pi_{11},t_1,f_1),\boldsymbol{v}_1)$$

in $\widetilde{\mu}^c$ *and all assignments* $\boldsymbol{w}_0$ *and* $\boldsymbol{w}_1$ *such that* $\pi_0(\boldsymbol{v}_0) =_{\rho(p_0)} \boldsymbol{w}_0$ *and* $\pi_1(\boldsymbol{v}_1) =_{\rho(p_1)} \boldsymbol{w}_1$, *there is an assignment* $\boldsymbol{w}$ *such that* $\pi(\boldsymbol{v}) =_{\rho(p)} \boldsymbol{w}$ *and*

$$(p_0,\boldsymbol{w}_0),\sigma_0,(p_1,\boldsymbol{w}_1),\sigma_1 \to (p,\boldsymbol{w}) \in \mu^c,$$

(iii) *(a)* (See Fig. 15 and 16.) *For all transitions*

$$((p_0,\boldsymbol{w}_0),\sigma_0),((s_0,\boldsymbol{u}),\sigma_1) \to (p,\boldsymbol{w}) \in \mu^c,$$

all types t and all valid selectors f for t which yield $\boldsymbol{w}$ in the above transition, all assignments $\boldsymbol{v}$ such that $\pi(\boldsymbol{v}) =_{\rho(p)} \boldsymbol{w}$, all pointers π, π_0, π_1, π_{00}, π_{01}, and all types t_0 and valid selectors f_0 for t_0 such that (π_0,π_1,π) comply with (t,f) and $(\pi_{00},\pi_{01},\pi_0)$ comply with (t_0,f_0), there is an assignment $\boldsymbol{v}_0$ such that $\pi_0(\boldsymbol{v}_0) =_{\rho(p_0)} \boldsymbol{w}_0$,

$$(((p,1,\pi,\pi_1,t,f),\boldsymbol{v}),\sigma_1) \in \widetilde{F}^c,$$

and
$(((p,0,\pi,\pi_0,t,f),\boldsymbol{v}),\sigma_0) \to$
$$((p_0,0,\pi_0,\pi_{00},t_0,f_0),\boldsymbol{v}_0,(p_0,1,\pi_0,\pi_{01},t_0,f_0),\boldsymbol{v}_0) \in \widetilde{\mu}^c\ .$$

Conversely, for all

$$(((p,1,\pi,\pi_1,t,f),\boldsymbol{v}),\sigma_1) \in \widetilde{F}^c,$$

and

$((p,0,\pi,\pi_0,t,f),\boldsymbol{v}),\sigma_0 \to$
$$((p_0,0,\pi_0,\pi_{00},t_0,f_0),\boldsymbol{v}_0,(p_0,1,\pi_0,\pi_{01},t_0,f_0),\boldsymbol{v}_0 \in \widetilde{\mu}^c,$$

and all assignment $\boldsymbol{w}_0$ *such that* $\boldsymbol{w}_0 =_{\rho(p_0)} \pi_0(\boldsymbol{v}_0)$*, there is an assignment* $\boldsymbol{w}$ *such that* $\boldsymbol{w} =_{\rho(p)} \pi(\boldsymbol{v})$ *and*

$$((p_0,\boldsymbol{w}_0),\sigma_0),((s_0,\boldsymbol{u}),\sigma_1) \to (p,\boldsymbol{w}) \in \mu^c.$$

(*b*) *For all transitions*

$$((s_0,\boldsymbol{u}),\sigma_0),((p_1,\boldsymbol{w}_1),\sigma_1) \to (p,\boldsymbol{w}) \in \mu^c,$$

all types t *and all valid selectors* f *for* t *which yield* $\boldsymbol{w}$ *in the above transition, all assignments* $\boldsymbol{v}$ *such that* $\pi(\boldsymbol{v}) =_{\rho(p)} \boldsymbol{w}$ *and for all pointers* π*,* π_0*,* π_1*,* π_{10}*,* π_{11}*, ad all types* t_1 *and valid selectors* f_1 *for* t_1 *such that* (π_0,π_1,π) *comply with* (t,f) *and* $(\pi_{10},\pi_{11},\pi_1)$ *comply with* (t_1,f_1)*, there is an assignment* $\boldsymbol{v}_1$ *such that* $\pi_1(\boldsymbol{v}_1) =_{\rho(p_1)} \boldsymbol{w}_1$*,*

