Regular Tree and Regular Hedge Languages over Unranked Alphabets: Version 1, April 3, 2001*

Anne Brüggemann-Klein† Makoto Murata‡ Derick Wood§

Abstract

We survey the basic results on regular tree languages over unranked alphabets; that is, we use an unranked alphabet for the labels of nodes, we allow unbounded, yet regular, degree nodes and we treat sequences of trees that, following Courcelle, we call hedges.

The survey was begun by the first and third authors. Subsequently, when they discovered that the second author had already written a summary of this view of tree automata and languages, the three authors decided to join forces and produce a consistent review of the area.

The survey is still unfinished because we have been unable to find the time to finish it. We are making it available in this unfinished form as a research report because it has, already, been heavily cited in the literature.

1 Introduction

The study of tree automata has a long (in computer science) history; see the survey of Thatcher [Tha87], and the texts of Gecseg and Steinby [GS84], and of the TATA group [CDG$^+$98]. We, however, take a path that is somewhat different from most of the

*The work of the first and third authors was supported partially by a joint DAAD-HK grant. In addition, the work of the third author was supported under a grant from the Research Grants Council of Hong Kong.

†Institut für Informatik, Technische Universität München, Arcisstr. 21, 80290 München, Germany. E-mail: brueggem@informatik.tu-muenchen.de

‡IBM Tokyo Research Laboratory, IUJ Research Institute, 1623-14 Shimotsuruma, Yamato-shi, Kanagawa-ken 242-8502, Japan. E-mail: mmurata@trl.ibm.co.jp

§Department of Computer Science, Hong Kong University of Science & Technology, Clear Water Bay, Kowloon, Hong Kong SAR. E-mail: dwood@ust.hk

research in this area. We use an unranked alphabet, we allow unbounded, yet regular, degree nodes and we treat sequences of trees that we call **hedges,** a term coined by Courcelle [Cou78, Cou89]. One of us has been looking for a name that is as simple as forest (which is used for sets of trees), but is more appropriate for sequences of trees. Hedge seems to best fit this requirement. Note that when we remove the root of a tree we are left with a hedge and, conversely, adding a new root to a hedge gives a tree.

One reason for taking a path less traveled by is that the Standard Generalized Markup Language (SGML) [ISO86] and the Extensible Markup Language (XML) [BPMM98], which are metalanguages for document grammars, introduce these requirements. Since most work on classes of documents generated by document grammars is grammatical in nature why do we consider tree and hedge automata rather than document grammars? We have investigated the use and theory of document grammars elsewhere [Woo95, BKW98, KW99]; however, grammars are not always the most appropriate tool for modeling applications. Tree and hedge automata provide a natural framework for investigating tree and hedge transformations, tree and hedge query languages, layout generation for trees and hedges, and context specification and evaluation [BKW00a, BKW00b, Mur95, Mur97, Mur98].

Although many of the fundamental results for extended hedge automata are proved by similar methods to those for the corresponding tree automaton results, they are not without interest. In addition, Thatcher both in published work [Tha70, Tha67a, TW68] and unpublished work [Tha67b] developed the basic theory of unranked tree automata and also introduced and investigated their regular extensions. This paper collects in a uniform presentation, both known results and new results for extended tree automata and also demonstrates how, if at all, these results carry over to extended hedge automata.

The paper currently has nine further sections. Section 2 introduces the basic notation and terminology for hedges, trees and tree automata, and Section 3 introduces hedge automata. Section 4 focusses on top-down and bottom-up automata. Section 5 introduces an algebraic approach to tree automata, Section 6 deals with local tree languages and Section 7 discusses characterizable languages. Section 8 introduces the notion of a top congruence and Section 9 uses that notion to prove that two-way tree automata are only as expressive as standard tree automata. Section 10 introduces tree regular expressions and Section 11 introduces endmarked tree automata. In Section 12 we review previous work and relate it to our contributions and in Section 13 we mention some open problems.

2 Notation and definitions

We define sequences of trees that we call **hedges** after Courcelle [Cou78, Cou89] and then extend the definition of tree automata to hedge automata along the lines of Murata's survey [Mur95]. Note that trees are hedges that have exactly one root.

We assume that the nodes of hedges and trees are labeled with symbols from some finite alphabet Σ. The Greek letter λ denotes the empty string over any alphabet.

We represent hedges and trees by sequences of terms, using the symbols in Σ as operators. Operators have no rank or arity, so they can have any number of operands, including none. Terms have at least one operator.

For example, the term sequence $a()b(0)a()$ represents a hedge of three trees each of height 1. We will often drop the terminating parentheses $()$ and use abc to represent the preceding hedge. Note that it also represents a string over Σ. The term $a(a(a()a())a(a()a()))$ represents a complete binary tree of height 3, all of whose nodes have the label a.

More formally, we define hedges over Σ inductively as follows:

- The empty term sequence is a hedge and it is denoted by λ
- If g is a hedge and $a \in \Sigma$, then $a(g)$ is a hedge
- If g and h are hedges, then gh is a hedge

The **width** $|h|$ of a hedge h is also defined inductively by: $|\lambda| = 0$; $|a(g)| = 1$; and $|gh| = |g| + |h|$. Immediately, a **tree** is a hedge of width one.

We denote symbols in Σ with a, strings over Σ (which we also call strings) with w, and sets of strings over Σ (which we also call string languages) with L.

We denote hedges with g or h, sets of hedges (which we also call hedge languages) with H, and sets of sets of hedges (which we also call families of hedge languages) with $\mathcal{H}$.

We denote trees with t, sets of trees (which we also call tree languages) with T, and sets of sets of trees (which we also call families of tree languages) with $\mathcal{T}$. Subscripted and superscripted variables have the same types as their base names.

Definition 2.1 We define the **set of nodes of a hedge** h as a set of strings of natural numbers and denote it by $\mathrm{nodes}(h)$. The definition is by induction on h:

1. For the hedge λ, $\mathrm{nodes}(\lambda) = \emptyset$.
2. For a tree $a(g)$, $\mathrm{nodes}(a(g)) = \{1\} \cup \{1 \cdot x \mid x \in \mathrm{nodes}(g)\}$.
3. For a hedge gh, $\mathrm{nodes}(gh) =$
$$\mathrm{nodes}(g) \cup \{(|g| + i_1) \cdot i_2 \cdots i_k \mid \ i_1 \cdots i_k \in \mathrm{nodes}(h), \text{ where } k \geq 0\}.$$

The nodes of a hedge (or of a tree) viewed as a term sequence (or as a term) correspond to occurrences of subterms. We denote nodes of hedges and trees with ν.

Definition 2.2 For each node ν of a hedge g, we define the set of **ν's children** to consist of all nodes $\nu \cdot i$ in nodes(g) and we denote it by children(ν).

Definition 2.3 A node ν of a hedge g is a **leaf** if and only if children$(\nu) = \emptyset$ and the set leaves(g) denotes the leaves of g.

We define the **roots** of a hedge h to be the set of all nodes $gn \in h$ such that $\nu \in \mathbf{N}$.

Note that the leaves and only the leaves of a hedge [nodes corresponding to subterms of the form $a()$] have an empty set of children.

Definition 2.4 For each node ν of a hedge g, the **label of ν in Σ** is denoted by label(ν). More precisely, for a hedge $g = g_1 \cdots g_n$ of width $n \geq 0$, we define:

1. The label of the node i of the hedge g to be $a_i \in \Sigma$, where $g_i = a_i(h_i)$,
2. The label of the node $i \cdot s$ in g to be the label of the node s in the tree g_i.

Definition 2.5 A **two-way nondeterministic tree automaton (2NTA)** is specified by a tuple (Q, δ, F), where Q is a finite set of states, $F \subseteq Q$ is a set of final states and $\delta \subseteq \Sigma \times Q^* \times Q \times \{u, d, s\}$ is a transition relation such that, for each a in Σ, q in Q and m in $\{u, d, s\}$, the set $\{w \in Q^* | (a, w, q, m) \in \delta\}$ is a regular set of strings over the alphabet Q (regularity of the transition relation).

We denote both hedge and tree automata with M.

We define the computations (or behavior) of a 2NTA on a tree as a sequence of configurations. A configuration assigns a state of the automaton to each node in what we call a cut of the tree.

Definition 2.6 A **cut** of a tree t is a subset of nodes(t) such that, for each leaf node ν of t, there is exactly one node (represented by a prefix of ν) on the path from the root to ν that is in the cut.

We denote cuts for both hedges and trees with C.

Definition 2.7 A **configuration** of a 2NTA $M = (Q, \delta, F)$ operating on a tree t is a map $c : C \longrightarrow Q$ from a cut C of t to the state set Q of M.

We denote configurations with c.

