Making Finite-State Methods Applicable to Languages Beyond Context-Freeness via Multi-dimensional Trees

Anna Kasprzik

University of Trier

Abstract. We provide a new term-like representation for multi-dimensional trees as defined by Rogers [8,9] which establishes them as a direct generalization of classical trees. As a consequence these structures can be used as input for finite-state applications based on classical tree language theory. Via the correspondence between string and tree languages these applications can then be conceived to be able to process even some language classes beyond context-freeness.

Key words: Finite-state methods, Multi-dimensional Trees, Regularization, Tree Adjoining Grammar

1 Introduction

It is well known that string languages that are recognizable by finite-state automata, the so-called regular languages, have a whole range of advantageous mathematical properties. However, due to the relatively restricted character of the latter, language classes that lie beyond regularity are often more interesting for applications based on formal language theory, even if the devices processing these languages (e.g., grammars or automata) are significantly more complex. Obviously, it would be of considerable use if one could reunite the advantages of regular and less restricted language classes by finding a way to handle these processes via regular mechanisms without giving up any of the expressive power.

Several such regularization methods have indeed been formulated, and at least two of them have been shown to be of use in the field of linguistics. A very prominent linguistic application of formal language theory is the area of natural language processing, i.e., conceiving the strings formed by a natural language as a formal language in order to treat them automatically. Unfortunately, the study of certain phenomena (e.g., cross-serial dependencies in Dutch or Swiss German) showed that some of these string sets are not context-free. Joshi [6] claimed the least class of formal languages containing all natural language string sets to be situated between the context-free and the context-sensitive languages, and named it the class of *mildly context-sensitive languages*. The string sets generated by the grammar formalism defined by Joshi [6] himself, Tree Adjoining Grammar, prototypically fulfil all the necessary conditions for this class. TAG is

considered the standard model for mild context-sensitivity and is the foundation of a considerable amount of current work in applied computational linguistics.

There are two methods of regularization for TAG (see [14]). Both methods are two-step approaches, i.e., they transform a TAG into a regular device (grammar or automaton) of some sort by representing its components in another shape and then reconstruct the intended objects from the objects generated or licensed by these devices via a simple process that can be carried out with regular means as well. One is based on an algebraic operation called Lifting (see [7]) which could be described as a way to write terms in a form that makes their internal structure more explicit, which, if the term is noted as a tree, has the side effect that all inner nodes are turned into leaves and thus become rewritable by substitution, which is a regular mechanism, and the other method (described by Rogers [8,9]) makes use of an additional dimension in space by representing the components of a TAG as three-dimensional trees which likewise has the consequence that all inner nodes are turned into leaves and can be expanded by substitution.

The theoretical foundation of the second method are *multi-dimensional trees*, which are structures built over tree domains of arbitrarily many dimensions. Just like ordinary two-dimensional trees, every multi-dimensional tree has a string associated with it, which is obtained by reducing the dimensions of the tree step by step to its leaves. The classes of string languages associated with the recognizable multi-dimensional tree languages ordered by number of dimensions form a (proper) infinite hierarchy properly contained in the context-sensitive class, with the classes of finite languages (associated with zero-dimensional point sets), regular languages (one-dimensional string sets), context-free languages (two-dimensional tree sets, as for the classical definition) and the mildly context-sensitive string languages generated by (non-strict) TAGs (three-dimensional tree sets, see [8,9]) as the first four steps. According to Rogers [8], this hierarchy coincides with Weir's Control Language Hierarchy [3].

It follows from these correspondencies that by processing recognizable higher-dimensional descriptions of non-regular string languages instead of the string sets themselves, finite-state methods become applicable again, and with them all the advantages and results pertaining to regularity. This can be a valuable insight in the area of natural language processing, but also in other areas based on formal language theory, e.g., grammatical inference: For instance, just as Angluin's learning algorithm for regular string languages [10] has been adapted to regular tree languages [11,12], thereby making context-free string languages learnable (wrt the underlying learning model), this adapted algorithm can be generalized to recognizable tree languages of arbitrarily many dimensions, making even string languages beyond context-freeness learnable in polynomial time (see [15]).

However, before such applications founded on formal tree languages can be generalized to arbitrarily many dimensions, there is a missing link to be provided: Most of them are not based on tree domains, as is Rogers' definition of multi-dimensional trees, but on the concept of trees as terms over a partitioned alphabet. In this paper we will give a new term-like representation for multi-dimensional trees, along with an adapted definition of finite-state automata for

these structures, and prove the equivalence of the two notations. As a consequence multi-dimensional trees can be seen and used as a direct generalization of classical trees, and the full range of beneficial results for regular (tree) languages as known from formal language theory in the spirit of the Chomsky Hierarchy can be exploited.

2 Preliminaries

2.1 Tree Basics

We presuppose familiarity with classical formal language theory (see for example [1]). We will give some basic notions regarding trees (see for example [2,5]).

A *ranked alphabet* is a finite set of symbols, each associated with a rank $n \in \mathbb{N}$. By Σ_n we denote the set of all symbols in Σ with rank n. Traditionally, every symbol has a single rank, but it is just as possible to admit several ranks for one symbol, as long as there is a maximal rank and the alphabet stays finite.

The set T_Σ of all trees over Σ is defined inductively as the smallest set of expressions such that $f[t_1, \ldots, t_n] \in T_\Sigma$ for every $f \in \Sigma_n$ and all $t_1, \ldots, t_n \in T_\Sigma$. A subset of T_Σ is called a tree language. $t_1, \ldots, t_n$ are the *direct subtrees* of the tree. The set $subtrees(t)$ consists of t itself and all subtrees of its direct subtrees.