$$(((p,0,\pi,\pi_0,t,f),\boldsymbol{v}),\sigma_0) \in \widetilde{F}^c$$

and

$$((p,1,\pi,\pi_1,t,f),\boldsymbol{v}),\sigma_1 \to (p_1,0,\pi_1,\pi_{10},t_1,f_1),\boldsymbol{v}_1,(p_1,1,\pi_1,\pi_{11},t_1,f_1),\boldsymbol{v}_1 \in \widetilde{\mu}^c.$$

Conversely, for all

$$((p,1,\pi,\pi_1,t,f),\boldsymbol{v}),\sigma_1 \to (p_1,0,\pi_1,\pi_{10},t_1,f_1),\boldsymbol{v}_1,(p_1,1,\pi_1,\pi_{11},t_1,f_1),\boldsymbol{v}_1 \in \widetilde{\mu}^c\ ,$$

and

$$(((p,0,\pi,\pi_0,t,f),\boldsymbol{v}),\sigma_0) \in \widetilde{F}^c,$$

and all assignments $\boldsymbol{w}_1$ *such that* $\boldsymbol{w}_1 =_{\rho(p_1)} \pi_1(\boldsymbol{v}_1)$*, there is an assignment* $\boldsymbol{w}$ *such that* $\boldsymbol{w} =_{\rho(p)} \pi(\boldsymbol{v})$ *and*

$$((s_0,\boldsymbol{u}),\sigma_0),((p_1,\boldsymbol{w}_1),\sigma_1) \to (p,\boldsymbol{w}) \in \mu^c.$$

(*iv*) (See Fig. 17 and 18.) *For all*

$$((s_0,\boldsymbol{u}),\sigma_0),((s_0,\boldsymbol{u}),\sigma_1) \to (p,\boldsymbol{w}) \in \mu^c,$$

all types t *and all valid selectors* f *for* t *which yield* $\boldsymbol{w}$ *in the above transition, all assignments* $\boldsymbol{v}$ *such that* $\pi(\boldsymbol{v}) =_{\rho(p)} \boldsymbol{w}$*, and for all pointers* π,π_0,π_1 *such that* (π_0,π_1,π) *comply with* (t,f)*,*

$$(((p,0,\pi,\pi_0,t,f),\boldsymbol{v}),\sigma_0) \in \widetilde{F}^c$$

and

$$(((p,1,\pi,\pi_1,t,f),\boldsymbol{v}),\sigma_1) \in \widetilde{F}^c.$$

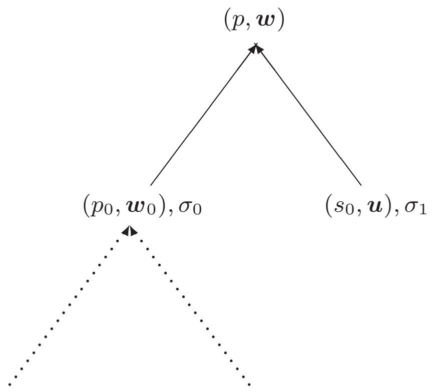

$(p, \boldsymbol{w})$
$(p_0, \boldsymbol{w}_0), \sigma_0$
$(s_0, \boldsymbol{u}), \sigma_1$