Let ν be a node of a tree t and let $c : C \longrightarrow Q$ be a configuration of the 2NTA M operating on t. If children$(\nu) \subseteq C$, then formally $c(children(\nu))$ is a subset of Q. We overload this notation so that $c(\text{children}(\nu))$ also denotes the sequence of states in Q which arises from the order of ν's children in t.

Definition 2.8

1. A **starting configuration** of a 2NTA $M = (Q, \delta, F)$ operating on a tree t is a configuration $c : C \longrightarrow Q$, where $C = \text{leaves}(t)$ and $c(\nu)$ is any state q in Q such that $(\text{label}(\nu), \lambda, q, u) \in \delta$.
2. A **halting configuration** is a configuration $c : C \longrightarrow Q$ such that $C = \{1\}$.
3. An **accepting configuration** is a halting configuration c such that $c(\{1\}) \in F$.

Definition 2.9

1. A 2NTA $M = (Q, \delta, F)$ operating on a tree t makes a transition from one configuration $c_1 : C_1 \longrightarrow Q$ to a second configuration $c_2 : C_2 \longrightarrow Q$ (symbolically $c_1 \longrightarrow c_2$) if and only if it makes an up transition, a down transition or a staying transition (defined in parts 2, 3 and 4).
2. M makes an **up transition** from c_1 to c_2 if and only if t has a node ν such that
   (a) $\text{children}(\nu) \subseteq C_1$,
   (b) $C_2 = (C_1 \setminus \text{children}(\nu)) \cup \{\nu\}$,
   (c) $(\text{label}(\nu), c_1(\text{children}(\nu)), c_2(\nu), u) \in \delta$ and
   (d) c_1 is identical to c_2 on their domains' common subset $C_1 \cap C_2$.
3. M makes a **down transition** from c_1 to c_2 if and only if t has a node ν such that
   (a) $\nu \in C_1$,
   (b) $C_2 = (C_1 \setminus \{\nu\} \cup \text{children}(\nu))$,
   (c) $(\text{label}(\nu), c_2(\text{children}(\nu)), c_1(\nu), d) \in \delta$ and
   (d) c_1 is identical to c_2 on their domains' common subset $C_1 \cap C_2$.
4. M makes a **staying transition** from c_1 to c_2 if and only if t has a node ν such that
   (a) $\nu \in C_1$,
   (b) $C_2 = C_1$,
   (c) $(\text{label}(\nu), c_1(\nu), c_2(\nu), s) \in \delta$ and
   (d) c_1 is identical to c_2 on their domains' common subset $C_1 \setminus \{\nu\}$, which is identical to $C_2 \setminus \{\nu\}$.

Definition 2.10

1. A **computation of a 2NTA** M on a tree t from configuration c to configuration c' is a sequence of configurations $c_1, \ldots, c_n$, $n \geq 1$, such that $c = c_1 \longrightarrow \cdots \longrightarrow c_n = c'$.
2. An **accepting computation of** M on t is a computation from a starting configuration to an accepting configuration.

For a 2NTA $M = (Q, \delta, F)$, the transition relation δ is a subset of $\Sigma \times Q^* \times Q \times \{u, d, s\}$. For a down transition (a, w, g, d) in δ, the interpretation is that M, when in state q and sitting on a node labeled a, can move down and label its children one after the other with the states in the string w, which is in Q^*. It might be confusing at a first glance that in the transition relation δ the next state for a down transition is given before the current state. We have chosen this notation for the sake of homogenuity. We are making this point here, since it may be particularly surprising when we define top-down tree automata.

Definition 2.11

1. A tree t is **recognizable** by an automaton M if and only if there is an accepting computation of M on t.
2. The **tree language** $T(M)$ that is recognizable by a 2NTA M is the set of trees that are recognizable by M.

Definition 2.12

1. A **nondeterministic bottom-up tree automaton (NBTA)** is a 2NTA $M = (Q, \delta, F)$, where δ contains only transitions whose last component is u. For an NBTA M, we can consider δ to be a subset of $\Sigma \times Q^* \times Q$, dropping the last, constant component in the Cartesian product.
2. A **deterministic bottom-up tree automaton (DBTA)** is an NBTA $M = (Q, \delta, F)$, where, for each a in Σ and w in Q^*, there is at most one q in Q such that $(a, w, q) \in \delta$.
3. A tree automaton without further qualification is always an NBTA.

The behaviour of a bottom-up tree automaton, which is also a 2NTA, is precisely the same as when it is viewed as a 2NTA.

Definition 2.13

1. A **nondeterministic top-down tree automaton (NTTA)** is a 2NTA $M = (Q, \delta, I)$, where δ contains only transitions whose last component is d and I is a **set of initial states.** For an NTTA M, we can consider δ to be a subset of $\Sigma \times Q^* \times Q$, dropping the last, constant component in the Cartesian product.
2. A **deterministic top-down tree automaton (DTTA)** is an NTTA $M = (Q, \delta, I)$, where I is a singleton set and, for each a in Σ and q in Q, there is, for each natural number n, at most one w of length n in Q^* such that $(a, w, q) \in \delta$.

The behavior of a top-down tree automaton $M = (Q, \delta, I)$ is slightly different. Computations between configurations are defined precisely the same as in the more general case of 2NTAs. The starting configuration for a tree t is, however, any configuration $c : \{1\} \longrightarrow I$, a halting configuration is any configuration $c : \text{leaves}(t) \longrightarrow Q$, and an accepting configuration is any halting configuration $c : \text{leaves}(t) \longrightarrow Q$ such that, for each leaf node ν of t, the condition $(\text{label}(\nu), \lambda, c(\nu)) \in \delta$ holds. For obvious reasons, in the case of a top-down tree automaton, the set I is called the set of initial states.

For a bottom-up as well as for a top-down tree automaton the language of recognizable trees consists of the trees for which the automaton can carry out a computation from a starting configuration to an accepting configuration.

Definition 2.14 A tree language is **tree regular** if and only if it is recognizable by a tree automaton, that is, by an NBTA.

3 Hedge automata

We extend the definitions of tree automata to hedge automata in the obvious way.

Definition 3.1 A **two-way nondeterministic hedge automaton (2NHA)** is specified by a tuple (Q, δ, F), where Q is a finite set of states, $F \subseteq Q$ is a set of final states and $\delta \subseteq \Sigma \times Q^* \times Q \times \{u, d, s\}$ is a transition relation such that, for each a in Σ, q in Q, and m in $\{u, d, s\}$, the set $\{w \in Q^* | (a, w, q, m) \in \delta\}$ is a regular set of strings over the alphabet Q (regularity of the transition relation).

We denote both hedge and tree automata with M.

We define the computations (or behavior) of a 2NHA on a hedge as a sequence of configurations. A configuration assigns a state of the automaton to each node in what we call a cut of the hedge.

Definition 3.2 A **cut of a hedge h** is a subset of nodes(h) such that, for each leaf node ν of h, there is exactly one node (represented by a prefix of ν) on the path from a root node of h to ν that is in the cut.

We denote cuts with C.

Definition 3.3 A **configuration** of a 2NHA $M = (Q, \delta, F)$ operating on a hedge h is a map $c : C \longrightarrow Q$ from a cut C of h to the state set Q of M.

We denote configurations with c.

Overloading: Let ν be a node of a hedge h and let $c : C \longrightarrow Q$ be a configuration of the 2NHA M operating on h. If children(ν) $\subseteq C$, then formally $c(children(\nu))$ is a subset of Q. We overload this notation so that c(children(ν)) also denotes *the sequence of states in* Q that arises from the order of ν's children in h.

Definition 3.4

1. A **starting configuration** of a 2NHA $M = (Q, \delta, F)$ operating on a hedge h is a configuration $c : C \longrightarrow Q$, where C = leaves(h) and $c(\nu)$ is any state q in Q such that (label(ν), $\lambda, q, u) \in \delta$.

2. A **halting configuration** is a configuration $c : C \longrightarrow Q$ such that $C = \{\text{roots}(h)\}$.
3. An **accepting configuration** is a halting configuration c such that $c(\text{roots}(h)) \in F^*$.

Definition 3.5

1. A 2NHA $M = (Q, \delta, F)$ operating on a hedge h makes a transition from one configuration $c_1 : C_1 \longrightarrow Q$ to a second configuration $c_2 : C_2 \longrightarrow Q$ (symbolically $c_1 \longrightarrow c_2$) if and only if it makes an up transition, a down transition or a staying transition (defined in parts 2, 3 and 4).
2. M makes an **up transition** from c_1 to c_2 if and only if h has a node ν such that
   (a) $\text{children}(\nu) \subseteq C_1$,
   (b) $C_2 = (C_1 \setminus \text{children}(\nu)) \cup \{\nu\}$,
   (c) $(\text{label}(\nu), c_1(\text{children}(\nu)), c_2(\nu), u) \in \delta$ and
   (d) c_1 is identical to c_2 on their domains' common subset $C_1 \cap C_2$.
3. M makes a **down transition** from c_1 to c_2 if and only if h has a node ν such that
   (a) $\nu \in C_1$,
   (b) $C_2 = (C_1 \setminus \{\nu\} \cup \text{children}(\nu))$,
   (c) $(\text{label}(\nu), c_2(\text{children}(\nu)), c_1(\nu), d) \in \delta$ and
   (d) c_1 is identical to c_2 on their domains' common subset $C_1 \cap C_2$.
4. M makes a **staying transition** from c_1 to c_2 if and only if h has a node ν such that
   (a) $\nu \in C_1$,
   (b) $C_2 = C_1$,
   (c) $(\text{label}(\nu), c_1(\nu), c_2(\nu), s) \in \delta$ and
   (d) c_1 is identical to c_2 on their domains' common subset $C_1 \setminus \{\nu\}$, which is identical to $C_2 \setminus \{\nu\}$.

