Equivalence Problems for Tree Transducers: A Brief Survey

Equivalence Problems for Tree Transducers: A Brief Survey

Sebastian Maneth
School of Informatics
University of Edinburgh
smaneth@inf.ed.ac.uk

The decidability of equivalence for three important classes of tree transducers is discussed. Each class can be obtained as a natural restriction of deterministic macro tree transducers (MTTs): (1) no context parameters, i.e., top-down tree transducers, (2) linear size increase, i.e., MSO definable tree transducers, and (3) monadic input and output ranked alphabets. For the full class of MTTs, decidability of equivalence remains a long-standing open problem.

1 Introduction

The macro tree transducer (MTT) was invented independently by Engelfriet [20, 29] and Courcelle [13, 14] (see also [37]). As a model of syntax-directed translations, MTTs generalize the attribute grammars of Knuth [46]. Note that one (annoying) issue of attribute grammars is that they can be circular; MTTs always terminate. Macro tree transducers are a combination of context-free tree grammars, invented by Rounds [57] and also known as "macro tree grammars" [34], and the top-down tree transducer of Rounds and Thatcher [58, 67]: the derivation of the grammar is (top-down) controlled by a given input tree. In terms of a top-down transducer, the combination is obtained by allowing nesting of state calls in the rules (similar to the nesting of nonterminals in the productions of context-free tree grammars). Top-down tree transducers generalize to trees the finite state (string) transducers (also known as "generalized sequential machines", or GSMs, see [38, 8]). In terms of formal languages, compositions of MTTs give rise to a large hierarchy of string languages containing, e.g., the IO and OI hierarchies (at level one they include the indexed languages of Aho [1]), see [22]. MTTs can be applied in many scenarios, e.g., to type check XML transformations (they can simulate the k-pebble transducers of Milo, Suciu, and Vianu [51]), see [23, 49, 50], or to efficiently implement streaming XQuery transformations [42, 52]. In terms of functional programs, MTTs are particularly simple programs that do primitive recursion over an input tree and only produce trees as output. Applications in programming languages exist include [68, 69, 52].

Equivalence of nondeterministic transducers is undecidable, already for restricted string transducers [40]. We therefore only consider deterministic transducers. What is known about the equivalence problem for deterministic macro tree transducers? Unfortunately not much in the general case. Only a few subcases are known to be decidable. Here we describe three of them:

(1) top-down tree transducers

(2) linear size increase transducers

(3) monadic tree transducers.

The first one was solved long ago by Esik [30], but was revived through the "earliest canonical normal form" by Engelfriet, Maneth, and Seidl [26]. The latter implies PTIME equivalence check for total top-down tree transducers. The second one is solved by the decidability of equivalence for deterministic MSO

Z. Ésik and Z. Fülöp (Eds.): Automata and Formal Languages 2014 (AFL 2014)
EPTCS 151, 2014, pp. 74–93, doi:10.4204/EPTCS.151.5

tree transducers of Engelfriet and Maneth [25]. The same authors have shown that every MTT of linear size increase is effectively equal to an MSO transducer [24]. Hence, decidability of equivalence follows for MTTs of linear size increase. Here we give a direct proof using MTTs. The third result is about MTTs over monadic trees. These are string transducers with copying. The decidability of their equivalence problem follows through a relationship with L-systems [28]; in particular, with the sequence equivalence problem of HDT0L systems. The latter was first proved decidable by Culik II and Karumhäki [18].

2 Preliminaries

We deal with finite, ordered, ranked trees. In such a tree, each node is labeled by a symbol from a ranked alphabet such that the rank of the symbol is equal to the number of children of the node. Formally, a *ranked alphabet* consists of a finite set Σ together with a mapping $\mathsf{rank}_\Sigma : \Sigma \to \mathbb{N}$ associating to each symbol its rank. We write $\sigma^{(k)}$ to denote that the rank of σ is equal to k. By $\Sigma^{(k)}$ we denote the subset of symbols of Σ that have rank k. Let Σ be a ranked alphabet. The *set of all trees over* Σ, denoted T_Σ, is the smallest set of strings T such that if $k \geq 0$, $t_1, \ldots, t_k \in T$, and $\sigma \in \Sigma^{(k)}$, then also $\sigma(t_1, \ldots, t_k) \in T$. For a tree of the form $a()$ we simply write a. For a set A, we denote by $T_\Sigma(A)$ the set of all trees over $\Sigma \cup A$ such that the rank of each $a \in A$ is zero. Let $a_1, \ldots, a_n$ be distinct symbols in $\Sigma^{(0)}$ and let $t_1, \ldots, t_n \in T_\Sigma$ such that none of the leaves in t_j are labeled by a_i for $1 \leq i \leq n$. Then by $[a_i \leftarrow t_i \mid i \in \{1, \ldots, n\}]$ we denote the *tree substitution* that replaces each leaf labeled a_i by the tree t_i. Thus $d(a,b,a)[a \leftarrow c(b), b \leftarrow a]$ denotes the tree $d(c(b), a, c(b))$.

Let $t \in T_\Sigma$ for some ranked alphabet Σ. We denote the *nodes of* t by their Dewey dotted decimal path and define the set $V(t)$ of nodes of t as $\{\varepsilon\} \cup \{i.u \mid 1 \leq i \leq k, u \in V(t_i)\}$ if $t = \sigma(t_1, \ldots, t_k)$ with $k \geq 0$, $\sigma \in \Sigma^{(k)}$, and $t_1, \ldots, t_k \in T_\Sigma$. Thus, ε denotes the root node, and $u.i$ denotes the ith child of the node u. For a node $u \in V(t)$ we denote by $t[u]$ its label, and, for a tree t' we denote by $t[u \leftarrow t']$ the tree obtained from t by replacing its subtree rooted at u by the tree t'. The *size* of t, denoted $|t|$, is its number $|V(t)|$ of nodes. The *height* of t, denoted $\mathsf{height}(t)$, is defined as $\mathsf{height}(\sigma(t_1, \ldots, t_k)) = 1 + \max\{\mathsf{height}(t_i) \mid 1 \leq i \leq k\}$ for $k \geq 0$, $\sigma \in \Sigma^{(k)}$, and $t_1, \ldots, t_k \in T_\Sigma$.

We fix the set of *input variables* $X = \{x_1, x_2, \ldots\}$ and the set of *formal context parameters* $Y = \{y_1, y_2, \ldots\}$. For $n \in \mathbb{N}$ we define $X_n = \{x_1, \ldots, x_n\}$ and $Y_n = \{y_1, \ldots, y_n\}$.

A *deterministic finite-state bottom-up tree automaton* is a tuple $A = (Q, \Sigma, \delta, Q_f)$ where Q is a finite set of states, $Q_f \subseteq Q$ is the set of final states, and for every $\sigma \in \Sigma^{(k)}$ and $k \geq 0$, δ_σ is a function from Q^k to Q. The transition function δ is extended to a mapping from T_Σ to Q in the obvious way, and the set of trees accepted by A is $L(A) = \{s \in T_\Sigma \mid \delta(s) \in Q_f\}$.

3 Macro Tree Transducers

Definition 1 A *(deterministic) macro tree transducer* M is a tuple $(Q, \Sigma, \Delta, q_0, R)$ such that Q is a ranked alphabet of states with $Q^{(0)} = \emptyset$, Σ and Δ are ranked alphabets of input and output symbols, respectively, $q_0 \in Q^{(1)}$ is the initial state, and for every $q \in Q^{(m+1)}$, $m \geq 0$, and $\sigma \in \Sigma^{(k)}$, $k \geq 0$, the set of rules R contains at most one rule of the form

$$q(\sigma(x_1, \ldots, x_k), y_1, \ldots, y_m) \to t$$

where $t \in T_{\Delta \cup Q}(X_k \cup Y_m)$ such that if a node in t has its label in Q, then its first child has its label in X_k. A rule as above is called a (q, σ)-rule and its right-hand side is denoted by $\mathsf{rhs}(q, \sigma)$.

The translation $\tau_M : T_\Sigma \to T_\Delta$ (often just denoted M) realized by an MTT M is a partial function recursively defined as follows. For each state q of rank $m+1$, $M_q : T_\Sigma \to T_\Delta(Y_m)$ is the translation of M starting in state q, i.e., "starting" with a tree $q(s, y_1, \ldots, y_m)$ where $s \in T_\Sigma$. For instance, for $a \in \Sigma^{(0)}$, $M_q(a)$ simply equals $\mathsf{rhs}(q,a)$. In general, for an input tree $s = \sigma(s_1, \ldots, s_k)$, $M_q(s)$ is obtained from $\mathsf{rhs}(q,\sigma)$ by repeatedly replacing a subtree of the form $q'(x_i, t_1, \ldots, t_n)$ with $t_1, \ldots t_n \in T_\Delta(Y_m)$ by the tree $M_{q'}(s_i)[y_j \leftarrow t_j \mid 1 \le j \le n]$. The latter tree will also be written $M_{q'}(s_i, t_1, \ldots, t_n)$. We define $\tau_M = M_{q_0}$.

By the definition above, all our MTTs are deterministic and we refer to them as "macro tree transducers" and denote their class of translations by MTT. An MTT is *total* if there is exactly one rule of the above form. If each state of an MTT is of rank one, then it is a *top-down tree transducer*. The class of translations realized by (deterministic) top-down tree transducers is denoted by T. In Lemmas 3 and 5 we make use of nondeterministic top-down tree transducers and mark the corresponding class there by the letter "N". A transducer is nondeterministic if there are several (q,σ)-rules in R. An MTT is *monadic* if its input and output ranked alphabets are monadic, i.e., only contain symbols of rank 1 and 0. An MTT M is of *linear size increase* if there is a constant c such that $|\tau_M(s)| \le c \cdot |s|$ for every $s \in T_\Sigma$.

As an example, consider the transducer M consisting of the rules in Figure 1. The alphabets of this

$$\begin{array}{lcl} q_0(d(x_1,x_2)) & \to & d(q(x_1,1(e)),q(x_2,2(e))) \\ q_0(a) & \to & a(e) \\ q(d(x_1,x_2),y_1) & \to & d(q(x_1,1(y_1)),q(x_2,2(y_1))) \\ q(a,y_1) & \to & a(y_1) \end{array}$$

Figure 1: The macro tree transducer M adds reverse Dewey paths to leaves

transducer are $Q = \{q_0^{(1)}, q^{(2)}\}$, $\Sigma = \{d^{(2)}, a^{(0)}\}$, and $\Delta = \{d^{(2)}, a^{(1)}, 1^{(1)}, 2^{(1)}, e^{(0)}\}$. The MTT M translates a binary tree into the same tree, but additionally adds under each leaf the reverse (Dewey) path of the node. For instance, $s = d(d(a,a),a)$ is translated into $d(d(a(1(1(e))),a(2(1(e)))),a(2(e)))$ as can be verified in this computation of M

$$\begin{array}{l} M(d(d(a,a),a)) = \\ d(M_q(d(a,a),1(e)),M_q(a,2(e))) = \\ d(d(M_q(a,1(1(e))),M_q(a,2(1(e)))),a(2(e))) = \\ d(d(a(1(1(e))),a(2(1(e)))),a(2(e))). \end{array}$$

Note that M is *not* of linear size increase. Hence, it is not MSO definable. Since the translation is neither top-down nor monadic, it falls into a class of MTTs for which we do not know a procedure to decide equivalence. As an exercise, the reader may wonder whether there is an MTT that is similar to M, but outputs Dewey paths below leaves (instead of their *reverses*). In contrast, consider input trees with only exactly one a-leaf (and all other leaves labeled differently). Then an MTT of linear size increase can output under the unique a-leaf its reverse Dewey path, i.e., this translation is MSO definable. Is it now possible with an MTT to output the non-reversed Dewey path?