Fig. 15. An application of a transition of A at a leaf node

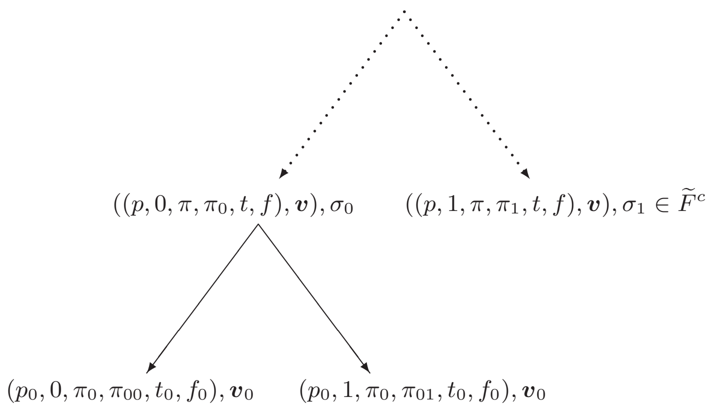

$((p, 0, \pi, \pi_0, t, f), \boldsymbol{v}), \sigma_0$
$((p, 1, \pi, \pi_1, t, f), \boldsymbol{v}), \sigma_1 \in \widetilde{F}^c$
$(p_0, 0, \pi_0, \pi_{00}, t_0, f_0), \boldsymbol{v}_0$
$(p_0, 1, \pi_0, \pi_{01}, t_0, f_0), \boldsymbol{v}_0$

Fig. 16. Reversing and "splitting" the transition of A in Fig. 15 into a transition and a final relation

Conversely, for all pairs of final relations

$$(((p, 0, \pi, \pi_0, t, f), \boldsymbol{v}), \sigma_0), (((p, 1, \pi, \pi_1, t, f), \boldsymbol{v}), \sigma_1) \in \widetilde{F}^c,$$

there is an assignment $\boldsymbol{w}$ *such that* $\boldsymbol{w} =_{\rho(p)} \pi(\boldsymbol{v})$ *and*

$$((s_0, \boldsymbol{u}), \sigma_0), ((s_0, \boldsymbol{u}), \sigma_1) \to (p, \boldsymbol{w}) \in \mu^c.$$

We postpone the proof of the lemmas to the end of this appendix and prove the "if" part of Theorem 1 first.

Proof. (of the "if" part of Theorem 1.) We prove that $L(A) = L(\widetilde{A})$ by showing how to convert an accepting run of A on a Σ-tree $\boldsymbol{\sigma} : T \to \Sigma$, into an accepting run of $\widetilde{A}$ on $\boldsymbol{\sigma}$, and vice versa.

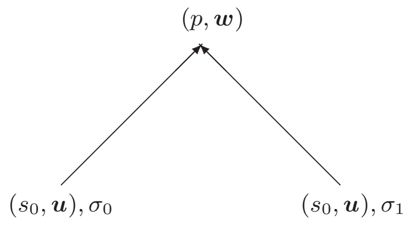

$(p, \boldsymbol{w})$
$(s_0, \boldsymbol{u}), \sigma_0$
$(s_0, \boldsymbol{u}), \sigma_1$

Fig. 17. An application of a transition of A at two leaf nodes

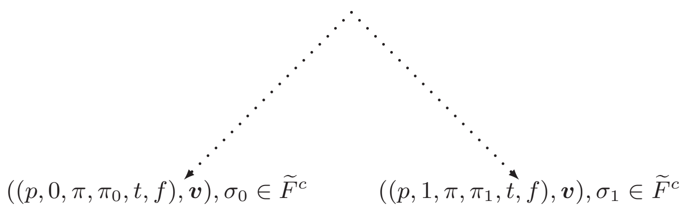

$((p, 0, \pi, \pi_0, t, f), \boldsymbol{v}), \sigma_0 \in \widetilde{F}^c$
$((p, 1, \pi, \pi_1, t, f), \boldsymbol{v}), \sigma_1 \in \widetilde{F}^c$

Fig. 18. Reversing and "splitting" the transition of A in Fig. 17 into two final relations of $\widetilde{A}$

For an accepting run $R : T \to S^c$, $R(n) = (p_n, \boldsymbol{w}_n)$, $n \in T$, of A on $\boldsymbol{\sigma}$ we construct an accepting run $\widetilde{R} : T \to S^c$ of $\widetilde{A}$ on $\boldsymbol{\sigma}$ top-down, i.e., from the root to the leaves, by induction for which we shall employ the following notation. For an interior node $n \in T$,

– t_n and f_n are the type and the valid selector used in the transition

$$(R(n0), \boldsymbol{\sigma}(n0)), (R(n1), \boldsymbol{\sigma}(n1)) \to R(n);$$

– π_{n0}, π_{n1}, and π_n are pointers such that $(\pi_{n0}, \pi_{n1}, \pi_n)$ complies with (t_n, f_n); and
– $\pi_\epsilon = \pi_{id}$.

