Input Strictly Local Tree Transducers

Jing Ji(✉) and Jeffrey Heinz

The Department of Linguistics and The Institute of Advanced Computational Science, Stony Brook University, Stony Brook, USA
{jing.ji,jeffrey.heinz}@stonybrook.edu

Abstract. We generalize the class of input strictly local string functions (Chandlee et al. 2014) to tree functions. We show they are characterized by a subclass of frontier-to-root, deterministic, linear tree transducers. We motivate this class from the study of natural language as it provides a way to distinguish local syntactic processes from non-local ones. We give examples illustrating this kind of analysis.

Keywords: Strictly local · Computational syntax · Tree transducers

1 Introduction

Locally Testable sets of strings in the strict sense (Strictly Local, SL) are a subclass of the regular languages with interesting properties [16,20]. Rogers [18] presents a generalization of SL to sets of trees and shows they characterize the derivations of context-free languages. Chandlee et al. [2,3] generalize SL formal languages in another direction. They present classes of strictly local string-to-string functions. In this paper, we generalize the SL class to a class of functions over trees. In particular, we present a characterization in terms of frontier-to-root, deterministic, linear tree transducers [5,7].

One motivation comes from computational and theoretical linguistics, where the goal of one program is to identify and understand the minimally powerful classes of formal grammars which can describe aspects of natural language [4]. To this end, subregular sets and functions over strings have been used to distinguish and characterize phonological generalizations [11]. More recent research has begun studying natural language syntax from the perspective of subregular sets and functions over trees, as opposed to strings [9,10].

One rationale for studying subclasses of regular string/tree sets and relations is that it is known that finite-state methods are sufficient to describe aspects of natural language. For phonology and morphology, finite-state methods over strings appear sufficient [1,17]. For syntax, finite-state methods over trees similarly appear sufficient. Rogers [19] showed that a syntactic theory of English can be understood in terms of Monadic Second Order (MSO) definable constraints over trees. Languages with more complex constructions can be understood in terms of regular tree languages undergoing regular tree transductions

A. Leporati et al. (Eds.): LATA 2020, LNCS 12038, pp. 369–381, 2020.
https://doi.org/10.1007/978-3-030-40608-0_26

[8,14]. Tree transducers also have found broad application in machine translation [13,15]. It remains an open question, however, whether the full power of regular computations are necessary [11].

Another rationale for identifying subregular classes of languages is that learning problems may be easier to solve in the sense of requiring less and time and resources than otherwise [12].

By defining and characterizing the Input Strictly Local class of tree transducers, we hope to take a first step in developing a more fine-grained perspective on the syntactic transformations present in natural languages. The structure of the paper is as follows.

Section 2 defines trees and associated properties and functions based on their recursive structure. In this way we follow the tree transducer literature [5,7]. However, we note that we do *not* adopt the convention of ranked alphabets. Instead we obtain their effects by bounding the largest number of children a tree in some tree set can have and by requiring that the pre-image of the transition function of the tree automata is finite. While this is unconventional, we believe it simplifies our presentation and proofs. Section 2 also reviews strictly local treesets and reviews the proof of the abstract characterization of them [18].

Section 3 presents the main theoretical results. Deterministic, frontier-to-root, finite-state, linear tree transducers (abbreviated DFT) are defined, Input Strictly Local (ISL) tree functions are defined abstractly and then characterized in terms DFTs. Section 4 concludes.

2 Preliminaries

Assume a finite alphabet Σ and let Σ^* denote the set of all strings of finite length that can be obtained via concatenation of the elements of Σ. We denote the empty string with λ.

Consider an alphabet Σ and symbols $[\,]$ which do not belong to it. A tree is defined inductively as follows:

- **Base Case:** For each $a \in \Sigma$, $a[\,]$ is a tree. The tree $a[\,]$ is also called a *leaf*. We also write $a[\lambda]$ for $a[\,]$.
- **Inductive Case:** If $a \in \Sigma$ and $t_1 t_2 \dots t_n$ is a string of trees of length n ($n \geq 1$), then $a[t_1 t_2 \dots t_n]$ is a tree.

For a trees $t = a[t_1 t_2 \dots t_n]$, the trees $t_1, t_2, \dots t_n$ are the *children* of t and t_i denotes the ith child. Σ^T denotes the set of all trees of finite size from Σ.

The depth, size, yield, root, branch, and the set of subtrees of a tree t, written $\mathtt{dp}(t)$, $|t|$, $\mathtt{yld}(t)$, $\mathtt{root}(t)$, $\mathtt{branch}(t)$ and $\mathtt{sub}(t)$, respectively, are defined as follows. For all $a \in \Sigma$:

- If $t = a[\,]$, then $\mathtt{dp}(t) = 1$, $|t| = 1$, $\mathtt{yld}(t) = a$, $\mathtt{root}(t) = a$, $\mathtt{branch}(t) = 0$, and $\mathtt{sub}(t) = \{t\}$.
- If $t = a[t_1 t_2 \dots t_n]$ then $\mathtt{dp}(t) = \mathtt{max}\{\mathtt{dp}(t_i) | 1 \leq i \leq n\} + 1$, and $|t| = 1 + \sum_{i=1}^{n} |t_i|$, and $\mathtt{yld}(t) = \mathtt{yld}(t_1)\mathtt{yld}(t_2) \dots \mathtt{yld}(t_n)$, and $\mathtt{root}(t) = a$, and $\mathtt{branch}(t) = n$, and $\mathtt{sub}(t) = \bigcup\{\mathtt{sub}(t_i) | 1 \leq i \leq n\} \cup \{t\}$.

The roots of the subtrees of a tree t are called *nodes*. The root of a tree is also called its *root node*. Leaves are also called *frontier nodes*.

The branching degree of a tree t is $\texttt{branch\_degree}(t) = \texttt{max}\{\texttt{branch}(u) \mid u \in \texttt{sub}(t)\}$. Let Σ_n^T denotes the set of trees $\{t \in \Sigma^T \mid \texttt{branch\_degree}(t) \leq n\}$.

Example 1. Suppose $\Sigma = \{\text{S, a, b}\}$. $S\,[\,a\;S\,[\,a\;b\,]\;b\,]$ denotes a tree rooted in S with `branch_degree` of 3.