Let $\Box$ be a special symbol of rank 0 (leaf label). A tree $c \in T_{\Sigma \cup \{\Box\}}$ in which $\Box$ occurs exactly once is called *context*, the set of all contexts over Σ is denoted by C_Σ. For $c \in C_\Sigma$ and $s \in T_\Sigma$, $c[[s]]$ denotes the tree obtained by substituting s for $\Box$ in c. The depth of c is the length of the path from the root to $\Box$.

A *(total, deterministic) bottom-up finite-state tree automaton (fta)* is a tuple $\mathcal{A} = (\Sigma, Q, \delta, F)$ where Σ is the ranked input alphabet, Q is the finite set of states, δ is the transition function assigning to every $f \in \Sigma_n$ and all $q_1, \ldots, q_n \in Q$ a state $\delta(q_1 \cdots q_n, f) \in Q$, and $F \subseteq Q$ is the set of accepting states. The transition function extends to trees: $\delta : T_\Sigma \longrightarrow Q$ is defined such that if $t = f[t_1, \ldots, t_n] \in T_\Sigma$ then $\delta(t) = \delta(\delta(t_1) \cdots \delta(t_n), f)$. The language accepted by $\mathcal{A}$ is $L(\mathcal{A}) = \{t \in T_\Sigma | \delta(t) \in F\}$. Such a tree language is called *regular*.

It is well known that the Myhill-Nerode theorem carries over to regular tree languages: Let $L \subseteq T_\Sigma$. Given two trees $s, s' \in T_\Sigma$, let $s \sim_L s'$ iff for every $c \in C_\Sigma$, either both of $c[[s]]$ and $c[[s']]$ are in L or none of them is. Obviously, $\sim_L$ is an equivalence relation on T_Σ. The equivalence class containing $s \in T_\Sigma$ is denoted by $[s]_L$. The *index* of L is the cardinality of $\{[s]_L | s \in T_\Sigma\}$. The Myhill-Nerode theorem states that L is a regular tree language iff L is of finite index iff L is the union of all equivalence classes $[s]_L$ with $s \in L$. It follows from this that for every fta $\mathcal{A}$, $L(\mathcal{A})$ is of finite index. Conversely, if a tree language is of finite index, we can easily build an fta $\mathcal{A}_L$ recognizing L, with the states being the equivalence classes of L, $F = \{[s]_L | s \in L\}$, and, given some $f \in \Sigma_k$ and states $[s_1]_L, \ldots, [s_k]_L$, $\delta_L([s_1]_L, \ldots, [s_k]_L, f) = [f[s_1, \ldots, s_k]]_L$. Moreover, this fta is the unique minimal fta recognizing L, up to a bijective renaming of states.

The Pumping lemma for regular tree languages (see [5] for a proof):

Lemma 1. *For any regular tree language $T \subseteq T_\Sigma$ there is a number $n \geq 1$ such that, if $t \in T_\Sigma$ has height $k \geq n$, then, for some $s \in T_\Sigma$ and $p, q \in C_\Sigma$, $t = q[[p[[s]]]]$ where p has depth ≥ 1 and $q[[\underbrace{p[[\ldots p[[s]]\ldots]]}_{k \text{ times}}]] \in T_\Sigma$ for all $k \geq 0$.*

2.2 Tree Adjoining Grammar

We will now give a definition for the grammar formalism Tree Adjoining Grammar, which was designed under linguistic considerations:

Definition 1. *A TAG is a 5-tuple $\langle \Sigma, N, I, A, S \rangle$, where Σ is the (non-ranked)* terminal labeling alphabet*, N is the (non-ranked)* nonterminal labeling alphabet *with $N \cap \Sigma = \emptyset$, S is the start symbol with $S \in N$, I is a finite set of* initial trees *where the root is labeled with S, and A is a finite set of* auxiliary trees*.*

Nonterminals label inner nodes, terminals label all leaf nodes but one, which is labeled by the nonterminal also labeling the root of the tree. This leaf is referred to as the *foot node*. Initial and auxiliary trees are referred to as *elementary trees*. New trees can be built by *adjunction*: A node in a tree is replaced by an auxiliary tree and the subtree formerly rooted at that node is attached to the foot node of the auxiliary tree.

A TAG can be enriched by associating a pair of constraints with every node, stating if adjunction is required or not (*obligatory adjunction (OA) constraint*), and which auxiliary trees may be adjoined at that node (*selective adjunction (SA) constraint*). These constraints obliterate the roles of nonterminal and terminal symbols and the start symbol, and hence the distinction between initial and auxiliary trees as well. Rogers [9] defines *non-strict* TAGs:

Definition 2. *A non-strict TAG is a pair $\langle E, I \rangle$ where E is a finite set of elementary trees in which each node is associated with a label from some alphabet, an SA constraint (a subset of E), and an OA constraint (Boolean valued). $I \subseteq E$ is a distinguished non-empty subset. Every elementary tree has a foot node.*

Example 1. Let $G = \langle \{\alpha, \beta\}, \{\alpha\} \rangle$ be a TAG (over the alphabet $\{a, b, c, d, S\}$). The only initial tree α and auxiliary tree β are given in Figure 1. Constraints at the inner nodes and the foot node are: $OA = 0$ and $SA = \{\beta\}$ for the ones without a bar, $OA = 0$ and $SA = \emptyset$ for the ones labeled with '$\bar{S}$'. The bar stands for null adjunction, no adjunction is allowed at these nodes. G generates the (non-context-free) string language $a^n b^n c^n d^n$.