By definition, for the root node ϵ, $\widetilde{R}(\epsilon) = \widetilde{s_0}^{\,c} = (\widetilde{s_0}, \widetilde{\boldsymbol{u}})$, and assume that $\widetilde{R}(n)$ has been constructed for a non-leaf node $n \in T$. Then

– $\widetilde{R}(n0) = ((p_n, 0, \pi_n, \pi_{n0}, t_n, f_n), \boldsymbol{v}_n)$ and
– $\widetilde{R}(n1) = ((p_n, 1, \pi_n, \pi_{n1}, t_n, f_n), \boldsymbol{v}_n)$,

where $\pi_{id}(\boldsymbol{v}_n) =_{\rho(p_n)} \boldsymbol{w}_n$ is defined below..

– The assignment $\boldsymbol{v}_\epsilon$ is provided by part (i) of Lemma 4, and
– for non-root interior node n, the assignment $\boldsymbol{v}_n$ is provided by part (ii) of Lemma 4.

By parts (iii) and (iv) of Lemma 4, $\widetilde{R}$ is an accepting run of $\widetilde{A}$ on $\boldsymbol{\sigma}$.

The converse direction can shown in a similar manner. That is, an accepting run R of A on $\boldsymbol{\sigma}$ is constructed from an accepting run $\widetilde{R}$ of $\widetilde{A}$ by applying the converse direction of Lemma 4. $\square$

It remains to prove Lemma 4.

Proof. (of Lemma 4.) We will prove only part (ii) of the lemma. The proofs of the other parts are very similar.

Let

$$(p_0, \boldsymbol{w}_0), \sigma_0, (p_1, \boldsymbol{w}_1), \sigma_1 \rightarrow (p, \boldsymbol{w}) \in \mu^c.$$

That is, there exist

- assignments $\boldsymbol{w}'_0 = w'_{0,1}w'_{0,2}\cdots w'_{0,r}$ and $\boldsymbol{w}'_1 = w'_{1,1}w'_{1,2}\cdots w'_{1,r}$ such that $\boldsymbol{w}'_0 =_{\rho(p_0)} \boldsymbol{w}_0$ and $\boldsymbol{w}'_1 =_{\rho(p_1)} \boldsymbol{w}_1$;
- a transition $(p_0, k_0), (p_1, k_1) \rightarrow p \in \mu$ such that $w'_{0,k_0} = \sigma_0$ and $w'_{1,k_1} = \sigma_1$; and
- a merging relation $((p_0, k_0), (p_1, k_1), t, f) \in \tau$ such that $(\boldsymbol{w}'_0, \boldsymbol{w}'_1, \boldsymbol{w})$ is an instance of (t, f).

Let $\pi_0, \pi_1, \pi, \pi_{00}, \pi_{01}, \pi_0, \pi_{10}, \pi_{11}, \pi_1$ be pointers and t_0, t_1 and f_0, f_1 be types and the corresponding valid selectors such that (π_0, π_1, π) complies with (t, f), $(\pi_{00}, \pi_{01}, \pi_0)$ complies with (t_0, f_0), and $(\pi_{10}, \pi_{11}, \pi_1)$ complies with (t_1, f_1).

By definition, $\widetilde{\mu}$ contains both

$$(p, 0, \pi, \pi_0, t, f), \pi_0(k_0) \rightarrow (p, 0, \pi_0, \pi_{00}, t_0, f_0), (p, 1, \pi_0, \pi_{01}, t_0, f_0)$$

and

$$(p, 0, \pi, \pi_1, t, f), \pi_1(k_1) \rightarrow (p, 0, \pi_1, \pi_{10}, t_1, f_1), (p, 1, \pi_1, \pi_{11}, t_1, f_1)$$

Let $\boldsymbol{v}$ be an assignment such that $\pi(\boldsymbol{v}) =_{\rho(p)} \boldsymbol{w}$. By the definition of the reassignments $\widetilde{\rho}((p, 0, \pi, \pi_0, t, f))$ and $\widetilde{\rho}((p, 1, \pi, \pi_0, t, f))$, there are assignments

- $\boldsymbol{v}_0 =_{\widetilde{\rho}((p,0,\pi,\pi_0,t,f))} \boldsymbol{v}$ such that $\pi_0(\boldsymbol{v}_0) = \boldsymbol{w}'_0$, and
- $\boldsymbol{v}_1 =_{\widetilde{\rho}((p,1,\pi,\pi_0,t,f))} \boldsymbol{v}$ such that $\pi_1(\boldsymbol{v}_1) = \boldsymbol{w}'_1$.