One of the most useful properties of MTTs is the effective preservation of regular tree languages by their inverses. This is used inside the proofs of several results presented here. For instance, to prove that for every MTT there is an equivalent one which is nondeleting in the parameters (= strict), one can use the above property as follows: given a state q of rank $m+1$ and a subset $A \subseteq Y_m$, the language $T_\Delta(A)$ is regular, and hence $M_q^{-1}(T_\Delta(A))$ is the regular set of inputs for which q outputs only parameters in the set A. Thus, by using regular look-ahead we can determine which parameters are used, and can change the

rules to call an appropriate state q' which is only provided the parameters in A which it uses. Regular look-ahead is explained in Section 4.1.

The following result is stated in Theorem 7.4 of [29]. A similar proof as below is given at the end of [23]. A slightly simpler proof, for a slightly larger class, is presented by Perst and Seidl for macro forest transducers [54].

Lemma 1 Let M be an MTT with output alphabet Δ and let $R \subseteq T_\Delta$ be a regular tree language (given by a bottom-up tree automaton B). Then $\tau_M^{-1}(R)$ is effectively regular. In particular, the domain $\mathsf{dom}(\tau_M)$ is effectively regular.

Proof. Let $M = (Q, \Sigma, \Delta, q_0, R)$ and $B = (P, \Delta, \delta, P_f)$. We construct the new automaton $A = (S, \Sigma, \delta', S_f)$. The states of A are mappings α that associate with each state $q \in Q^{(m+1)}$ a mapping $\alpha(q) : P^m \to P$. The set S_f consists of all $\alpha \in S$ such that $\alpha(q_0) \in P_f$. For $a \in \Sigma^{(0)}$ we define $\delta'_a() = \alpha$ such that for every $q \in Q^{(m+1)}$ and $p_1, \ldots, p_m$, $(\alpha(q))(p_1, \ldots, p_m) = \delta^*(\mathsf{rhs}(q,a)[y_j \leftarrow p_j \mid j \in [m]])$. Here, δ^* is the extension of δ to trees in $T_\Delta(P)$ by the rule $\delta(p) = p$ for all $p \in P$. Now let $b \in \Sigma^{(k)}$ with $k \geq 1$. For $\alpha_1, \ldots, \alpha_k \in S$ we define $\delta'_b(\alpha_1, \ldots, \alpha_k) = \alpha$ where for every $q \in Q^{(m+1)}$ and $p_1, \ldots, p_m$, $(\alpha(q))(p_1, \ldots, p_m) = \underline{\delta}^*(\mathsf{rhs}(q,b)[y_j \leftarrow p_j \mid j \in [m]])$. Now, $\underline{\delta}^*$ is the extension of δ^* of above to symbols $q' \in Q^{(n+1)}$ by $\delta(q'(x_i, p'_1, \ldots, p'_n)) = (\alpha_i(q'))(p'_1, \ldots, p'_n)$ for every $p'_1, \ldots, p'_n \in P$. □

3.1 Bounded Balance

Many algorithms for deciding equivalence of transducers are based on the notion of bounded balance. Intuitively, two transducers have bounded balance if the difference of their outputs on any "partial" input is bounded by a constant. For instance, for two D0L systems $G_i = (\Sigma, h_i, \sigma)$ with $i = 1, 2$, Culik II defines [16] the balance of a string $w \in \Sigma^*$ as the difference of the lengths of $h_1(w)$ and $h_2(w)$. He shows that equivalence is decidable for D0L systems that have bounded difference. In a subsequent article [17], Culik II and Fris show that any two equivalent D0L systems in normal form have bounded difference, thus giving the first solution to the famous D0L equivalence problem.

For a tree transducer M a partial input is an input tree which contains exactly on distinguished leaf labeled x, and x is a fresh symbol not in Σ. More precisely, a partial input is a tree $p = s[u \leftarrow x]$ where s is in the domain of M and u is a node of s. Since the transducer has no rules for x, the computation on the input p "blocks" at the x-labeled node. Thus, the tree $M(p)$ may contain subtrees of the form $q(x, t_1, \ldots, t_m)$ where q is a state of rank $m+1$. For instance, consider the transducer M shown in the left of Figure 2 and consider the partial input tree $s = a(a(x))$. Then

$$M(a(a(x))) = d(M(a(x)), M(a(x))) = d(d(q_0(x), q_0(x)), d(q_0(x), q_0(x))).$$

It should be clear that if we replace each $q_0(x)$ by $M_{q_0}(s')$ so that $s' \in T_\Sigma$ is a tree with $\hat{s} = s[x \leftarrow s'] \in \mathsf{dom}(M)$, then we obtain the output $M(\hat{s})$ of M on input tree $\hat{s}$. For instance, we may pick $s' = e$ above; then we obtain $d(d(e,e), d(e,e))$ which indeed equals $M(a(a(e)))$.

Consider the trees $M_1(p)$ and $M_2(p)$ for two MTTs M_1 and M_2. What is the balance of these two trees? There are two natural notions of balance: either we compare the sizes of $M_i(p)$, or we compare their heights. Given two trees t_1, t_2 we define their *size-balance* (for short, *s-balance*) as $||t_1| - |t_2||$ and their *height-balance* (for short, *h-balance*) as $|\mathsf{height}(s_1) - \mathsf{height}(s_2)|$. Two transducers M_1, M_2 have *bounded s-balance* (resp. h-balance) if there exists a $c > 0$ such that for any partial input p the s-balance (resp. h-balance) of $M_1(p)$ and $M_2(p)$ is at most c. Obviously, bounded h-balance implies bounded s-balance, but not vice versa.

Let M and N be equivalent MTTs. Do M and N have bounded size-balance? To see that this is in general *not* the case, it suffices to consider two simple top-down tree transducers: M translates a monadic partial input of the form $a^n(x)$ into the full binary tree of height n containing 2^n occurrences of the subtree $q_0(x)$. The transducer N translates $a^n(x)$ into the full binary tree of height $n-1$ with 2^{n-1} occurrences of the subtree $p(x)$. The rules of M and N are shown in Figure 2. Clearly, M and N are of bounded

$$
\begin{array}{lll}
M: & q_0(a(x_1)) & \to \; d(q_0(x_1),q_0(x_1)) \\
 & q_0(e) & \to \; e
\end{array}
\qquad\qquad
\begin{array}{lll}
N: & p_0(a(x_1)) & \to \; p(x_1) \\
 & p_0(e) & \to \; e \\
 & p(a(x_1)) & \to \; d(p(x_1),p(x_1)) \\
 & p(e) & \to \; d(e,e)
\end{array}
$$

Figure 2: Equivalent top-down tree transducers M and N with unbounded size-balance

height-balance (with constant $c = 1$). But, their size-balance is not bounded.

In a similar way it can be seen that equivalent MTTs need not have bounded height-balance. This even holds for monadic MTTs: Consider the transducers M' and N' with rules shown in Figure 3. Their

$$
\begin{array}{lll}
M': & q_0(a(x_1)) & \to \; q(x_1,q_0(x_1)) \\
 & q_0(e) & \to \; e \\
 & q(a(x_1),y_1) & \to \; q(x_1,y_1) \\
 & q(e,y_1) & \to \; a(a(y_1))
\end{array}
\qquad
\begin{array}{lll}
N': & p_0(a(x_1)) & \to \; p(x_1,p(x_1,p_0(x_1))) \\
 & p_0(e) & \to \; e \\
 & p(a(x_1),y_1) & \to \; p(x_1,y_1) \\
 & p(e,y_1) & \to \; a(y_1)
\end{array}
$$

Figure 3: Equivalent monadic macro tree transducers M' and N' with unbounded height-balance

height-balance on input $a^n(x)$ is equal to n; for instance, for the tree $a(a(x))$ we obtain

$$
\begin{array}{ll}
M'(a(a(x))) = & N'(a(a(x))) = \\
M'_q(a(x),M'(a(x))) = & N'_p(a(x),N'_p(a(x),N'(a(x)))) = \\
q(x,q(x,q_0(x))) & p(x,p(x,p(x,p(x,p_0(x))))))
\end{array}
$$

which are trees of height 3 and 5, respectively. We may verify that replacing x by e results in equal trees:

$$M'_q(e,M'_q(e,M(e))) = M'_q(e,M'_q(e,e)) = M'_q(e,a(a(e))) = a(a(a(a(e)))).$$

And for N' we obtain that

$$
\begin{aligned}
N'_p(e,N'_p(e,N'_p(e,N'_p(e,N(e))))) = N'_p(e,N'_p(e,N'_p(e,N'_p(e,e)))) = N'_p(e,N'_p(e,N'_p(e,a(e)))) = \\
N'_p(e,N'_p(e,a(a(e)))) = N'_p(e,a(a(a(e)))) = a(a(a(a(e)))).
\end{aligned}
$$

Let us now show that equivalent top-down tree transducers have bounded height-balance.

Lemma 2 Equivalent top-down tree transducers effectively have bounded height-balance.

Proof. Let M_1, M_2 be equivalent top-down tree transducers with sets of states Q_1, Q_2, respectively. Note that they have the same domain D. Let $s \in D$ and $u \in V(s)$. Consider the two trees $\xi_i = M_i(s[u \leftarrow x])$ for $i = 1,2$. Let s' be a smallest input tree such that $s[u \leftarrow s'] \in D$. It should be clear that the height of s' is bounded by some constant d. In fact, let d be the height of a smallest tree in the set $(\cap_{q \in Q} \mathsf{dom}(M_{1,q}) \cap (\cap_{q' \in Q'} \mathsf{dom}(M_{2,q'}))$, for any subsets $Q \subseteq Q_1$ and $Q' \subseteq Q_2$. Since s' is in such a set, its height is at most d.