3 Multi-dimensional trees and automata

In this section we will introduce multi-dimensional trees and some related concepts as presented by Rogers [8,9].

Starting from a definition of ordinary trees based on two-dimensional tree domains, Rogers [8,9] generalizes the concept both downwards (to strings and points) and upwards and defines *labeled multi-dimensional trees* based on a hierarchy of multi-dimensional tree domains:

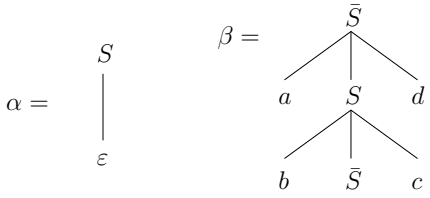

Fig. 1. A TAG generating the (non-context-free) string language $a^n b^n c^n d^n$.

Definition 3. *Let* ${}^d1$ *be the class of all* d*th-order sequences of* 1*s:* ${}^0 1 := \{1\}$*, and* ${}^{d+1}1$ *is the smallest set satisfying (i)* $\langle\rangle \in {}^{d+1}1$*, and (ii) if* $\langle x_1, \ldots, x_l\rangle \in {}^{d+1}1$ *and* $y \in {}^d1$*, then* $\langle x_1, \ldots, x_l, y\rangle \in {}^{d+1}1$*. Let* $\mathbb{T}^0 := \{\emptyset, \{1\}\}$ *(point domains). A* $(d{+}1)$*-dimensional tree domain is a set of hereditarily prefix closed* $(d{+}1)$*st-order sequences of* 1*s, i.e.,* $\mathsf{T} \in \mathbb{T}^{d+1}$ *iff*

- $\mathsf{T} \subseteq {}^{d+1}1$,
- $\forall s, t \in {}^{d+1}1 : s \cdot t \in \mathsf{T} \Rightarrow s \in \mathsf{T}$,
- $\forall s \in {}^{d+1}1 : \{w \in {}^d1 | s \cdot \langle w\rangle \in \mathsf{T}\} \in \mathbb{T}^d$.

A Σ*-labeled* $\mathsf{T}d$ *(*d-dimensional tree*) is a pair* $\langle T, \tau\rangle$*) where* T *is a* d*-dimensional tree domain and* $\tau : T \longrightarrow \Sigma$ *is an assignment of labels in the (non-partitioned) alphabet* Σ *to nodes in* T*. We will denote the class of all* Σ*-labeled* $\mathsf{T}d$ *as* $\mathbb{T}^d_\Sigma$*.*

Every d-dimensional tree can be conceived to be built up from one or more d-dimensional *local trees*, that is, trees of depth at most one in their major dimension. Each of these smaller trees consists of a root and an arbitrarily large $(d-1)$-dimensional "child tree" consisting of the root's children (a formal definition of the set $\mathbb{T}^{d,loc}_\Sigma$ of all local trees over some alphabet Σ would be for example $\mathbb{T}^{d,loc}_\Sigma = \{\langle T, \tau\rangle | \langle T, \tau\rangle$ is a Σ-labeled $\mathsf{T}d$, and $\forall s \in T : |s| \leq 1\}$). Local strings (i.e., one-dimensional trees), for example, consist of a root and a point as its child tree. Local two-dimensional trees consist of a root and a string. Local three-dimensional trees would have a pyramidal form, with a two-dimensional tree as its base. There are also trivial local trees (consisting of a single root), and even empty ones. Composite trees can be built from local ones by identifying the root of one local tree with a node in the child tree of another (and adapting the addresses in order to incorporate them into the newly created tree domain). Figure 2 shows examples of local and composite trees for the first four steps of the hierarchy (only some composite trees are labeled, and in the three-dimensional case, only the addresses of nodes that do not appear in the rightmost local tree as well are given, for clarity. ε_d denotes an empty sequence of order d).

Rogers [9] also defines automata for labeled $\mathsf{T}d$s:

Definition 4. *A* $\mathsf{T}d$ *automaton with finite state set* Q *and (non-ranked) alphabet* Σ *is a finite set of triples* $\mathcal{A}^d \subseteq \Sigma \times Q \times \mathbb{T}^{d-1}_Q$*.*

The interpretation of a triple $\langle \sigma, q, \mathcal{T}\rangle \in \mathcal{A}^d$ is that if a node of a $\mathsf{T}d$ is labeled with σ and $\mathcal{T}$ encodes the assignment of states to its children, then that node