Definition 3.6

1. A computation of a 2NHA M on a hedge h from configuration c to configuration c' is a sequence of configurations $c_1, \ldots, c_n$, $n \geq 1$, such that $c = c_1 \longrightarrow \cdots \longrightarrow c_n = c'$.
2. An accepting computation of M on h is a computation from a starting configuration to an accepting configuration.

For a 2NHA $M = (Q, \delta, F)$, the transition relation δ is a subset of $\Sigma \times Q^* \times Q \times \{u, d, s\}$. For a down transition (a, w, g, d) in δ, the interpretation is that M, when in state q and sitting on a node labeled a, can move down and label its children one after the other with the states in the string w, which is in Q^*.

Definition 3.7

1. A hedge h is **recognizable** by an automaton M if and only if there is an accepting computation of M on h.
2. The **hedge language** $T(M)$ that is recognizable by a 2NHA M is the set of hedges that are recognizable by M.

Definition 3.8

1. A **nondeterministic bottom-up hedge automaton (NBHA)** is a 2NHA $M = (Q, \delta, F)$, where δ contains only transitions whose last component is u. For an NBHA M, we can consider δ to be a subset of $\Sigma \times Q^* \times Q$, dropping the last, constant component in the Cartesian product.
2. A **deterministic bottom-up hedge automaton (DBHA)** is an NBHA $M = (Q, \delta, F)$, where, for each a in Σ and w in Q^*, there is at most one q in Q such that $(a, w, q) \in \delta$.
3. A hedge automaton without further qualification is always an NBHA.

The behaviour of a bottom-up hedge automaton, which is also a 2NHA, is precisely the same as when it is viewed as a 2NHA.

Definition 3.9

1. A **nondeterministic top-down hedge automaton (NTHA)** is a 2NHA $M = (Q, \delta, I)$, where δ contains only transitions whose last component is d and I is a set of initial states. For an NTHA M, we can consider δ to be a subset of $\Sigma \times Q^* \times Q$, dropping the last, constant component in the Cartesian product.
2. A **deterministic top-down hedge automaton (DTHA)** is an NTHA $M = (Q, \delta, I)$, where I is a singleton set and, for each a in Σ and q in Q, there is, for each natural number n, at most one w of length n in Q^* such that $(a, w, q) \in \delta$.

For a top-down hedge automaton $M = (Q, \delta, I)$, the situation is slightly different. Computations between configurations are defined precisely the same as in the more general case of 2NHAs. The starting configuration for a hedge h is, however, any configuration $c : \{\text{root}(t)\} \longrightarrow I$, a halting configuration is any configuration $c : \text{leaves}(t) \longrightarrow Q$, and an accepting configuration is any halting configuration $c : \text{leaves}(t) \longrightarrow Q$ such that, for each leaf node ν of h, the condition $(\text{label}(\nu), \lambda, c(\nu)) \in \delta$ holds.

For a bottom-up as well as for a top-down hedge automaton, the language of recognizable hedges consists of the hedges for which the automaton can carry out a computation from a starting configuration to an accepting configuration.

Definition 3.10 A hedge language is **hedge regular** if and only if it is recognizable by a hedge automaton, that is, by an NBHA.

4 Languages recognizable by bottom-up and top-down tree automata

The following results have analogs in the theory of tree automata over ranked alphabets. Each proof is based on a proof for the ranked-alphabet case. *These results can be extended to hedge automata in a straightforward manner.*

Theorem A *The tree languages that are recognizable by nondeterministic top-down tree automata are precisely the tree-regular languages; that is, nondeterministic bottom-up and nondeterministic top-down tree automata recognize precisely the same tree languages.*

PROOF Let $M = (Q, \delta, F)$ be a NBTA and let $M' = (Q, \delta, I)$ be the dual NTTA, whose sets of states and transitions are the same as M's and whose set of initial states is the same as M's set of final states. We note that, for any tree t, the starting configurations of M on t are precisely the accepting configurations of M' on t and that the accepting configurations of M on t are precisely the starting configurations of M' on t. Furthermore, for any two configurations c_1 and c_2 on t, M makes a transition from c_1 to c_2 when operating on t if and only if M' makes the reverse transition fom c_2 to c_1. Hence, accepting computations of M on t are precisely the reverse of accepting computations of M' on t. Therefore, M and M' recognize the same tree language. $\square$

Theorem B *Each tree-regular language is recognizable by a deterministic bottom-up tree automaton; that is, deterministic bottom-up and nondeterministic bottom-up tree automata recognize precisely the same tree languages.*

PROOF We apply the well-known subset construction for string automata or tree automata over ranked alphabets to tree automata over unranked alphabets.

Let $M = (Q, \delta, F)$ be a NBTA. We define a DBTA $M' = (Q', \delta', F')$ that recognizes the same tree language as does M.

Let Q' be the powerset of Q and let F' be the set $\{S \subseteq Q | S \cap F \neq \emptyset\}$.

For each symbol a in Σ, each string $X = X_1 \cdots X_n$ over Q', $n \geq 0$, and each state S in Q', we add (a, X, S) to δ' if and only if

$$S = \{q : \text{there are } q_i \text{ in } X_i,\ 1 \leq i \leq n, \text{ such that } (a, q_1 \cdots q_n, q) \in \delta\}.$$

In particular, $\delta'(a, \lambda, S)$ if and only if

$$S = \{q | \delta(a, \lambda, q)\}.$$

To demonstrate the regularity of the transition relation, we need to show that, for each a in Σ and each $S \in Q'$, the set

$$L'_{a,S} = \{X_1 \cdots X_n \text{ in } Q'^* | \delta'(a, X_1 \cdots X_n, S)\}$$

is regular.

First, note that $X_1 \cdots X_n \in L'_{a,S}$ if and only if the following two conditions hold:

1. For each q in S, there are q_i in X_i, $1 \leq i \leq n$, such that $(a, q_1 \cdots q_n, q) \in \delta$.
2. For each q_i in X_i, $1 \leq i \leq n$, there is a q in S such that $(a, q_1 \cdots q_n, q) \in \delta$.

Since, for each a in Σ and each q in Q, the set

$$\{q_1 \cdots q_n \text{ in } Q^* | (a, q_1 \cdots q_n, q) \in \delta\}$$

is a regular string language, one may be tempted to conclude that $L'_{a,S}$ is also regular since it appears to be built from unions and intersections of regular languages. Unfortunately, these unions and intersections are unbounded. Therefore, we need to use a more sophisticated argument that involves string morphisms and, in particular, the well-known fact that morphic images of regular sets are also regular. Additionally, we use the $\cap$-morphic image

$$h^\cap(L) = \{w | \text{there is } v \text{ such that } h(v) = w \in L\}$$

of a language L with respect to a morphism h. Note that the $\cap$-morphic images of regular sets are also regular.

Now, observe that

$$L''_{a,q} = \{(q_1, X_1) \cdots (q_n, X_n) | (a, q_1 \cdots q_n, q) \in \delta \text{ and } q_i \in X_i\}$$

is a regular string language over the alphabet

$$Q'' = \{(q, X) | q \in X \in Q'\}.$$

Furthermore, consider the morphism $h : Q''^* \longrightarrow Q'^*$ that maps each (q, X) in Q'' to its second component X.

Since $L'_{a,S}$ is the intersection of the two sets

$$\bigcap_{q \in S} h(L''_{a,q})$$

and

$$h^\cap(\bigcup_{q \in S} L''_{a,q}),$$

we can conclude that $L'_{a,S}$ is string regular.

Having established the regularity of the transition relation of $M' = (Q', \delta', F')$, it is now obvious that M' is indeed an DBTA and that M and M' recognize the same tree language. □

Theorem C *There are tree languages that are tree regular but are not recognizable by any deterministic top-down tree automaton.*

PROOF Any DTTA that recognizes the two trees $a(a(), a())$ and $a(b(), b())$ also recognizes the trees $a(a(), b())$ and $a(b(), a())$. On the other hand, there is an NTTA that recognizes the two-element tree language $\{a(a(), a()), a(b(), b())\}$. $\square$

5 Monoid view of tree automata

We can reprove some of the results in the Sections 4 and 3 using a more algebraic definition of a tree automaton in which the regularity of the state sequences is captured by using the morphic characterization of regular languages.