This bound d can be computed because by Lemma 1 the sets $\mathsf{dom}(M_{1,q})$ and $\mathsf{dom}(M_{2,q'})$ are effectively regular, and regular tree languages are effectively closed under intersection [11]. In fact, it is not difficult to see that we can choose $d = 2^{|Q_1|+|Q_2|}$. Hence, there is a constant c such that $\mathsf{height}(M_{i,q_i}(s')) < c$ for any $q_i \in Q_i$ appearing in ξ_i. Clearly we can take $c = d \cdot h$, where h is the maximal height of the right-hand side of any rule of M_1 and M_2. This means that $|\mathsf{height}(\xi_1) - \mathsf{height}(\xi_2)| \leq c$ because $\xi_1\Theta_1 = \xi_2\Theta_2$ and the substitutions $\Theta_i = [q(x) \leftarrow M_{i,q}(s') \mid q \in Q_i]$ increase the height of ξ_i by at most c. □

If the transducers M_1, M_2 of Lemma 2 are total, then $d = 1$ and c is the maximal size of the right-hand side of any rule for an input leaf symbol, i.e., $c = \max\{\mathsf{height}(\mathsf{rhs}(M_i, q_i, a)) \mid i \in \{1,2\}, q_i \in Q_i, a \in \Sigma^{(0)}\}$.

Let us consider an example of two equivalent top-down tree transducers M and N with output paths

$$
\begin{array}{lllll}
M: & q_0(a(x_1)) & \to & d(q(x_1), q_0(x_1)) \\
 & q_0(e) & \to & e \\
 & q(a(x_1)) & \to & q'(x_1) \\
 & q(e) & \to & e \\
 & q'(a(x_1)) & \to & a(a(q(x_1))) \\
 & q'(e) & \to & a(e)
\end{array}
\qquad
\begin{array}{lllll}
N: & p_0(a(x_1)) & \to & p(x_1) \\
 & p_0(e) & \to & e \\
 & p(a(x_1)) & \to & d(a(p'(x_1)), p(x_1)) \\
 & p(e) & \to & d(e,e) \\
 & p'(a(x_1)) & \to & a(p'(x_1)) \\
 & p'(e) & \to & e
\end{array}
$$

Figure 4: Equivalent top-down tree transducers M and N

of different height. The rules of M and N are given in Figure 4. Let us consider the input tree $s = aaaa(x)$. We omit some parentheses in monadic input trees. We obtain

$$
\begin{aligned}
M(s) = d(M_q(aaax), M(aaax)) = d(M_{q'}(aax), d(M_q(aax), M(aax))) = \\
d(a(a(M_q(ax))), d(M_{q'}(ax), d(M_q(ax), M(ax)))) = \\
d(a(a(q'(x))), d(a(a(q(x))), d(q'(x), d(q(x), q_0(x)))))
\end{aligned}
$$

Similarly, for the transducer N we obtain

$$
\begin{aligned}
N(s) = N_p(aaax) = d(a(N_{p'}(aax)), N_p(aax)) = d(a(a(N_{p'}(ax))), d(a(N_{p'}(ax)), N_p(ax))) = \\
d(a(a(a(p'(x)))), d(a(a(p'(x))), d(a(p'(x)), p(x)))).
\end{aligned}
$$

As the reader may verify, if x is replaced by the leaf e, then indeed the output trees $M(aaaae)$ and $N(aaaae)$ are the same, i.e., the transducers are equivalent. Let us compare the trees $M(aaaax)$ and $N(aaaax)$. On the one hand, the transducer M is "ahead" of the transducer N in the output branch 2.2.2. It has already produced a d-node at that position, while N has not (and is in state p at that position). On the other hand, N is ahead of M at two other positions in the output: at the node 1.1.1 the transducer N has produced an a-node already, while M at that node is in state q', and, at node 2.2.1 the transducer M has output an a-node, while also here M is in state q'.

4 Decidable Equivalence Problems

4.1 Top-Down Tree Transducers

It was shown by Ésik [30] that the bounded height-difference of top-down tree transducers can be used to decide equivalence.

Theorem 1 ([30]) Equivalence of top-down tree transducers is decidable.

Proof. We follow the version of the proof given by Engelfriet [20]. Consider two equivalent top-

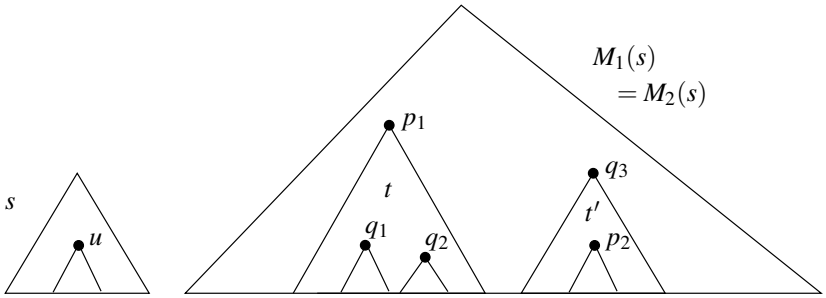

Figure 5: Two equivalent top-down tree transducers

down tree transducers M_1 and M_2. By Lemma 2 they have bounded height-balance by some constant c. Consider the trees $M_1(s[u \leftarrow x])$ and $M_2(s[u \leftarrow x])$. An "overlay" of these two trees is shown in Figure 5 (this is a copy of Figure 10 of [20]). At the node where M_2 is in state p_1, the transducer M_1 has already produced the tree t, i.e., at this node M_1 is "ahead" of M_2 by the amount t. Similarly, at the q_3-labeled node, M_2 is ahead of M_1 by the amount t'. Clearly, the height of t and t' is bounded by c. Hence, there are only finitely many such trees t and t'. We can construct a top-down tree automaton A which in its states keeps track of all such "difference trees" t and t', while simulating the runs of M_1 and M_2. It checks if the outputs are consistent, and rejects if either the outputs are different or if the height of a difference tree is too large. Finally, we check if A accepts the language $D = \mathsf{dom}(M_1) = \mathsf{dom}(M_2)$; this is decidable because D is regular by Lemma 1, and equivalence of regular tree languages is decidable (see [11]). □

Note that Ésik [30] shows that even for single-valued (i.e., functional) nondeterministic top-down tree transducers, equivalence is decidable. It is open whether or not equivalence is decidable for k-valued nondeterministic top-down tree transducers (but believed to be decidable along the same lines as for bottom-up tree transducers [63], cf. the text below Theorem 5). A top-down tree transducer is letter-to-letter if the right-hand side of each rule contains exactly one output symbol in Δ. It was shown by Andre and Bossut [6] that equivalence is decidable for nondeleting *nondeterministic* letter-to-letter top-down tree transducers. An interesting generalization of Theorem 1 is given by Courcelle and Franchi-Zannettacci [15]. They show that equivalence is decidable for "separated" attribute grammars which are evaluated in two independent phases: first a phase that computes all inherited attributes, followed by a phase that computes all synthesized attributes (top-down tree transducers are the special case of synthesized attributes only).

Top-down Tree Transducers with Regular Look-Ahead. Regular look-ahead means that the transducer (MTT or T) comes with a (complete) deterministic bottom-up automaton (without final states), called the "look-ahead automaton of M". A rule of the look-ahead transducer is of the form

$$q(\sigma(x_1,\dots,x_k),\dots) \to t \quad \langle p_1,\dots,p_k\rangle$$

and is applicable to an input tree $\sigma(s_1,\dots,s_k)$ only if the look-ahead automaton recognizes s_i in state p_i for all $1 \le i \le k$. Given two top-down tree transducers with regular look-ahead M_1, M_2, we can transform them into ordinary transducers (without look-ahead) N_1, N_2 such that the resulting transducers

are equivalent if and only if the original ones are. This is done by changing the input alphabet so that for every original input symbol $\sigma \in \Sigma$ of rank k, it now contains the symbols $\langle \sigma, p_1, \ldots, p_k, q_1, \ldots, q_k \rangle$ for all possible look-ahead states p_i of M_1 and q_i of M_2. Thus, for every $\sigma \in \Sigma^{(k)}$, the new input alphabet has $|P|^k |Q|^k$-many symbols. It is easy to see that Lemma 2 also holds for transducers with look-ahead, by additionally requiring that the domain D of the N_i is intersected with all input trees that represent correct runs of the look-ahead automata. Thus, Theorem 1 also holds for top-down tree transducers with look-ahead.

Theorem 2 Equivalence of top-down tree transducers with regular look-ahead is decidable.

Canonical Normal Form. Consider two equivalent top-down tree transducers M_1, M_2 and let D be their domain. As the example transducers M and N with rules in Figure 4 show, for a partial input tree $s[u \leftarrow x]$, there may be positions in the output trees where M_1 is ahead of M_2, and other positions where M_2 is ahead of M_1. Such a scenario is also depicted in Figure 5.

We say that M_1 is *earlier* than M_2, if for every $s \in D$ and $u \in V(s)$, the tree $M_2(s[u \leftarrow x])$ is a prefix of the tree $M_1(s[u \leftarrow x])$. A tree t is a prefix of a tree t' if for every $u \in V(t)$ with $t[u] \in \Delta$ it holds that $t'[u] = t[u]$. The question arises whether for every top-down translation there is an equivalent unique earliest transducer M such that $M_q \neq M_{q'}$ for $q \neq q'$; we call such a transducer a *canonical transducer*. The question was answered affirmative by Engelfriet, Maneth, and Seidl [26]. We only state this result for total transducers.

Theorem 3 Let M be a total top-down tree transducer. An equivalent canonical transducer can be constructed in polynomial time.

Proof. The canonical transducers are top-down tree transducers without an initial state, but with an *axiom tree* $A \in T_{\Delta \cup Q}(\{x_0\})$. This means that the translation on input tree $s \in T_\Sigma$ starts with the tree $A[x_0 \leftarrow s]$ (instead of $q_0(s)$ for ordinary transducers).

Starting with M, we define its axiom $A = q_0(x_0)$. In a first step, an earliest transducer is constructed: if there is a state q and an output symbol δ (of rank k) such that $\mathsf{rhs}(q, \sigma)[\varepsilon] = \delta$ for every input symbol σ, then M is *not* earliest. Intuitively, the symbol δ should be produced earlier, at each call of the state q. Thus, the construction replaces $q(x_i)$ in all right-hand sides (and in the axiom A) by $\delta(\langle q,1\rangle(x_i), \ldots, \langle q,k\rangle(x_i))$ where the $\langle q, j\rangle$ are new states. For every σ, $\mathsf{rhs}(\langle q, j\rangle, \sigma)$ is defined as the j-th subtree of the root of $\mathsf{rhs}(q, \sigma)$ – beware, this right-hand side may have changed due to the replacement above. Finally we remove q and its rules. This step is repeated until it cannot be applied anymore. In this case M has become earliest and no q and δ exists such that $M_q(s) = \delta(\ldots)$ for all input trees s of q. It should be clear that the earliest step can be carried out in polynomial time. In the second step, equivalent states are merged to obtain the canonical transducer; the corresponding equivalence relation on states is computed using fixed point iteration in cubic time (with respect to the size of M). It is computed in such a way that if $q \neq q'$, then $M_q \neq M_{q'}$. □

Note that the availability of a canonical ("minimal") transducer has many advantages. For instance, it makes possible to formulate a Myhill-Nerode like theorem which, in turn, makes possible Gold-style learning of top-down tree transducers (in polynomial time), as shown by Lemay, Maneth, and Niehren [47].