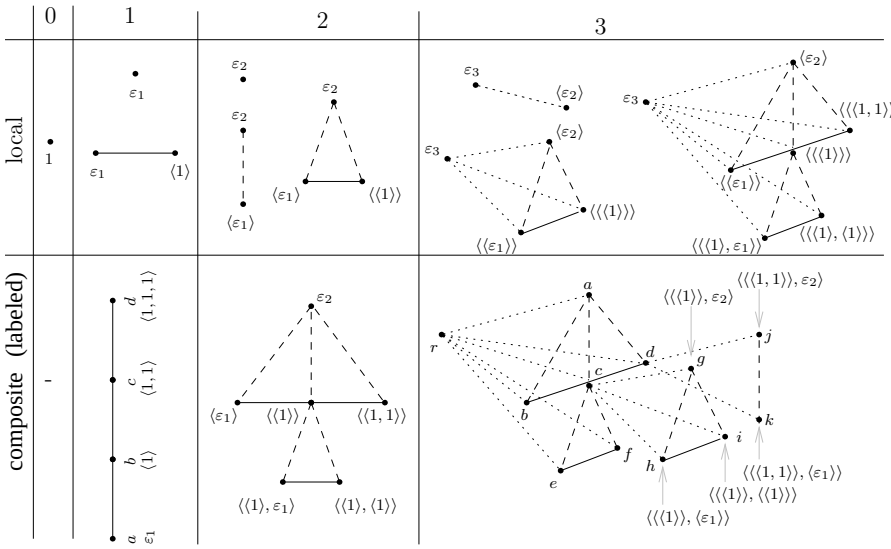

Fig. 2. Local and composite trees for $d = 0, 1, 2, 3$

may be assigned state q. A *run* of a $\mathsf{T}d$ automaton on a Σ-labeled $\mathsf{T}d$ $\mathcal{T} = \langle T, \tau \rangle$ is a mapping $r : T \longrightarrow Q$ in which each assignment is licensed by $\mathcal{A}^d$. Note that this implies that a leaf labeled with σ may be assigned state q only if there is a triple $\langle \sigma, q, \emptyset \rangle \in \mathcal{A}^d$, where $\emptyset$ is the empty $\mathsf{T}(d-1)$. If $F \in Q$ is the set of accepting states, then the set of (finite) Σ-labeled $\mathsf{T}d$ recognized by $\mathcal{A}^d$ is that set for which there is a run of $\mathcal{A}^d$ that assigns the root a state in F.

$\mathsf{T}1$ automata correspond to finite-state automata for strings, i.e., they recognize the regular languages. $\mathsf{T}2$ automata correspond to (non-deterministic) finite-state automata for trees, i.e., they recognize the regular tree languages.

One of the most important concepts in connection with multi-dimensional trees is that of the *yield* of a tree. The yield of a two-dimensional tree is the string formed by its leaf labels. In Rogers' [9] words, it is a projection of the tree onto the next lower level, i.e., its dimensions are reduced by one. $\mathsf{T}d$s with $d \geq 3$ have several yields, one for each dimension that is taken away, down to the one-dimensional string yield. Note that when taking the yield of a tree with $d \geq 3$, some thought has to go into the question of how to interweave the child trees of its local components to form a coherent $(d-1)$-dimensional tree, since there are often several possibilities. Rogers solves this by introducing special nodes called *heads* and defines them such that in the child tree of every local component there is a unique path of heads leading from the root to a leaf. This leaf is called the *foot* of the child tree and marks the splicing point, i.e., the point where the yield of the subtree containing it should be connected to the remaining part of the overall yield. See [9] for the exact definition.

As is well known, the class of the string yields of languages recognized by (two-dimensional) finite-state tree automata are the context-free languages. The

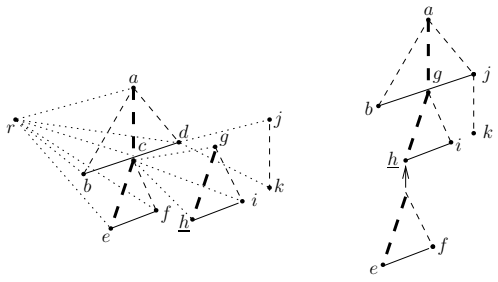

Fig. 3. Ambiguity in the yield for $d \geq 3$, resolved by marked foot nodes

class of the string yields of d-dimensional tree languages for $d \geq 3$ are situated between the classes of context-free and context-sensitive languages in the Chomsky Hierarchy, where for every d the class of string yields of the d-dimensional tree languages is properly contained in the next (i.e., for $d + 1$).

Via the yield operation, Rogers has established a link between T3s and TAGs by proving the equivalence of T3 recognizing automata and non-strict TAGs:

Theorem 1 ([9]). *A set of Σ-labeled two-dimensional trees is the yield of a recognizable set of Σ-labeled* T3 *iff it is generated by a non-strict TAG.*

The representation of a TAG as three-dimensional trees obviously constitutes a regularization: Trees are now constructed by adding local trees at the frontier of another tree (see Figure 4), which is a regular process, instead of expanding nodes at the interior. As follows from Theorem 1, the trees generated by the original TAG can be extracted from the T3s using the yield operation.

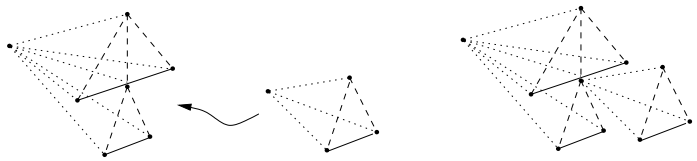

Fig. 4. Adjunction in TAG expressed via three-dimensional trees

Rogers conjectures that there may also be potential linguistic applications for structures of more than three dimensions, and gives an amelioration of the standard TAG account of modifiers using four dimensions (see [8]).

In the next section we will introduce a new notation for multi-dimensional trees that is a generalization of the one on which (classical) finite-state tree automata are based, i.e., a representation that allows multi-dimensional trees to be noted as expressions over a partitioned alphabet.

4 Multi-dimensional trees as terms

We will use finite d-dimensional tree labeling alphabets Σ^d where each symbol $f \in \Sigma^d$ is associated with at least one unlabeled $(d-1)$-dimensional tree t specifying the admissible child structure for a root labeled with f (as before it is possible to admit several child structure trees for one symbol). t can be given in any form suitable for trees, as long as it is compatible with the existence of an empty tree. For consistency's sake we will use the definition of multi-dimensional trees given below and write t as an expression over a special kind of "alphabet" containing just one symbol ρ for which any child structure is admissible.

