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 longstanding open problem.

Keywords: Equivalence; tree transducer.

The macro tree transducer (mtt) was invented independently by Engelfriet [26, 37] and Courcelle [18, 17] (see also [46]). As a model of syntax-directed translations, mtts generalize the attribute grammars of Knuth [56]. Note that one (annoying) issue of attribute grammars (and attributed tree transducers [45]) is that they can be circular, and that testing for circularity is expensive (it is exptime-complete [54]). In contrast, mtts always terminate.

There are two ways in which macro tree transducers can be seen as a combination of context-free tree grammars, invented by Rounds [71] and also known as “macro tree grammars” [42], and the top-down tree transducer of Rounds and Thatcher [72, 81]. (1) In terms of the context-free tree grammar, the derivation of the grammar is (top-down) controlled by a given input tree (see [27] for the idea of grammars controlled by input storage). (2) In terms of a top-down tree transducer, the state calls of the transducer can appear in a nested way (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 gsm, see [47, 9]).

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 of Damm [23] (at level one they include the indexed languages of Aho [1]), see also [29]. The latter hierarchies can be obtained by restricting higher-order recursive program schemes.
to their “safe” variant, see Ong’s survey [66].

Macro tree transducers can be applied in many scenarios, recently for instance, to type-check XML transformations (they can simulate the $k$-pebble transducers of Milo, Suciu, and Vianu [64]), see [30, 61, 62], or to efficiently implement streaming XQuery transformations [51, 65]. In terms of functional programs, MTTS are particularly simple programs that use primitive recursion over an input tree to produce output trees by top-concatenation. Applications in programming languages include for instance [82, 83, 65, 7].

Let us now focus our attention on the equivalence problem for tree transducers. For nondeterministic transducers, Griffiths [49] shows that equivalence is undecidable already for very restricted string transducers. Therefore we 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. This survey discusses three of them:

1. top-down tree transducers,
2. linear size increase transducers, and
3. monadic tree transducers.

For the first class, decidability was established already in 1980 by Ésik [38]. The idea of Ésik’s proof is to show that equivalent top-down tree transducers have “bounded balance”, i.e., the difference of their outputs on any partial input is bounded by a constant (which can be computed from the transducers). It then suffices to construct a tree automaton that keeps track of the differences of the outputs and rejects if they become larger than the bound; see Section 3.

The equivalence problem for top-down tree transducers was recently revived by Engelfriet, Maneth, and Seidl through the “earliest canonical normal form” [33]. They show that every top-down tree translation has a canonical representation by a fixed transducer (up to state renaming). This canonical transducer always produces output as early as possible, and has no two equivalent states. While Ésik’s procedure had not given rise to complexity results, the earliest canonical normal form implies that equivalence of total top-down tree transducers can be decided in polynomial time.

The second class of MTTS, those of linear size increase, is equal to the class of MSO tree transducers [31]; the paper also shows how to decide whether a given MT is MSO definable, and if so how to construct an equivalent MSO transducer. The latter class is the restriction to trees of the MSO graph transducer of Courcelle (see [15]) and Engelfriet, see [16] for a recent survey. For MSO tree transducers it is shown by Engelfriet and Maneth [32] that equivalence is decidable. The proof is based on the Parikh-property of the ranges of such transducers, in a similar way as Gurari’s proof for two-way string transducers [50]. In this survey we present a proof that works directly on macro tree transducers and does not go through MSO.

The third class, of MTTS over monadic trees, can be seen as a class of string-
to-string translations obtained by string transducers with copying. The decidability of equivalence for this class follows through a relationship with L-systems [36]; in particular, with the sequence equivalence problem of HDT0L systems. The latter was first shown decidable by Culik II and Karhumäki [22]. Later proofs are given by Ruohonen [73] and Honkala [52].

1. 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 each node such labeled. Formally, a ranked alphabet consists of a finite set Σ together with a mapping rankΣ : Σ → {0, 1, . . .} associating to each symbol its rank. We write σ(k) to denote that the rank of σ is equal to k. By Σ(k) we denote the subset of symbols of Σ that have rank k. We write rankΣ0∪Σ to denote the subset of symbols of Σ that have rank 0. Let Σ be a ranked alphabet. The set of all trees over Σ, denoted TΣ, is the smallest set of strings T such that if k ≥ 0, t1, . . . , tk ∈ T, and σ ∈ Σ(k), then also σ(t1, . . . , tk) ∈ T. For a tree of the form a() we simply write a. For a set A with A ∩ Σ = ∅, we denote by TΣ∪A the set of all trees over Σ∪A such that the rank of each a ∈ A is zero. Let a1, . . . , an be distinct symbols in Σ(0) and let t1, . . . , tn be trees in TΣ. Then [a1 ← t1 | i ∈ {1, . . . , n}] denotes the tree substitution that replaces each leaf labeled a_i by the tree t_i; recall that a leaf is a node with no children. Thus d(a, b, a)[a ← c(b), b ← a] denotes the tree d(c(b), a, c(b))

Let t ∈ TΣ for some ranked alphabet Σ. We identify the nodes of t by their Dewey dotted decimal paths and define the set V(t) of nodes of t as {ε} ∪ {i.u | 1 ≤ i ≤ k, u ∈ V(t_i)} if t = σ(t1, . . . , tk) with k ≥ 0, σ ∈ Σ(k), and t1, . . . , tk ∈ TΣ. Thus, ε denotes the root node, and u.i denotes the i-th child of the node u. For a node u ∈ V(t) we denote by t[u] its label, and, for a tree t’ we denote by t[u ← 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 height(t), is one plus the length of the longest path in V(t), i.e., height(t) = 1 + max{|u| | u ∈ V(t)|}.

A deterministic finite-state bottom-up tree automaton is a tuple A = (Q, Σ, δ, Q_f) where Q is a finite set of states, Q_f ⊆ Q is the set of final states, and for every σ ∈ Σ(k) and k ≥ 0, δσ is a partial function from Qk to Q. The transition function δ is extended to a partial function from TΣ to Q in the obvious way, and the set of trees accepted by A is L(A) = {s ∈ TΣ | δ(s) ∈ Q_f}. If L = L(A) for some deterministic finite-state bottom-up tree automaton A, then L is called a regular tree language. The class of all regular tree languages is denoted by REGT.

We fix the disjoint sets of input variables X = {x_1, x_2, . . .} and of formal context parameters Y = {y_1, y_2, . . .}, and will assume that all other ranked alphabets are disjoint with X and Y. For n ∈ {1, 2, . . .} we define X_n = {x_1, . . . , x_n} and Y_n = {y_1, . . . , y_n}.

Let f : A → B be a partial function. The domain of f, i.e., the set {a ∈ A | f(a) is defined}, is denoted by dom(f).
1.1. Macro Tree Transducers

This section introduces macro tree transducers and proves the basic property that their inverse translations effectively preserve the regular tree languages.

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 \( Q^{(0)} = \emptyset \), \( \Sigma \) and \( \Delta \) are ranked alphabets of input and output symbols, respectively (with \( \Delta \) disjoint from \( Q \)), \( q_0 \in Q^{(1)} \) 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

\[
q(\sigma(x_1, \ldots, x_k), y_1, \ldots, y_m) \rightarrow t
\]

where \( t \in T_{\Delta \cup \Sigma}(X_k \cup Y_m) \) such that a node in \( t \) has its label in \( X_k \) if and only if it is the first child of a node that has its label in \( Q \). A rule as above is called a \((q, \sigma)\)-rule and its right-hand side is denoted by \( \text{rhs}(q, \sigma) \). The set \( R \) contains no rules of any other form than the above (thus, \( |R| \leq |Q| \cdot |\Sigma| \)).

The translation \( \tau_M : T_{\Sigma} \rightarrow T_{\Delta} \) realized by an \( \text{mtt} \) \( M \) is the partial function recursively defined as follows. For each state \( q \) of rank \( m + 1 \), \( M_q : T_{\Sigma} \rightarrow T_{\Delta}(Y_m) \) 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 \( \text{rhs}(q, a) \). In general, for an input tree \( s = \sigma(s_1, \ldots, s_k) \), \( M_q(s) \) is obtained from \( \text{rhs}(q, \sigma) \) 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 \( M_{q'}(s_j)[y_j \leftarrow t_j \mid 1 \leq j \leq n] \). The latter tree will also be written as \( M_{q'}(s_i, t_1, \ldots, t_n) \). We define \( \tau_M = M_{q_0} \) and often write \( M(s) \) instead of \( \tau_M(s) \). The domain of \( M \), denoted \( \text{dom}(M) \), is \( \text{dom}(\tau_M) \).

Observe that in the replacement of trees \( q'(x_i, t_1, \ldots, t_n) \) defined above, it is required that the trees \( t_i \) do not contain states. This bottom-up kind of replacement is often called “inside-out” (io) or “eager”, or “call-by-value”.

Two \( \text{mtts} \) \( M_1 \) and \( M_2 \) are equivalent if they realize the same translation, i.e., if \( \tau_{M_1} = \tau_{M_2} \).