Consider the two transducers M and N with the rules given in Figure 4. To construct a canonical equivalent transducer for M according to Theorem 3, we observe that state q' of M is *not* earliest: the root equals a for the right-hand sides of all q-rules. We replace $q'(x_1)$ by $a(\langle q', 1\rangle(x_1))$ in the (q,a)-rule,

and introduce the two rules $\langle q',1\rangle(a(x_1)) \to a(q(x_1))$ and $\langle q',1\rangle(e) \to e$. We remove q' and have obtained an earliest transducer. The canonical transducer is constructed by realizing that the states $\langle q',1\rangle$ and q are equivalent and hence can be merged. The rules of the canonical transducer can thus be given as

$$\begin{array}{lcl} q_0(a(x_1)) & \to & d(q(x_1),q_0(x_1)) \\ q_0(e) & \to & e \\ q(a(x_1)) & \to & a(q(x_1)) \\ q(e) & \to & e. \end{array}$$

To construct the canonical transducer for N, we observe that state p is not earliest: the root equals d in both rules. We thus replace $p(x_1)$ everywhere by $d(p_1(x_1),p_2(x_1))$ where p_1,p_2 are new states. After this replacement, the current (p,a)-rule is:

$$p(a(x_1)) \to d(a(p'(x_1)),d(p_1(x_1),p_2(x_1))).$$

Thus, new a-rules are $p_1(a(x_1)) \to a(p'(x_1))$ and $p_2(a(x_1)) \to d(p_1(x_1),p_2(x_1))$. The e-rules are $p_1(e) \to e$ and $p_2(e) \to e$. The resulting transducer is earliest. We now compute that $p_1 \equiv p'$ and that $p_2 \equiv p_0$. We merge these pairs of states and obtain the same transducer (up to renaming of states) as the canonical one of M above. Hence, M and N are equivalent.

As a consequence of Theorem 3 we obtain that equivalence of total top-down tree transducers can be decided in polynomial time.

Theorem 4 Equivalence of total top-down tree transducers can be decided in polynomial time.

The earliest normal form has also certain "disadvantages". For instance, it does not preserve linearity (or nondeletingness) of the transducer. Consider for $\Sigma = \{a^{(2)},e^{(0)}\}$ the rules $q(a(x_1,x_2)) \to d(q(x_1),q(x_2))$ and $q(e) \to d(e,e)$. When making earliest, these rules are removed and a rule such as $q_1(f(x_1)) \to q(x_1)$ is replaced by $q_1(f(x_1)) \to d(\langle q,1\rangle(x_1),\langle q,2\rangle(x_1))$ which is non-linear. It is also deleting: $\langle q,1\rangle(a(x_1,x_2)) \to q(x_1)$.

4.2 Bottom-Up Tree Transducers

As a corollary of Theorem 2 we obtain that also for deterministic bottom-up tree transducers, equivalence is decidable. This follows from the fact that every deterministic bottom-up tree transducer can be transformed into an equivalent deterministic top-down tree transducer with regular look-ahead [19].

Theorem 5 Equivalence of deterministic bottom-up tree transducers is decidable.

Proof. A deterministic bottom-up tree transducer is a tuple $B = (\Sigma,\Delta,Q,Q_f,R)$ where $Q_f \subseteq Q$ is the set of final states and R contains for every $k \geq 0$, $\sigma \in \Sigma^{(k)}$, and $q_1,\ldots,q_k \in Q$ at most one rule of the form $\sigma(q_1(x_1),\ldots,q_k(x_k)) \to q(t)$ where t is a tree in $T_\Delta(X_k)$. We construct in linear time a deterministic bottom-up tree automaton which for every rule as above has the transition $\delta_\sigma(q_1,\ldots,q_k) \to q$. This automaton serves as the look-ahead automaton of a top-down tree transducer with the unique state p. For a rule as above, the transducer has the rule

$$p(\sigma(x_1,\ldots,x_k)) \to t[x_i \leftarrow p(x_i) \mid i \in [k]] \quad \langle q_1,\ldots,q_k\rangle.$$

It should be clear that the resulting top-down tree transducer with look-ahead T (which has only the single state p) is equivalent to the given bottom-up tree transducer B. □

The equivalence problem for bottom-up tree transducers was first solved by Zachar [70]. It was shown by Seidl [60] that equivalence can be decided in polynomial time for single-valued (i.e., functional) nondeterministic bottom-up tree transducers. Note that this also follows from Theorem 8 and the

(polynomial time) construction in the proof of Theorem 5. This result was extended to finite-valued nondeterministic bottom-up tree transducers by Seidl [61]. For nondeterministic letter-to-letter bottom-up tree transducers, equivalence was shown decidable by Andre and Bossut [5]; such transducers contain exactly one output symbol in the right-hand side of each rule. They reduce the problem to the equivalence of bottom-up relabelings which was solved by Bozapalidis [9]. For deterministic bottom-up tree transducers the effective existence of a canonical normal form, similar in spirit to the earliest normal form of top-down tree transducers, was shown by Friese, Seidl, and Maneth [36]. They show that this normal form can be constructed in polynomial time, if each state of the given transducer produces either none or infinitely many outputs; hence, equivalence is decidable in polynomial time for such transducers. Friese presents in her PhD thesis [35] a Myhill-Nerode theorem for bottom-up tree transducers.

4.3 Linear Size Increase mtts

It was shown by Engelfriet and Maneth [24] that total deterministic mtts of linear size increase characterize the total deterministic MSO definable tree translations. In fact, even any composition of total deterministic mtts, when restricted to linear size increase, is equal to an MSO definable translation, as shown by Maneth [48]. The MSO definable tree translations are a special instance of the MSO definable graph translations, introduced by Courcelle and Engelfriet, see [12]. Decidability of equivalence for deterministic MSO graph-to-string translations on a context-free graph language was proved by Engelfriet and Maneth [25]. It implies decidable equivalence also for MSO tree translations. We present a proof of the latter here that only uses MTTs and avoids going through MSO.

The idea of the proof stems from Gurari's proof [41] of the decidability of equivalence for 2DGSM. In a nutshell: the ranges of all the above translations are Parikh. A language is *Parikh* if its set of Parikh vectors is equal to the set of Parikh vectors of a regular language. Let $\Sigma = \{a_1, \ldots, a_m\}$ be an alphabet. The *Parikh vector* of a string $w \in \Sigma^*$ is the n-tuple $(i_1, \ldots, i_m)$ of natural numbers i_j such that for $1 \leq j \leq m$, i_j equals the number of occurrences of a_j in w. For a language that is Parikh, it is decidable whether or not it contains a string with Parikh vector $(n, n, \ldots, n)$ for some natural number n. This property is used to prove equivalence as follows. Given two tree-to-string transducers M_1, M_2 we first change M_i to produce a new end marker \$ at the end of each output string. Then, given the regular domain language D of M_1 and M_2, and two distinct output letters a, b we construct the Parikh language

$$L^{a,b} = \{a^m b^n \mid \exists s \in D : M_1(s)/m = a, M_2(s)/n = b\}.$$

Here w/m denotes the m-th letter in the string w. We now decide if there is an n such that $a^n b^n \in L^{a,b}$, using the fact that $L^{a,b}$ is Parikh. If such an n exists, then the transducers M_1, M_2 are *not* equivalent. If, for all possible a, b, no such n exists, then we know that the transducers M_1, M_2 are equivalent.

It was shown by Engelfriet, Rozenberg, and Slutzki in Corollary 3.2.7 of [28] that ranges of *nondeterministic* finite-copying top-down tree transducers with regular look-ahead (for short, N-$\mathrm{T}^{\mathrm{R}}_{\mathrm{fc}}$s) possess the Parikh property. The nondeterminism of this result is useful for defining the language $L^{a,b}$, because we need to nondeterministically choose a and b positions m and n of the output strings. A nondeterministic top-down tree transducer M is *finite-copying* if there is number c such that for every $s \in T_\Sigma$ and $u \in V(s)$, the number of occurrences of states (more precisely, subtrees $q(x)$ such that q is a state of M) in the tree $M(s[u \leftarrow x])$ is $\leq c$. We denote the class of translations of nondeterministic finite-copying top-down tree transducers by N-$\mathrm{T}^{\mathrm{R}}_{\mathrm{fc}}$.

For a tree t we denote by yt its *yield*, i.e., the string of its leaf labels from left to right. For a class X of tree translations we denote by yX the corresponding class of *tree-to-yield* translations. The tree-to-yield translations of top-down tree transducers can be obtained by top-down tree-to-*string* transducers which

have strings over output symbols and state calls $q(x_i)$ in the right-hand sides of their rules. We repeat the argument given in [28]. By REGT we denote the class of regular tree languages, i.e., those languages recognized by (deterministic) finite-state bottom-up tree automata.

Lemma 3 Languages in $y\text{N-T}^{\text{R}}_{\text{fc}}(\text{REGT})$ are Parikh.

Proof. Let M be a $y\text{N-T}^{\text{R}}_{\text{fc}}$ transducer and let $R \in \text{REGT}$. A top-down transducer is *linear* if no x_i appears more than once in any of the right-hand sides of its rules. We construct a linear transducer M' such that $\mathsf{dom}(M') = \mathsf{dom}(M)$ and the string $M'(s)$ is a permutation of the string $M(s)$, for every $s \in \mathsf{dom}(M)$. The new transducer computes in its states the state sequences of M, i.e., the sequence of states that are translating the current input node. Since M is finite-copying, there effectively exists a bound c on the length of the state sequences. For a new state $\langle q_1, \ldots, q_n \rangle$ with $n \leq c$ the right-hand side of a rule is obtained by simply concatenating the right-hand sides of the corresponding rules for q_i. It is well known that linear top-down tree transducers preserve regularity and hence the language $M'(R)$ is in $y\text{REGT}$, i.e., it is the yield language of a regular tree language. The latter is obviously a context-free language (cf. Theorem 3.8 of [67]) which is Parikh by Parikh's theorem [53]. □

A macro tree transducer M is *finite-copying* if there exist constants k and n such that

(1) for every input tree $s' = s[u \leftarrow x]$ with $s \in T_\Sigma$ and $u \in V(s)$, the number of occurrences of states in $M(s')$ is $\leq k$ and

(2) for every state q of rank $m+1$, $1 \leq j \leq m$, and $s \in T_\Sigma$, the number of occurrences of y_j in $M_q(s)$ is $\leq n$.