Since (π_0, π_1, π) comply with (t, f), $\pi(\boldsymbol{v}_0) = \boldsymbol{w} = \pi(\boldsymbol{v}_1)$. Thus, both

$$((p, 0, \pi, \pi_0, t, f), \boldsymbol{v}), \sigma_0 \rightarrow (p_0, 0, \pi_0, \pi_{00}, t_0, f_0), \boldsymbol{v}_0, (p_0, 1, \pi_0, \pi_{01}, t_0, f_0), \boldsymbol{v}_0$$

and

$$((p, 1, \pi, \pi_1, t, f), \boldsymbol{v}), \sigma_1 \rightarrow (p_1, 0, \pi_1, \pi_{10}, t_1, f_1), \boldsymbol{v}_1, (p_1, 1, \pi_1, \pi_{11}, t_1, f_1), \boldsymbol{v}_1$$

are in $\widetilde{\mu}^c$.

For the proof of converse part of the lemma, let

$$((p, 0, \pi, \pi_0, t, f), \boldsymbol{v}), \sigma_0 \rightarrow (p_0, 0, \pi_0, \pi_{00}, t_0, f_0), \boldsymbol{v}_0, (p_0, 1, \pi_0, \pi_{01}, t_0, f_0), \boldsymbol{v}_0 \in \widetilde{\mu}^c$$

and

$$((p, 1, \pi, \pi_1, t, f), \boldsymbol{v}), \sigma_1 \rightarrow (p_1, 0, \pi_1, \pi_{10}, t_1, f_1), \boldsymbol{v}_1, (p_1, 1, \pi_1, \pi_{11}, t_1, f_1), \boldsymbol{v}_1 \in \widetilde{\mu}^c$$

where $\boldsymbol{v}_0 =_{\widetilde{\rho}((p,0,\pi,\pi_0,t,f))} \boldsymbol{v}$, $\boldsymbol{v}_1 =_{\widetilde{\rho}((p,1,\pi,\pi_1,t,f))} \boldsymbol{v}$, and (π_0, π_1, π), $(\pi_{00}, \pi_{01}, \pi_0)$, and $(\pi_{10}, \pi_{11}, \pi_1)$ comply with (t, f), (t_0, f_0), and (t_1, f_1), respectively.

Therefore, by definition, there are transitions

$$(p, 0, \pi, \pi_0, t, f), \pi_0(k_0) \to (p_0, 0, \pi_0, \pi_{00}, t_0, f_0), (p_0, 1, \pi_0, \pi_{01}, t_0, f_0) \in \widetilde{\mu}$$

and

$$(p, 1, \pi, \pi_1, t, f), \pi_1(k_1) \to (p_1, 0, \pi_1, \pi_{10}, t_1, f_1), (p_1, 1, \pi_1, \pi_{11}, t_1, f_1) \in \widetilde{\mu}$$

such that $v_{0,\pi_0(k_0)} = \sigma_0$ and $v_{1,\pi_1(k_1)} = \sigma_1$.

By the definition of $\widetilde{\mu}$,

$$(p_0, k_0), (p_1, k_1) \to p \in \mu$$

and

$$((p_0, k_0), (p_1, k_1), t, f) \in \tau.$$

Let $\boldsymbol{w}_0, \boldsymbol{w}_1$ be assignments such that $\boldsymbol{w}_0 =_{\rho(p_0)} \pi_0(\boldsymbol{v})$ and $\boldsymbol{w}_1 =_{\rho(p_1)} \pi_1(\boldsymbol{v})$, and let denote $\boldsymbol{w}'_0 = \pi_0(\boldsymbol{v})$ and $\boldsymbol{w}'_1 = \pi_1(\boldsymbol{v})$.