The basic result on which this section is based is as follows:

Proposition 5.1 *A language $L \subseteq \Sigma^*$ is regular if and only if there are a finite monoid K, a morphism $\alpha : \delta^* \longrightarrow K$ and a subset $R \subseteq K$ such that $L = \alpha^{-1}(R)$.*

Definition 5.1 A hedge automaton is specified by $M = (Q, \alpha, R, \mu, R_f)$, where Q is a finite set of states, R is a finite monoid, α is a map $\Sigma \times R \longrightarrow Q$, μ is a map $Q^* \longrightarrow R$, where $\mu(p\ q) = \mu(p)\mu(q)$, and R_f is a subset of R.

Obviously, given a hedge h, we can compute a state in R denoted by $M(h)$. Now, h is accepted by M if and only if $M(h) \in R_f$.

Two hedges g and h are equivalent if and only if $M(g) = M(h)$ $(\in R)$.

In addition, following the ideas of Pair and Quere [PQ68], of Takahashi [Tak75] and of Murata [Mur01], we can introduce the notion of a **binoid** (it is closed under two operations) and establish the following result.

Proposition 5.2 *A hedge language $L \subseteq \Sigma^*$ is regular if it is the inverse morphic image of a subset of some finite binoid, where the morphism is from the set of hedges to this finite binoid.*

THIS SECTION IS NOT YET FINISHED.

6 Local tree languages and morphic images

Definition 6.1 A **tree grammar** G over an alphabet Σ is specified by a tuple (P, I), where P is a subset of $\Sigma \times \Sigma^*$, for each a in Σ, the set $\{w | a \longrightarrow w \text{ in } P\}$ is string-regular, and I is a nonempty subset of Σ. We call P the set of productions and I the set of start symbols. *These results can be extended to hedge grammars in a straightforward manner.*

A tree grammar is very similiar to an extended context-free grammar; there are just two differences: First, a tree grammar can have more than one start symbol and, second, there is no distinction between terminal and nonterminal symbols in tree grammars.

We denote tree grammars with G.

Definition 6.2 The **derivation trees** of a tree grammar $G = (P, I)$ with root label a, $a \in \Sigma$, are defined inductively: For each production $a \longrightarrow w$, $w = a_1 \cdots a_n$, $n \geq 0$, and, for any derivation trees t_1 with root label $a_1, \ldots, t_n$ with root label a_n, the tree $a(t_1 \cdots t_n)$ is a derivation tree of G with root label a. A derivation tree of G is a derivation tree with a root label in I.

Note that a-labeled leaves in a derivation tree correspond to productions $a \longrightarrow \lambda$ in the grammar.

Definition 6.3 A tree language is **tree local** if and only if it is the set of derivation trees for a tree grammar.

Definition 6.4 A tree morphism is based on a map from one alphabet to another and is extended to trees in the standard way.

Theorem D *A tree language is tree regular if and only if it is the morphic image of a tree-local tree language.*

PROOF The following observation helps to clarify the connection between tree automata and tree grammars:

One-way tree automata visit each node of a tree exactly once during an accepting computation, so the whole computation can be represented by adding an additional Q-label to each Σ-labeled node.

More precisely, let $M = (Q, \delta, I)$ be an NTTA and let t be a tree. We add a second, Q-component to the labels of t, constructing a number of $(\Sigma \times Q)$-labeled trees, as follows:

1. Each state q in I is added as a second component to the root of t.
2. If q has been added to an a-labeled node ν of t, ν has precisely n children and $(a, q_1 \cdots q_n, q) \in \delta$, then q_i is added to the ith child of ν, $1 \leq i \leq n$.

We consider only those $(\Sigma \times Q)$-labeled trees t' constructed from t such that every node has been additionally labeled and every (a, q)-labeled leaf satisfies the condition $(a, \lambda, q) \in \delta$.

There is a one-to-one correspondence between accepting computations of M on t and the $(\Sigma \times Q)$-labelings of t, which we call M's accepting computation trees corresponding to t.

In particular, if t is not recognizable by M, then no corresponding accepting computation $(\Sigma \times Q)$-tree can be constructed from t and conversely.

After these preliminaries, we now proceed to the proof proper.

Given an NTTA $M = (Q, \delta, I)$, define the grammar $G = (P, \Sigma \times I)$ over the alphabet $\Sigma' = \Sigma \times Q$ as follows:

$$P = \{(a, q) \longrightarrow (a_1, q_1) \cdots (a_n, q_n) | n \geq 0,\ a_1, \ldots, a_n \in \Sigma,\ (a, q_1 \cdots q_n, q) \in \delta\}.$$

The syntax trees of G correspond precisely to the accepting computation trees of M. Therefore, we define a projection h on Σ' that gives the first, label component yo obtain the intended result; namely, the h-images of G's syntax trees are precisely the trees recognizable by M.

Conversely, given a tree grammar $G = (P, I)$, define the NTTA $M = (\Sigma', \delta, I)$ as follows: For each a in Σ, $q_1 \cdots q_n$ in Σ'^*, q in Σ, $(a, q_1 \cdots q_n, q) \in \delta$ if and only if

1. $h(q) = a$.
2. The production $q \longrightarrow q_1 \cdots q_n$ is in P.

The accepting computation trees of M are labeled with pairs (a, q) in $\Sigma \times \Sigma'$ where $h(q) = a$. If we drop the first, Σ-labeled components from the accepting computation trees of M, the results are precisely the syntax trees of G. Hence, the trees recognizable by M are precisely the h-images of G's syntax trees. $\Box$

Theorem E *For each tree-regular tree language there is a uniquely defined minimal tree-local superset.*

Theorem F *Tree-local tree languages do not form a Boolean algebra.*

7 Characterizable tree languages

These results can be extended to hedge languages in a straightforward manner.

Definition 7.1 An algebra $\mathcal{A} = (A, S)$ consists of a set A and a set of functions S. Each function s in S has a rank $r(f)$ in $\mathbf{N}_0$ so that $s : A^{r(s)} \longrightarrow A$.

Definition 7.2 Let $\mathcal{A} = (A, S)$ be an algebra. An equivalence relation $\sim$ on A is called an S-congruence if and only if any s in S and $a_1, \ldots, a_{r(s)}, b_1, \ldots, b_{r(s)}$ in A satisfy the following condition: If $a_i \sim b_i$, $1 \leq i \leq r(s)$, then $s(a_1, \ldots, a_{r(s)}) \sim s(b_1, \ldots, b_{r(s)})$; that is, an $\mathcal{A}$-congruence is an equivalence relation on A that is compatible with the functions in S.

Definition 7.3 An equivalence relation is of finite index if it has a finite number of equivalence classes.

Definition 7.4 Let $\mathcal{A} = (A, S)$ be an algebra. A subset L of A is S-characterizable if and only if there is a finite S-congruence on A such that L is the union of some of its equivalence classes.

Proposition 7.1 *Let $\mathcal{A} = (A, S)$ be an algebra. For any finite number of S-characterizable subsets of A there is a single S-congruence that characterizes all of them simultaneously.*

Proposition 7.2 *Let $\mathcal{A} = (A, S)$ be an algebra. The subsets of A that are S-characterizable form a Boolean algebra.*

Definition 7.5 For a in Σ let the string functions p_a and s_a be defined as follows: $p_a : \Sigma^* \longrightarrow \Sigma^*$, $p_a(w) = aw$, $s_a : \Sigma^* \longrightarrow \Sigma^*$, $s_a(w) = wa$. Furthermore, let $\cdot$ denote string catenation.

The following well-known characterization for string languages is commonly attributed to Myhill and Nerode:

Proposition 7.3 *Let S be any of the following three sets of functions: $\{\cdot\}$, $\{p_a | a \in \Sigma\}$ and $\{s_a | a \in \Sigma\}$. Then, a subset L of Σ^* is S-characterizable if and only if L is string regular.*

A $\{p_a | a \in \Sigma\}$-congruence is called a left congruence and a $\{s_a | a \in \Sigma\}$-congruence is called a right congruence on Σ^*.

Definition 7.6 For trees t and t', where $t' = a(t_1 \cdots t_n)$, we define $t \downarrow t'$ as the tree $a(t, t_1 \cdots t_n)$, inserting t as the first child of t'.

A theorem on trees in the spirit of Myhill and Nerode.

Theorem G *A set of trees T is tree regular if and only if T is $\downarrow$-characterizable.*

Proof In the first part of the proof let T be a tree-regular set of trees and let $M = (Q, \delta, F)$ be a tree automaton that recognizes T. We construct an equivalence relation $\approx$ that $\downarrow$-characterizes T.