Recall that an MTT M is of linear size increase if there is a constant c such that $|\tau_M(s)| \leq c \cdot |s|$ for every $s \in T_\Sigma$. We denote the class of translations realized by MTTs of linear size increase by MTT_{lsi}.

Lemma 4 $(y\text{MTT}_{\text{lsi}}) \subseteq y\text{T}^{\text{R}}_{\text{fc}}$.

Proof. It was shown in [24] how to construct a finite-copying macro tree transducer with look-ahead, for a given macro tree transducer of linear size increase. The construction goes through several normal forms which make sure that the transducer generates only finitely many copies; most essentially, the "proper" normal form: each state produces infinitely many output trees, and, each parameter is instantiated by infinitely many trees. By using regular look-ahead finitely many different trees can be determined and outputted directly. The idea of the proper normal form was used already by Aho and Ullmann for top-down tree transducers [2].

It was shown in Lemmas 6.3 and 6.6 of [21] that M can be changed into an equivalent transducer which is "special in the parameters". This means that it is linear and nondeleting in the parameters, i.e., each parameter y_j of a state q appears *exactly once* in the right-hand side of each (q, σ)-rule. The idea is to simply provide multiple parameters, whenever parameters are copied, and to use regular look-ahead in order to determine which parameters are deleted. This was mentioned above Lemma 1 already. For a $y\text{MTT}^{\text{R}}$ transducer that is special in the parameters, it was shown in Lemma 13 of [22] how to construct an equivalent $y\text{T}^{\text{R}}$ transducer. The parameters of the yMTT can be removed by outputting the strings between them directly. Since each parameter appears once, the final string $M_q(s)$ is divided into $m+1$ string chunks w_j (where $m+1$ is the rank of q): $w_0, y_1 w_1, \ldots, y_m w_m$. We leave further details as an exercise, and suggest to start with the case that all y_j appear in strictly increasing order at the leaves of any $M_q(s)$. It is not difficult to see that the construction preserves finite-copying. □

Lemma 5 Let M_1, M_2 be $y\mathrm{T}^{\mathrm{R}}_{\mathrm{fc}}$ transducers with input and output alphabets Σ and Δ, and let $a, b \in \Delta$ with $a \neq b$. Let $D \subseteq T_\Sigma$ be a regular tree language. The language $L^{a,b} = \{a^m \# b^n \mid \exists s \in D : M_1(s)/m = a, M_2(s)/n = b\}$ is Parikh.

Proof. Let us assume that the state sets Q_1, Q_2 of the transducers M_1, M_2 are disjoint. The initial state of M_1, M_2 is q_0 and p_0, respectively. We first construct a $y\mathrm{N}\text{-}\mathrm{T}^{\mathrm{R}}_{\mathrm{fc}}$ transducer M_1' such that

$$M_1'(s) = \{ua \mid u \in \Delta^*, \exists v \in \Delta^* : uav = M_1(s)\}.$$

Its state set is $Q_0 = Q_1 \cup \{q_a \mid q \in Q_1\}$ and its initial state is $q_{0,a}$. It has all rules of M_1 and, moreover, for every rule $q(\sigma(x_1,\ldots,x_k)) \to w \quad \langle \cdots \rangle$ of M_1, whenever $w = uav$ it has the rule $q_a(\sigma(x_1,\ldots,x_k)) \to ua \quad \langle \cdots \rangle$, and whenever $w = uq'(x_i)v$ it has the rule $q_a(\sigma(x_1,\ldots,x_k)) \to uq'_a(x_i) \quad \langle \cdots \rangle$. From M_1' one obtains a $y\mathrm{N}\text{-}\mathrm{T}^{\mathrm{R}}_{\mathrm{fc}}$ transducer M_1'' such that

$$M_1''(s) = \{a^m \mid M_1(s)/m = a\}$$

by simply changing all symbols of Δ into a in the rules of M_1'. Similarly, one obtains a transducer M_2'' such that $M_2''(s) = \{b^n \mid M_2(s)/n = b\}$. Finally, a $y\mathrm{N}\text{-}\mathrm{T}^{\mathrm{R}}_{\mathrm{fc}}$ transducer M is defined such that $M(s) = \{a^m \# b^n \mid M_1(s)/m = a, M_2(s)/n = b\}$. Its state set is $\{r_0\} \cup Q_1' \cup Q_2'$ with initial state r_0. The look-ahead automaton of M is the product automaton of the look-ahead automata of M_1 and M_2. The set of rules of M is the union of those of M_1'' and M_2'', adapted to the new look-ahead appropriately. Moreover, for $\sigma \in \Sigma^{(k)}$, $k \geq 0$, and rules $q_{0,a}(\sigma(x_1,\ldots,x_k)) \to u \quad \langle q_1',\ldots,q_k' \rangle$ and $p_{0,b}(\sigma(x_1,\ldots,x_k)) \to w \quad \langle p_1',\ldots,p_k' \rangle$, we let

$$r_0(\sigma(x_1,\ldots,x_k)) \to u \# w \quad \langle (q_1',p_1'),\ldots,(q_k',p_k') \rangle$$

be a rule of M. Obviously, $M(s)$ equals the concatenation $M_1''(s) \# M_2''(s)$, and is finite-copying. Since $M(D) = L^{a,b}$ it follows by Lemma 3 that $L^{a,b}$ is Parikh. □

Theorem 6 *Equivalence of deterministic macro tree transducers of linear size increase is decidable.*

Proof. Let M_1, M_2 be MTT transducers of linear size increase. We first check that the domains of M_i coincide. This is decidable because $\mathsf{dom}(M_i)$ is effectively regular by Lemma 1. If not then the transducers are not equivalent and we are finished. Otherwise, let D be their domain. We may consider M_i as tree-to-string transducers, by considering the tree in the right-hand side of each rule as a string (which uses additional terminals symbols for denoting the tree structure such as opening and closing parentheses and commas). Thus, by Lemma 4 (which is effective) we may in fact assume that M_1 and M_2 are $y\mathrm{T}^{\mathrm{R}}_{\mathrm{fc}}$ transducers. Let Δ be the output alphabet of M_i and let \$ be a new symbol not in Δ. We change M_i so that each output string is followed by the \$ symbol. This can easily be done by first splitting the initial state q_0 so that it appears in the right-hand side of no rule, and then adding \$ to the end of each q_0-rule. It now holds that M_1 and M_2 are *not equivalent* if and only if there exist $a, b \in \Delta$ with $a \neq b$, $s \in D$, and a number n such that $M_1(s)/n = a$ and $M_2(s)/n = b$. The latter holds if the intersection of $L^{a,b}$ of Lemma 5 with the language $E = \{a^n \# b^n \mid n \in \mathbb{N}\}$ is nonempty. Since $L^{a,b}$ is Parikh by Lemma 5, we obtain decidability because semilinear sets are closed under intersection [39, 38] and have decidable emptiness. But, there is a much easier proof: $E \cap L$ is context-free, because E is (by the well-known "triple construction", see, e.g., Theorem 6.5 of [44]), where L is a regular language with the same Parikh vectors as $L^{a,b}$. The result follows since context-free grammars have decidable emptiness. □

4.4 Monadic mtts

Recall that a macro tree transducer is monadic if both its input and output alphabet are monadic, i.e., consist of symbols of rank one and rank zero only. We will reduce the equivalence problem for monadic MTT transducers to the sequence equivalence problem of HDT0L systems. An MTT is *nondeleting* if for every state q of rank $m+1$, $1 \leq j \leq m$, and input symbol σ, the parameter y_j occurs in $\mathsf{rhs}(q,\sigma)$. A monadic MTT $M = (Q, \Sigma, \Delta, q_0, R)$ is *normalized* if

(N0) it is nondeleting

(N1) each state is of rank two or one, i.e., $Q = Q^{(2)} \cup Q^{(1)}$ and

(N2) there is only one input and output symbol of rank zero, i.e., $\Sigma^{(0)} = \Delta^{(0)} = \{\bot\}$

Note that for total transducers (N1) is a consequence of (N0) because a $(q,\bot)$-rule can only contain at most one parameter occurrence.

HDT0L systems An instance of the HDT0L sequence equivalence problem consists of finite alphabets Σ and Δ, two strings $w_1, w_2 \in \Sigma^*$, homomorphisms $h_j, g_j : \Sigma^* \to \Sigma^*$, $1 \leq j \leq n$, and homomorphisms $h, g : \Sigma^* \to \Delta^*$. To solve the problem we have to determine whether or not

$$h(h_{i_k}(\cdots h_{i_1}(w_1)\cdots)) = g(g_{i_k}(\cdots g_{i_1}(w_2)\cdots))$$

holds true for all $k \geq 0$, $1 \leq i_1, \ldots, i_k \leq n$. This problem is known to be decidable. It was first proven by Culik II and Karhumäki [18], using Ehrenfeucht's Conjecture and Makanin's algorithm. A later proof of Ruohonen [59] is based on the theory of metabelian groups. Yet another, very short, proof was given by Honkala [43] which only relies on Hilbert's Basis Theorem. We now show that the equivalence problem for total monadic MTT transducers can be reduced to the sequence equivalence problem for HDT0L systems. For a monadic tree $s = a_1(\cdots a_n(e)\cdots)$ we denote by $\mathsf{strip}(s)$ the string $a_1 \cdots a_n$.

Lemma 6 *Equivalence of total monadic normalized* MTT*s on a regular input language is decidable.*

Proof. We first solve the problem without a given input tree language. Let $M_1 = (Q_1, \Gamma, \Pi, q_0, R_1)$ and $M_2 = (Q_2, \Gamma, \Pi, p_0, R_2)$ be total monadic normalized macro tree transducers such that Q_1 is disjoint from Q_2. Let $Q = Q_1 \cup Q_2$. We define an instance of the HDT0L sequence equivalence problem. The string alphabets Σ, Δ are defined as $\Sigma = \Pi^{(1)} \cup Q$ and $\Delta = \Pi^{(1)}$. We define homomorphisms h_a, g_a for every input symbol $a \in \Gamma^{(1)}$. For $\pi \in \Pi^{(1)}$ let $h_a(\pi) = g_a(\pi) = \pi$. Let $q \in Q$. If $q \in Q_1$ then let $h_a(q) = \mathsf{strip}(\mathsf{rhs}_{M_1}(q,a))$, and otherwise let $h_a(q) = q$. If $q \in Q_2$ then let $g_a(q) = \mathsf{strip}(\mathsf{rhs}_{M_2}(q,a))$, and otherwise let $g_a(q) = q$. For trees $t \in T_{\Pi \cup Q}(X_k \cup Y_m)$ we define the mapping strip by $\mathsf{strip}(\pi(t)) = \pi \cdot \mathsf{strip}(t)$ for $\pi \in \Gamma^{(1)}$, $\mathsf{strip}(q(x_1,t)) = q \cdot \mathsf{strip}(t)$ for $q \in Q^{(2)}$, $\mathsf{strip}(q(x_1)) = q$ for $q \in Q^{(1)}$, and $\mathsf{strip}(\bot) = \mathsf{strip}(y_1) = \varepsilon$, where "$\cdot$" denotes string concatenation. The final homomorphisms h, g are defined as $h(q) = \mathsf{strip}(\mathsf{rhs}_{M_1}(q,\bot))$ if $q \in Q_1$, and otherwise $h(q) = q$, and $g(q) = \mathsf{strip}(\mathsf{rhs}_{M_2}(q,\bot))$ if $q \in Q_2$, and otherwise $g(q) = q$. Last but not least, let $w_1 = q_0$ and $w_2 = p_0$. This ends the construction of the HDT0L instance. Consider an input tree $s = a_1(\cdots a_n(\bot)\cdots) \in T_\Gamma$. It should be clear that $h(h_{a_n}(\cdots h_{a_1}(w_1)\cdots)) = \mathsf{strip}(M_1(s))$ and that $g(g_{a_n}(\cdots g_{a_1}(w_2)\cdots)) = \mathsf{strip}(M_2(s))$. Thus, this instance of the HDT0L sequence equivalence problem solves the equivalence problem of the two transducers M_1 and M_2.