Since $w_{0,k_0} = v_{\pi_0(k_0)} = \sigma_0$ and $w_{1,k_1} = v_{\pi_1(k_1)} = \sigma_1$,

$$(p_0, \boldsymbol{w}_0), \sigma_0, (p_1, \boldsymbol{w}_1), \sigma_1 \to (p, \boldsymbol{w}) \in \mu^c$$

where $\boldsymbol{w}$ is an assignment such that $(\boldsymbol{w}'_0, \boldsymbol{w}'_1, \boldsymbol{w})$ is an instance of (t, f).

Thus, the proof will be complete if we show that $\pi(\boldsymbol{v}) =_{\rho(p)} \boldsymbol{w}$. Let $i \notin \rho(p)$. Since (π_0, π_1, π) complies with (t, f), and $(\boldsymbol{w}'_0, \boldsymbol{w}'_1, \boldsymbol{w})$ is an instance of (t, f),

$$v_{\pi(i)} = \begin{cases} v_{\pi_0(j)} = v_{0,\pi_0(j)} = w'_{0,j} = w_i, \text{ where } f(j) = (0, i) \\ v_{\pi_1(j)} = v_{1,\pi_1(j)} = w'_{1,j} = w_i, \text{ where } f(j) = (1, i) \end{cases}$$

and the converse part of the lemma follows. □

C Closure Properties

In this section we establish some basic closure properties of the tree languages defined defined by ↓-NR-FMAs. The proofs are pretty standard and can be easily modified for all other models of tree automata introduced in this paper.

Let $A_i = \langle S_i, s_{0,i}, \boldsymbol{u}_i, \rho_i, \mu_i, F_i \rangle$, $i = 1, 2$, be ↓-FMAs. Without loss of generality, we assume that A_1 and A_2 possess the following properties.

1. The sets of states S_1 and S_2 are disjoint.
2. Both the initial states $s_{0,1}$ and $s_{0,2}$ are not accessible from the other states of the corresponding automaton. That is, there is no transition of the form $(q, k) \to (s_{0,i}, q')$ or $(q, k) \to (q', s_{0,i})$, $i = 1, 2$.
3. Both automata have the same initial assignment of the form $\theta_1 \cdots \theta_m \#^r$, and registers $1, \ldots, m$ are never reset.[18] This property can be verified similarly to the proof of [7, Lemma 5.1].

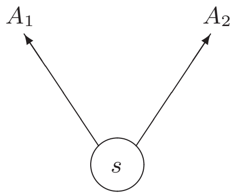


Fig. 19. The diagram of A

Closure under union. We construct an automaton A that accepts $L(A_1) \cup L(A_2)$ as the "union" of A_1 and A_2. The initial state of A is a new state s. ¿From this state A can simulate either of A_1 and A_2, as illustrated in Fig. 19.

The initial assignment of A is $\theta_1 \cdots \theta_m \#^{2r}$. Except for the input symbols from $\{\theta_1, \theta_2, \ldots, \theta_m\}$, A_1 is simulated by the registers $(m+1)$ to $(m+r)$, while A_2 is simulated by the last r registers. The transitions of A_2 are renamed from $(q,k) \to (q_0, q_1)$ to $(q, k+r) \to (q_0, q_1)$ for all $k = m+1, m+2, \cdots, m+r$. For the inputs from $\{\theta_1, \theta_2, \ldots, \theta_m\}$, both A_1 and A_2 use the first m registers. Finally, $\rho(s) = \rho(s_1) \cup \{k + r : k \in \rho(s_2)\}$.

Closure under concatenation.[19] An automaton A accepting $L(A_1)L(A_2)$ results in "extending" A_1 with A_2. That is, A starts by simulating A_1 until it reaches a final relation. Then it continues by simulating A_2. This is achieved by changing all the final relations $(q,k) \in F_1$ to $(q,k) \to (s_{0,2}, s_{0,2})$. The set of final relations of A is F_2.