Let N^* be the set of all sequences of finite length of positive natural numbers. For $\vec{n} = \langle n_1, n_2, \ldots, n_m\rangle \in N^*$ (m ≥ 1), the subtree of t at $\vec{n}$ is written $t.\vec{n}$, and it is defined inductively:

- **Base Case:** $t.\vec{n} = t$ iff $\vec{n} = \lambda$.
- **Inductive Case:** Suppose $t = a[t_1 t_2 \ldots t_n]$ and $\vec{n} \neq \lambda$. Then $t.\vec{n} = t.\langle n_1, n_2 \ldots n_m\rangle = t_{n_1}.\langle n_2, n_3 \ldots n_m\rangle$.
- **Note:** $t.\vec{n}$ is undefined otherwise.

These sequences are the Gorn addresses of the subtrees of t. For example, The first child of t is given by $t.\langle 1\rangle$ (if it exists); the second child by $t.\langle 2\rangle$ (if it exists); the second child of the first child by $t.\langle 1, 2\rangle$ (if it exists); and of course $t.\langle\,\rangle = t$.

The Gorn addresses provide a natural ordering of the subtrees of t in terms of the length-lexicographic ordering. For distinct $\vec{n} = \langle n_1, n_2, \ldots, n_k\rangle, \vec{m} = \langle m_1, m_2, \ldots, m_\ell\rangle$, $\vec{n}$ precedes $\vec{m}$ iff either $k < \ell$, or $k = \ell$ and $n_1 < m_1$, or $k = \ell$ and $n_1 = m_1$ and $\langle n_2, \ldots, n_k\rangle < \langle m_2, \ldots, m_\ell\rangle$. This essentially orders subtrees of t such that the ones closer to the root of t are ordered earlier, and those 'on the same level' in t are ordered 'left to right.' We make use of this ordering in our proof of Theorem 1.

The largest common subtrees of a set of trees T, denoted $\texttt{lcs}(T)$, is $\{d \in \bigcap_{t\in T} \texttt{sub}(t) \mid \forall d' \in \bigcap_{t\in T} \texttt{sub}(t),\ |d'| \leq |d|\}$.

The k-stem ($k \geq 1$) of a tree t, written $\texttt{stem}_k$(t), is defined as follows.

- **Base Case:** For all $a \in \Sigma$, if $t = a[\,]$, then $\texttt{stem}_k(t) = a[\,]$.
- **Inductive Case:** For all a $\in \Sigma$, if t = a$[t_1 t_2 \ldots t_n]$, then
  - $\texttt{stem}_1(t) = \texttt{root}(t)[\,]$, and
  - $\texttt{stem}_k(t) = a[\texttt{stem}_{k-1}(t_1)\texttt{stem}_{k-1}(t_2)\ldots\texttt{stem}_{k-1}(t_n)]$.

The stems of a tree t, denoted $\texttt{stem}(t)$ is the set $\{\texttt{stem}_k(t) \mid k \geq 1\}$.

Example 2. The 2-stems of the tree in Example 1 is $\{S\,[\,a\;S\;b\,], S\,[\,a\;b\,], a[\,], b[\,]\}$.

It is useful to incorporate boundary markers into the roots and leaves of trees. Informally, given a Σ-tree t, boundary markers are added above the root and below the leaves. Formally, we employ symbols $\rtimes, \ltimes \notin \Sigma$ for this purpose. We let $\hat{\Sigma} = \Sigma \cup \{\rtimes, \ltimes\}$.

Thus for all $a \in \Sigma$, $t \in \Sigma^T$, let $\texttt{add}_{\rtimes}(t) = \rtimes[\,t\,]$, and $\texttt{add}_{\ltimes}(a[\,]) = a[\ltimes[\,]]$, and $\texttt{add}_{\ltimes}(a[t_1 \ldots t_n]) = a[\ltimes(t_1)\cdots\ltimes(t_n)]$. Then for any Σ-tree t, its augmented counterpart $\hat{t} = \texttt{add}_{\rtimes}(\texttt{add}_{\ltimes}(t))$.

The k-factors of a tree t are defined as the set of k-depth stems of subtrees of $\hat{t}$. For all $t \in \Sigma^T$, let $F_k(t) = \bigcup\{\texttt{stem}_k(u) \mid u \in \texttt{sub}(\hat{t})\}$.

We lift the definition of k-factors to treesets in the natural way. For all T $\subseteq \Sigma^T$, $F_k(T) = \bigcup_{t\in T} F_k(t)$.

Example 3. The 2-factors of the tree in Example 1 is the set $\{\rtimes[\,S\,[\,]\,],$ $S\,[\,a\,[\,]\,S\,[\,]\,b\,[\,]\,],\ S\,[\,a\,[\,]\,b\,[\,]\,],\ a\,[\,\rtimes[\,]\,],\ b\,[\,\rtimes[\,]\,],\rtimes[\,]\}$.

A strictly k-local grammar $G = (\Sigma, S)$ where S is a finite subset of $F_k(\Sigma^T)$ and the tree language of G is defined as: $\mathbb{L}((\Sigma, S)) = \{t \mid F_k(t) \subseteq S\}$.

Note that since S is finite, there exists a smallest number n such that $S \subseteq \hat{\Sigma}_n^T$. It follows that $\mathbb{L}((\Sigma, S))$ is of branching degree n. A treeset $T \subseteq \Sigma^T$ is strictly k-local if there exists a k and a strictly k-local grammar G such that $\mathbb{L}(G) = T$. Such treesets form exactly strictly k-local treesets (SL_k). Strictly local stringsets are a special case of strictly local treesets where all the branching degree is 1; so every node (except leaves) are unary branching.

Strictly 2-local treesets have been called local treesets in previous literature [18]. Every Strictly 2-local tree language can be generated by a context free grammar [7,18].

Comparable to the characterization of strictly local string sets, which is Suffix Substitution Closure [20], each strictly 2-local tree language satisfies Subtree Substitution Closure[18]. To explain this characterization, we first introduce the notion of subtree-substitution.

For $t, s \in \Sigma^T$ and $\vec{n} = \langle n_1, n_2, \ldots, n_m\rangle \in N^*$ (m $\geq$ 1), the operation of substituting the subtree of t at $\vec{n}$ by s, written as $t.\vec{n} \leftarrow s$, is defined as follows.

- **Base Case:** $t.\vec{n} \leftarrow s = s$ iff $\vec{n} = \lambda$.
- **Inductive Case:** If t $= a[t_1 t_2 \ldots t_n]$ then $t.\vec{n} \leftarrow s = a[t_1 t_2 \ldots (t_{n_1}.$ $\langle n_2, n_3 \ldots n_m\rangle \leftarrow s) \ldots t_n]$.

We also define substitution of all the subtrees of t rooted at x ($x \in \Sigma$) by s, which we write as $t \overset{x}{\leftarrow}$s.