By the definition above, all our \( \text{mtts} \) are deterministic. We refer to them as “macro tree transducers” and denote their class of translations by \( \text{mtt} \). Observe that in the literature this class is usually denoted \( \text{mtt}_{\text{io}} \) or \( \text{mtt}_{\text{io}} \). An \( \text{mtt} \) is total if for every state \( q \) and input symbol \( \sigma \) it has exactly one \((q, \sigma)\)-rule. If each state of an \( \text{mtt} \) \( M \) is of rank one, then \( M \) is called a top-down tree transducer. The class of translations realized by (deterministic) top-down tree transducers is denoted by \( T \).

In Lemmas 10 and 12 we make use of nondeterministic top-down tree transducers and mark the corresponding class there by the letter “N”. A transducer is nondeterministic if it has several \((q, \sigma)\)-rules. An \( \text{mtt} \) is monadic if its input and output ranked alphabets are monadic, i.e., only contain symbols of rank 1 and 0. An \( \text{mtt} \) \( M \) is of linear size increase if there is a number \( c \) such that \( |\tau_M(s)| \leq c \cdot |s| \) for every \( s \in T_{\Sigma} \).

Figure 1 shows the rules of an \( \text{mtt} \) \( M \). The alphabets of \( M \) are \( Q = \{q_0^{(1)}, q_0^{(2)}\} \),
\[ \Sigma = \{d, e\}, \text{ and } \Delta = \{d^2, a^{(1)}, 1^{(1)}, 2^{(1)}, e^{(0)}\}. \]

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, \( s = d(a, a), a \) is translated into \( d(d(a(1(e))), a(2(1(e))), a(2(e))) \) as can be seen by this computation of \( M \):

\[
\begin{align*}
M(d(a, a), a) &= d(M_q(d(a, a), 1(e)), M_q(a, 2(e))) \\
&= d(d(M_q(a, 1(1(e))), M_q(a, 2(1(e)))), a(2(e))) \\
&= d(d(a(1(1(e))), a(2(1(e))), a(2(e)))
\end{align*}
\]

Note that \( M \) is not of linear size increase. Hence, its translation is not MSO definable. Since the transducer is neither top-down nor monadic, it falls into a class of MTTs for which we do not know a procedure to decide equivalence. Note that there is no top-down tree transducer equivalent to \( M \); this follows from the facts that output path languages of top-down tree transducers are regular (by Theorem 4 of [72]) and that the output path language \( L \) of \( M \) is non-regular (path languages are over \( \Delta^{(0)} \) and symbols \( a_i \) with \( a \in \Delta^{(k)} \) and \( 1 \leq i \leq k \); the intersection of \( L \) with \( d^i_1 \) equals \( \{d^n_1 e \mid n \geq 0\} \) which is non-regular, hence \( L \) is non-regular).

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 basic and most useful properties of MTTs is the effective preservation of regular tree languages by their inverse translations. We will apply this property several times throughout this paper. Recall from the Preliminaries the definition of a bottom-up tree automaton.

**Lemma 2.** Let \( M \) be an MTT with output alphabet \( \Delta \) 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 \( \text{dom}(\tau_M) \) is effectively regular.

**Proof.** Let \( M = (Q, \Sigma, \Delta, q_0, R) \) and \( B = (P, \Delta, \delta, P_f) \). Without loss of generality we assume that \( \delta_d \) is a mapping from \( P^k \) to \( P \) for every \( d \in \Delta^{(k)} \) and \( k \geq 0 \). We construct the tree automaton \( A = (S, \Sigma, \delta', S_f) \) such that \( L(A) = \tau_M^{-1}(L(B)) \). The
states of \( A \) are partial functions \( \alpha \) 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 \( \alpha \in S \) for which \( \alpha(q_0) \in P_f \).

Let \( s \) be an input tree in \( T_S \). The automaton \( A \) is constructed in such a way that if \( \delta(s) = \alpha \), then \( \alpha \) captures the behavior of \( B \) on the trees \( M_q(s) \) for all states \( q \) of \( M \). Let \( q \in Q^{(m+1)} \). Recall that \( M_q(s) \) is a tree in \( T_\Delta(Y_m) \). The behavior of \( B \) on a tree \( \xi \) in \( T_\Delta(Y_m) \) is expressed by a mapping from \( P^m \) to \( P \); if the tree in parameter \( y_j \) is accepted by \( B \) in state \( p_j \), then \( (\alpha(q))(p_1, \ldots, p_m) \) is the state that \( B \) reaches at the root of \( \xi \). Thus,

\[
(\alpha(q))(p_1, \ldots, p_m) = \delta^*(M_q(s)[y_j \leftarrow p_j \mid j \in \{1, \ldots, m\}])
\]

where \( \delta^* \) is the extension of \( \delta \) to trees in \( T_\Delta(P) \) by \( \delta^*(p) = p \) for all \( p \in P \).

Formally, for \( a \in \Sigma^{(0)} \) we define \( \delta^*_a() = \alpha \) such that for every \( q \in Q^{(m+1)} \) and \( p_1, \ldots, p_m \in P \), \( (\alpha(q))(p_1, \ldots, p_m) = \delta^*(\text{rhs}(q,a)[y_j \leftarrow p_j \mid j \in \{1, \ldots, m\}]) \).

For \( b \in \Sigma^{(k)} \) with \( k \geq 1 \) and \( \alpha_1, \ldots, \alpha_k \in S \) we define \( \delta^*_b(\alpha_1, \ldots, \alpha_k) = \alpha \) where for every \( q \in Q^{(m+1)} \) and \( p_1, \ldots, p_m \in P \), \( (\alpha(q))(p_1, \ldots, p_m) = \delta^*(\text{rhs}(q,b)[y_j \leftarrow p_j \mid j \in \{1, \ldots, m\}]) \). The mapping \( \delta^* \) is the extension of \( \delta^* \) to symbols \( q' \in Q^{(m+1)} \) by \( \delta^*(q'(x_1, t_1, \ldots, t_n)) = (\alpha_1(q'))(\delta^*(t_1), \ldots, \delta^*(t_n)) \) for every \( t_1, \ldots, t_n \in T_\Delta(P) \).[\( \square \]

What is the number \( |S| \) of states of the tree automaton \( A \) in the construction of the proof of Lemma 2? Let \( m \) be the maximal rank of the states of the \( \text{MTT} \) \( M \).

Since \( |F|^{|E|} \) equals the number of functions from \( E \) to \( F \), we obtain \( |S| \leq |P|^{|Q|P|^m} \), i.e., the bound is doubly exponential in \( m \). Already for top-down tree transducers (i.e., \( m = 1 \)) it follows from the results of Martens and Neven [63] that constructing the automaton \( A \) is \( \text{EXPTIME-complete} \).

The result of Lemma 2 is stated in Theorem 7.4 of [37]. A proof similar to the one above is given at the end of [30]. A slightly different proof, for a larger class, is presented by Perst and Seidl for macro forest transducers [68].

2. 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. As an example, let \( G_i = (\Sigma, h_i, \alpha_i) \) with \( i = 1, 2 \) be two \( \text{DOL} \) systems. A \( \text{DOL} \) system is a string rewriting system consisting of an alphabet \( \Sigma \), a homomorphism \( h : \Sigma \to \Sigma \), and an axiom string \( \alpha \in \Sigma^* \). It produces the language \( \{h^n(\alpha) \mid n \geq 0\} \). Culik II defines in [20] the balance of a string \( w \in \Sigma^* \) as the difference in length of the outputs \( h_1(w) \) and \( h_2(w) \). He shows that equivalence is decidable for \( \text{DOL} \) systems that have bounded balance. In a subsequent article [21], Culik II and Fris show that any two equivalent \( \text{DOL} \) systems in normal form have bounded balance, thus giving the first solution to the famous \( \text{DOL} \) equivalence problem.

For a macro tree transducer \( M \) a partial input is an input tree that contains exactly one distinguished leaf labeled \( x \), where \( x \) is a fresh symbol not in \( \Sigma \). More
precisely, a partial input is a tree $s_x = s[u \leftarrow x]$ 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 $s_x$ “blocks” at the $x$-labeled node. Thus, the tree $M(s_x)$ can 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 the partial input tree $s_x = a(a(x))$. 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 $s \in T^n$ is a tree with $s' = s_x[u \leftarrow s] \in \text{dom}(M)$, then we obtain the output $M(s')$ of $M$ on input tree $s'$. E.g., for $s = e$ we obtain $d(d(e, e), d(e, e))$ which equals $M(a(a(e)))$.