Let Σ_t^d for $d \geq 1$ be the set of all symbols associated with t and Σ^0 a set of constant symbols. The set $\mathbb{T}_{\Sigma^d}$ of all d-dimensional trees can then be defined inductively as follows:

Definition 5. *Let ε^d be the empty d-dimensional tree. Then*

- $\mathbb{T}_{\Sigma^0} := \{\varepsilon^0\} \cup \Sigma^0$, *and*
- *for $d \geq 1$: $\mathbb{T}_{\Sigma^d}$ is the smallest set such that $\varepsilon^d \in \mathbb{T}_{\Sigma^d}$ and $f[t_1, \ldots, t_n]_t \in \mathbb{T}_{\Sigma^d}$ for every $f \in \Sigma_t^d$, n the number of nodes in t, $t_1, \ldots, t_n \in \mathbb{T}_{\Sigma^d}$ and $t_1, \ldots, t_n$ are rooted breadth-first in that order[1] at the nodes of t.*

Note how naturally this generalization comprises the concept of rank for labeling symbols in two-dimensional trees: For every symbol f in Σ_s^2 for some string s, s encodes the rank of f in its length, and specifies that the children of a node labeled with f should be ordered linearly (as is normal in two-dimensional trees). For $d \geq 3$ the term "rank" of some symbol $g \in \Sigma_t^d$ still makes sense if we indicate the number of nodes in t by it – this way most of the results for two-dimensional trees can be read directly as applying to multi-dimensional trees in general.

For some tree $t_p = f[t_1, \ldots, t_n]_t$ with $f \in \Sigma_t^d$, $t_1, \ldots, t_n$ are the direct subtrees of the tree, and the rest of the usual tree terminology can be applied in a similar manner. Also, for some fixed d, let $\square$ be a special symbol associated with ε^{d-1} (leaf label). A tree $c \in \mathbb{T}_{\Sigma^d \cup \{\square\}}$ in which $\square$ occurs exactly once is still called a context, and $c[[s]]$ for $c \in C_{\Sigma^d}$ and $s \in \mathbb{T}_{\Sigma^d}$ is defined via substitution as before.

Our new notation is equivalent to the one by Rogers in the following sense: For every recognizable set $L^R \subseteq \mathbb{T}_\Sigma^d$ of d-dimensional trees over some alphabet Σ in Rogers' notation there is a translation $\Phi : L^R \longrightarrow \mathbb{T}_{\Sigma^d}$ characterized by:

- For $d = 0$: $\langle \emptyset, \emptyset \rangle \mapsto \varepsilon^0$ and, for some $a \in \Sigma$, $\langle \{1\}, \{1 \mapsto a\} \rangle \mapsto a$.
- For $d \geq 1$: $\langle \emptyset, \emptyset \rangle \mapsto \varepsilon^d$, $\langle \{\langle\rangle\}, \{\langle\rangle \mapsto a\} \rangle \mapsto a$ for some $a \in \Sigma$, and, for some $f \in \Sigma$, $\langle \{\langle\rangle\} \cup T_x, \{\langle\rangle \mapsto f\} \cup \tau_x \rangle \mapsto f[\Phi(\langle T_1, \tau_1 \rangle), \ldots, \Phi(\langle T_n, \tau_n \rangle)]_t$ with $t = \Phi(\langle T_t, \tau_t \rangle)$ where T_t is the set of first elements of the members of T_x and τ_t is the unique function $\tau_t : T_t \longrightarrow \{\rho\}$, and $T_i = \{z | \langle a_i \rangle \cdot z \in T_x\}$ for $1 \leq i \leq n$ where a_i is the ith element in the sequence obtained by ordering the members of T_t inductively by length. τ_i is defined by $\tau_i(z) = \tau_x(\langle a_i \rangle \cdot z)$.

[1] This is an ad hoc settlement, any other spatial arrangement would do as well.

Σ^d is obtained as follows: For each term $t_p \in \Phi(L^R)$ and each subterm $f[t_1, \ldots, t_n]_t$ of t_p, $f \in \Sigma^d_t$. We have restricted ourselves to recognizable sets of trees (i.e., that are built from a finite set of local trees) because otherwise Σ^d may be infinite, which is due to the fact that Rogers uses non-partitioned labeling alphabets so that in theory arbitrarily many roots labeled with the same symbol can have completely different child structures.

For every set $L^N \subseteq \mathbb{T}_{\Sigma^d}$ of d-dimensional trees in the notation given above there is a translation $\Psi : L^N \longrightarrow \mathbb{T}^d_\Sigma$ characterized by the following (the construction of Σ from Σ^d is trivial):

- For $d = 0$: $\varepsilon^0 \mapsto \langle \emptyset, \emptyset \rangle$ and, for $a \in \Sigma^0$, $a \mapsto \langle \{1\}, \{1 \mapsto a\} \rangle$.
- For $d \geq 1$: $\varepsilon^d \mapsto \langle \emptyset, \emptyset \rangle$, $a \mapsto \langle \{\langle\rangle\}, \{\langle\rangle \mapsto a\} \rangle$ for some $a \in \Sigma^d_{\varepsilon^{d-1}}$, and, for some $f \in \Sigma^d_s$, $f[s_1, \ldots, s_m]_s \mapsto \langle \{\langle\rangle\} \cup T_y, \{\langle\rangle \mapsto f\} \cup \tau_y \rangle$ where $T_y = \bigcup_{1 \leq i \leq m,\ x \in S_i} \langle b_i \rangle \cdot x$ with $\langle S_i, \sigma_i \rangle = \Psi(s_i)$ for all i with $1 \leq i \leq m$ and $\langle S_s, \sigma_s \rangle = \Psi(s)$ and b_i is the ith element in the sequence obtained by ordering the elements of S_s inductively by length. τ_y is defined by $\tau_y(\langle b_i \rangle \cdot z) = \sigma_i(z)$.