- **Base Case:** If root(t) = x, $t \overset{x}{\leftarrow}$s = s.
- **Base Case:** If root(t) $\neq$ x and t $= a[\,]$ (a $\in \Sigma$), $t \overset{x}{\leftarrow}$s = t.
- **Inductive Case:** If root(t) $\neq$ x and t $= a[t_1 t_2 \ldots t_n]$ (a $\in \Sigma$), $t \overset{x}{\leftarrow}$s $=$ $a[s_1 s_2 \ldots s_n]$ where $s_i = t_i \overset{x}{\leftarrow}$s ($1 \leq i \leq n$).

Rogers [18] proves the following result and we repeat the proof to set the stage for the sequel.

Theorem 1 (Subtree Substitution Closure). *A treeset $T \subseteq \Sigma^T$ is strictly 2-local iff there is n such that T is of branching degree n and for all $A, B \in T$, whenever there exist two vectors $\vec{n_1}, \vec{n_2} \in \vec{N}$, such that* $\texttt{root}(A.\vec{n_1}) = \texttt{root}(B.\vec{n_2})$ *then $A.\vec{n_1} \leftarrow B.\vec{n_2} \in T$.*

Proof. If T is strictly 2-local, then there exists a corresponding strictly 2-local grammar G that satisfies $\mathbb{L}(G) = T$. Thus there exists a finite set $S \subset F_k(\Sigma^T)$ such that $\mathbb{L}((\Sigma, S)) = T$.

Consider any $A, B \in T$ and $\vec{n_1}, \vec{n_2} \in \vec{N}$ such that $\texttt{root}(A.\vec{n_1}) = \texttt{root}(B.\vec{n_2})$. Let $t = A.\vec{n_1} \leftarrow B.\vec{n_2}$. We show $t \in T$. First notice that $F_2(A) \subseteq S$ and $F_2(B) \subseteq S$ because $A, B \in T$ and $T = \mathbb{L}((\Sigma, S))$. Next consider any element $u \in F_2(t)$. By definition of t and 2-factor, u must be a 2-stem of a subtree of

$A.\vec{n_1} \leftarrow B.\vec{n_2}$. If u is the 2-stem of a subtree of $B.\vec{n_2}$ then $u \in F_2(B) \subset S$. If not, then u is a 2-stem of a subtree of A and so $u \in F_2(A) \subset S$. Either way, $u \in S$ and so $F_2(t) \subseteq S$. It follows that $t \in T$.

Conversely, consider a treeset T such that whenever there exist two vectors $\vec{n_1}, \vec{n_2} \in \vec{N}$, such that $\texttt{root}(A.\vec{n_1}) = \texttt{root}(B.\vec{n_2})$ then $A.\vec{n_1} \leftarrow B.\vec{n_2} \in T$. We refer to this property as the SSC. To show T is Strictly 2-Local, we present a finite set $S \subset F_k(\Sigma^T)$ such that $\mathbb{L}((\Sigma, S)) = T$. Let S = $F_2(T)$. Since T is of branching degree n, S is finite. In order to prove $\mathbb{L}((\Sigma, S)) = T$, we need to show both $\mathbb{L}((\Sigma, S)) \subseteq T$ and $T \subseteq \mathbb{L}((\Sigma, S))$. It is obvious that $T \subseteq \mathbb{L}((\Sigma, S))$ because for any t $\in T$, $F_2(t) \subseteq S = F_2(T)$.

The following proves that $\mathbb{L}((\Sigma, S)) \subseteq T$ by recursive application of SSC. Consider any $t \in \mathbb{L}((\Sigma, S))$. Let $t_1 = t.\vec{n}_1, t_2 = t.\vec{n_2}, \ldots t_m = t.\vec{n}_m$ be an enumeration of the m subtrees of t by their Gorn addresses in length-lexicographic order. (Note that $t_1 = t$).

The base step of the induction is to choose a tree $s_0 \in T$ that has the same root as t. Such a $s_0 \in T$ exists because $\rtimes[\texttt{root}(t)[\,]] \in S$.

Next we assume by the induction hypothesis that $s_{i-1} \in T$ and we will construct s_i which is also in T. For each $1 \leq i \leq m$, if t_i is a leaf then let $u = t_i[\rtimes[\,]]$, otherwise let $u = \texttt{stem}_2(t_i)$. Choose a tree $x \in T$ such that $u \in F_2(x)$. Such a tree $x \in T$ exists because $u \in S = F_2(T)$. It follows there is $\vec{m}$ such that $\texttt{stem}_2(x.\vec{m}) = u$. Let $s_i = s_{i-1}.\vec{n_i} \leftarrow x.\vec{m}$. Since $\texttt{root}(s_{i-1}.\vec{n_i}) = \texttt{root}(x.\vec{m})$ and $s_{i-1}, x \in T$, it follows that $s_i \in T$ by SSC. Informally, this construction ensures the nodes and children of s_i are identical to those of t from the root of t to the root of the subtree t_i.

Since each s_i is built according to s_{i-1} and $s_0 \in T$ we conclude that $s_m \in T$. Furthermore, since the subtrees are ordered length-lexicographically and we substitute a 2-stem of a subtree of t to build s_i, it follows that $s_m = t$. As t was arbitrary in $\mathbb{L}((\Sigma, S))$, we obtain $\mathbb{L}((\Sigma, S)) \subseteq T$. □

The catenation operation of two trees $u \cdot t$ is defined by substitution in the leaves. Let \$ be a new symbol, i.e., \$ $\notin \Sigma$. Let $\Sigma^T_{\$}$ denote the set of all trees over $\Sigma \cup \$$ which contain exactly one occurrence of label \$ in the leaves. The operation of catenation is defined inductively:

- **Base Case:** For $t \in \Sigma^T$, $\$[\,] \cdot t = t$.
- **Base Case:** For all $a \in \Sigma$, if $u = a[\,]$, $u \cdot t = a[\,]$.
- **Inductive Case:** For all $a \in \Sigma$, if $u = a[t_1 t_2 \ldots t_n]$, $u \cdot t = a[(t_1 \cdot t)(t_2 \cdot t) \ldots (t_n \cdot t)]$.

Example 4. Suppose Σ = {S, a, b}. Let u = S [a[ ] \$[ ] b[ ]] and t = S [a[ ] b[ ]]. u·t = S [(a[ ]·t) (\$[ ]·t) (b[ ]·t)] = S [a[ ] S [a[ ] b[ ]] b[ ]].