The regularity condition for the transition relation of M ensures that, for each a in Σ and q in Q, the set

$$\{w \text{ in } Q^* | (a, w, q) \in \delta\}$$

is string regular. Since there are only finitely many such sets, we can find a single $\{p_a | a \in \Sigma\}$-congruence $\sim$ on Σ that characterizes them all simultaneously.

A **necklace** of a tree t is an equivalence class $[w]_\sim$, $w \in Q^*$, such that M, when operating on t, can generate w as the sequence of states for the children of t's root.

We define trees t_1 and t_2 to be equivalent ($t_1 \approx t_2$) if and only if t_1 and t_2 have identical root labels and identical sets of necklaces.

First of all, $\approx$ is an equivalence relation on the set of trees and it has finite index.

Furthermore, if $t_1 \approx t_2$ and $t_1 \in T$, then $t_2 \in T$. The reason is that the sets of states that M obtains for the roots of t_1 and t_2, when operating on t_1 and t_2, respectively, only depend on the root labels and on the sets of necklaces.

It remains to demonstrate that $\approx$ is a $\downarrow$-congruence. Let $t_1 \approx t_2$ and $t_3 \approx t_4$. We can see immediately that $t_1 \downarrow t_3$ and $t_2 \downarrow t_4$ have identical root labels. Furthermore, the necklaces of $t_1 \downarrow t_3$ are precisely the necklaces $[p_q(w)]_\sim$, where q is a state that M obtains for t_1's root, when M is operating on t_1 and $[w]_\sim$ is a necklace of t_3. The analogous argument holds for t_2 and t_4, so the sets of necklaces for $t_1 \downarrow t_3$ and $t_2 \downarrow t_4$ are the same.

Thus, we have demonstrated that $\approx$ $\downarrow$-characterizes T.

For the second proof step, let $\approx$ be an equivalence relation that $\downarrow$-characterizes the set T of trees. We construct a tree automaton $M = (Q, \delta, F)$ that recognizes T.

First, let $Q = \{[t]_\approx | t \text{ is a tree}\}$. Since $\approx$ has finite index, the state set Q is finite.

Second, let $F = \{[t]_\approx | t \in T\}$. Since $T = \bigcup\{[t]_\approx | t \in T\}$, if $t_1 \approx t_2$ and $t_1 \in T$, then $t_2 \in T$.

Finally, let

$$\delta = \{(a, [t_1]_\approx \cdots [t_n]_\approx, [a(t_1 \cdots t_n)]_\approx)\}.$$

Since $a(t_1 \cdots t_n) = t_1 \downarrow (\ldots (t_{n-1} \downarrow (t_n \downarrow a)) \cdots))$ and since $\approx$ is a $\downarrow$-congruence, the transition relation δ is well defined.

The tree automaton $M = (Q, \delta, F)$ is deterministic. When operating on the tree t, the automaton M yields $[t]_\approx$ as the state for t's root. Hence, M recognizes t.

We finally demonstrate the regularity of M's transition relation δ. Choose, for each state q in Q, a representative tree t_q in q such that $q = [t_q]_\approx$. For each a in Σ and each tree t, we demonstrate that the set

$$\{q_1 \cdots q_n | a(t_{q_1} \cdots t_{q_n}) \approx t\}$$

is string regular over Q, using the Myhill–Nerode theorem for string languages. We define the equivalence relation $\sim$ over Q-strings as follows:

$$q_1 \cdots q_m \sim q'_1 \cdots q'_n$$

if and only if

$$a(t_{q_1}\cdots t_{q_m}) \approx a(t_{q'_1}\cdots t_{q'_n}).$$

Since $\approx$ is of finite index, so is $\sim$.

Next we demonstrate that $\sim$ is a p_q-congruence for each q in Q. Let $q_1\cdots q_m \sim q'_1\cdots q'_n$. Then,

$$a(t_{q_1}\cdots t_{q_m}) \approx a(t_{q'_1}\cdots t_{q'_n})$$

and, since $\approx$ is a $\downarrow$-congruence,

$$\begin{aligned} a(t_q, t_{q_1}\cdots t_{q_m}) &= t_q \downarrow a(t_{q_1}\cdots t_{q_m}) \\ &\approx t_q \downarrow a(t_{q'_1}\cdots t_{q'_n}) \\ &= a(t_q, t_{q'_1}\cdots t_{q'_n}). \end{aligned}$$

Hence,

$$p_q(q_1\cdots q_m) = qq_1\ldots q_m \sim qq'_1\cdots q'_n = p_q(q'_1\cdots q'_n).$$

Finally, we demonstrate that $\sim$ characterizes the set

$$\{q_1\cdots q_n | a(t_{q_1}\cdots t_{q_n})\}.$$

It is, however, obvious since, for each $q_1\cdots q_m$ that belongs to this set and for each $q'_1\cdots q'_n$ that is $\sim$-equivalent to $q_1\cdots q_m$, the string $q'_1\cdots q'_n$ belongs to this set as well. Hence,

$$\{q_1\cdots q_n | a(t_{q_1}\cdots t_{q_n})\}$$

is the union of $\sim$-equivalence classes. □

Theorem H *The tree-regular tree languages form a Boolean algebra.*

8 Top congruences

These results can be extended to hedges in a straightforward manner.

Definition 8.1 A **pointed tree** (sometimes also called a tree with a handle or a handled tree) is a tree over an extended alphabet $\Sigma \cup \{X\}$ so that precisely one node is labeled with the variable X and that node is a leaf.

Definition 8.2 If t is a pointed tree and t' is a (pointed or nonpointed) tree, we can catenate t and t' by replacing the node labeled X in t with the root of t'. The result is the (pointed or nonpointed) tree tt'.

Definition 8.3 Let T be a tree language. Trees t_1 and t_2 are **top congruent** with respect to T ($t_1 \sim_T t_2$ or simply $t_1 \sim t_2$) if and only if, for each pointed tree t, the following condition holds:

$tt_1 \in T$ if and only if $tt_2 \in T$.

The tree functions p_t that map each tree t' to $p_t(t') = tt'$ are, for pointed trees t, the tree analog of the string functions p_a, $a \in \Sigma$. The top congruence for trees is the tree analog of the left congruence for strings.

Lemma 8.1 *The top congruence is an equivalence relation on trees; it is a congruence with respect to catenations of trees with pointed or nonpointed trees.*

PROOF To verify the congruence condition, let t be a pointed tree. We have to demonstrate that $t_1 \sim_T t_2$ implies that $tt_1 \sim_T tt_2$. Since tree catenation is associative, for any trees t_1 and t_2 such that $t_1 \sim_T t_2$ and for any pointed tree t',

$$t'(tt_1) = (t't)t_1 \sim_T (t't)t_2 = t'(tt_2).$$

Hence, $tt_1 \sim_T tt_2$. □

Definition 8.4 The **top index** of a tree language T is the number of $\sim_T$-equivalence classes.

Lemma 8.2 *Each tree-regular tree language has finite top index.*

PROOF Let T be the tree-regular language of a DBTA M. For any pair t_1 and t_2 of trees, if the automaton M when operating on t_1 yields the same states in a halting configuration as it does when operating on t_2, then $t_1 \sim_T t_2$. Since M's state set is finite, the equivalence relation $\sim_T$ is of finite index. □

A string language is regular if and only if it has finite index; however, that a tree language has finite top index is insufficient for it to be regular.

Example Consider the tree language

$$L = \{a(b^i c^i) : i \geq 1\}.$$

Clearly, L has finite top index, but it is not regular.

A second condition, regularity of **local views,** must also be satisfied.

Definition 8.5 Let T be a tree language, a be a symbol in Σ, let t be a pointed tree and T_f be a finite set of trees. Then the local view of T with respect to t, a, and T_f is the string language

$$V_{t,a,T_f}(T) = \{t_1 \cdots t_n \in T_f^* | ta(t_1 \cdots t_n) \in T\}$$

over the alphabet T_f. For the purposes of local views we treat the trees in the finite set T_f as symbols in the alphabet T_f; the trees in T_f are primitive entities that can be catenated to give strings over T_f. Note that we are not catenating trees.

Because one argument that we use in the proof of Lemma 8.4 occurs several times in the paper, we factor it out as Lemma 8.3.

Lemma 8.3 *Let Σ and Σ' be finite alphabets, F be a subset of $\Sigma \times \Sigma'$ and B be a string-regular language over Σ'. Then, the string language*

$$\{a_1 \cdots a_n \mid (a_1, b_1), \ldots, (a_n, b_n) \in F \text{ and } b_1 \cdots b_n \in B\}$$

is regular.

Proof Using projection to the second component as a morphism, we see that the set

$$A = \{(a_1, b_1) \cdots (a_n, b_n) \mid (a_1, b_1), \ldots, (a_n, b_n) \in F \text{ and } b_1 \cdots b_n \in B\}$$

is the inverse morphic image of the regular language B; hence, A is regular.