Let $D \subseteq T_\Gamma$ be a regular input tree language. We wish to decide whether $M_1(s) = M_2(s)$ for every $s \in D$. We assume that D is given by a deterministic finite-state automaton A that runs top-down on the unary symbols in $\Gamma^{(1)}$. We further assume that $A = (R, \Gamma^{(1)}, r_0, \delta, R_f)$ is complete, i.e., for every state $r \in R$ and every symbol $a \in \Gamma^{(1)}$, $\delta(r,a)$ is defined (and in R). Note that r_0 is the initial state and

$R_f \subseteq R$ is the set of final states. Let $\Sigma = \Pi^{(1)} \cup Q$ as before and define $\Sigma' = \{\langle r,b\rangle \mid r \in R, b \in \Sigma\}$ and $\Delta = \Pi^{(1)}$. Our HDT0L instance is over Σ' and Δ. Let $a \in \Gamma^{(1)}$, $r \in R$, and $r' = \delta(r,a)$. For $\pi \in \Pi^{(1)}$ let $h_a(\langle r,\pi\rangle) = g_a(\langle r,\pi\rangle) = \langle r',\pi\rangle$. Let $q \in Q_1$ and $p \in Q_2$. Define

$$\begin{aligned} h_a(\langle r,q\rangle) &= \mathsf{strip}(\mathsf{rhs}_{M_1}(q,a))[b \leftarrow \langle r',b\rangle \mid b \in \Sigma] \\ g_a(\langle r,p\rangle) &= \mathsf{strip}(\mathsf{rhs}_{M_2}(p,a))[b \leftarrow \langle r',b\rangle \mid b \in \Sigma]. \end{aligned}$$

Let $h_a(\langle r,p\rangle) = \langle r,p\rangle$ and $g_a(\langle r,q\rangle) = \langle r,q\rangle$. The final homomorphisms g,h are defined as follows. If $r \in R_f$ then let $h(\langle r,q\rangle) = \mathsf{strip}(\mathsf{rhs}_{M_1}(q,\bot))$, $g(\langle r,p\rangle) = \mathsf{strip}(\mathsf{rhs}_{M_2}(p,\bot))$, and let $h(\langle r,b\rangle) = b$ and $g(\langle r,b\rangle) = b$ for the remaining cases. If $r \notin R_f$ then let $h(\langle r,b\rangle) = g(\langle r,b\rangle) = \varepsilon$ for every $b \in \Sigma$. The initial strings are defined as $w_1 = \langle r_0,q_0\rangle$ and $w_2 = \langle r_0,p_0\rangle$.

Consider an input tree $s = a_1(\cdots a_n(\bot)\cdots) \in T_\Gamma$ and let $1 \le j \le n$. It should be clear that if $\delta^*(r_0,a_1\cdots a_j) = r$, i.e., A arrives in state r after reading the prefix $a_1\cdots a_j$, then

$$h_{a_j}(\cdots h_{a_1}(w_1)\cdots) = \mathsf{strip}(M_1(a_1\cdots a_j(x)))[\pi \leftarrow \langle r,\pi\rangle \mid \pi \in \Delta][q \leftarrow \langle r,q\rangle \mid q \in Q_1]$$

and similarly for g and M_2. Thus each and every symbol of a sentential form is labeled by the current state of the automaton A. Hence, if $s \notin D$, then every symbol in $u_1 = h_{a_n}(\cdots h_{a_1}(w_1)\cdots)$ and in $u_2 = g_{a_n}(\cdots g_{a_1}(w_2)\cdots)$ is labeled by some state $r \notin R_f$. This implies that $h(u_1) = g(u_2) = \varepsilon$, i.e., the final strings are equal whenever $s \notin D$. If on the contrary $s \in D$ then every symbol in u_i is labeled by a final state and therefore $h(u_i) = \mathsf{strip}(M_i(s))$ as before. □

Input and output symbols of rank zero of a given transducer become symbols of rank one in the corresponding normalized transducer. For a monadic tree $t = a_1(\cdots a_n(e)\cdots)$ we denote by $\mathsf{expand}(t)$ the tree $a_1(\cdots a_n(e(\bot))\cdots)$.

Lemma 7 *For every monadic* MTT^R *transducer M a normalized* MTT^R *transducer N can be constructed such that $\tau_N = \{(\mathsf{expand}(s),\mathsf{expand}(t)) \mid (s,t) \in \tau_M\}$.*

Proof. Using regular look-ahead we first make M nondeleting. As mentioned in the proof of Lemma 4, this construction was given in the proof of Lemma 6.6 of [21]. Now, every parameter that appears in the left-hand side of a rule, also appears in the right-hand side. Since the final output tree is monadic, the resulting transducer satisfies (N1) above. Finally, we define the MTT^R transducer N which has input and output alphabets $\Sigma' = \Sigma^{(1)} \cup \{a'^{(1)} \mid a \in \Sigma^{(0)}\}$ and $\Delta' = \Delta^{(1)} \cup \{a'^{(1)} \mid a \in \Delta^{(0)}\}$. For input symbols in $\Sigma^{(1)}$ the transducer N has exactly the same rules as M. Let $q \in Q$ and $a \in \Sigma^{(0)}$ such that $\mathsf{rhs}(q,a)$ is defined. Then we let

$$q(a'(x_1)) \to \mathsf{rhs}(q,a)[b \leftarrow b'(\bot) \mid b \in \Delta^{(0)}].$$

be a rule of N. Regular look-ahead can be used to ensure that only trees of the form $\mathsf{expand}(s)$ are in the domain of N. □

Obviously, two monadic MTT transducers are equivalent if and only if their normalized versions are equivalent. Hence, it suffices to consider the equivalence problem of normalized monadic MTT transducers.

Theorem 7 *Equivalence of monadic macro tree transducers with regular look-ahead is decidable.*

Proof. Let M_1,M_2 be monadic macro tree transducers with regular look-ahead and let Σ be their input alphabet. Let A_1,A_2 be the look-ahead automata of M_1,M_2. By Lemma 7 we may assume that

M_1 and M_2 are normalized. We first check if the domains of M_1 and M_2 coincide. If not then the transducers are not equivalent and we are finished. Otherwise, let D be their domain. We define two total monadic MTTs N_1, N_2 *without* look-ahead. Let P_1, P_2 be the sets of states of A_1, A_2, respectively. The input alphabet of N_i is defined as $\Sigma' = \{\langle \sigma, p_1, p_2 \rangle \mid \sigma \in \Sigma^{(1)}, p_1 \in P_1, p_2 \in P_2\}$. An input symbol $\langle \sigma, p_1, p_2 \rangle$ denotes that the look-ahead automata at the child of the current node are in states p_1 and p_2, respectively. Thus, the $(q, \langle \sigma, p_1, p_2 \rangle)$-rule of N_1 is defined as the (q, σ)-rule with look-ahead $\langle p_1 \rangle$ of M_1, and the $(q, \langle \sigma, p_1, p_2 \rangle)$-rule of N_2 is defined as the (q, σ)-rule with look-ahead $\langle p_2 \rangle$ of M_2. Finally, we make N_1 and N_2 total (in some arbitrary way).

For a tree t in $T_{\Sigma'}$ we denote by $\gamma(t)$ the tree in T_Σ obtained by changing every label $\langle \sigma, p, p' \rangle$ into the label σ. Let $E \subseteq T_{\Sigma'}$ be the regular tree language consisting of all trees t such that

(1) $s = \gamma(t)$ is in D,

(2) the second components of the labels in t constitute a correct run of A_1 on s, and

(3) the third components of the labels in t constitute a correct run of A_2 on s.

Clearly, for the resulting transducers N_i it holds that N_1 and N_2 are equivalent on E if and only if M_1 is equivalent to M_2. Hence decidability of equivalence follows from Lemma 6. □

Note that macro tree transducers with monadic output alphabet are essentially the same as top-down tree-to-string transducers (see Lemma 7.6 of [21]). For the latter, the equivalence problem was stated already in 1980 by Engelfriet [20] as a big open problem. This problem remains open, but, as this section has shown, at least for the restricted case of monadic input, we obtain decidability. Note further that the connection between L-systems and tree transducers is well known and was studied extensively in [28].

5 Complexity

In Section 4 we already mentioned one complexity result, viz. Theorem 4, which states that equivalence can be decided in polynomial time for total top-down tree transducers. How about top-down tree transducers (TS) in general? It was mentioned in the Conclusions of [7] that checking equivalence of TS can be done in double exponential time, using the procedure of [26].

Without giving details we now present a proof that strengthens both results above (and which also works for transducers with look-ahead). We show that equivalence for TS can be decided in EXPSPACE, and for total TS in NLOGSPACE. For a top-down tree transducer M and trees s, t with $t = M(s)$, it holds that each node v in the output tree t is produced by one particular node u in s. The latter is called v's origin. It means that $M(s[u \leftarrow x])$ does not have a Δ-node v, while v is a Δ node in $M[s \leftarrow a(x, \ldots, x)]$ where $a = s[u]$.

Theorem 8 Equivalence of top-down tree transducers with regular look-ahead is decidable in EXPSPACE, and for total transducers in NLOGSPACE.