Compare the trees $M_1(s_x)$ and $M_2(s_x)$ for two mttts $M_1$ and $M_2$ and a partial input $s_x$. What is the balance of these two trees? There are two natural notions of balance: either we compare the sizes of $M_1(s_x)$, 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 $|\text{height}(t_1) - \text{height}(t_2)|$. Two transducers $M_1, M_2$ have bounded $s$-balance (resp. $h$-balance) if there exists a number $c > 0$ such that for any partial input $s_x$ the $s$-balance (resp. $h$-balance) of $M_1(s_x)$ and $M_2(s_x)$ is at most $c$. Obviously, bounded $h$-balance implies bounded $s$-balance, but not vice versa.

$$M_1 : q_0(a(x_1)) \rightarrow d(q_0(x_1), q_0(x_1))$$
$$q_0(e) \quad \rightarrow e$$

$$M_2 : p_0(a(x_1)) \rightarrow p(x_1)$$
$$p_0(e) \quad \rightarrow e$$
$$p(a(x_1)) \quad \rightarrow d(p(x_1), p(x_1))$$
$$p(e) \quad \rightarrow d(e, e)$$

Fig. 2. Equivalent top-down tree transducers $M_1$ and $M_2$ with unbounded size-balance

Let $M_1$ and $M_2$ be equivalent mttts. Do $M_1$ and $M_2$ have bounded size-balance? To see that this is in general not the case, consider the two top-down tree transducers $M_1$ and $M_2$ with rules shown in Figure 2. On (monadic) input $s_x = a^n(x)$, $M_1$ outputs a full binary tree of height $n$ containing $2^n$ occurrences of the subtree $q_0(x)$, while $M_2$ outputs a similar tree but of height $n - 1$. Clearly, $M_1$ and $M_2$ are of bounded height-balance (with constant $c = 1$), but their size-balance is not bounded.

Before we prove in Lemma 3 that equivalent top-down tree transducers have bounded height-balance, we show that for equivalent mttts this is not the case: There are monadic mttts $M_1$ and $M_2$ such that their height-balance on input $a^n(x)$ equals $n$. The rules of $M_1$ and $M_2$ are given in Figure 3. For instance, on the partial
input of $M$ are trees of height 4 and 6, respectively. If we replace $x$ where $D$ is bounded height-balance.

**Lemma 3.** Equivalent deterministic 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 $M_1$ and $M_2$ have the same domain $D$. Let $s \in D$ and $u \in V(s)$. Define the two trees $\xi_i = M_i(s[u \leftarrow x])$ for $i = 1, 2$. Let $s'$ be an input tree of smallest height 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 number

$$\max \{ \min \{ \text{height}(t) \mid t \in D(Q'_1, Q'_2) \} \mid Q'_i \subseteq Q_i, i = 1, 2 \}$$

where $D(Q'_1, Q'_2) = \cap\{ \text{dom}(M_{1,q_1}) \mid q_1 \in Q'_1 \} \cap (\cap\{ \text{dom}(M_{2,q_2}) \mid q_2 \in Q'_2 \}$ for $Q'_i \subseteq Q_i$ and $i = 1, 2$. Then $s' \in D(Q'_1, Q'_2)$ where, for $i = 1, 2, Q'_i$ is the set of states appearing in $\xi_i$. Thus, $\text{height}(s')$ is in the set of which $d$ is the maximum, i.e., $\text{height}(s') \leq d$. This bound $d$ can be computed because by Lemma 2 the sets $\text{dom}(M_{i,q_i})$ for $i = 1, 2$ are effectively regular, and regular tree languages are effectively closed under intersection [14]. In fact, it is not difficult to see that we can
choose $d = 2^{|Q_1|+|Q_2|}$. Hence, there is a number $c$ such that $\text{height}(M_{i,q_i}(s')) < c$ for any $q_i \in Q_1$ appearing in $\xi_i$. 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 $|\text{height}(\xi_1) - \text{height}(\xi_2)| \leq c$ because $\xi_1 \Theta_1 = \xi_2 \Theta_2$ and the substitutions $\Theta_i = [q(x) \leftarrow M_{i,q_i}(s') \mid q \in Q_i]$ increase the height of $\xi_i$ by at most $c$.

If the transducers $M_1, M_2$ of Lemma 3 are total, then $d = 1$ and $c$ is the maximal height of the right-hand side of any rule for an input leaf symbol, i.e., $c = \max\{\text{height}(\text{rhs}(M_{i,q_i}, a)) \mid i \in \{1, 2\}, q \in Q_i, a \in \Sigma^{(0)}\}$.

\[
\begin{align*}
M_1: q_0(a(x_1)) &\rightarrow d(q(x_1), q_0(x_1)) & M_2: p_0(a(x_1)) &\rightarrow p(x_1) \\
q_0(e) &\rightarrow e & p_0(e) &\rightarrow e \\
q(a(x_1)) &\rightarrow q'(x_1) & p(a(x_1)) &\rightarrow d(a(p'(x_1)), p(x_1)) \\
q(e) &\rightarrow e & p(e) &\rightarrow d(e, e) \\
q'(a(x_1)) &\rightarrow a(a(q(x_1))) & p'(a(x_1)) &\rightarrow a(p'(x_1)) \\
q'(e) &\rightarrow a(e) & p'(e) &\rightarrow e
\end{align*}
\]

Fig. 4. Equivalent top-down tree transducers $M_1$ and $M_2$

Figure 4 shows the rules of two top-down tree transducers $M_1$ and $M_2$. These transducers are equivalent and hence have bounded height-balance. The purpose of the example is to show that on the same partial input, $M_1$ produces an output path that $M_2$ has not produced, and $M_2$ produces an output path that $M_1$ has not produced. We denote the monadic tree $a(\cdots a(\_\cdots))$ with $n$ occurrences of $a$ by $a^n(\_)$, where $\_$ denotes any symbol of rank zero. Let $s_x = a^4(x)$ be a partial input tree.

\[
\begin{align*}
M_1(s_x) &= d(M_{1,q_1}(a^3(x)), M_1(a^4(x))) \\
&= d(M_{1,q_1}(a^2(x)), d(M_{1,q}(a^2(x)), M_1(a^2(x)))) \\
&= d(a^2(M_{1,q_1}(a(x))), d(M_{1,q}(a(x)), d(M_{1,q_1}(a(x)), M_1(a(x)))) \\
&= d(a^2(q(x)), d(a^2(q(x)), d(q(x), q_0(x))))
\end{align*}
\]

For the transducer $M_2$ we obtain

\[
\begin{align*}
M_2(s_x) &= M_{2,p_2}(a^3(x)) \\
&= d(a(M_{2,p_2}(a^2(x))), M_{2,p_2}(a(x))) \\
&= d(a^2(M_{2,p_2}(a(x))), d(M_{2,p_2}(a(x)), M_{2,p_2}(a(x))) \\
&= d(a^3(p'(x)), d(a^2(p'(x)), d(a(p'(x)), p(x))))
\end{align*}
\]

As the reader may verify, if $x$ is replaced by the leaf $e$, then indeed the output trees $M_1(a^4(e))$ and $M_2(a^4(e))$ are the same, i.e., the transducers are equivalent. Let us compare the trees $M_1(a^4(x))$ and $M_2(a^4(x))$. On the one hand, the transducer $M_1$ is
“ahead” of the transducer $M_2$ on the output branch 2.2.2. It has already produced a $d$-node at that position, while $M_2$ has not (and is in state $p$ at that position). On the other hand, $M_2$ is ahead of $M_1$ at two other positions in the output: at the node 1.1.1 the transducer $M_2$ has produced an $a$-node already, while $M_1$ at that node is in state $q'$, and, at node 2.2.1 the transducer $M_2$ has output an $a$-node, while also here $M_1$ is in state $q'$.

3. Equivalence of Top-Down and Bottom-Up Tree Transducers

In this section we first show that top-down tree transducers have decidable equivalence. The proof uses the fact that such transducers have bounded height-balance, according to Lemma 3. We then extend this result to top-down tree transducers with regular look-ahead, by arguing their bounded height-balance. An alternative proof for the decidability of top-down tree transducers is given in Section 3.3. It is based on the idea of a canonical normal form. For total transducers this method implies a polynomial time equivalence check. Finally, Section 3.4 establishes the decidability of equivalence for bottom-up tree transducers. This is done by presenting the well-known result that such transducers can be simulated by top-down tree transducers with regular look-ahead.

3.1. Top-Down Tree Transducers

Ésik [38] uses the bounded height-balance property of top-down tree transducers to decide their equivalence. This proof is given now, following the presentation of Engelfriet [26].

**Theorem 4 ([38])** Equivalence of deterministic top-down tree transducers is decidable.

**Proof.** Let $M_1$ and $M_2$ be two equivalent top-down tree transducers. By Lemma 3

![Diagram of overlapping output trees of two equivalent top-down tree transducers](image-url)

Fig. 5. Overlapping output trees of two equivalent top-down tree transducers

they have bounded height-balance by some constant $c$. Consider the trees $M_1(s)[u \leftarrow t] = M_2(s)[u \leftarrow t']$. Since $M_1$ and $M_2$ are equivalent, there exists a sequence of state transitions in $M_1$ that simulates the state transitions in $M_2$ on the input $s$. This sequence of state transitions in $M_1$ is bounded by some constant $c$. Therefore, $M_1(s)[u \leftarrow t]$ and $M_2(s)[u \leftarrow t']$ have bounded height-balance by some constant $c$. Since $M_1$ and $M_2$ are equivalent, they have the same output on input $s$. Therefore, $M_1(s)[u \leftarrow t] = M_2(s)[u \leftarrow t']$.
x) and \(M_2(s[u \leftarrow x])\). An “overlay” of these two trees is shown in Figure 5. 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 \(D = \text{dom}(M_1) = \text{dom}(M_2)\); this is decidable because \(D\) is regular by Lemma 2, and equivalence of regular tree languages is decidable (see [14]).