Both translations traverse the input structure recursively, which includes, for every symbol, a recursion through the tree specifying the admissible child structure for that symbol, which in turn entails recursions through the dimensions down to zero (as the child structure tree is translated, too). We will now show the equivalence of the two notations by proving that $\Psi(\Phi(t_p)) = t_p$ for all $t_p \in L_1$ for some arbitrary recognizable $L_1 \subseteq \mathbb{T}^d_\Sigma$ and $\Phi(\Psi(t_q)) = t_q$ for all $t_q \in L_2$ for some arbitrary $L_2 \subseteq \mathbb{T}_{\Sigma^d}$ for corresponding alphabets Σ and Σ^d.

(1) $\Psi(\Phi(t_p)) = t_p$ for all $t_p \in L_1$: For $d = 0$ this is clear. We will prove the claim for $d \geq 1$ by induction on the depth of t_p. For depth 0 ($t_p = \langle \emptyset, \emptyset \rangle$) and 1 ($t_p = a$ for some $a \in \Sigma$) this is also clear. Assume that the claim holds for all $d_1 < d$ and all d-dimensional trees with depth k for some $k \geq 0$. Assume that $t_p = \langle \{\langle\rangle\} \cup T_x, \{\langle\rangle \mapsto f\} \cup \tau_x \rangle$ for some $f \in \Sigma$ has depth $k + 1$.

$$
\begin{aligned}
t_p &= \langle \{\langle\rangle\} \cup \{\langle a_1 \rangle\} \cdot T_1 \cup \ldots \cup \{\langle a_n \rangle\} \cdot T_n, \\
&\qquad \{\langle\rangle \mapsto f\} \cup \bigcup_{z \in T_1} (\langle a_1 \rangle \cdot z \mapsto \tau_1(z)) \cup \ldots \cup \bigcup_{z \in T_n} (\langle a_n \rangle \cdot z \mapsto \tau_n(z)) \rangle \\
&\qquad \text{(definition of } T_x \text{ and } \tau_x \text{, with } T_i, \tau_i, a_i \text{ defined as above)} \\
\Psi(\Phi(t_p)) &= \langle \{\langle\rangle\} \cup \langle b_1 \rangle \cdot S_1 \cup \ldots \cup \langle b_m \rangle \cdot S_m, \\
&\qquad \{\langle\rangle \mapsto f\} \cup \bigcup_{z \in S_1} (\langle b_1 \rangle \cdot z \mapsto \sigma_1(z)) \cup \ldots \cup \bigcup_{z \in S_m} (\langle b_m \rangle \cdot z \mapsto \sigma_m(z)) \rangle \\
&\qquad \text{(definition of } T_y \text{ and } \tau_y \text{, with } S_i, \sigma_i, b_i \text{ defined as above)} \\
\Psi(\Phi(t_p)) &= \Psi(f[\Phi(\langle T_1, \tau_1 \rangle), \ldots, \Phi(\langle T_n, \tau_n \rangle)]_{\Phi(\langle T_t, \tau_t \rangle)}) \text{ (definition of } \Phi\text{).}
\end{aligned}
$$

By the induction hypothesis we know that $\langle S_s, \sigma_s \rangle = \Psi(s) = \Psi(\Phi(\langle T_t, \tau_t \rangle)) = \langle T_t, \tau_t \rangle$ and consequently $n = m$ and $a_i = b_i$ for all $1 \leq i \leq n, m$. In the same way we know that $\langle S_i, \sigma_i \rangle = \Psi(\Phi(\langle T_i, \tau_i \rangle)) = \langle T_i, \tau_i \rangle$ for all i with $1 \leq i \leq n, m$, and thus $\Psi(\Phi(t_p)) = t_p$. ■

(2) $\Phi(\Psi(t_q)) = t_q$ for all $t_q \in L_2$: Again, for $d = 0$ this is clear. We will prove the claim for $d \geq 1$ by induction on the depth of t_q. For depth 0 ($t_q = \varepsilon^d$) and 1 ($t_q = a$ for some $a \in \Sigma^d_{\varepsilon^{d-1}}$) this is also clear. Assume that the claim holds for all $d_1 < d$ and all d-dimensional trees with depth k for some $k \geq 0$. Assume that $t_q = f[s_1, \ldots, s_m]_s$ for some $f \in \Sigma^d_s$ has depth $k+1$.

$$\begin{aligned}\Phi(\Psi(t_q)) &= \Phi(\langle\{\langle\rangle\} \cup \{\langle b_1\rangle\} \cdot S_1 \cup \ldots \cup \{\langle b_m\rangle\} \cdot S_m,\\ &\qquad \{\langle\rangle \mapsto f\} \cup \bigcup_{z \in S_1} (\langle b_1\rangle \cdot z \mapsto \sigma_1(z)) \cup \ldots \cup \bigcup_{z \in S_m} (\langle b_m\rangle \cdot z \mapsto \sigma_m(z))\rangle)\\ &\qquad \text{(definition of } T_y \text{ and } \tau_y\text{, with } S_i\text{, } \sigma_i\text{, } b_i \text{ defined as above)}\\ &= f[\Phi(\langle T_1, \tau_1\rangle), \ldots, \Phi(\langle T_n, \tau_n\rangle)]_{\Phi(\langle T_t, \tau_t\rangle)} \text{ (definition of } \Phi).\end{aligned}$$