Using projection to the first component of A as a morphism, we see that the required language which is the morphic image of A is also regular. □

We continue now with the proof of Lemma 8.4.

Lemma 8.4 *All local views of each tree-regular tree language are regular string languages.*

Proof Let T be the tree language of an NBTA $M = (Q, \delta, F)$, a be a symbol in Σ, t be a pointed tree and T_f be a finite set of trees.

For each q in Q, define $M_q = (Q, \delta_q, F)$ such that $\delta_q = \delta \cup \{(X, \lambda, q)\}$; that is, M_q is identical to M with the exception of one additional tuple in M_q's transition relation. The automaton M_q operates on $\Sigma \uplus \{X\}$-labeled trees.

A computation of M on a tree tt' can be divided into two parts: a computation of M on the tree t' resulting in some state q at the root of t', and a computation of M_q on the pointed tree t. The automaton M recognizes the tree tt' if and only if M, when operating on t', makes a sequence of transitions from any starting configuration to a halting configuration such that t''s root label is q and if M_q recognizes t.

For any t' in T_f, let $Q_{t'}$ be the set of states q in Q so that M, when operating on t', makes a sequence of transitions from any starting configuration to a halting configuration such that the root label of t' is q. Lemma 8.3 and the regularity of M's transition relation imply that the local view V_{t,a,T_f}, rewritten as

$$\{t_1 \cdots t_n \mid \text{for some } q_i \in Q_{t_i} \ (a, q_1 \cdots q_n, q) \in \delta \text{ and } M_q \text{ recognizes } t\},$$

is string regular. □

Example Let

$$T = \{c(t_1 \cdots t_n) | \text{label}(\text{root}(t_1)) \cdots \text{label}(\text{root}(t_n)) \in \{a^l b^l | l \geq 1\}\}.$$

The tree language T is has top index four. Two of its equivalence classes are the sets of trees whose root labels are a or b; the other two are T and the set of trees that are not in T, but have the root label c. The local view of T with respect to the pointed tree $X()$, symbol c, and the finite set of trees $\{a(), b()\}$ is the nonregular set of strings $\{a^l b^l | l \geq 1\}$. Hence, T has finite top index but it is not tree regular.

Theorem I *A tree language is tree regular if and only if it has finite top index and all its local views are regular string languages.*

Proof Lemmas 8.2 and 8.4 establish the two if clauses; thus, we need prove only the only-if clause. Let T be a tree language of finite top index such that all its local views are regular string languages. Let $\sim$ denote the top congruence with respect to T and let $[t]$ denote the equivalence class of a tree t with respect to the top congruence. that recognizes T. We now construct an NBTA $M = (Q, \delta, F)$ for T.

The first part of the proof is a straightforward adaption of its string-language counterpart. The second part deals with the regularity of the transition relation.

Let $Q = \{[t] | t \text{ is a tree}\}$ and $F = \{[t] | t \in T\}$. Note that $t_1 \sim t_2$ implies that $t_1 \in T$ if and only if $t_2 \in T$; thus, $[t]$ consists of either only trees in T or only trees not in T. Finally, let δ consist of the triples

$$(a, [t_1] \cdots [t_n], [a(t_1 \cdots t_n]).$$

Observe that δ is well defined; that is, if $t_1 \sim t'_1 \cdots t_n \sim t'_n$, then $a(t_1 \cdots t_n) \sim a(t'_1 \cdots t'_n)$. The argument is incremental. Let t be a pointed tree; then,

$$\begin{aligned}
ta(t_1 \cdots t_n) \in T &\Leftrightarrow ta(X \cdot t_2 \cdots t_n)t_1 \in T \\
&\Leftrightarrow ta(X \cdot t_2 \cdots t_n)t'_1 \in T \\
&\Leftrightarrow ta(t'_1 \cdot t_2 \cdots t_n) \in T \\
&\Leftrightarrow ta(t'_1 \cdot X \cdot t_3 \cdots t_n)t_2 \in T \\
&\Leftrightarrow ta(t'_1 \cdot X \cdot t_3 \cdots t_n)t'_2 \in T \\
&\Leftrightarrow ta(t'_1 \cdot t'_2 \cdot t_3 \cdots t_n) \in T \\
&\Leftrightarrow \vdots \\
&\Leftrightarrow ta(t'_1 \cdots t'_n) \in T.
\end{aligned}$$

Hence, $a(t_1, \cdots, t_n) \sim a(t'_1, \cdots, t'_n)$. Furthermore, M is deterministic.

Finally, M recognizes T, since M, when operating on a tree t from the starting configuration, reaches exactly one halting configuration, which corresponds to state $[t]$. Hence, M recognizes t if and only if $[t] \in F$; that is, if $t \in F$.

The second task is to demonstrate the regularity of M's transition relation δ. For each q in Q, let t_q in q be a representative of q; that is, $[t_q] = q$. Furthermore, let $T_f = \{t_q | q \in Q\}$. Finally, let $\hat{T}$ be a finite set of pointed trees that discriminates between the finitely many $\sim$-equivalence classes. More precisely, $t_1 \sim t_2$ if and only if, for each t in $\hat{T}$, the tree tt_1 is in T if and only if tt_2 is in T.

Now, $(a, q_1 \cdots q_n, q) \in \delta$ if and only if $a(t_{q_1} \cdots t_{q_n}) \sim t_q$; that is, if and only if, for each t in $\hat{T}$, the tree $ta(t_{q_1} \cdots t_{q_n})$ being in T is equivalent to the tree tt_q being in T. Hence, $(a, q_1 \cdots q_n, q) \in \delta$ if and only if $q_1 \cdots q_n$ is in the intersection of the sets $V_{t,a,T_f}(T)$ such that $t \in \hat{T}$ and $tt_q \in T$, and of the complements of the sets $V_{t,a,T_f}(T)$ such that $t \in \hat{T}$ and $tt_q \notin T$. This intersection, however, is a string-regular set, which establishes the regularity of M's transition relation using string morphism. $\square$

At first glance it may appear that the local-view condition for tree-regular tree languages is a condition on infinitely many trees. But, if we exchange a tree t_1 in a finite set T_f by an equivalent—with respect to top congruence—tree t_2, then $V_{a,t,(T_f \setminus \{t_1\}) \cup \{t_2\}}(T)$ is the morphic image of $V_{a,t,T_f}(T)$ under a string isomorphism. Hence, if T has finite top index, we need to check the local-view condition for only a finite number of tree sets T_f.

9 Languages recognizable by two-way tree automata

These results can be extended to two-way hedge automata in a straightforward manner.

We establish the following theorem.

Theorem J *The tree language of every nondeterministic two-way tree automaton is tree regular.*

Lemma 9.1 *The tree languages of all nondeterministic two-way tree automata have finite top index.*

PROOF Let $M = (Q, \delta, F)$ be a 2NTA that recognizes the tree language T. We demonstrate that T is of finite top index and that all its local views are string regular.

For each tree t let Q_t be the union of the two sets

$$\{q \in Q \mid c_1 \longrightarrow c_2,\ c_1 \text{ a starting configuration for } M \text{ on } t,\ c_2 \text{ a halting configuration for } M \text{ on } t,\ c_2(\mathrm{root}(t)) = q\}$$

and

$$\{(q_1, q_2) \in Q \times Q \mid c_1 \longrightarrow c_2,\ c_1 \text{ and } c_2 \text{ halting configurations for } M \text{ on } t,\ c_1(\mathrm{root}(t)) = q_1,\ c_2(\mathrm{root}(t)) = q_1\}.$$

Clearly the set $\{Q_t | t$ is a tree is finite. Furthermore, if $Q_{t_1} = Q_{t_2}$, then $t_1 \sim_T t_2$. Hence, T has finite top index. $\Box$

Lemma 9.2 *The tree languages of all nondeterministic two-way tree automata have only regular local views.*

PROOF Let t be a pointed tree, a be a symbol in Σ and T_f be a finite set of trees.

We demonstrate that the local view $V_{t,a,T_f}(T)$ of T with respect to t, a, and T_f, namely, the string language

$$\{t_1 \cdots t_n \in T_f^* | ta(t_1 \cdots t_n) \in T\}$$

is string regular. The proof is in three steps.