Proof. We sketch the proof for transducers without look-ahead. Since both complexity classes are closed under complement, it suffices to consider nonequivalence. Consider two top-down tree transducers M_1 and M_2. The idea (as in the finite-copying case) is to guess (part of) an input tree s and a node v of the output trees $t_1 = M_1(s)$ and $t_2 = M_2(s)$ such that $t_1[v] \neq t_2[v]$. It suffices to guess the two *origins* of v with respect to M_1 and M_2: nodes u_1 and u_2 of s, respectively. More precisely, it suffices to guess the paths from the root of s to u_1 and u_2, and the path from the root of t_1 and t_2 to v, where we may assume

that all proper ancestors of v have the same label in t_1 and t_2. When guessing the path from the root of s to the least common ancestor of u_1 and u_2, the path in t_1 can be ahead of the path in t_2, or vice versa, so the difference between these paths must be stored. But it suffices to keep the length of this difference to be at most exponential in the sizes of M_1 and M_2, due to the bounded height-balance of M_1 and M_2 in case they are equivalent. In the proof of Lemma 2 the height of the smallest tree s' is at most exponential, and hence the height of its translation is at most exponential. Hence the difference between the paths in t_1 and t_2 can be stored in exponential space.

If M_1 and M_2 are total, then the difference between the paths in t_1 and t_2 is at most a path in a right-hand side of a rule, which can be kept in logarithmic space. Logarithmic space is also needed to do all the guesses, of course. The same proof as above also holds for transducers with regular look-ahead. □

Theorem 9 *Equivalence of top-down tree transducers is* EXPTIME*-hard.*

Proof. It is well known that testing intersection emptiness of n deterministic top-down tree automata $A_1,\ldots,A_n$ is EXPTIME-complete. This was shown by Seidl [62], cf. also [11]. Let Σ be the ranked alphabet of the A_i. We define the top-down tree transducer $M_1 = (\{q_0,\ldots,q_n\},\Sigma,\Sigma\cup\{\delta^{(n)}\},q_0,R)$. We consider each A_i as a partial identity transducer with start state q_i, and add the corresponding rules to R. Thus, $M_{1,q_i} = \{(s,s) \mid s \in L(A_i)\}$. Let $\sigma \in \Sigma$ be an arbitrary symbol of rank ≥ 1, and let e be an arbitrary symbol in $\Sigma^{(0)}$. We add these two rules to R:

$$\begin{array}{lcl} q_0(\sigma(x_1,\ldots)) & \to & \delta(q_1(x_1),q_2(x_1),\ldots,q_n(x_1)) \\ q_0(e) & \to & e \end{array}$$

The transducer $M_2 = (\{p\},\Sigma,\Sigma,p,\{p(e)\to e\})$ realizes the translation $\tau_{M_2} = \{(e,e)\}$. If the intersection of the $L(A_i)$ is empty, then there is no tree $s \in T_\Sigma$ such that $M_{q_i}(s)$ is defined for all $i \in \{1,\ldots,n\}$, i.e., the first rule displayed above is never applicable. Hence, in this case also $\tau_{M_1} = \{(e,e)\}$, i.e., the transducers M_1, M_2 are equivalent. If the intersection is non-empty, then there is an input tree s such that $M(s) = \delta(s,s,\ldots,s)$. Thus, M_1 is equivalent to M_2 if and only if the intersection of the $L(A_i)$ is empty. □

5.1 Streaming Tree Transducers

The (deterministic) streaming tree transducers of Alur and d'Antoni are a new model with the same expressive power as deterministic MSO tree translations which in turn realize the same translations as deterministic macro tree translations of linear size increase. The idea of the model is to use a finite set of variables which hold partial outputs. These variables are updated during a single depth-first left-to-right traversal of the input tree. It is stated in Theorem 20 of [3] that equivalence of streaming tree transducers can be decided in exponential time. The idea of the proof is the same as the one in Theorem 6: construct a context-free language $L^{a,b}$ and use its Parikhness to check if a^nb^n is in the language. For them, $L^{a,b}$ is represented by a pushdown automaton A, the number of states of which is exponential in the number of variables of the given streaming tree transducer. They mention that checking if a^nb^n is in $L(A)$ can be done in NPTIME using [31, 64].

Theorem 10 ([3]) *Equivalence of streaming tree transducers is decidable in* CO-NEXPTIME.

For the transducers that map strings to nested strings, that is, for streaming string-to-tree transducers their construction yields a PSPACE bound (Theorem 21 of [3]).

Theorem 11 ([3]) *Equivalence of streaming string-to-tree transducers is decidable in* PSPACE.

5.2 Visibly Pushdown Transducers

Visibly pushdown languages were defined by Alur and Madhusudan [4] as a particular subclass of the context-free languages. In fact, they are just regular tree languages in disguise. Visibly pushdown transducers were introduced by Raskin and Servais [56]. They translate well-nested input strings into strings, during one left-to-right traversal of the input. If the output strings are nested as well, then they describe tree transformations. The expressive power of the resulting tree transformations is investigated by Caralp, Filiot, Reynier, Servais, and Talbot [10]. Such transducers cannot copy nor swap the order of input trees. Thus, they are MSO definable. But they are incomparable to the top-down or bottom-up tree translations, because they can translate a tree into its yield (string of leaf labels from left to right).

Theorem 12 ([32]) *Equivalence of functional visibly pushdown transducers is* EXPTIME*-complete. For total such transducers the problem is in* PTIME.

The EXPTIME-completeness result extends to the case of regular look-ahead, as shown in Section 8.4 of [33, 65]. Staworko, Laurence, Lemay, and Niehren [66] have considered the equivalence problem for deterministic visibly pushdown transducers and show that it can be reduced in PTIME to the homomorphic equivalence problem on context-free grammars. The latter was shown by Plandowski [45, 55] to be solvable in PTIME. They show in [66] that for several related classes the problem is in PTIME, for instance, linear and order-preserving deterministic top-down and bottom-up tree transducers.

Theorem 13 ([66]) *Equivalence of deterministic visibly pushdown transducers is decidable in* PTIME.

6 Conclusion

We discussed the decidability of equivalence for three incomparable subclasses of deterministic macro tree transducers: top-down tree transducers, linear size increase MTTs, and monadic MTTs. For top-down tree transducers the proof either uses its bounded height-balance property and constructs an automaton that keeps track of the balance. Alternatively, such transducers may be transformed into their canonical normal form and then be checked for isomorphism. For these decision procedures it is not "harmful" that a top-down tree transducer can copy a lot and be of exponential size increase, because the multiple copies of equivalent transducers must be well-nested into each other (cf. Figure 5). This nesting property is not present for MTTs, and in particular the bounded height-balance does not hold for MTTs, even not for monadic ones. Thus, other techniques are needed in these two cases. For the linear size increase subclass of MTTs we may use the Parikh property of the corresponding output languages: the two transducers are merged ("twinned") to output $a^m b^n$ if, on the same input, one transducer produces at position m of its output the letter a while the other transducer produces at position n the letter b. Since this output language is Parikh, we may decide if it contains $a^n b^n$ which implies that the transducers are not equivalent (because $a \neq b$). For monadic MTTs we use yet another technique: we simulate the transducers by HDT0L sequences. Since the sequence equivalence problem for HDT0L systems is decidable (not detailed here), the result follows. It remains a deep open problem whether or not equivalence is decidable for arbitrary deterministic macro tree transducers. Even for MTTs with monadic output, which are the same as deterministic top-down tree-to-string transducers, it is open whether or not equivalence is decidable. Note that the availability of a canonical normal form is a much stronger result than the decidability of equivalence: for instance, equivalence is easily decided for top-down transducers with look-ahead, but, for such transducers we only know a canonical normal form in the total case for a fixed look-ahead automaton [27]. In fact, even to decide whether or not a given T^{R} is equivalent to a T is a difficult open problem; it was solved recently for a subclass of T^{R}s [27].

References

[1] A. V. Aho (1968): *Indexed Grammars - An Extension of Context-Free Grammars*. *J. ACM* 15(4), pp. 647–671, doi:10.1145/321479.321488.

[2] A. V. Aho & J. D. Ullman (1971): *Translations on a Context-Free Grammar*. *Information and Control* 19(5), pp. 439–475, doi:10.1016/S0019-9958(71)90706-6.

[3] R. Alur & L. D'Antoni (2011): *Streaming Tree Transducers*. *CoRR* abs/1104.2599.

[4] R. Alur & P. Madhusudan (2004): *Visibly pushdown languages*. In: *STOC*, pp. 202–211, doi:10.1145/1007352.1007390.

[5] Y. Andre & F. Bossut (1995): *The Equivalence Problem for Letter-to-Letter Bottom-up Tree Transducers is Solvable*. In: *TAPSOFT*, pp. 155–171, doi:10.1007/3-540-59293-8_193.

[6] Y. Andre & F. Bossut (1998): *On the Equivalence Problem for Letter-to-Letter Top-Down Tree Transducers*. *Theor. Comput. Sci.* 205(1-2), pp. 207–229, doi:10.1016/S0304-3975(97)00080-7.

[7] M. Benedikt, J. Engelfriet & S. Maneth (2013): *Determinacy and Rewriting of Top-Down and MSO Tree Transformations*. In: *MFCS*, pp. 146–158, doi:10.1007/978-3-642-40313-2_15.

[8] J. Berstel (1979): *Transductions and context-free languages*. *Teubner, Stuttgart.*

[9] S. Bozapalidis (1992): *Alphabetic Tree Relations*. *Theor. Comput. Sci.* 99(2), pp. 177–211, doi:10.1016/0304-3975(92)90348-J.

[10] M. Caralp, E. Filiot, P.-A. Reynier, F. Servais & J.-M. Talbot (2013): *Expressiveness of Visibly Pushdown Transducers*. In: *TTATT*, pp. 17–26, doi:10.4204/EPTCS.134.3.

[11] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, C. Löding, D. Lugiez, S. Tison & M. Tommasi (2007): *Tree Automata Techniques and Applications*. Available at: http://www.grappa.univ-lille3.fr/tata.

[12] B. Courcelle & J. Engelfriet (2012): *Graph Structure and Monadic Second-Order Logic - A Language-Theoretic Approach*. *Encyclopedia of mathematics and its applications* 138, Cambridge University Press, doi:10.1017/CBO9780511977619.

[13] B. Courcelle & P. Franchi-Zannettacci (1982): *Attribute Grammars and Recursive Program Schemes I*. *Theor. Comput. Sci.* 17, pp. 163–191, doi:10.1016/0304-3975(82)90003-2.

[14] B. Courcelle & P. Franchi-Zannettacci (1982): *Attribute Grammars and Recursive Program Schemes II*. *Theor. Comput. Sci.* 17, pp. 235–257, doi:10.1016/0304-3975(82)90024-X.

[15] B. Courcelle & P. Franchi-Zannettacci (1982): *On the Equivalence Problem for Attribute Systems*. *Information and Control* 52(3), pp. 275–305, doi:10.1016/S0019-9958(82)90786-0.

[16] K. Culik II (1976): *On the Decidability of the Sequence Equivalence Problem for D0L-Systems*. *Theor. Comput. Sci.* 3(1), pp. 75–84, doi:10.1016/0304-3975(76)90066-9.