Notice that the classical catenation of strings can be viewed as a special case of catenation of trees with unary branching. This operation can also be used to represent subtrees. For $t \in \Sigma^T \cup \Sigma^T_{\$}$, if $t = u \cdot s$, then s is a subtree of t.

If $U \subseteq \Sigma^T_{\$}$ and $T \subseteq \Sigma^T \cup \Sigma^T_{\$}$, then $U \cdot T = \{u \cdot t \mid u \in U, t \in T\}$. Furthermore, for any $t \in T$ and any tree language $T \subseteq \Sigma^T$, the quotient of t w.r.t. T is defined

as $\mathtt{qt}_T(t) = \{u \in \Sigma_{\$}^T \mid u \cdot t \in T\}$. Canonical finite-state tree recognizers can be defined in terms of these quotients.

3 Input Strictly Local Tree Transducers

In this section we define functions that map trees to trees. After reviewing some basic terminology, we introduce deterministic, frontier-to-root, linear, finite-state Tree Transducers (DFT). We then define Input Strictly Local Tree Transducers (ISLTT) in a grammar-independent way, and then prove they correspond exactly to a type of DFTs. Examples are provided along the way.

A function f with domain X and co-domain Y can be written $f : X \to Y$. The image of f is the set $\{f(x) \in Y | x \in X, f(x) \text{ is defined}\}$ and the pre-image of f is the set $\{x \in X | f(x) \text{ is defined}\}$. Tree transducers compute functions that map trees to trees $f : \Sigma_n^T \to \Gamma^T$.

DFTs are defined as a tuple $(Q, \Sigma, \Gamma, F, \delta)$, where Q is a finite set of states, $F \subseteq Q$ is a set of final states, and δ is a transition function that maps a sequence of states paired with an element of Σ to a state and a *variably-leafed tree*. A variably-leafed tree is a tree which may include variables in the leaves of the tree. Let $X = \{x_1, x_2, \ldots\}$ be a countable set of variables. If Σ is a finite alphabet then $\Sigma^T[X]$ denotes the set of trees t formed with the alphabet $\Sigma \cup X$ such that if the root of a subtree s of t is a variable then s is a leaf (so variables are only allowed in leaves). Thus formally the transition function is $\delta : Q^* \times \Sigma \to \Gamma^T[X] \times Q$. Importantly, the pre-image of the transition function must be finite. We sometimes write $(q_1 q_2 \ldots q_m, a, t, q) \in \delta$ to mean $\delta(q_1 q_2 \ldots q_m, a) = (t, q)$.

In the course of computing a tree transduction, the variables in variably-leafed trees are substituted with trees. Assume $t_1, t_2, \ldots t_m \in \Gamma^T$ and $s \in \Gamma^T[X]$, which is a variable leafed tree with any subset of the variables $\{x_1, x_2, ..., x_m\}$. We define a substitution function ϕ such that $\phi(t_1 t_2 \ldots t_m, s) = s \overset{x_i}{\leftarrow} t_i$ for $1 \leq i \leq m$.

We define the process of transducing a tree recursively using a function π, which maps Σ_n^T to $Q \times \Gamma^T$, which itself is defined inductively with δ.

- **Base Case:** $\pi(a[\,]) = (q, v)$ iff $\delta(\lambda, a) = (v, q)$
- **Inductive Case:** $\pi(a[t_1 t_2 \ldots t_m]) = (q, \phi(v_1 v_2 \ldots v_m, s))$ iff $\delta(q_1 q_2 \ldots q_m, a) = (s, q)$ and $\pi(t_i) = (q_i, v_i)$ for each $1 \leq i \leq m$.

The tree-to-tree function the transducer M recognizes is the set of pairs $\mathbb{L}(M) = \{(t, s) \mid t \in \Sigma_n^T, s \in \Gamma^T, \pi(t) = (q, s), q \in F\}$. We also write $M(t) = s$ whenever $(t, s) \in \mathbb{L}(M)$.

A DFT is *linear* provided whenever $\delta(q_1 q_2 \ldots q_m, a) = (s, q)$, no variable occurs more than once in s.

Example 5. Wh-movement refers to a syntactic analysis of question words such as English *what* and *who*. It is common to analyze this as a relation between tree structures [21]. The input structure describes the relation of the wh-word to its verb (cf. "John thinks Mary believes Bill buys what?") and the yield of the

output structure reflects the pronunciation (cf. "What does John think Mary believe Bill buys").

We use a simplified transformation to make the point. In the alphabet, S represents the root node of a input tree, W stands for a wh-word and P for everything else (P is for phrase). A transducer of wh-movement can be constructed as a tuple $M_{wh} = (Q, \Sigma, F, \delta)$ where $Q = \{q_w, q_p, q_s\}$, $F = \{q_s\}$, $\Sigma = \{S, P, W\}$, and $\delta = \{(\lambda, P, P[\,], q_p), (\lambda, W, W[\,], q_w), (q_p q_p, P, P[x_1 x_2], q_p), (q_w q_p, P, P[x_1 x_2], q_w), (q_p q_w, P, P[x_1 x_2], q_w), (q_p q_w, S, S[W[\,]S[x_1 x_2]], q_s), (q_w q_p, S, S[W[\,]S[x_1 x_2]], q_s), (q_p q_p, S, S[x_1 x_2], q_s)\}$.

Figure 1 illustrates some of the transformations computed by the finite-state machine M_{wh}. The tree with a wh-word in Fig. (1a) is transformed into the tree in Fig. (1b). (M_{wh} keeps the original wh-word in-situ but it could easily be removed or replaced with a trace). The trees in Fig. (1c) and (d) are the same because there is no wh-word in the input tree and so M_{wh} leaves it unchanged.

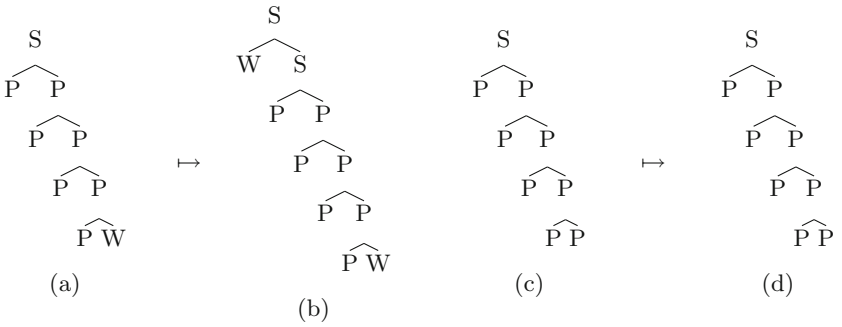

Fig. 1. M_{wh} maps the tree in (a) to the tree in (b) and likewise maps the tree in (c) to itself in (d).