The first step is to recognize that V_{t,a,T_f} is a finite union of finite intersections of the following sets X_p and X_{pq}, $p, q \in Q$:

$$\begin{array}{rcl} X_p & = & \{t_1 \cdots t_n \in T_f^* | \\ & & \quad c_1 \longrightarrow c_2, \\ & & \quad c_1 \text{ is starting configuration of } M \text{ on } a(t_1 \cdots t_n), \\ & & \quad c_2 \text{ is halting configuration of } M \text{ on } a(t_1 \cdots t_n), \\ & & \quad c_2(\text{root}(a(t_1 \cdots t_n))) = p \text{ and} \\ & & \quad \text{there is no other halting configuration in the computation } c_1 \longrightarrow c_2\} \end{array}$$

and

$$\begin{array}{rcl} X_{p,q} & = & \{t_1 \cdots t_n \in T_f^* | \\ & & \quad c_1 \longrightarrow c_2, \\ & & \quad c_1 \text{ and } c_2 \text{ are halting configuration of } M \text{ on } a(t_1 \cdots t_n), \\ & & \quad c_1(\text{root}(a(t_1 \cdots t_n))) = p, \\ & & \quad c_2(\text{root}(a(t_1 \cdots t_n))) = q \text{ and} \\ & & \quad \text{there is no other halting configuration in the computation } c_1 \longrightarrow c_2\} \end{array}$$

Any computation on $ta(t_1 \cdots t_n)$ from a starting configuration to a halting configuration can be partitioned into those parts that concern only t and those parts that concern only $a(t_1 \cdots t_n)$. The parts that concern only $a(t_1 \cdots t_n)$ form a computation from a starting configuration to a halting configuration, followed by a number of computations from halting configurations to halting configurations.

Hence, the $a(t_1 \cdots t_n)$-related parts of any accepting computation of M on $ta(t_1 \cdots t_n)$ first go from a starting configuration to a halting configuration, having M in some state p at $a(t_1 \cdots t_n)$'s root, and then from halting configuration to halting configuration, leading M from some state p_i to some state q_i on $a(t_1 \cdots t_n)$'s root until M finally leaves $a(t_1 \cdots t_n)$

and does not return. This implies that $t_1 \cdots t_n$ is in $X_p \cap X_{p_1 q_1} \cap \cdots \cap X_{p_r q_r}$. The state sequence $p, p_1, q_1 \cdots p_r, q_r$ documents the behavior of M at the root of $a(t_1 \cdots t_n)$ during an accepting computation of M on the complete tree $ta(t_1 \cdots t_n)$.

For any other sequence of trees $t'_1 \cdots t'_m$ in $X_p \cap X_{p_1 q_1} \cap \cdots \cap X_{p_r q_r}$, we can construct an accepting computation of M on $ta(t'_1 \cdots t'_m)$ by patching the $a(t_1 \cdots t_n)$-related parts of the original computation with computations on $a(t'_1 \cdots t'_m)$ that have the same state-behavior at the root as $a(t_1 \cdots t_n)$ had. Since $t'_1 \cdots t'_m$ is in $X_p \cap X_{p_1 q_1} \cap \cdots \cap X_{p_r q_r}$, we can find such patches.

We conclude that the whole set $X_p \cap X_{p_1 q_1} \cap \cdots \cap X_{p_r q_r}$ is a subset of V_{t,a,T_f}.

Since there are only finitely many sets X_p and X_{pq}, the set V_{t,a,T_f} is a finite union of finite intersections of these.

The next two steps establish that X_p and X_{pq} are regular string languages.

First, a string $t_1 \cdots t_n$ is in X_p if and only if there are $p_1, \ldots, p_n$ in Q such that M, when operating on t_i beginning in a starting configuration, eventually reaches a halting configuration c such that $c(\mathit{root}(t_i)) = p_i$ and $(a, p_1 \cdots p_n, p, u) \in \delta$. Lemma 8.3 and the regularity of the transition relation δ imply that X_p is a regular string language.

Second, let X^s_{pq} be the subset of X_{pq} in which the computation $c_1 \longrightarrow c_2$ (compare the definition of X_{pq}) makes just one computation step and let X^m_{pq} be the subset of X_{pq} in which the computation $c_1 \longrightarrow c_2$ makes more than one computation step. Then, X_{pq} is the (not necessarily disjoint) union of X^s_{pq} and X^m_{pq}. We demonstrate that both subsets are string regular.

First, note that

$$X^s_{pq} = \{t_1 \cdots t_n \in T^*_f \mid (a, p, q, s) \in \delta\}.$$

Hence, X^s_{pq} depends only on a and is either empty or T^*_f. In both cases, X^s_{pq} is regular. Second, $t_1 \cdots t_n \in X^m_{pq}$ if and only if there are $p_1, \ldots, p_n, q_1, \ldots, q_n$ in Q such that $(a, p_1 \cdots p_n, p, d)$ is in δ and M, when operating on t_i, makes a computation from a halting configuration with root label p_i to a halting configuration with root label q_i and $(a, q_1 \cdots q_n, q, u) \in \delta$.

Lemma 8.3 and the regularity of the transition relation δ imply that X^m_{pq} is a regular string language. □

10 Hedge and tree regular expressions

THIS SECTION IS NOT EVEN STARTED, LET ALONE FINISHED. We will follow the ideas of Pair and Quere [PQ68], Takahashi [Tak75] and of Murata [Mur01].

11 Tree automata with endmarkers

These results can be extended to hedge automata in a straightforward manner.

We introduce two new symbols, $\top$ and $\bot$, that serve as endmarkers on trees. Let Σ^e be the disjoint union of Σ and $\{\top, \bot\}$.

Definition 11.1 A Σ^e-labeled tree is **endmarked** if and only if

1. Its root is labeled $\top$,
2. Its leaves are labeled $\bot$,
3. Its root is unary,
4. Its leaves have no siblings,
5. Each node that is not the root and not a leaf is labeled with a symbol in Σ, and
6. There is at least one Σ-labeled node.

Obviously, there is a one-to-one correspondence between Σ-labeled trees and endmarked trees. If we add to the Σ-labeled tree t a new root labeled $\top$ and to each leaf a new child labeled $\bot$, the result is the endmarked version t^e. For each tree language T we define the endmarked version $T^e = \{t^e | t \in T\}$.

Theorem K *If a tree language T over the extended alphabet Σ^e is tree regular, then so is its endmarked subset.*

PROOF Given a tree automaton $M = (Q, \delta, F)$ that recognizes the tree language T over the extended alphabet Σ^e, we define a tree automaton $M' = (Q', \delta', F')$ that recognizes precisely the endmarked trees in T.

We construct δ' from δ as follows:

First, we remove all transitions on the symbols $\top$ and $\bot$ and all transitions on leaves from δ; that is all transitions of the form (a, λ, q) whose second component is the empty string. We will add transitions later for $\bot$-labeled leaves or for $\top$-labeled nodes, the latter being transitions into a state that prevents further up moves. Therefore, any tree recognizable by M' has $\bot$-labeled leaves and a $\top$-labeled root; all other nodes are Σ-labeled.

Next, we now replace all transitions on $\bot$-labeled leaves, but we mark the target states so that further transitions can ensure that leaves have no siblings and that parents of leaves are Σ-labeled.

Finally, for each transition of the form $(\top, q, q') \in \delta$, where $q' \in F$, we add a transition $(\top, q, q_f)$, where q_f is a new and final state in Q'. Thus, we ensure that the roots of M'-recognizable trees are unary and $\top$-labeled.

More precisely, let Q' be the disjoint union of Q, $\{q_f\}$ and $Q \times \{q_s\}$, and let $F' = \{q_f\}$. Furthermore, define δ' to be

$$\begin{array}{rcl} \delta \cap (\Sigma \times Q^* \times Q) & \setminus & \{(a, \lambda, q) | a \in \Sigma^*,\ q \in Q\} \\ & \cup & \{(\bot, \lambda, (q, q_s)) | (\bot, \lambda, q) \in \delta\} \\ & \cup & \{(a, (q, q_s), q') | (a, q, q') \in \delta\} \\ & \cup & \{(\top, q, q_f) | (\top, q, q') \in \delta \text{ and } q' \in F\}. \end{array}$$

□

Theorem L *Let T be a tree language over the alphabet Σ. Then, T is tree regular if and only if T^e is tree regular.*

PROOF First, let T be tree regular and the tree automaton $M = (Q, \delta, F)$ recognize T. We define a tree automaton $M' = (Q', \delta', F')$ such that the endmarked trees that M' recognizes are precisely the trees in T^e.

Let Q' be the disjoint union of Q, $\{q_s\}$ and $F' = F$. Furthermore, let

$$\begin{array}{rcl} \delta' = \delta & \cup & \{(\bot, \lambda, q_s)\} \cup \{(a, q_s, q) | (a, \lambda, q) \in \delta\} \\ & \cup & \{(\top, q, q) | q \in Q\}. \end{array}$$

Second, let T^e be tree regular and the tree automaton $M = (Q, \delta, F)$ recognize T^e. We define a tree automaton $M' = (Q', \delta', F')$ such that M' recognizes T.