We know, by the relevant definitions, that $\langle T_t, \tau_t\rangle = \langle\{b_1, \ldots, b_m\}, \{b_1 \mapsto \rho, \ldots, b_m \mapsto \rho\}\rangle = \Psi(s)$ and thus $\Phi(\langle T_t, \tau_t\rangle) = \Phi(\Psi(s)) = s$ by the induction hypothesis, which also implies $m = n$. By the same reflection, $\langle T_i, \tau_i\rangle = \langle\{z | \langle b_i\rangle \cdot z \in \{\langle b_1\rangle\} \cdot S_1 \cup \ldots \cup \{\langle b_m\rangle\} \cdot S_m\}, \bigcup_{z \in S_i} (\langle b_i\rangle \cdot z \mapsto \sigma_i(z))\rangle = \Psi(s_i)$ and $\Phi(\langle T_i, \tau_i\rangle) = \Phi(\Psi(s_i)) = s_i$ for all i with $1 \leq i \leq n, m$. This concludes the proof. ■

We can now represent automata for multi-dimensional trees as a direct generalization of classical finite-state tree automata:

Definition 6. *A (total, deterministic) finite-state d-dimensional tree automaton is a quadruple $\mathcal{A}^d = (\Sigma^d, Q, \delta, F)$ with input alphabet Σ^d, finite set of states Q, set of accepting states $F \subseteq Q$ and transition function δ with $\delta(t(q_1, \ldots, q_n), f) \in Q$ for every $f \in \Sigma^d_t$ where $t(q_1, \ldots, q_n)$ encodes the assignment of states to the nodes of t (i.e., $t(q_1, \ldots, q_n)$ is isomorphic to t and its nodes are labeled with $q_1, \ldots, q_n$ breadth-first in that order if Q is taken as a partitioned alphabet in which every element is associated with all the child structures it occurs with in δ). The transition function extends to d-dimensional trees: $\delta : \mathbb{T}_{\Sigma^d} \longrightarrow Q$ is defined such that if $t_p = f[t_1, \ldots, t_n]_t \in \mathbb{T}_{\Sigma^d}$ then $\delta(t_p) = \delta(t(\delta(t_1), \ldots, \delta(t_n)), f)$. The set of trees accepted by $\mathcal{A}^d$ is $L(\mathcal{A}^d) = \{t_p \in \mathbb{T}_{\Sigma^d} | \delta(t_p) \in F\}$.*

The equivalence between this definition and the one by Rogers [9] is easy to see: For two corresponding automata $\mathcal{A}^d = (\Sigma^d, Q, \delta, F)$ and $\mathcal{A}^d_R \subseteq \Sigma_R \times Q_R \times \mathbb{T}^{d-1}_{Q_R}$ with the set of accepting states F_R in the two notations the set of states Q and Q_R and accepting states F and F_R coincide, the construction of Σ_R from Σ^d is trivial, and Σ^d is constructed from $\mathcal{A}^d_R$ as follows: $f \in \Sigma^d_t$ for all triples $\langle f, q, t_0\rangle \in \mathcal{A}^d_R$, where $t = \Phi(\langle T_0, \tau_{0+}\rangle)$ for $t_0 = \langle T_0, \tau_0\rangle$ and τ_{0+} is the unique function $\tau_{0+} : T_0 \longrightarrow \{\rho\}$. Most importantly, there is a one-to-one correspondence between the elements of $\mathcal{A}^d_R$ and δ: Every triple $\langle f, q, t_0\rangle \in \mathcal{A}^d_R$ can be translated to an assignment $\delta(\Psi(t_0), f) = q$ of $\mathcal{A}^d$, and every assignment $\delta(t(q_1, \ldots, q_n), f) = q$ of $\mathcal{A}^d$ to a triple $\langle f, q, \Phi(t(q_1, \ldots, q_n))\rangle \in \mathcal{A}^d_R$. From this and from the identical state sets it follows that $L(\mathcal{A}^d_R) = \Psi(L(\mathcal{A}^d))$ and $L(\mathcal{A}^d) = \Phi(L(\mathcal{A}^d_R))$.

With the term representation and the adapted definitions of contexts and automata given in this section, results pertaining to the class of regular string or tree languages as for instance the Myhill-Nerode theorem or the Pumping lemma (see Section 2.1) and all their consequences (like the existence of a

unique minimal finite-state automaton $\mathcal{A}_L^d$ recognizing L for every recognizable d-dimensional tree language L) carry over directly to multi-dimensional trees.

Finally, we will define a yield function for multi-dimensional trees in the new notation. As for $d \geq 3$ the yield is not unambiguous (see Figure 3), the structures have to be enriched with additional information. Assume that, for $d \geq 2$, in every tree $t_p \in \mathbb{T}_{\Sigma^d}$ every labeling symbol $f \in \Sigma^d$ is indexed with a set $S \subseteq \{2, \ldots, d\}$. If $x \in S$ then we call a node labeled by f_S a *foot node for the* $(x-1)$*-dimensional yield of* t_p. For every subtree t_q of t_p the distribution of these foot nodes must fulfil certain conditions:

(1) If t_q has depth 0 the index set in its root label must contain d, otherwise $t_q = f_S[t_1, \ldots, t_n]_t$ with $f \in \Sigma_t^d$, $S \subseteq \{2, \ldots, d\}$, and $t_1, \ldots, t_n \in \mathbb{T}_{\Sigma^d}$ must have exactly one direct subtree $t_i \in \{t_1, \ldots, t_n\}$ whose root labeling symbol is indexed with a set containing d and this subtree is attached to a *leaf* in t. In both cases, we will refer to that root as the d-dimensional foot node of t_q.
(2) The foot nodes are distributed in such a way that for every n-dimensional yield of t_p with $n < d$, condition (1) is fulfilled as well.