Next we describe the canonical form of deterministic tree transducers. The *quotient* of a tree $t \in \Sigma^T$ with respect to a tree-to-tree function $f : \Sigma^T \to \Gamma^T$ is a key idea. It will be useful to develop some notation for the *largest common subtree of the image under f of the set of trees which includes t as a subtree.* Let $\texttt{lcsi}_f(t) = \texttt{lcs}\Big(f\big(\Sigma^T_{\$} \cdot \{t\}\big)\Big)$. When f is understood from context, we just write $\texttt{lcsi}(t)$. Then the quotient is defined as follows:

$$\texttt{qt}_f(t) = \Big\{(u, v) \mid f(u \cdot t) = v \cdot s, s = \texttt{lcsi}_f(t)\Big\}. \tag{1}$$

When f is clear from context, we write $\texttt{qt}(t)$ instead of $\texttt{qt}_f(t)$.

It is worth noting that for a tree $t \in \Sigma^T_n$, the largest common subtree of the image of a linear transducer with the input of $\Sigma^T_{\$} \cdot \{t\}$ is unique if it exists because if there is more than one tree that belongs to $\texttt{lcs}(f(\Sigma^T_{\$} \cdot \{t\}))$, they must be produced by copying, which is not allowed by linear DFT. If trees $t_1, t_2 \in \Sigma^T$

have the same quotient with respect to a function f, they are quotient-equivalent with respect to f and we write $t_1 \sim_f t_2$. Clearly, $\sim_f$ is an equivalence relation which partitions Σ^T.

As in the string case, to each regular tree language T, there is a canonical DFT accepting T. The characterization given by the Myhill-Nerode theorem can be transferred to the tree case [6]. For any treeset T, the quotients of trees w.r.t. T can be used to partition Σ^T into a finite set of equivalence classes.

Analogous to the smallest subsequential finite state transducer for a subsequential function, we can construct the smallest linear DFT for a deterministic tree-to-tree function f and refer to this transducer as the canonical transducer for f, Ψ_f^c. For $t_1, t_2, \ldots, t_m \in \Sigma_n^T$ (m $\leq$ n) and a $\in \Sigma$, let the contribution of a w.r.t. $t_1 t_2 \ldots t_m$ be $\texttt{cont}_f(a, t_1 t_2 \ldots t_m) = v \in \Gamma^T[X]$, which satisfies

$$\phi\Big(\texttt{lcsi}(t_1)\texttt{lcsi}(t_2)\ldots\texttt{lcsi}(t_m), v\Big) = \texttt{lcsi}\Big(a[t_1 t_2 \ldots t_m]\Big). \tag{2}$$

The term $\texttt{cont}_f(a, t_1 t_2 \ldots t_m)$ is well-defined since each $\texttt{lcsi}(t_1), \texttt{lcsi}(t_2), \ldots$ $\texttt{lcsi}(t_m)$, and $\texttt{lcsi}(a[t_1 t_2 \ldots t_m])$ are unique.

Then the canonical DFT for a deterministic tree-to-tree function f is:

- $Q = \{\texttt{qt}_f(t) \mid\in \Sigma_n^T\}$,
- $F \subseteq Q$,
- For $a \in \Sigma$, there exists $v \in \Gamma^T$ that satisfies $(\lambda, a, v, \texttt{qt}_f(a[\,])) \in \delta$,
- For $t_1, t_2, \ldots, t_m \in \Sigma_n^T (m \leq n)$ and $a \in \Sigma$, $\Big(\texttt{qt}_f(t_1)\,\texttt{qt}_f(t_2)\ldots\texttt{qt}_f(t_m)$, a, $\texttt{cont}_f(a, t_1 t_2 \ldots t_m)$, $\texttt{qt}_f(a[t_1 t_2 \ldots t_m])\Big) \in \delta$.

The presentation here differs from Friese et al. [6], but the only thing we require in the proof of Theorem 2 below is the existence of the canonical DFT whenever $\sim_f$ is of finite index.

We define ISLTT as a subclass of linear DFTs.

Definition 1 (Input Strictly Local Tree-to-tree Function). *A function f is Input Strictly Local (ISL) if there is a k and n such that for all $t_1, t_2 \in \Sigma_n^T$, if $\texttt{stem}_{k-1}(t_1) = \texttt{stem}_{k-1}(t_2)$ then $\texttt{quotient}_f(t_1) = \texttt{quotient}_f(t_2)$.*

In the same way ISL string functions can be used to probe the locality properties of phonological processes, ISL tree functions can used to probe the locality properties of syntactic transformations.

To show that a syntactic transformation is *not* ISL one need only construct a counterexample to Definition 1.

Example 6. We can show the function computed by $M_w h$ from Example 5 is not ISL for any k because there is no bound on the distance the wh-word can 'travel.' Suppose there is a k and n such that for all $t_1, t_2 \in \Sigma_n^T$, if $\texttt{stem}_{k-1}(t_1) = \texttt{stem}_{k-1}(t_2)$ then $\texttt{qt}_f(t_1) = \texttt{qt}_f(t_2)$. Let $u_1 = u_2 \ldots = u_{k-1} = P[P\$]$. Also let u_k = P [P P], $s = S[P\$]$ and $w = P[PW]$. We construct two sentence structures: $s \cdot t_1$ and $s \cdot t_2$, where $t_1 = u_1 \cdot u_2 \ldots u_{k-1} \cdot w$ and $t_2 = u_1 \cdot u_2 \ldots u_{k-1} \cdot u_k$.

It is obvious that $\texttt{stem}_{k-1}(t_1) = \texttt{stem}_{k-1}(t_2)$. However, $\texttt{qt}_f(t_1) \neq \texttt{qt}_f(t_2)$ since $(s, s) \in \texttt{qt}_f(t_2)$ but $(s, s) \notin \texttt{qt}_f(t_1)$. As we can always find such a pair of trees t_1 and t_2 for any k, it is thus proved that wh-movement is not ISL for any k.

Our main result, Theorem 2 below, establishes an automata-theoretic characterization of ISL tree-to-tree functions. As we illustrate after the proof, one can show that a tree transformation is ISL using this theorem.

Theorem 2 (ISL Tree Transducers). *A function f is ISL iff there is some k and n such that f can be described with a DFT for which*

1. $Q = \{\texttt{stem}_{k-1}(t) \mid t \in \Sigma_n^T)\}$ *and* $F \subseteq Q$,
2. $\forall q_1 q_2 \dots q_m \in Q^*(1 \leq m \leq n)$, $a \in \Sigma$, $u \in \Gamma^T[X]$, *it is the case that* $(q_1 q_2 \dots q_m, a, u, q') \in \delta \Rightarrow q' = \texttt{stem}_{k-1}(a[q_1 q_2 \dots q_m])$.