Closure under Kleene star.[20] The construction is similar to the concatenation case. To accept $L(A_1)^*$, the automaton A simulate A_1 a number of times, which is achieved by adding the transition $(q,k) \to (s_{0,1}, s_{0,1})$ to μ_1 for each final relation $(q,k) \in F_1$ and setting $\rho(s_{0,1}) = \{m+1, \ldots, m+r\}$.

Closure under intersection. The construction here is a bit more involved. The basic idea is to use the (equivalent) tree automata model similar to the M-FMA introduced in [6, Sect. 3]. These automata are allowed to "consult" a number of registers at the same time.

Definition 7. (Cf. [20, Definition 5. pp. 20-22].) *A top-down M-finite-memory automaton (↓-M-FMA) is a system* $A = \langle S, s_0, \boldsymbol{u}, \rho, \mu, F \rangle$, *where*

[18] Consequently, only registers $m+1, \ldots, m+r$ may be reset.

[19] The concatenation of two tree languages L_1 and L_2 is defined by extending each leaf in every tree from L_1 with two child nodes in each of which a tree from L_2 is rooted. In terms of the corresponding definition in [3, Sect. 2.2, p. 52], we can view this as first extending each leaf in every tree from L_1 with two children labeled with a new symbol $\square$ and then applying $\cdot_\square$.

[20] For a tree language L, L^* is the collection of all iterated concatenations of $L \cup \{\square\}$, where $\square$ is the "empty tree," i.e., a single node labeled with the symbol $\square$. In terms of the corresponding definition in [3, Sect. 2.2, p. 54], $L^* = L^{*,\square}$ $(= \bigcup_{n \geq 0} L^{n,\square})$.

– *S is the finite set of states.*
– *s_0 is the initial state.*
– *$\boldsymbol{u} = \theta_1 \cdots \theta_m \underbrace{\# \cdots \#}_{r} \underbrace{\# \cdots \#}_{r} \in (\Sigma \cup \{\#\})^{(m+2r)}$ is the initial assignment.*
– *$\rho : S \to \{m+1, \ldots, m+r\} \times \{m+r+1, \ldots, m+2r\}$ is the reassignment function.*
– *μ is the set of transitions of the following form*
  - $(p, k) \to (p_0, p_1) \in S \times \{1, \ldots, m\} \times S \times S$,
  - $(p, (k_0, k_1)) \to (p_0, p_1) \in S \times \{m+1, \ldots, m+r\} \times \{m+r+1, \ldots, m+2r\} \times S \times S$.
– *F is the set of final relations of the following form*
  - $(p, k) \in S \times \{1, \ldots, m\}$,
  - $(p, (k_0, k_1)) \in S \times \{m+1, \ldots, m+r\} \times \{m+r+1, \ldots, m+2r\}$.

Similarly, the transition relation μ induces the following relation μ^c which is defined as follows. $(p, \boldsymbol{w}), \sigma \to (p_0, \boldsymbol{w}'), (p_1, \boldsymbol{w}')$ belongs to μ^c if and only if the following conditions are satisfied. $\boldsymbol{w}' = w'_1 \cdots w'_{m+2r}$, where $w'_i = w_i$, for all $i \notin \{i, j : (i, j) \in \rho(p)\}$ and

1. If $\sigma \in \{\theta_1, \ldots, \theta_m\}$, then $\boldsymbol{w}'_k = \theta$ and $(p, k, (p_0, p_1)) \in \mu$.
2. If $\sigma \notin \{\theta_1, \ldots, \theta_m\}$, then $\sigma = w'_{k_0} = w'_{k_1}$, for some $k_0 < k_1$ and the triple $(p, (k_0, k_1), (p_0, p_1))$ belongs to μ.

The relation F^c is defined as follows. A pair $((p, \boldsymbol{w}), \sigma) \in F^c$ if the following holds.

– If $\sigma \in \{\theta_1, \ldots, \theta_m\}$, then $\boldsymbol{w}'_k = \theta$ and $(p, k, (p_0, p_1)) \in F$.
– If $\sigma \notin \{\theta_1, \ldots, \theta_m\}$, then $\sigma = w'_{k_0} = w'_{k_1}$, where $m < k_0 < k_1$ and $(p, (k_0, k_1), (p_0, p_1))$ belong to F.