Note that \(\check{\text{Esik}}\) [38] 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 [77], cf. the text below Theorem 9). A top-down tree transducer is letter-to-letter if the right-hand side of each rule contains exactly one output symbol in \(\Delta\). It is shown by André and Bossut [6] that equivalence is decidable for nondeleting nondeterministic letter-to-letter top-down tree transducers. An interesting generalization of Theorem 4 is given by Courcelle and Franchi-Zannettacci [19]. They show that equivalence is decidable for “separated” attribute grammars which are evaluated in two independent phases: 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).

### 3.2. Regular Look-Ahead

A macro tree transducer can be equipped with regular look-ahead by associating with it a deterministic bottom-up tree automaton (without final states), called the “look-ahead automaton”. Formally a macro tree transducer with regular look-ahead (\(\text{mtt}^R\)) \(M\) is a tuple \((Q, \Sigma, \Delta, q_0, R, \delta)\) where \(P\) is the finite set of look-ahead states and \(\delta\) is a partial function from \(P^k\) to \(P\) for every \(\sigma \in \Sigma^{(k)}\). A rule of \(M\) is of the form

\[
q(\sigma(x_1, \ldots, x_k), y_1, \ldots, y_m) \rightarrow t \quad \langle p_1, \ldots, p_k \rangle
\]

and is applicable to an input tree \(\sigma(s_1, \ldots, s_k)\) only if the look-ahead automaton \((P, \Sigma, \delta, \_\_\_)\) accepts \(s_i\) in state \(p_i\) for all \(1 \leq i \leq k\). For every \(q, \sigma,\) and \(p_1, \ldots, p_k\) there is at most one such rule. If there is always exactly one such rule, and the functions \(\delta\) are total, then the \(\text{mtt}^R\) is called total. The class of translations realized by \(\text{mtt}^R\)s is denoted by \(\text{mtt}^R\), and the class of translations realized by top-down tree transducers with regular look-ahead is denoted by \(\text{t}^R\).

Note that \(\text{mtt}\) is a proper subclass of \(\text{mtt}^R\). The same holds for top-down tree transducers, i.e., \(\text{t} \subset \text{t}^R\). In contrast, total \(\text{mtts}\) are closed under regular look-
ahead, i.e., for every total \( \text{MTT}^R \) there (effectively) is an equivalent total \( \text{MTT} \). The idea of the proof is to simulate look-ahead by a special new “test state” which appears nested in a full tree. A transition of the look-ahead automaton then corresponds to the selection of a particular parameter of this test state, see Theorem 4.21 of [37] or the proof of Lemma 7.13 of [28].

Let \( N_1, N_2 \) be two top-down tree transducers with regular look-ahead, of which we assume (without loss of generality) that the \( \delta_\sigma \) are total functions. We can transform \( N_1, N_2 \) into ordinary transducers (without look-ahead) \( M_1, M_2 \) and a regular tree language \( R \), such that \( N_1 \) and \( N_2 \) are equivalent if and only if \( M_1 \) and \( M_2 \) are equivalent on \( R \) (i.e., \( \tau_{M_1}(s) = \tau_{M_2}(s) \) for every \( s \in R \)). This is done by changing the input alphabet so that for every original input symbol \( \sigma \in \Sigma^{(k)} \), it now contains the symbols \( \langle \sigma, p_1, \ldots, p_k, p'_1, \ldots, p'_k \rangle \) for all possible look-ahead states \( p_i \in P_1 \) of \( N_1 \) and \( p'_i \in P_2 \) of \( N_2 \). Thus, for every \( \sigma \in \Sigma^{(k)} \), the new input alphabet has \( |P_1|^k |P_2|^k \)-many symbols. Obviously, \( N_i \) can be simulated by \( M_i \) on this new input alphabet, provided \( M_i \)'s input is restricted to the regular tree language \( R \) of new input trees that represent correct runs of the look-ahead automata. It is not difficult to see that Lemma 3 and Theorem 4 also hold for transducers restricted to a regular input tree language \( R \), by additionally requiring that the domain \( D \) of \( M_i \) is intersected with \( R \). Thus, Theorem 4 also holds for top-down tree transducers with regular look-ahead.

**Theorem 5.** Equivalence of deterministic top-down tree transducers with regular look-ahead is decidable.

### 3.3. Canonical Normal Form

It is proved already in 1979 by Choffrut [12] (see also [13]) that subsequential string transducers possess a canonical normal form. The idea is to pull the output symbols “upstream” within the transducer, so that the longest common prefix of outputs generated by one state is the empty string. This implies that output is always produced as early as possible. Engelfriet, Maneth, and Seidl [33] extend this idea to top-down tree transducers. It should be noted 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. Both are established for top-down tree transducers by Lemay, Maneth, and Niehren [59].

Let \( M_1, M_2 \) be two equivalent top-down tree transducers and let \( D \) be their domain. 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 \( t'[u] = t[u] \). Thus, if \( M_1 \) is earlier than \( M_2 \), then it can be that \( M_1 \) produces more output than \( M_2 \) on the same partial input (while \( M_2 \) never produces more output than \( M_1 \) on the same partial input).

If a transducer \( M \) is earliest, then for every of its states \( q \) it holds that there
is no output symbol $\delta$ such that $M_q(T_{\Sigma}) = \delta(\ldots)$. Thus, there must be at least two distinct output symbols that appear at the roots of output trees in $M_q(T_{\Sigma})$. This carries through to the rules of an earliest transducer: there is no $\delta$ such that the root of the right-hand side of each $q$-rule equals $\delta$. A transducer $M$ is canonical if it is earliest and $M_q \neq M_{q'}$ for $q \neq q'$. We state the next result only for total transducers.

**Theorem 6.** Let $M$ be a total deterministic top-down tree transducer. An equivalent canonical transducer can be constructed in polynomial time.

**Proof.** Let $M = (Q, \Sigma, \Delta, q_0, R)$. 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 $\delta$ (of rank $k$) such that $\text{rhs}(q, \sigma)[\epsilon] = \delta$ for every input symbol $\sigma$, then $M$ is not earliest. Intuitively, the symbol $\delta$ should be produced earlier, whenever the state $q$ is called in a rule. Thus, we replace $q(x_1)$ in each right-hand side (and in the axiom $A$) by

$$\delta(q_1(x_1), \ldots, q_k(x_1))$$

where the $(q, j)$ are new states. For every $\sigma$, $\text{rhs}((q, j), \sigma)$ is defined as the $j$-th subtree of the root of $\text{rhs}(q, \sigma)$. Beware, this right-hand side may have changed due to the replacement above. Finally we remove $q$ and its rules. We repeat the whole process and check again if there is a state for which each right-hand side has the same root output symbol, and if so remove the state as explained. We repeat until no such state exists anymore.

It should be clear that the repetition terminates with an earliest transducer. In fact, the number of states of the resulting earliest transducer is bounded by $|Q| \cdot r$ where $r$ is the maximal size of a right-hand side $\text{rhs}(q, a)$ of $M$ with $a \in \Sigma^{(0)}$. So, it is at most $|M|^2$, where the size $|M|$ of $M$ is defined as $|Q| + |\Sigma|$ plus the sum of the sizes of all right-hand sides.

In the second step, equivalent states are merged to obtain the canonical transducer. The corresponding equivalence relation $\equiv$ on states is computed using fixed-point iteration in cubic time (with respect to $|M|$). It is computed in such a way that $q \equiv q'$ if and only if the functions $M_q$ and $M_{q'}$ are equal.