The transducer is finite since Σ is finite and n bounds the branching degree of the pre-image of f which ensures the finiteness of both Q and δ.

Before our proof of the Theorem, we prove a lemma based on these remarks.

Remark 1. For all k,m $\in \mathbb{N}$ with k $\leq$ m, and for all $t \in \Sigma_n^T$, $\texttt{stem}_k(\texttt{stem}_m(t)) = \texttt{stem}_k(t)$ since both t and $\texttt{stem}_m(t)$ share the same k-stem from the root.

Remark 2. For all $k \in \mathbb{N}$, and for all $a \in \Sigma$ and $t_1, t_2, \dots t_m \in \Sigma_n^T (m \leq n)$, $\texttt{stem}_{k-1}(a[t_1 t_2 \dots t_m]) = \texttt{stem}_{k-1}(a[\texttt{stem}_{k-1}(t_1)\, \texttt{stem}_{k-1}(t_2) \dots \texttt{stem}_{k-1}(t_m)])$. This is a direct consequence of Remark 1.

Lemma 1. *Let Ψ be a ISLTT with the properties defined in Theorem 2. If $t \in \Sigma_n^T$ and $u \in \Gamma^T$, $\pi(t) = (q, u)$, then $q = \texttt{stem}_{k-1}(t)$.*

Proof. The proof is by induction on the depth of the trees to which π is applying. The base case follows from the facts that for $(\lambda, a, v, q) \in \delta$ iff $\pi(a[\,]) = (q, v)$ and $q = \texttt{stem}_{k-1}(a[\,])$.

Next assume for all $t_1, t_2, \dots, t_m \in \Sigma_n^T$ (m $\leq$ n) and $v_1, v_2 \dots, v_m \in \Gamma^T$ such that $\pi(t_1) = (q_1, v_1)$ implies $q_1 = \texttt{stem}_{k-1}(t_1)$, $\pi(t_2) = (q_2, v_2)$ implies $q_2 = \texttt{stem}_{k-1}(t_2), \dots, \pi(t_m) = (q_m, v_m)$ implies $q_m = \texttt{stem}_{k-1}(t_m)$. We show that $\forall a \in \Sigma$ that there is a $v \in \Gamma^T[X]$ such that $\pi(a[t_1 t_2 \dots t_m]) = \big(q, \phi(v_1 v_2 \dots v_m, v)\big)$ and $q = \texttt{stem}_{k-1}(a[t_1 t_2 \dots t_m])$. Based on the assumption, we know that $\pi(t_1) = (\texttt{stem}_{k-1}(t_1), v_1)$, $\pi(t_2) = (\texttt{stem}_{k-1}(t_2), v_2)$, $\dots$, $\pi(t_m) = (\texttt{stem}_{k-1}(t_m), v_m)$, so there exists $v \in \Gamma^T[X]$ such that $(\texttt{stem}_{k-1}(t_1)\texttt{stem}_{k-1}(t_2)\texttt{stem}_{k-1}(t_m), a, v, q) \in \delta$. By the construction, q is defined to be equal to $\texttt{stem}_{k-1}(a[\texttt{stem}_{k-1}(t_1)\texttt{stem}_{k-1}(t_2)\texttt{stem}_{k-1}(t_m)])$, which by Remark 2, equals $\texttt{stem}_{k-1}(a[t_1 t_2 \dots t_m])$.

Now we can prove the theorem.

Proof (Theorem 2). ($\Leftarrow$) Assume $k \in \mathbb{N}$ and let f be a function described by $\Psi = \{\text{Q}, \Sigma, \Gamma, \text{F}, \delta\}$ constructed as in Theorem. Let $t_1, t_2 \in \Sigma_n^T$ such that $\texttt{stem}_{k-1}(t_1) = \texttt{stem}_{k-1}(t_2)$. By Lemma 1, both t_1 and t_2 lead to the same state, so $\texttt{qt}_f(t_1) = \texttt{qt}_f(t_2)$. Therefore, f is k-ISL.

($\Rightarrow$) Consider any ISL tree-to-tree function f. Then there is some k and n such that $\forall t_1, t_2 \in \Sigma_n^T$, we have $\mathtt{stem}_{k-1}(t_1) = \mathtt{stem}_{k-1}(t_2) \Rightarrow \mathtt{qt}_f(t_1) = \mathtt{qt}_f(t_2)$. We show that the corresponding ISL tree transducer Ψ_f^{ISL} exists. Since $\mathtt{stem}_{k-1}(\Sigma_n^T)$ is a finite set, the equivalence relation $\sim_f$ partitions Σ^T into at most $\mathtt{stem}_{k-1}(\Sigma_n^T)$ blocks. Thus there exists a canonical linear DFT $\Psi_f^c = \{Q_c, F_c, \Sigma, \Gamma, \delta_c\}$. π_c is the process function derived from δ_c that maps Σ_n^T to $Q_c \times \Gamma^T$.

Construct $\Psi = \{\mathrm{Q}, \mathrm{F}, \Sigma, \Gamma, \delta\}$ as follows:

- $\mathrm{Q} = \mathtt{stem}_{k-1}(\Sigma_n^T)$
- $\forall q \in Q, q \in F$ iff $\mathtt{qt}(q) \in F_c$.
- For $a \in \Sigma$ and $v \in \Gamma^T[X]$, $(\lambda, a, v, q) \in \delta$ iff $(\lambda, a, v, \mathtt{qt}(q)) \in \delta_c$.
- $\forall q_1 q_2 \dots q_m \in Q^*(1 \leq m \leq n), a \in \Sigma, u \in \Gamma^T[X]$, we have $(q_1 q_2 \dots q_m, a, v, \mathtt{stem}_{k-1}(a[q_1 q_2 \dots q_m])) \in \delta$ if and only if $(\mathtt{qt}(q_1)\mathtt{qt}(q_2)\dots\mathtt{qt}(q_m), a, v, \mathtt{qt}(a[q_1 q_2 \dots q_m])) \in \delta_c$.

Ψ is ISL by construction, as the states and transitions of Ψ meet requirements (1) and (2) of Theorem 2.