[17] K. Culik II & I. Fris (1977): *The Decidability of the Equivalence Problem for D0L-Systems*. *Information and Control* 35(1), pp. 20–39, doi:10.1016/S0019-9958(77)90512-5.

[18] K. Culik II & J. Karhumäki (1986): *A new proof for the D0L Sequence Equivalence Problem and its implications*, doi:10.1007/978-3-642-95486-3_5. In G. Rozenberg & A. Salomaa, editors: *The book of L*, Springer, Berlin, pp. 63–74.

[19] J. Engelfriet (1977): *Top-down Tree Transducers with Regular Look-ahead*. *Mathematical Systems Theory* 10, pp. 289–303, doi:10.1007/BF01683280.

[20] J. Engelfriet (1980): *Some open questions and recent results on tree transducers and tree languages*. In R. V. Book, editor: *Formal Language Theory; Perspectives and Open Problems*, Academic Press, New York.

[21] J. Engelfriet & S. Maneth (1999): *Macro Tree Transducers, Attribute Grammars, and MSO Definable Tree Translations*. *Inf. Comput.* 154(1), pp. 34–91, doi:10.1137/S0097539701394511.

[22] J. Engelfriet & S. Maneth (2002): *Output String Languages of Compositions of Deterministic Macro Tree Transducers*. *J. Comput. Syst. Sci.* 64(2), pp. 350–395, doi:10.1006/jcss.2001.1816.

[23] J. Engelfriet & S. Maneth (2003): *A comparison of pebble tree transducers with macro tree transducers*. Acta Inf. 39(9), pp. 613–698, doi:10.1007/s00236-003-0120-0.

[24] J. Engelfriet & S. Maneth (2003): *Macro Tree Translations of Linear Size Increase are MSO Definable*. SIAM J. Comput. 32(4), pp. 950–1006, doi:10.1137/S0097539701394511.

[25] J. Engelfriet & S. Maneth (2006): *The equivalence problem for deterministic MSO tree transducers is decidable*. Inf. Process. Lett. 100(5), pp. 206–212, doi:10.1016/j.ipl.2006.05.015.

[26] J. Engelfriet, S. Maneth & H. Seidl (2009): *Deciding equivalence of top-down XML transformations in polynomial time*. J. Comput. Syst. Sci. 75(5), pp. 271–286, doi:10.1016/j.jcss.2009.01.001.

[27] J. Engelfriet, S. Maneth & H. Seidl (2013): *Look-Ahead Removal for Top-Down Tree Transducers*. CoRR abs/1311.2400.

[28] J. Engelfriet, G. Rozenberg & G. Slutzki (1980): *Tree Transducers, L Systems, and Two-Way Machines*. J. Comput. Syst. Sci. 20(2), pp. 150–202, doi:10.1016/0022-0000(80)90058-6.

[29] J. Engelfriet & H. Vogler (1985): *Macro Tree Transducers*. J. Comput. Syst. Sci. 31(1), pp. 71–146, doi:10.1016/0022-0000(85)90066-2.

[30] Z. Ésik (1981): *Decidability results concerning tree transducers I*. Acta Cybern. 5(1), pp. 1–20.

[31] J. Esparza (1997): *Petri Nets, Commutative Context-Free Grammars, and Basic Parallel Processes*. Fundam. Inform. 31(1), pp. 13–25, doi:10.3233/FI-1997-3112.

[32] E. Filiot, J.-F. Raskin, P.-A. Reynier, F. Servais & J.-M. Talbot (2010): *Properties of Visibly Pushdown Transducers*. In: MFCS, pp. 355–367, doi:10.1007/978-3-642-15155-2_32.

[33] E. Filiot & F. Servais (2012): *Visibly Pushdown Transducers with Look-Ahead*. In: SOFSEM, pp. 251–263, doi:10.1007/978-3-642-27660-6_21.

[34] M. J. Fischer (1968): *Grammars with Marcro-like Productions*. Ph.D. thesis, Harvard University.

[35] S. Friese (2011): *On Normalization and Type Checking for Tree Transducers*. Ph.D. thesis, Institut für Informatik, Technische Universität München. Available at http://mediatum.ub.tum.de/doc/1078090/1078090.pdf.

[36] S. Friese, H. Seidl & S. Maneth (2011): *Earliest Normal Form and Minimization for Bottom-up Tree Transducers*. Int. J. Found. Comput. Sci. 22(7), pp. 1607–1623, doi:10.1142/S012905411100891X.

[37] Z. Fülöp & H. Vogler (1998): *Syntax-Directed Semantics - Formal Models Based on Tree Transducers*. Monographs in Theoretical Computer Science. An EATCS Series, Springer, doi:10.1007/978-3-642-72248-6.

[38] S. Ginsburg (1966): *The Mathematical Theory of Context-Free Languages*. McGraw-Hill.

[39] S. Ginsburg & E. H. Spanier (1964): *Bounded ALGOL-like languages*. Trans. Amer. Math. Soc 113, pp. 333–368, doi:10.2307/1994067.

[40] T. V. Griffiths (1968): *The Unsolvability of the Equivalence Problem for Lambda-Free Nondeterministic Generalized Machines*. J. ACM 15(3), pp. 409–413, doi:10.1145/321466.321473.

[41] E. M. Gurari (1982): *The Equivalence Problem for Deterministic Two-Way Sequential Transducers is Decidable*. SIAM J. Comput. 11(3), pp. 448–452, doi:10.1137/0211035.

[42] S. Hakuta, S. Maneth, K. Nakano & H. Iwasaki (2014): *XQuery Streaming by Forest Transducers*. In: ICDE, pp. 417–428.

[43] J. Honkala (2000): *A short solution for the HDT0L sequence equivalence problem*. Theor. Comput. Sci. 244(1-2), pp. 267–270, doi:10.1016/S0304-3975(00)00158-4.

[44] J. E. Hopcroft & J. D. Ullman (1979): *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley.

[45] J. Karhumäki, W. Plandowski & W. Rytter (1995): *Polynomial Size Test Sets for Context-Free Languages*. J. Comput. Syst. Sci. 50(1), pp. 11–19, doi:10.1006/jcss.1995.1002.

[46] D. E. Knuth (1968): *Semantics of Context-Free Languages*. Mathematical Systems Theory 2(2), pp. 127–145, doi:10.1007/BF01692511.

[47] A. Lemay, S. Maneth & J. Niehren (2010): *A learning algorithm for top-down XML transformations*. In: *PODS*, pp. 285–296, doi:10.1145/1807085.1807122.

[48] S. Maneth (2003): *The Macro Tree Transducer Hierarchy Collapses for Functions of Linear Size Increase*. In: *FSTTCS*, pp. 326–337, doi:10.1007/978-3-540-24597-1_28.

[49] S. Maneth, A. Berlea, T. Perst & H. Seidl (2005): *XML type checking with macro tree transducers*. In: *PODS*, pp. 283–294, doi:10.1145/1065167.1065203.

[50] S. Maneth, T. Perst & H. Seidl (2007): *Exact XML Type Checking in Polynomial Time*. In: *ICDT*, pp. 254–268, doi:10.1007/11965893_18.

[51] T. Milo, D. Suciu & V. Vianu (2003): *Typechecking for XML transformers*. *J. Comput. Syst. Sci.* 66(1), pp. 66–97, doi:10.1016/S0022-0000(02)00030-2.

[52] K. Nakano & S.-C. Mu (2006): *A Pushdown Machine for Recursive XML Processing*. In: *APLAS*, pp. 340–356, doi:10.1007/11924661_21.

[53] R. Parikh (1966): *On Context-Free Languages*. *J. ACM* 13(4), pp. 570–581, doi:10.1145/321356.321364.

[54] T. Perst & H. Seidl (2004): *Macro forest transducers*. *Inf. Process. Lett.* 89(3), pp. 141–149, doi:10.1016/j.ipl.2003.05.001.

[55] W. Plandowski (1994): *Testing Equivalence of Morphisms on Context-Free Languages*. In: *ESA*, pp. 460–470.

[56] J.-F. Raskin & F. Servais (2008): *Visibly Pushdown Transducers*. In: *ICALP (2)*, pp. 386–397, doi:10.1007/978-3-540-70583-3_32.

[57] W. C. Rounds (1969): *Context-Free Grammars on Trees*. In: *STOC*, pp. 143–148, doi:10.1145/800169.805428.

[58] W. C. Rounds (1970): *Mappings and Grammars on Trees*. *Mathematical Systems Theory* 4(3), pp. 257–287, doi:10.1007/BF01695769.

[59] K. Ruohonen (1986): *Equivalence problems for regular sets of word morphisms*, doi:10.1007/978-3-642-95486-3_33. In G. Rozenberg & A. Salomaa, editors: *The book of L*, Springer, Berlin, pp. 393–401.

[60] H. Seidl (1992): *Single-Valuedness of Tree Transducers is Decidable in Polynomial Time*. *Theor. Comput. Sci.* 106(1), pp. 135–181, doi:10.1016/0304-3975(92)90281-J.

[61] H. Seidl (1994): *Equivalence of Finite-Valued Tree Transducers Is Decidable*. *Mathematical Systems Theory* 27(4), pp. 285–346, doi:10.1007/BF01192143.

[62] H. Seidl (1994): *Haskell Overloading is DEXPTIME-Complete*. *Inf. Process. Lett.* 52(2), pp. 57–60, doi:10.1016/0020-0190(94)00130-8.

[63] H. Seidl (2014): *Private Communication*.

[64] H. Seidl, T. Schwentick, A. Muscholl & P. Habermehl (2004): *Counting in Trees for Free*. In: *ICALP*, pp. 1136–1149, doi:10.1007/978-3-540-27836-8_94.

[65] F. Servais (2011): *Visibly Pushdown Transducers*. Ph.D. thesis, Université Libre de Bruxelles.

[66] S. Staworko, G. Laurence, A. Lemay & J. Niehren (2009): *Equivalence of Deterministic Nested Word to Word Transducers*. In: *FCT*, pp. 310–322, doi:10.1007/978-3-642-03409-1_28.

[67] J. W. Thatcher (1970): *Generalized Sequential Machine Maps*. *J. Comput. Syst. Sci.* 4(4), pp. 339–367, doi:10.1016/S0022-0000(70)80017-4.

[68] H. Vogler (1991): *Functional Description of the Contextual Analysis in Block-Structured Programming Languages: A Case Study of Tree Transducers*. *Sci. Comput. Program.* 16(3), pp. 251–275, doi:10.1016/0167-6423(91)90009-M.

[69] J. Voigtländer (2005): *Tree transducer composition as program transformation*. Ph.D. thesis, Technical University Dresden.

[70] Z. Zachar (1979): *The solvability of the equivalence problem for deterministic frontier-to-root tree transducers*. *Acta Cybern.* 4(2), pp. 167–177.