Let $M_1$ and $M_2$ be the two top-down tree transducers with their rules given in Figure 4. To construct a canonical equivalent transducer for $M_1$ according to Theorem 6, we observe that state $q'$ of $M_1$ is not earliest: the root equals $a$ for the right-hand sides of all $q'$-rules. We replace $q'(x_1)$ by $a(q'(1)(x_1))$ in the $(q, a)$-rule, and introduce the two rules $(q', 1)(a(x_1)) \rightarrow a(q(x_1))$ and $(q', 1)(e) \rightarrow e$. 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

\[
\begin{align*}
q_0(a(x_1)) &\to d(q(x_1), q_0(x_1)) \\
q_0(e) &\to e \\
q(a(x_1)) &\to a(q(x_1)) \\
q(e) &\to e.
\end{align*}
\]

To construct the canonical transducer for \( M_2 \), we observe that state \( p \) is not earliest: the root equals \( d \) in both rules. We thus replace \( p(x_1) \) everywhere by \( d(\langle p, 1 \rangle(x_1), \langle p, 2 \rangle(x_1)) \) where \( \langle p, 1 \rangle, \langle p, 2 \rangle \) are new states. After this replacement, the current \( (p, a) \)-rule is:

\[
\begin{align*}
p(a(x_1)) &\to d(a(p'(x_1)), d((p, 1)(x_1), (p, 2)(x_1))).
\end{align*}
\]

Thus, the new \( a \)-rules and \( e \)-rules are

\[
\begin{align*}
\langle p, 1 \rangle(a(x_1)) &\to a(p'(x_1)) \\
\langle p, 2 \rangle(a(x_1)) &\to d((p, 1)(x_1), (p, 2)(x_1)) \\
\langle p, 1 \rangle(e) &\to e \\
\langle p, 2 \rangle(e) &\to e.
\end{align*}
\]

The resulting transducer is earliest. We now compute that \( \langle p, 1 \rangle \equiv p' \) and that \( \langle p, 2 \rangle \equiv p_0 \). We merge these pairs of states and obtain the same transducer (up to renaming of states) as the canonical one of \( M_1 \) above. Hence, \( M_1 \) and \( M_2 \) are equivalent.

As a consequence of Theorem 6 we obtain that equivalence of total top-down tree transducers can be decided in polynomial time.

**Theorem 7.** Equivalence of total deterministic top-down tree transducers can be decided in polynomial time.

As a final observation we mention that the earliest normal form has certain “disadvantages”. As an example: it does not preserve linearity (nor input-nondeletion) of the transducer. Linearity means that every \( x_i \) appears at most once in every right-hand side. Input-nondeletion means that every \( x_i \) that appears in the left-hand side of a rule, also appears in the right-hand side of the rule. Consider for \( \Sigma = \{ a^{(2)}, e^{(0)}, f^{(0)} \} \) the rules

\[
\begin{align*}
q(a(x_1, x_2)) &\to d(q(x_1), q(x_2)) \\
q(e) &\to d(e, e)
\end{align*}
\]

together with the axiom \( q(x_0) \). Making this transducer earliest, the axiom is changed into \( d((q, 1)(x_0), (q, 2)(x_0)) \) and the following rules are added (for \( i = 1, 2 \)):

\[
\begin{align*}
(q, i)(a(x_1, x_2)) &\to d((q, 1)(x_i), (q, 2)(x_i)) \\
(q, i)(e) &\to e
\end{align*}
\]

The resulting transducer is non-linear and input-deleting (viz. the first new rule).
3.4. Bottom-Up Tree Transducers

A bottom-up tree transducer processes an input tree in a bottom-up fashion, in a way similar to a bottom-up tree automaton. Additionally, it can produce output in each transition step. It can delete or copy output that it has produced (bottom-up) already, by using the variables $x_i$ appropriately. Formally, a deterministic bottom-up tree transducer is a tuple $B = (Q, \Sigma, \Delta, Q_f, R)$ where $Q$ is a finite set of states, $\Sigma$ and $\Delta$ are ranked alphabets of input and output symbols, $Q_f \subseteq Q$ is the set of final states, and $R$ is the set of rules which 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

$$\sigma(q_1(x_1), \ldots, q_k(x_k)) \rightarrow q(t)$$

where $q \in Q$ and $t$ is a tree in $T_\Delta(X_k)$.

Note that the class of translations realized by deterministic bottom-up tree transducers is incomparable to the class of deterministic top-down tree translations [24]. This of course already holds on monadic trees, i.e., strings: the translation that copies an input string until the first occurrence of some fixed symbol is top-down but not bottom-up (a similar example shows the other direction).

It is proved in Theorem 3.2 of [25] that every deterministic bottom-up tree transducer can be transformed into an equivalent deterministic top-down tree transducer with regular look-ahead. The proof uses several composition and decomposition results. Here we give a direct construction.

**Lemma 8.** For every deterministic bottom-up tree transducer an equivalent deterministic top-down tree transducer with regular look-ahead can be constructed in linear time.

**Proof.** Let $B = (Q, \Sigma, \Delta, Q_f, R)$ be a deterministic bottom-up tree transducer. We define an equivalent deterministic top-down tree transducer with look-ahead $N = (\{p_0, p\}, \Sigma, \Delta, p_0, R', Q, \delta)$. The look-ahead automaton of $N$ is constructed so that it mimics the state behavior of $B$. Since $B$ is deterministic, the states of the look-ahead automaton at the children of an input node plus the label of the node tell us precisely which rule of $B$ is applied at the node. Thus, the top-down part of $N$ need not keep track of $B$’s states. It uses two states $p_0$ and $p$ that have the same right-hand sides in their rules. The initial state $p_0$ ensures that the look-ahead automaton is in a final state at the root of the input tree. Formally, let $\sigma \in \Sigma^{(k)}$, $k \geq 0$, and $q_1, \ldots, q_k \in Q$. If the rule $\sigma(q_1(x_1), \ldots, q_k(x_k)) \rightarrow q(t)$ is in $R$, then define $\delta_\sigma(q_1, \ldots, q_k) = q$ and let the following rule be in $R'$:

$$p(\sigma(x_1, \ldots, x_k)) \rightarrow t[x_i \leftarrow p(x_i) \mid i \in \{1, \ldots, k\}] \quad (q_1, \ldots, q_k).$$

If $q \in Q_f$ then let additionally the following rule be in $R'$:

$$p_0(\sigma(x_1, \ldots, x_k)) \rightarrow t[x_i \leftarrow p(x_i) \mid i \in \{1, \ldots, k\}] \quad (q_1, \ldots, q_k).$$

It should be clear that $N$ can be constructed in linear time, and that $N$ is equivalent to $B$. \qed
As a consequence of Lemma 8 and Theorem 5 we obtain that equivalence is decidable for deterministic bottom-up tree transducers.

**Theorem 9.** Equivalence of deterministic bottom-up tree transducers is decidable.

The equivalence problem for bottom-up tree transducers was first solved by Zachar [84]. It is shown by Seidl [74] that equivalence can be decided in polynomial time for single-valued (i.e., functional) nondeterministic bottom-up tree transducers. Note that for deterministic transducers this also follows from Theorem 18 and the (polynomial time) construction in the proof of Lemma 8. This result is extended to finite-valued nondeterministic bottom-up tree transducers by Seidl [75]. For nondeterministic letter-to-letter bottom-up tree transducers, equivalence is 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 is solved by Bozapalidis [10]. For deterministic bottom-up tree transducers the effective existence of a canonical normal form, similar in spirit to the canonical normal form of top-down tree transducers, is shown by Friese, Seidl, and Maneth [44]. They prove 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 [43] a Myhill-Nerode theorem for bottom-up tree transducers.

### 4. Equivalence of Linear Size Increase MTTs

It is shown by Engelfriet and Maneth [31] 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 [60]. The MSO definable tree translations are a special instance of the MSO definable graph translations, introduced by Courcelle and Engelfriet, see [16]. Decidability of equivalence for deterministic MSO graph-to-string translations on a context-free graph language is proved by Engelfriet and Maneth [32]. It implies decidable equivalence also for deterministic MTO tree translations, and hence for MTTs of linear size increase. We now present a proof of the latter result that is purely based on MTTs and does not use results about MSO.

The idea of the proof stems from Gurari’s proof [50] of the decidability of equivalence for two-way deterministic gsm. 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 symbol in the string $w$. We now decide if there is an $n$ such that $a^m b^n \in L^{a,b}$, 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 the transducers $M_1, M_2$ are equivalent.

It is shown by Engelfriet, Rozenberg, and Slutzki in Corollary 3.2.7 of [36] that ranges of nondeterministic finite-copying top-down tree transducers with regular look-ahead (for short, N-$R_{fc}^R$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 $\leq c$. A similar definition holds for transducers with regular look-ahead. We denote the class of translations of nondeterministic (resp. deterministic) finite-copying top-down tree transducers with regular look-ahead by N-$R_{fc}^R$ (resp. $R_{fc}^R$).

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 [36]. Recall from the Preliminaries that REGT denotes the class of all regular tree languages.

**Lemma 10.** Languages in $yN-R_{fc}^R(\text{REGT})$ are Parikh.

**Proof.** Let $M$ be a $yN-R_{fc}^R$ transducer and let $R \in \text{REGT}$. A top-down tree 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 $\text{dom}(M') = \text{dom}(M)$ and the string $M'(s)$ is a permutation of the string $M(s)$, for every $s \in \text{dom}(M)$. 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 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 [81]) which is Parikh by Parikh’s theorem [67].
A macro tree transducer $M$ is finite-copying if there exist numbers $k$ and $n$ such that

1. For every input tree $s' = s[u \leftarrow x]$ with $s \in T$ 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 $s \in T$, 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 number $c$ such that $|\tau M(s)| \leq c \cdot |s|$ for every $s \in T$. We denote the class of translations realized by MTTs of linear size increase by $\mathrm{MTT}_{\mathrm{lsi}}$.