The following proof show that Ψ computes the same function as Ψ_f^c by showing that Ψ and Ψ_f^c generate the same function. In other words we show $\forall t \in \Sigma_n^T$, $u \in \Gamma^T$, $\pi(t) = (\mathtt{stem}_{k-1}(t), u)$ iff $\pi_c(t) = (\mathtt{qt}(t), u)$ and $\mathtt{stem}_{k-1}(t) \in F$ iff $\mathtt{qt}(t) \in F_c$.

First, we show that $\pi(t) = (\mathtt{stem}_{k-1}(t), u)$ iff $\pi_c(t) = (\mathtt{qt}(t), u)$. Clearly, the base case is satisfied. For all $a \in \Sigma$ and $v \in \Gamma^T[X]$, $(\lambda, a, v, q) \in \delta$ iff $(\lambda, a, v, \mathtt{qt}(q)) \in \delta_c$. Thus $\pi_c(a[\,]) = (\mathtt{qt}(a[\,]), v)$ and $\pi(a[\,]) = (\mathtt{stem}_{k-1}(a[\,]), v)$.

Next assume that there exist $t_1, t_2, \dots, t_m \in \Sigma_n^T$ and $u_1, u_2, \dots, u_m \in \Gamma^T$ such that $\pi(t_i) = (\mathtt{stem}_{k-1}(t_i), u_i)$ iff $\pi_c(t_i) = (\mathtt{qt}(t_1), u_i)$ for each $1 \leq i \leq m$. We show $\forall a \in \Sigma$ and $\forall v \in \Sigma^T[X]$ such that $\pi(a[t_1 t_2 \dots t_m]) = (\mathtt{stem}_{k-1}(a[t_1 t_2 \dots t_m])$, we have $(\phi(u_1 u_2 \dots u_m], v))$ iff $\pi_c(a[t_1 t_2 \dots t_m]) = (\mathtt{qt}(a[t_1 t_2 \dots t_m]), \phi(u_1 u_2 \dots u_m], v))$.

Suppose $\pi_c(a[t_1 t_2 \dots t_m]) = (\mathtt{qt}(a[t_1 t_2 \dots t_m]), (\phi(u_1 u_2 \dots u_m, v)))$. By assumption, $\pi_c(t_i) = (\mathtt{qt}(t_1), u_i)$ for each $1 \leq i \leq m$. Hence, $\big(\mathtt{qt}(t_1) \dots \mathtt{qt}(t_m), a, v, \mathtt{qt}(a[t_1 t_2 \dots t_m])\big) \in \delta_c$.

Let $q_i = \mathtt{stem}_{k-1}(t_i)$ for each $1 \leq i \leq m$. Observe that each $\mathtt{stem}_{k-1}(t_i) = \mathtt{stem}_{k-1}(q_i)$ by Remark 1. Consequently, since f is k-ISL, $\mathtt{qt}(t_i) = \mathtt{qt}(q_i)$. Similarly, $\mathtt{stem}_{k-1}(a[t_1 t_2 \dots t_m]) = \mathtt{stem}_{k-1}(a[q_1 q_2 \dots q_m])$ and so $\mathtt{qt}(a[t_1 t_2 \dots t_m]) = \mathtt{qt}(a[q_1 q_2 \dots q_m])$. By substitution then, we have $\pi_c(t_i) = (\mathtt{qt}(q_i), u_i)$ for each $1 \leq i \leq m$ and $\big(\mathtt{qt}(q_1)\mathtt{qt}(q_2)\dots\mathtt{qt}(q_m), a, v, \mathtt{qt}(a[q_1 q_2 \dots q_m])\big) \in \delta_c$.

By construction of Ψ, $\big(q_1 q_2 \dots q_m, a, x, \mathtt{stem}_{k-1}(a[q_1 q_2 \dots q_m])\big) \in \delta$. Since $\pi(t_i) = (\mathtt{stem}_{k-1}(t_i), u_i)$ for each $1 \leq i \leq m$, it follows that $\pi(a[t_1 t_2 \dots t_m]) = \big(\mathtt{stem}_{k-1}(a[q_1 q_2 \dots q_m]), (\phi(u_1 u_2 \dots u_m, v))\big)$ which equals $\big(\mathtt{stem}_{k-1}(a[t_1 t_2 \dots t_m]), (\phi(u_1 u_2 \dots u_m, v))\big)$.

Conversely, consider any $a \in \Sigma$ and $v \in \Sigma^T[X]$ and suppose $\pi(a[t_1 t_2 \dots t_m]) = \big(\mathtt{stem}_{k-1}(a[t_1 t_2 \dots t_m]), (\phi(u_1 u_2 \dots u_m], v))\big)$. By assumption, $\pi(t_i)$ equals $(\mathtt{stem}_{k-1}(t_i), u_i)$ for each $1 \leq i \leq m$. Thus $\big(\mathtt{stem}_{k-1}(t_1)\mathtt{stem}_{k-1}(t_2)\dots$

$\texttt{stem}_{k-1}(t_m), a, v, \texttt{stem}_{k-1}(a[t_1 t_2 \dots t_m])\big) \in \delta$. Let $q_i = \texttt{stem}_{k-1}(t_i)$ for each $1 \leq i \leq m$ as before. It follows that $\texttt{stem}_{k-1}(t_i) = \texttt{stem}_{k-1}(q_i)$, so $\texttt{qt}(t_i) = \texttt{qt}(q_i)$. Likewise, $\texttt{stem}_{k-1}(a[t_1 t_2 \dots t_m]) = \texttt{stem}_{k-1}(a[q_1 q_2 \dots q_m])$, so $\texttt{qt}(a[t_1 t_2 \dots t_m]) = \texttt{qt}(a[q_1 q_2 \dots q_m])$. Therefore, $\big(\texttt{stem}_{k-1}(q_1)\texttt{stem}_{k-1}(q_2) \dots \texttt{stem}_{k-1}(q_m), a, v, \texttt{stem}_{k-1}(a[q_1 q_2 \dots q_m])\big) \in \delta$.

By construction of Ψ, this means $\big(\texttt{qt}(q_1)\texttt{qt}(q_2) \dots \texttt{qt}(q_m), a, v, \texttt{qt}(a[q_1 q_2 \dots q_n])\big) \in \delta_c$. Since $\pi_c(t_i) = (\texttt{qt}(t_i), u_i)$ for each i by assumption, it follows that $\pi_c(a[t_1 t_2 \dots t_m]) = \big(\texttt{qt}(a[q_1 q_2 \dots q_n]), (\phi(u_1 u_2 \dots u_n, v)\big)$.