Proposition 6. *For each $\downarrow$-M-NR-FMA A there exists an $\downarrow$-NR-FMA A' such that $L(A) = L(A')$.*

Proof. We convert A into a standard $(m + r^2)$-register $\downarrow$-NR-FMA A' whose initial assignment is $\theta_1 \cdots \theta_m \#^{r^2}$.

Let $\varphi : \{m+1, \ldots, m+r\} \times \{m+r+1, \ldots, m+2r\} \to \{m+1, \ldots, m+r^2\}$ be a one-to-one function. The simulation of A by A' is done by referring each pair of registers (k_0, k_1) to a single register $\varphi(k, k')$ of A'. Formally, $A' = \langle S', s'_0, \boldsymbol{u}', \rho', \mu', F' \rangle$ is defined as follows.

– $S' = S$ and $s'_0 = s_0$.
– $\boldsymbol{u}' = \theta_1 \cdots \theta_m \#^{r^2}$.
– $\rho(q_1, q_2) = \varphi(\rho(q_1) \times \{m+r+1, \ldots, m+2r\} \cup \{m+1, \ldots, m+r\} \times \rho(q_2))$.
– μ' consists of the following two types of transitions:
  - for every $((q_1, q_2)(k_1, k_2)) \to ((q'_1, q''_1)(q_2, q''_2)) \in \mu$ it contains

$$((q, q'), \varphi((k, k'))) \to ((q_0, q'_0), (q_1, q'_1)),$$

  - for every $((q_1, q_2), k) \to ((q_1', q_2''), (q_2, q_2'')) \in \mu$, it contains

$$((q_1, q_2), k) \to ((q_1', q_2''), (q_2, q_2''))$$

    itself.
- F' consists of the following two types of relations:
  - for every $((q_1, q_2), (k_1, k_2)) \in F$ it contains $((q_1, q_2), \varphi(k_1, k_2))$, and
  - for every $((q_1, q_2), k) \in F$ it contains $((q_1, q_2), k)$ itself. $\square$

Now we construct an $\downarrow$-M-FMA A that accepts $L(A_1) \cap L(A_2)$ by simultaneously simulating A_1 and A_2. This is done by defining A as the product of A_1 and A_2.

The precise description of A is as follows. Like in the case of the closure under union, the initial assignment of A is $\theta_1 \cdots \theta_m \#^{2r}$ and the automata A_1 and A_2 are simulated on the registers $\{1, \ldots, m+r\}$ and $\{1, \ldots, m\} \cup \{m+r+1 \ldots, m+2r\}$, respectively. That is, $A = \langle S, s_0, \boldsymbol{u}, \rho, \mu, F \rangle$ is defined as follows.

- $S = S_1 \times S_2$.
- $s_0 = (s_{0,1}, s_{0,2})$.
- $\boldsymbol{u} = \theta_1 \cdots \theta_m \#^{2r}$.
- $\rho(q_1, q_2) = \{\rho_1(q_1), \rho_2(q_2) + r\}$.
- μ consists of the following two types of transitions:
  - for every $(q_1, k_1) \to (q_1', q_1'') \in \mu_1$ and every $(q_2, k_2) \to (q_2', q_2'') \in \mu_2$ such that $m \leq k_1, k_2 \leq m + r$, it contains $((q_1, q_2), (k_1, k_2 + r)) \to (q_1', q_2')(q_1'', q_1'')$, and
  - for every $(q_1, k) \to (q_1', q_1'') \in \mu_1$ and every $(q_2, k) \to (q_2', q_2'') \in \mu_2$ such that $1 \leq k \leq m$, it contains $((q_1, q_2), k) \to (q_1', q_2')(q_1'', q_1'')$.
- F consists of the following two types of relations:
  - for every $(q_1, k_1) \in F_1$ and every $(q_2, k_2) \in F_2$ such that $m \leq k_1, k_2 \leq m + r$, it contains $((q_1, q_2), (k_1, k_2 + r))$, and
  - for every $(q_1, k) \in F_1$ and every $(q_2, k) \in F_2$ such that $1 \leq k \leq m$, it contains $((q_1, q_2), k)$.