**Lemma 11.** $y^{\mathrm{MTT}_{\mathrm{lsi}}} \subseteq y^{\mathrm{R}_{\mathrm{fc}}}$.

**Proof.** It is shown in [31] how to construct a finite-copying macro tree transducer with regular 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 is used already by Aho and Ullman for top-down tree transducers [2].

It is shown in Lemmas 6.3 and 6.6 of [28] 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 $(q, \sigma)$-rule. The idea is to provide multiple parameters, whenever parameters are copied, and to use regular look-ahead in order to determine which parameters are deleted. For a $y^{\mathrm{MTT}_{\mathrm{R}}}$ transducer that is special in the parameters, it is shown in Lemma 13 of [29] how to construct an equivalent $y^{\mathrm{R}}$ transducer. The parameters of the $y^{\mathrm{MTT}}$ can be removed by outputting the strings between them directly. Since each parameter appears once, the final string $M_q(s)$ is divided into $m + 1$ chunks $w_0, y_1 w_1, \ldots, y_m w_m$ (in any order), where $m + 1$ is the rank of $q$ and $w_i$ is an output string. 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.$\square$

**Lemma 12.** Let $M_1, M_2$ be $y^{\mathrm{R}_{\mathrm{fc}}}$ transducers with input and output alphabets $\Sigma$ and $\Delta$, and let $a, b \in \Delta$ with $a \neq b$. Let $D \subseteq T$ be a regular tree language. The language $L^{ab} = \{a^n b^n \mid \exists s \in D : M_1(s)/m = a, M_2(s)/n = b\}$ 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 $y^{\mathrm{N}_{\mathrm{R}_{\mathrm{fc}}}}$ transducer $M'_1$ such that $M'_1(s) = \{ua \mid u \in \Delta^*, \exists v \in \Delta^* : uav = M_1(s)\}$. 


A Survey on Decidable Equivalence Problems for Tree Transducers

Its state set is \( Q'_1 = Q_1 \cup \{ q_a \mid q \in Q_1 \} \) and its initial state is \( q_{0,a} \). It has all rules of \( M_1 \) and, moreover, for every rule
\[
q(\sigma(x_1, \ldots, x_k)) \rightarrow w \quad (\cdots)
\]
of \( M_1 \), whenever \( w = uav \) it has the rule
\[
q_a(\sigma(x_1, \ldots, x_k)) \rightarrow ua \quad (\cdots)
\]
and whenever \( w = uq'(x_i)v \) it has the rule
\[
q_a(\sigma(x_1, \ldots, x_k)) \rightarrow uq'_a(x_i) \quad (\cdots).
\]
From \( M'_1 \) one obtains a \( yN\text{-}TT^R \_E \) transducer \( M''_1 \) such that
\[
M''_1(s) = \{ \sigma^n \mid M_1(s)/n = a \}
\]
by simply changing all symbols of \( \Delta \) 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 \( yN\text{-}TT^R \_E \) transducer \( M \) is defined such that \( M(s) = \{ a^n b^n \mid M_1(s)/m = a, M_2(s)/n = b \} \). 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 \( q_{0,a}(\sigma(x_1, \ldots, x_k)) \rightarrow u \langle q'_1, \ldots, q'_k \rangle \) and \( p_{0,b}(\sigma(x_1, \ldots, x_k)) \rightarrow w \langle p'_1, \ldots, p'_k \rangle \), we let
\[
\langle \sigma(x_1, \ldots, x_k) \rangle \rightarrow uv \langle \langle q'_1, p'_1 \rangle, \ldots, (q'_k, p'_k) \rangle
\]
be a rule of \( M \). Obviously, \( M(s) \) equals the concatenation \( M''_1(s)M''_2(s) \), and is finite-copying. Since \( M(D) = L_n^a,b \) it follows by Lemma 10 that \( L_n^a,b \) is Parikh.

**Theorem 13.** 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 \( \text{dom}(M_i) \) is effectively regular by Lemma 2. If not then the transducers are not equivalent and we are finished. Otherwise, let \( D \) be their domain. We can view \( M_i \) as a tree-to-string transducer, by regarding the tree in the right-hand side of each rule as a string (which uses additional terminal symbols for denoting the tree structure such as opening and closing parentheses and commas). Thus, by Lemma 11 (which is effective) we can in fact assume that \( M_1 \) and \( M_2 \) are \( yT^R \_E \) transducers. Let \( \Delta \) be the output alphabet of \( M_i \) and let \$ be a new symbol not in \( \Delta \). 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 \( a, b \in \Delta \cup \{ \$ \} \) 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 and only if the intersection
of $L^{a,b}$ of Lemma 12 with the language $E = \{ w \in \{a,b\}^* \mid \#_a(w) = \#_b(w) \}$ is nonempty ($\#_a(w)$ denotes the number of occurrences of the symbol $a$ in $w$). Since $L^{a,b}$ is Parikh by Lemma 12, we obtain decidability because semilinear sets are closed under intersection \cite{48,47} and have decidable emptiness. But, there is an easier proof: $L^{a,b} \cap E$ is nonempty if and only if $L \cap E$ is nonempty, where $L$ is a regular language with the same Parikh vectors as $L^{a,b}$. Since $E$ is context-free, so is $L \cap E$ (by the well-known “triple construction”, see, e.g., Theorem 6.5 of \cite{53}). The result follows since context-free grammars have decidable emptiness.

Recall from the discussion in Section 3.2 that MTT is a proper subclass of MTT$^R$. In fact, the proper inclusion already holds for their domains: the domains of MTT$^R$s are the regular tree languages (see Corollary 5.6 of \cite{37}). Thus, as an example, no MTT exists that has domain $\{f(a,b), f(b,a)\}$. This implies that also the class of MTT$s of linear size increase is a proper subclass of the class of MTT$^R$s of linear size increase.

It is not difficult however to extend the result of Theorem 13 to MTT$^R$s of linear size increase. For every MTT$^R$ $M$ we can construct a total MTT$^R$ $M'$ such that $\tau_{M'}(s) = \bot$ if $\tau_M(s)$ is undefined, and $\tau_{M'}(s) = \tau_M(s)$ otherwise (where $\bot$ is a new symbol of rank zero). This is done by first constructing a tree automaton for $\text{dom}(M)$ according to Lemma 2, and then incorporating this automaton into the look-ahead automaton of $M'$. As mentioned in Section 3.2 there effectively exists an MTT $M''$ (without look-ahead) that is equivalent to $M'$. Clearly, two given MTT$^R$s $M_1$ and $M_2$ are equivalent if and only if the corresponding total MTT's $M_1''$ and $M_2''$ are equivalent. For MTT$s of linear size increase the latter is decidable by Theorem 13.

**Theorem 14.** Equivalence of MTT$^R$s of linear size increase is decidable.

## 5. Equivalence of 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 $\sigma$, the parameter $y_j$ occurs in $\text{rhs}(q, \sigma)$. A monadic MTT $M = (Q, \Sigma, \Delta, q_0, R)$ is **normalized** if

(A) it is nondeleting,

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

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

Note that for total monadic transducers (B) is a consequence of (A) because a $(q, \sigma)$-rule with $\sigma \in \Sigma^{(0)}$ can contain at most one parameter occurrence.

**HDT0L systems.** An instance of the HDT0L sequence equivalence problem consists of alphabets $\Sigma$ and $\Delta$, two strings $w_1, w_2 \in \Sigma^*$, homomorphisms $h_j, g_j$:
\[ \Sigma^* \to \Sigma^*, \quad 1 \leq j \leq n, \text{ and homomorphisms } h, g : \Sigma^* \to \Delta^*. \] To solve the problem we have to determine whether or not
\[ h(h_n(\cdots h_1(w_1)) \cdots) = g(g_n(\cdots g_1(w_2)) \cdots) \]
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 proved by Culik II and Karhumäki [22], using Ehrenfeucht’s Conjecture and Malanin’s algorithm. A later proof of Ruohonen [73] is based on the theory of metabelian groups. Yet another, very short, proof is given by Honkala [52] 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 \( s = a_1(\cdots a_n(\perp) \cdots) \) we denote by strip\( (s) \) the string \( a_1 \cdots a_n. \)

**Lemma 15.** Equivalence of total monadic normalized MTTs 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 \( \Sigma, \Delta \) are defined as \( \Sigma = \Pi(1) \cup Q \) and \( \Delta = \Pi(1). \) We define homomorphisms \( h_n, g_n \) for every input symbol \( a \in \Gamma(1). \) Let \( a \in \Gamma(1). \) For \( \pi \in \Pi(1) \) let \( h_n(\pi) = g_n(\pi) = \pi. \) Let \( q \in Q. \) If \( q \in Q_1 \) then let \( g_n(q) = \text{strip}(\text{rhs}_{M_1}(q, a)), \) and otherwise let \( h_n(q) = q. \) If \( q \in Q_2 \) then let \( g_n(q) = \text{strip}(\text{rhs}_{M_2}(q, a)), \) and otherwise let \( g_n(q) = q. \) For trees \( t \in T_{\Pi(1)} \) we define the mapping \( \text{strip} \) by \( \text{strip}(\pi(t)) = \pi \cdot \text{strip}(t) \) for \( \pi \in \Pi(1), \) \( \text{strip}(g(x_1, t)) = q \cdot \text{strip}(t) \) for \( q \in Q(2), \) \( \text{strip}(\perp) = \epsilon \) and \( \text{strip}(\perp) = \epsilon,\) where \( \cdot \) denotes string concatenation. The final homomorphisms \( h, g \) are defined as \( h(q) = \text{strip}(\text{rhs}_{M_1}(q, \perp)) \) if \( q \in Q_1, \) and otherwise \( h(q) = \epsilon, \) and \( g(q) = \text{strip}(\text{rhs}_{M_2}(q, \perp)) \) if \( q \in Q_2, \) and otherwise \( g(q) = \epsilon. \) Moreover, \( h(\pi) = g(\pi) = \pi \) for all \( \pi \in \Pi(1). \) Last but not least, let \( w_1 = q_0 \) and \( w_2 = p_0. \) This ends the construction of the HDT0L instance.