For $d \geq 2$, the direct yield of a tree $t_p \in \mathbb{T}_{\Sigma^d}$ is then defined recursively as

$$yd_{d-1}(t_p) = \begin{cases} \varepsilon^{d-1} & \text{for } t_p = \varepsilon^d, \\ a_S & \text{for } t_p = a_S \text{ with } a \in \Sigma^d_{\varepsilon^{d-1}} \text{ and } S \subseteq \{2, \ldots, d\}, \\ op_{t_p}(t_1) & \text{for } t_p = f_S[t_1, \ldots, t_n]_t \text{ with } t_1, \ldots, t_n \in \mathbb{T}_{\Sigma^d},\ f \in \Sigma_t^d, \\ & t \neq \varepsilon^{d-1}, \text{ and } S \subseteq \{2, \ldots, d\}, \end{cases}$$

where $op_{t_p}(t_i)$ for $t_i \in \{t_1, \ldots, t_n\}$ is defined as the $(d-1)$-dimensional tree that is obtained by replacing the d-dimensional foot node of t_i in $yd_{d-1}(t_i)$ by $e_R[op_{t_p}(t_j), \ldots, op_{t_p}(t_k)]_{t_x}$, where e_R with $e \in \Sigma^d$ and $R \subseteq \{2, \ldots, d\}$ is the label of the foot node, t_x is the $(d-2)$-dimensional child structure of the node at which t_i is attached in t and $t_j, \ldots, t_k$ are the direct subtrees of t_p that are attached (breadth-first in that order) at the nodes of t_x.

The result $yd_{d-1}(t_p)$ is a $(d-1)$-dimensional tree over an alphabet Σ^{d-1} containing at least all the node labels in $yd_{d-1}(t_p)$, each associated at least with the child structures it occurs with. Obviously, the string yield of a d-dimensional tree for $d \geq 2$ can be obtained by taking the direct yield $d-1$ times.

5 Conclusion

We have provided a new, term-like representation for multi-dimensional trees which establishes them as a direct generalization of classical trees. As a consequence multi-dimensional trees can now be used as an input for (slightly adapted) finite-state applications based on classical formal (tree) language theory, for example in the areas of grammatical inference (shown in [15]) or natural language processing. Via the concept of the yield of a multi-dimensional tree this also means that these applications can now be conceived to be able to process even some language classes that lie beyond context-freeness as well.

Due to lack of space we have not furnished the full possible system of concepts linked to recognizable multi-dimensional tree languages, but of course further notions such as regular grammars can be formulated for these structures as well. Also, various results can be ameliorated such as a less complex-looking, regular version of the Pumping lemma for the string languages generated by TAGs [13] relying on the correspondence to three-dimensional trees (see Section 3).

Another interesting project we propose for the near future would be to check whether any implementations of known finite-state applications based on formal tree languages can be adapted to multi-dimensional trees, or even if with this generalization new implementations have become possible.

References

1. Hopcroft, J.E., Ullman, J.D.: Introduction to automata theory, languages, and computation. Addison-Wesley (1979)
2. Comon, H., Dauchet, M., Gilleron, R., Jacquemard, F., Lugiez, D., Tison, S., Tommasi, M.: Tree Automata Techniques and Applications. Available on: www.grappa.univ-lille3.fr/tata (2005)
3. Weir, D.J.: A geometric hierarchy beyond context-free languages. Theoretical Computer Science 104(2), 235–261 (1992)
4. F. Gécseg, M. Steinby: Tree automata. Akademiai Kiado (1984)
5. F. Gécseg, M. Steinby: Tree languages. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, vol. 3, chapter 1, pp. 1–68. Springer (1997)
6. Joshi, A.K.: Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural description. In: Dowty, D., Karttunen, L., Zwicky, A. (eds.) Natural Language Processing. Cambridge University Press (1985)
7. Morawietz, F.: Two-Step Approaches to Natural Language Formalisms. Studies in Generative Grammar, vol. 64. Mouton de Gruyter (2003)
8. Rogers, J.: Syntactic Structures as Multi-dimensional Trees. Research on Language and Computation 1, 265–305 (2003)
9. Rogers, J.: wMSO Theories as Grammar Formalisms. Theoretical Computer Science 293, 291–320 (2003)
10. Angluin, D.: Learning regular sets from queries and counterexamples. Information and Computation 75(2), 87–106 (1987)
11. Sakakibara, Y.: Learning context-free grammars from structural data in polynomial time. Theorteical Computer Science 76(2–3), 223–242 (1990)
12. Drewes, F., Högberg, J.: Learning a Regular Tree Language from a Teacher. In: Ésik, Z., Fülöp, Z. (eds.) Developments in Language Theory 2003. LNCS, vol. 2710, pp. 279–291. Springer (2003)
13. Vijayashanker, K.: A Study of Tree-Adjoining Grammars. PhD thesis, University of Pennsylvania (1987)
14. Kasprzik, A.: Two Equivalent Regularizations of Tree Adjoining Grammars. Technical Report 08-1, University of Trier. Available on: urts117.uni-trier.de/cms/index.php?id=15939 (2008)
15. Kasprzik, A.: A learning algorithm for multi-dimensional trees, or: Learning beyond context-freeness. Technical Report 08-2, University of Trier. Available on: urts117.uni-trier.de/cms/index.php?id=15939 (2008)