We need to further show that $\texttt{stem}_{k-1}(t) \in F$ iff $\texttt{qt}(t) \in F_c$. By construction, we know that $q \in F$ iff $\texttt{qt}(q)$ belongs to F_c. Thus $\texttt{stem}_{k-1}(t) \in F$ iff $\texttt{qt}(\texttt{stem}_{k-1}(t)) \in F_c$. By Remark 1, $\texttt{stem}_{k-1}(t) = \texttt{stem}_{k-1}(\texttt{stem}_{k-1}(t))$. Hence $\texttt{qt}(t) = \texttt{qt}(\texttt{stem}_{k-1}(t))$. Therefore, $\texttt{stem}_{k-1}(t) \in F$ iff $\texttt{qt}(t) \in F_c$.

This concludes the proof that Ψ and Ψ_f^c generate the same function. □

As mentioned earlier, the value of Theorem 2 is that it can be used to establish that certain tree transformations are ISL by presenting a transducer for the transformation which satisfies the properties specified by the theorem.

Example 7. This example shows that reversing the branch order of a regular tree set $T \subseteq \Sigma_n^T$ is ISL. We illustrate with the classic tree language whose yield is the string language $a^n b^n$. In other words we wish to show that the transformation that maps $t_1 = S[a[\,]b[\,]]$ to $t_1' = S[b[\,]a[\,]]$ and $S[a[\,]t_1 b[\,]]$ to $S[b[\,]t_1' a[\,]]$ and so on is ISL.

The DFT can be represented as a tuple (Q, Σ, F, δ) where the states are expressed by the 1-stems of the subtrees of the pre-image: $Q = \{a[\,], b[\,], S[\,]\}$, and $F = \{S[\,]\}$, and $\Sigma = \{a, b, S\}$, and $\delta = \{(\lambda, a, a[\,], a[\,]), (\lambda, b, b[\,], b[\,]), (a[\,]b[\,], S, S[x_2, x_1], S[\,]), (a[\,]S[\,]b[\,], S, S[x_3 x_2 x_1], S[\,])\}$.

The reader can verify that this transducer correctly reverses the branch order of the trees in its pre-image. Further, this construction shows the function is ISL since it satisfies the requirements in Theorem 2.

4 Conclusion

This paper took a first step in characterizing local syntactic transformations by generalizing Input Strictly Local string functions to trees. Future work includes defining Output SL tree functions (cf. [3]) and studying whether these classes of tree functions can be learned more quickly and with fewer resources, and characterizing subclasses of tree transducers which characterize the types of non-local processes found in syntax and machine translation.

References

1. Beesley, K., Kartunnen, L.: Finite State Morphology. CSLI Publications, Stanford (2003)

2. Chandlee, J., Eyraud, R., Heinz, J.: Learning strictly local subsequential functions. Trans. Assoc. Comput. Linguist. **2**, 491–503 (2014)
3. Chandlee, J., Eyraud, R., Heinz, J.: Output strictly local functions. In: Kuhlmann, M., Kanazawa, M., Kobele, G.M. (eds.) Proceedings of the 14th Meeting on the Mathematics of Language (MoL 2015), Chicago, USA, pp. 112–125, July 2015
4. Chomsky, N.: The Minimalist Program. The MIT Press, Cambridge (1995)
5. Comon, H., et al.: Tree automata techniques and applications (2007). http://tata.gforge.inria.fr/. Release 12 Oct 2007
6. Friese, S., Seidl, H., Maneth, S.: Minimization of deterministic bottom-up tree transducers. In: Gao, Y., Lu, H., Seki, S., Yu, S. (eds.) DLT 2010. LNCS, vol. 6224, pp. 185–196. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14455-4_18
7. Gécseg, F., Steinby, M.: Tree Automata. Akadémiai Kiadó, Budapest (2015). http://arxiv.org/abs/1509.06233. Originally published in 1984
8. Graf, T.: Closure properties of minimalist derivation tree languages. In: Pogodalla, S., Prost, J.-P. (eds.) LACL 2011. LNCS (LNAI), vol. 6736, pp. 96–111. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22221-4_7
9. Graf, T.: Curbing feature coding: strictly local feature assignment. In: Proceedings of the Society for Computation in Linguistics (SCiL) 2020 (2020, to appear)
10. Graf, T., Shafiei, N.: C-command dependencies as TSL string constraints. In: Jarosz, G., Nelson, M., O'Connor, B., Pater, J. (eds.) Proceedings of SCiL 2019, pp. 205–215 (2019)
11. Heinz, J.: The computational nature of phonological generalizations. In: Hyman, L., Plank, F. (eds.) Phonological Typology, Chap. 5, pp. 126–195. Phonetics and Phonology, De Gruyter Mouton (2018)
12. Heinz, J., de la Higuera, C., van Zaanen, M.: Grammatical Inference for Computational Linguistics. Synthesis Lectures on Human Language Technologies, Morgan and Claypool (2015)
13. Knight, K., May, J.: Applications of weighted automata in natural language processing. In: Droste, M., Kuich, W., Vogler, H. (eds.) Handbook of Weighted Automata, Chap. 14. EATCS, pp. 571–596. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01492-5_14
14. Kobele, G.M.: Minimalist tree languages are closed under intersection with recognizable tree languages. In: Pogodalla, S., Prost, J.-P. (eds.) LACL 2011. LNCS (LNAI), vol. 6736, pp. 129–144. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22221-4_9
15. Maletti, A.: Survey: tree transducers in machine translation. In: Bordihn, H., Freund, R., Hinze, T., Holzer, M., Kutrib, M., Otto, F. (eds.) Proceedings of the 2nd International Workshop on Non-Classical Models of Automata and Applications. `books@ocg.at`, vol. 263, pp. 11–32. Österreichische Computer Gesellschaft (2010)
16. McNaughton, R., Papert, S.: Counter-Free Automata. MIT Press, Cambridge (1971)
17. Roark, B., Sproat, R.: Computational Approaches to Morphology and Syntax. Oxford University Press, Oxford (2007)
18. Rogers, J.: Strict LT_2: regular :: local : recognizable. In: Retoré, C. (ed.) LACL 1996. LNCS, vol. 1328, pp. 366–385. Springer, Heidelberg (1997). https://doi.org/10.1007/BFb0052167
19. Rogers, J.: A Descriptive Approach to Language-Theoretic Complexity. CSLI Publications, Stanford (1998)

20. Rogers, J., Pullum, G.: Aural pattern recognition experiments and the subregular hierarchy. J. Log. Lang. Inf. **20**, 329–342 (2011)
21. Sportiche, D., Koopman, H., Stabler, E.: An Introduction to Syntactic Analysis and Theory. Wiley, Hoboken (2013)