Let \( s = a_1(\cdots a_n(\perp) \cdots) \in T_{\Gamma} \) be an input tree. It should be clear that
\[ h(h_n(\cdots h_1(w_1)) \cdots) = \text{strip}(M_1(s)) \]
and that
\[ g(g_n(\cdots g_1(w_2)) \cdots) = \text{strip}(M_2(s)). \]

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 \( \Sigma' = \{ (r, b) \mid r \in R, b \in \Sigma \} \) and \( \Delta = \Pi^{(1)} \). Our \text{HDTOL} instance is over \( \Sigma' \) and \( \Delta \). Let \( a \in \Gamma^{(1)}, r \in R \), and \( r' = \delta(r, a) \). For \( \pi \in \Pi^{(1)} \) let \( h_u((r, \pi)) = g_u((r, \pi)) = (r', \pi) \). Let \( q \in Q_1 \) and \( p \in Q_2 \). Define

\[
\begin{align*}
    h_u((r, q)) &= \text{strip}(\text{rhs}_{M_1}(q, a))[b \leftarrow (r', b) \mid b \in \Sigma] \\
    g_u((r, p)) &= \text{strip}(\text{rhs}_{M_2}(p, a))[b \leftarrow (r', b) \mid b \in \Sigma].
\end{align*}
\]

Let \( h_u((r, p)) = (r, p) \) and \( g_u((r, q)) = (r, q) \). The final homomorphisms \( g, h \) are defined as follows. If \( r \in R_f \) then let

\[
\begin{align*}
    h((r, q)) &= \text{strip}(\text{rhs}_{M_1}(q, \bot)) \\
    g((r, p)) &= \text{strip}(\text{rhs}_{M_2}(p, \bot))
\end{align*}
\]

and in the remaining cases let

\[
\begin{align*}
    h((r, b)) &= b \\
    g((r, b)) &= b.
\end{align*}
\]

If \( r \not\in R_f \) then let

\[
\begin{align*}
    h((r, b)) &= g((r, b)) = \epsilon
\end{align*}
\]

for every \( b \in \Sigma \). The initial strings are defined as \( w_1 = (r_0, q_0) \) and \( w_2 = (r_0, p_0) \).

Let \( s = a_1(a_1\cdots a_j) \in T_\Gamma \) be an input tree and let \( 1 \leq j \leq n \). 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

\[
\begin{align*}
    h_u(a_1\cdots a_j)(w_1) \cdots &= \text{strip}(M_1(a_1\cdots a_j(x)))[\pi \leftarrow (r, \pi) \mid \pi \in \Delta][q \leftarrow (r, q) \mid q \in Q_1]
\end{align*}
\]

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 \( s \not\in D \), then every symbol in \( u_1 = h_u(a_1\cdots a_j(w_1)) \cdots \) and in \( u_2 = g_u(a_1\cdots a_j(w_2)) \cdots \) is labeled by some state \( r \not\in R_f \). This implies that \( h(u_1) = g(u_2) = \epsilon \), i.e., the final strings are equal whenever \( s \not\in D \). If on the contrary \( s \in D \) then every symbol in \( u_i \) is labeled by a final state and therefore \( h(u_i) = \text{strip}(M_f(s)) \) as before.

\[
\begin{align*}
    M_1: q_0(a(x_1)) \rightarrow c(d(q(x_1, q_0(x_1)))) & \quad M_2: p_0(a(x_1)) \rightarrow c(p(x_1, d(p_0(x_1)))) \\
    q_0(\bot) \rightarrow \bot & \quad p_0(\bot) \rightarrow \bot \\
    q(a(x_1), y_1) \rightarrow q(x_1, c(d(y_1))) & \quad p(a(x_1), y_1) \rightarrow d(c(p(x_1, y_1))) \\
    q(\bot, y_1) \rightarrow y_1 & \quad p(\bot, y_1) \rightarrow y_1
\end{align*}
\]

Fig. 6. Equivalent total monadic normalized MTTs \( M_1 \) and \( M_2 \)
As an example, we apply the construction in the proof of Lemma 15 to the two total monadic normalized mtt\textsubscript{s} M\textsubscript{1} and M\textsubscript{2} with their rules shown in Figure 6. We define the homomorphisms h\textsubscript{a}, g\textsubscript{a}, h, g. For \( \pi \in \{c, d\} \) let \( h\textsubscript{a}(\pi) = g\textsubscript{a}(\pi) = h(\pi) = g(\pi) = \pi \). For the state \( q_0 \) let \( h\textsubscript{a}(q_0) = \text{strip}(\text{rhs}_{M\textsubscript{1}}(q_0, a)) = \text{strip}(c(d(q(x, q_0(x)))))) = cdq_0 \) and \( g\textsubscript{a}(q_0) = q_0 \). For the state \( q \) we let \( h\textsubscript{a}(q) = \text{strip}(\text{rhs}_{M\textsubscript{1}}(q, a)) = \text{strip}(g(x_1, c(d(y_1)))) = qcd \) and \( g\textsubscript{a}(q) = q \). Similarly, for the state \( p_0 \) we obtain \( g\textsubscript{a}(p_0) = \text{strip}(\text{rhs}_{M\textsubscript{2}}(p_0, a)) = \text{strip}(c(p(x, d(p_0(x)))))) = cpdp_0 \) and \( h\textsubscript{a}(p_0) = p_0 \). For the state \( p \) we obtain \( g\textsubscript{a}(p) = \text{strip}(\text{rhs}_{M\textsubscript{2}}(p, a)) = \text{strip}(d(c(p(x_1, y_1)))) = dcp \) and \( h\textsubscript{a}(p) = p \). Finally, for \( \beta \in \{q_0, q, p_0, p\} \) we define \( h(\beta) = g(\beta) = \epsilon \). Consider now the input tree \( s = a^4(\perp) \). We compute

\[
\begin{align*}
  h(h\textsubscript{a}(h\textsubscript{a}(h\textsubscript{a}(q_0)))) &= h(h\textsubscript{a}(h\textsubscript{a}(cdq_0)))) \\
  &= h(h\textsubscript{a}(cdq)) \\
  &= h(cdp) \\
  &= cd \\
  &= (cd)^2.
\end{align*}
\]

Similarly we compute

\[
\begin{align*}
  g(g\textsubscript{a}(g\textsubscript{a}(g\textsubscript{a}(p_0)))) &= g(g\textsubscript{a}(cpdp_0)) \\
  &= g(cdp) \\
  &= cdp \\
  &= (cd)^2.
\end{align*}
\]

As the reader may verify, these are precisely the strings obtained by stripping the monadic trees \( M\textsubscript{1}(s) \) and \( M\textsubscript{2}(s) \), respectively.

We now show that every monadic mtt\textsuperscript{R} transducer can be transformed into a normalized monadic mtt\textsuperscript{R} transducer. This is done in such a way that input and output symbols of rank zero of the given monadic mtt\textsuperscript{R} transducer become symbols of rank one in the constructed normalized transducer. For a monadic tree \( t = a\textsubscript{1}(\cdots a\textsubscript{m}(\epsilon) \cdots) \) we denote by \text{expand}(t) the tree \( a\textsubscript{1}(\cdots a\textsubscript{m}(\epsilon(\perp)) \cdots) \).

**Lemma 16.** For every monadic mtt\textsuperscript{R} transducer \( M \) a normalized mtt\textsuperscript{R} transducer \( N \) can be constructed such that \( \tau_N = \{ (\text{expand}(s), \text{expand}(t)) \mid (s, t) \in \tau_M \} \).