Let Q' be the disjoint union of Q, $\{q_f\}$ and $F' = \{q_f\}$. Furthermore, let

$$\begin{array}{rcl} \delta' = \delta & \cap & (\Sigma \times Q^* \times Q) \\ & \cup & \{(a, \lambda, q) | (\bot, \lambda, q') \in \delta \text{ and } (a, q', q) \in \delta\} \\ & \cup & \{(a, w, q_f) | (a, w, q) \in \delta \text{ and } (\top, q, q') \in \delta \text{ and } q' \in F\}. \end{array}$$

□

12 Previous work

Work on tree automata and tree-regular languages can be divided into two categories, one dealing with ranked and the other with unranked alphabets.

The bulk of the literature belongs to the first category. Gécseg and Steinby [GS84] have written a comprehensive book on tree automata and tree transducers over ranked alphabets. In this standard reference trees are algebraic expressions and tree automata are algebras. The Handbook of Formal Languages [RS97] provides an updated and more concise version by the same authors.

Tree-regular languages over unranked alphabets seem to have been investigated first by Thatcher [Tha67a]. Two of Thatcher's papers [Tha67a, TW65] state a number of results on tree automata that carry over directly from the theory of string automata. Neither of the two papers provides the proofs, though, which seem to be included in a technical report [Tha67b]. On the other hand, tree automata were in the air in the mid to late 1960's; thus, other researchers had similar ideas. In particular, Arbib and Giveón [Ao68], Brainerd [Bra69], Doner [Don70], and Pair and Quere [PQ68].

The characterization of tree-regular languages as morphic images of tree-local sets essentially goes back to Thatcher [Tha67a] and was rediscovered by [Tak75].

Thatcher [Tha67a] indicates that results on semigroup automata, congruences, minimality, and decomposition also carry through from string languages to tree languages. This is, in 1969, cited as ongoing work; we have not been able to find any publications of this work, though.

Thatcher [Tha70] provides a light introduction to the theory of tree automata and tree-regular languages. This work indicates how the theory is applicable to context-free language problems, for example to structural equivalence. It also addresses tree transformations. The paper starts out with tree languages over unranked alphabets, but quickly restricts itself to the ranked-alphabet case. In addition, Thatcher's survey also provides a useful introduction to the area [Tha87].

Murata [Mur95] surveys definitions and results on tree-regular languages. His terminology goes back to Thatcher [Tha67a]. Murata extends his survey to hedge languages since he is interested in tree transformations and hedges often are the natural results of them. Murata covers tree automata, tree expressions, and regular tree grammars.

Barrero [Bar91] argues that in application areas such as pattern recognition and grammatical inference it is sensible to consider unranked alphabets. He developes a theory of tree languages over unranked alphabets. Although familiar with Thatcher's work, Barrero demands that for tree automata the number of transitions be finite. In this framework, tree languages are not closed under complement. Barrero also addresses tree grammars.

Moriya [Mor94] demonstrates that two-way tree automata are equipotent to one-way tree automata. The alphabet of tree labels is not ranked; however, like in Barrero's work [Bar91], the transition relation is finite. Moriya's proof is based on crossing sequences, as is the corresponding proof for string automata. Moriya also addresses one-way and two-way pushdown tree automata; for this type of automata, two-way ones are more powerfull.

Takahashi [Tak75] establishes a uniform mathematical framework to study regular sets of strings, trees, and hedges, covering both ranked and unranked alphabets. She characterizes string-, tree-, and hedge-regular sets as morphic images of local sets and as congruence-characterizable sets. Takahashi applies the theory of hedge-regular languages to the structural study of context-free languages.

13 Concluding remarks

THIS SECTION IS NOT EVEN STARTED, LET ALONE FINISHED.

References

[Ao68] M.A. Arbib and Y. Giveón. Algebraic automata I: Parallel programming as a prolegomena to the categorical approach. *Information and Control*, 12:331–345, 1968.

[Bar91] A. Barrero. Unranked tree languages. *Pattern Recognition*, 24(1):9–18, 1991.

[BKW98] A. Brüggemann-Klein and D. Wood. One-unambiguous regular languages. *Information and Computation*, 142(2):182–206, May 1998.

[BKW00a] A. Brüggemann-Klein and D. Wood. Caterpillars: A context specification technique. *Markup Languages*, 2(1):81–106, 2000.

[BKW00b] A. Brüggemann-Klein and D. Wood. The regularity of two-way nondeterministic tree automata languages. Technical Report HKUST-TCSC-2000-07, HKUST, Theoretical Computer Science Center, Computer Science Department, Hong Kong SAR, 2000. URL: http://www.cs.ust.hk/tcsc/RR/index_3.html.

[BPMM98] T. Bray, J. P. Paoh, and C. M.Sperberg-McQueen. Extensible markup language (XML) 1.0. http://www.w3.org/TR/1998/REC-xml-19980210/, February 1998.

[Bra69] W.S. Brainerd. Tree generating regular systems. *Information and Control*, 14:217–231, 1969.

[CDG+98] H. Comon, M. Daucher, R. Gilleron, S. Tison, and M. Tommasi. Tree automata techniques and applications, 1998. Available on the Web from l3ux02.univ-lille3.fr in directory tata.

[Cou78] B. Courcelle. A representation of trees by languages. *Theoretical Computer Science*, 7:25–55, 1978.

[Cou89] B. Courcelle. On recognizable sets and tree automata. In H. Aït-Kaci and M. Nivat, editor, *Resolution of Equations in Algebraic Structures*, volume 1, pages 93–126. Academic Press, San Diego, CA, 1989.

[Don70] J. Doner. Tree acceptors and some of their applications. *Journal of Computer and System Sciences*, 4(5):406–451, 1970.

[GS84] F. Gécseg and M. Steinby. *Tree Automata*. Akadémiai Kiadó, Budapest, 1984.

[ISO86] ISO 8879: Information processing—Text and office systems—Standard Generalized Markup Language (SGML), October 1986. International Organization for Standardization.

[KW99] P. Kilpeläinen and D. Wood. SGML and XML document grammars and exceptions. Technical Report HKUST-TCSC-1999-01, HKUST, Theoretical Computer Science Center, Computer Science Department, Hong Kong SAR, 1999. URL: http://www.cs.ust.hk/tcsc/RR/index_2.html.

[Mor94] E. Moriya. On two-way tree automata. *Information Processing Letters*, 50:117–121, 1994.

[Mur95] M. Murata. Forest-regular languages and tree-regular languages. Unpublished manuscript, 1995.

[Mur97] M. Murata. Transformation of documents and schemas by patterns and contextual conditions. In C. Nicholas and D. Wood, editors, *Proceedings of the Third International Workshop on Principles of Document Processing (PODP 96)*, pages 153–169, Heidelberg, 1997. Springer-Verlag. Lecture Notes in Computer Science 1293.

[Mur98] M. Murata. Data model for document tranformation and assembly. In E.V. Munson, C. Nicholas, and D.Wood, editors, *Proceedings of the Fourth International Workshop on Principles of Digital Document Processing (PODDP 98)*, pages 140–152, Heidelberg, Germany, 1998. Springer-Verlag. Lecture Notes in Computer Science 1481.

[Mur01] M. Murata. Extended path expressions for xml, 2001. Unpublished manuscript.

[PQ68] C. Pair and A. Quere. Définition et etude des bilangages réguliers. *Information and Control*, 13:565–593, 1968.

[RS97] G. Rozenberg and A. Salomaa, editors. *Handbook of Formal Languages, Volume 3, Beyond Words*. Springer-Verlag, Berlin, Heidelberg, New York, 1997.

[Tak75] M. Takahashi. Generalization of regular sets and their application to a study of context-free languages. *Information and Control*, 27(1):1–36, January 1975.

[Tha67a] J. W. Thatcher. Characterizing derivation trees of context-free grammars through a generalization of finite automata theory. *Journal of Computer and System Sciences*, 1:317–322, 1967.

[Tha67b] J. W. Thatcher. A further generalization of finite automata. Technical Report RC 1846, IBM Thomas J. Watson Research Center, Yorktown Heights, New York, 1967.

[Tha70] J. W. Thatcher. There's a lot more to finite automata theory than you would have thought. Technical Report RC 2852 (#13407), IBM Thomas J. Watson

Research Center, Yorktown Heights, New York, 1970.

[Tha87] J.W. Thatcher. Tree automata: An informal survey. In A.V. Aho, editor, *Currents in the Theory of Computing*, pages 143–172. Prentice-Hall, Engle-Wood Cliffs, NJ, 1987.

[TW65] J. W. Thatcher and J. B. Wright. Abstract 65T-469. *Notices of the American Mathematical Society*, 12:820, 1965.

[TW68] J.W. Thatcher and J.B. Wright. Generalized finite automata theory with an application to a decision problem of second-order logic. *Mathematical Systems Theory*, 2(1):57–81, 1968.

[Woo95] D. Wood. Standard Generalized Markup Language: Mathematical and philosophical issues. In J. van Leeuwen, editor, *Computer Science Today*, pages 344–365. Springer-Verlag, Heidelberg, 1995. Lecture Notes in Computer Science 1000.