**Proof.** Using regular look-ahead we first make \( M \) nondeleting. As mentioned in the proof of Lemma 11, this construction is given in the proof of Lemma 6.6 of [28]. We denote the resulting transducer by \( M' \). Every parameter that appears in the left-hand side of a rule of \( M' \) also appears in the right-hand side of that rule. Since the final output tree is monadic, the resulting transducer satisfies condition
(B) given at the beginning of Section 5. Finally, we define the $\text{mtt}^R$ transducer $N$ which has input and output alphabets $\Sigma' = \Sigma^{(1)} \cup \{a^{(1)} \mid a \in \Sigma^{(0)}\} \cup \{\perp^{(0)}\}$ and $\Delta' = \Delta^{(1)} \cup \{a^{(1)} \mid a \in \Delta^{(0)}\} \cup \{\perp^{(0)}\}$. 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 $\text{rhs}(q,a)$ is defined. Then we let

$$\text{rhs}_N(q,a') = \text{rhs}_{M'}(q,a)[b \leftarrow b'('\perp') \mid b \in \Delta^{(0)}].$$

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

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

**Theorem 17.** Equivalence of monadic deterministic macro tree transducers with regular look-ahead is decidable.

**Proof.** Let $N_1, N_2$ be monadic macro tree transducers with regular look-ahead and let $\Sigma$ be their input alphabet. Let $A_1, A_2$ be the look-ahead automata of $N_1, N_2$. By Lemma 16 we can assume that $N_1$ and $N_2$ are normalized. We first check if the domains of $N_1$ and $N_2$ 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 $M_1, M_2$ without look-ahead. Let $P_1, P_2$ be the sets of states of $A_1, A_2$, respectively.

The input alphabet of $M_i$ 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 $M_1$ is defined as the $(q,\langle \sigma, p_1, p_2 \rangle)$-rule of $N_1$, and the $(q,\langle \sigma, p_1, p_2 \rangle)$-rule of $M_2$ is defined as the $(q,\langle \sigma, p_1, p_2 \rangle)$-rule of $N_2$. Finally, we make $M_1$ and $M_2$ total (in some arbitrary way).

For a tree $t$ in $T_{\Sigma'}$ we denote by $\gamma(t)$ the tree in $T_{\Sigma}$ obtained by changing every label $\langle \sigma, p, p' \rangle$ into the label $\sigma$. 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 $M_i$ it holds that $M_1$ and $M_2$ are equivalent on $E$ if and only if $N_1$ is equivalent to $N_2$. Hence decidability of equivalence follows from Lemma 15.

The connection between L-systems and tree transducers is well known and is studied extensively in [36].
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 [28]). For the latter, the decidability of the equivalence problem is stated already in 1980 by Engelfriet [26] as a difficult open problem. New recent progress has been made on this problem: First, Filiot, Maneth, Reynier, and Talbot [?] show that under “origin semantics”, equivalence is decidable for top-down tree-to-string transducers. Origin semantics means that two transducers are equivalent if for all input trees \( s \), \( M_1(s) = M_2(s) = t \) and, the origin of every node \( v \) in \( t \) (i.e., the input node in \( s \) responsible for generating \( v \)) is the same for \( M_1 \) and \( M_2 \). They use the additional structural constraints of origin semantics to reduce the problem to the hdtol sequence equivalence problem.

Second, the big open problem of equivalence for top-down tree-to-string transducers has now finally been solved by Seidl, Maneth, and Kemper [?]. The proof is based on multivariate linear algebra and uses Hilbert’s Basis Theorem. They also show that the growth equivalence and Abelian equivalence problems for macro tree transducers are decidable.

6. Complexity

In Section 3.3 we already mentioned one complexity result, viz. Theorem 7, 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 is mentioned in the Conclusions of [8] that checking equivalence of Ts can be done in double exponential time, using the procedure of [33].

We now present a proof that strengthens both results above (and which also works for transducers with regular look-ahead). We show that equivalence for Ts can be decided in \textsc{expspace}, and for total Ts in \textsc{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 \( M(s[u ← x]) \) does not have a \( \Delta \)-node \( v \), while \( v \) is a \( \Delta \)-node in \( M(s[u ← a(x,\ldots,x)]) \) where \( a = s[u] \).

**Theorem 18.** Equivalence of deterministic top-down tree transducers with regular look-ahead is decidable in \textsc{expspace}, and for total transducers in \textsc{nlogspace}.

**Proof.** We explain the proof for transducers without look-ahead. Since both complexity classes are closed under complement, it suffices to consider non-equivalence. 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 \( t_1[v] ≠ t_2[v] \). 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
S. Maneth

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 3 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 19.** Equivalence of deterministic top-down tree transducers is \textsc{exptime}-hard.

**Proof.** It is well known that testing intersection emptiness of \( n \) deterministic top-down tree automata \( A_1, \ldots, A_n \) is \textsc{exptime}-complete. This is shown by Seidl [76], cf. also [14]. Let \( \Sigma \) 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, \( M_1, q_i = \{(s, s, \ldots, s) \mid s \in L(A_i)\} \). Let \( \sigma \in \Sigma \) be an arbitrary symbol of rank \( \geq 1 \), and let \( e \) be an arbitrary symbol in \( \Sigma^{(0)} \). We add these two rules to \( R \):

\[
q_0(\sigma(x_1, \ldots)) \rightarrow \delta(q_1(x_1), q_2(x_1), \ldots, q_n(x_1))
\]

\[
q_0(e) \rightarrow e
\]

The transducer \( M_2 = (\{p\}, \Sigma, \Sigma, p, \{p(e) \rightarrow e\}) \) 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. \( \square \)

6.1. **Streaming Tree Transducers**

The (deterministic) streaming tree transducers of Alur and d’Antoni are a new model with the same expressive power as deterministic \textsc{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 non-equivalence of streaming tree transducers can be decided in nondeterministic exponential time. The idea of the proof is the same as the one in Theorem 13: construct a context-free language \( L_{a^nb^n} \) 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 $a^n b^n$ is in $L(A)$ can be done in $\text{nptime}$ using [39,78].

**Theorem 20 ([3])** Equivalence of deterministic streaming tree transducers is decidable in $\text{co-nexptime}$.

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

**Theorem 21 ([3])** Equivalence of deterministic streaming string-to-tree transducers is decidable in $\text{pspace}$.

### 6.2. Visibly Pushdown Transducers

Visibly pushdown languages are 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 are introduced by Raskin and Servais [70]. 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 [11]. Such transducers cannot copy nor swap the order of input trees. Thus, they are $\text{mso}$ 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 22 ([40])** Equivalence of functional visibly pushdown transducers is $\text{exptime}$-complete. For total such transducers the problem is in $\text{Ptime}$.

The $\text{exptime}$-completeness result extends to the case of regular look-ahead, as shown in Section 8.4 of [41,79]. Staworko, Laurence, Lemay, and Niehren [80] have considered the equivalence problem for deterministic visibly pushdown transducers and show that it can be reduced in $\text{ptime}$ to the homomorphic equivalence problem on context-free grammars. The latter is shown by Plandowski [55,69] to be solvable in $\text{ptime}$. They show in [80] that for several related classes the problem is in $\text{ptime}$, for instance, linear and order-preserving deterministic top-down and bottom-up tree-to-string transducers.

**Theorem 23 ([80])** Equivalence of deterministic visibly pushdown transducers is decidable in $\text{Ptime}$.

For linear and order-preserving deterministic top-down tree-to-string transducers a canonical normal form is presented by Laurence, Lemay, Niehren, Staworko, and Tommasi [57]. A Myhill-Nerode characterization and Gold-style learning for this class is investigated by the same authors in [58].
7. Conclusion

We discussed the decidability of equivalence for three incomparable subclasses of deterministic macro tree transducers: top-down tree transducers, linear size increase \(\text{mtt}\)s, and monadic \(\text{mtt}\)s. For top-down tree transducers the proof uses their bounded height-balance property and constructs an automaton that keeps track of the balance. Alternatively, such transducers can 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 \(\text{mtt}\)s, and in particular the bounded height-balance does not hold for \(\text{mtt}\)s, even not for monadic ones. Thus, other techniques are needed in these two cases. For the linear size increase subclass of \(\text{mtt}\)s we can use the Parikh property of the corresponding output languages: the two transducers are merged (“twinned”) to output \(a^mb^n\) 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 can decide if it contains \(a^nb^n\) which implies that the transducers are not equivalent (because \(a \neq b\)). For monadic \(\text{mtt}\)s we use yet another technique: we simulate the transducers by \(\text{hdtol}\) sequences. Since the sequence equivalence problem for \(\text{hdtol}\) 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. For the subclass of linear order-preserving deterministic top-down tree-to-string transducers a canonical normal form presented in [57]. 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 regular look-ahead, but, for such transducers we only know a canonical normal form in the total case for a fixed look-ahead automaton [34]. In fact, even to decide whether or not a given top-down tree transducer with look-ahead is equivalent to a top-down tree transducer (without look-ahead) is a difficult open problem; it was solved recently for a subclass of top-down tree transducers with regular look-ahead [34] (see also [35]).

Acknowledgements

I wish to thank Joost Engelfriet and the anonymous referee for many helpful comments on a previous version of this paper.

References


