

Available online at www.sciencedirect.com

SCIENCE @ DIRECT®

Theoretical Computer Science 347 (2005) 276–287

Theoretical
Computer Sciencewww.elsevier.com/locate/tcs

Linear deterministic multi bottom-up tree transducers[☆]

Zoltán Fülöp^{a,*}, Armin Kühnemann^b, Heiko Vogler^b^a*Department of Computer Science, University of Szeged, Árpád tér 2., H-6720 Szeged, Hungary*^b*Faculty of Computer Science, Technical University of Dresden, D-01062 Dresden, Germany*

Received 10 September 2004; received in revised form 14 January 2005; accepted 22 July 2005

Communicated by M. Ito

Abstract

In general, top-down and bottom-up tree transducers lead to incomparable classes of tree transformations, both for the nondeterministic and the deterministic case. If deterministic top-down tree transducers are extended by the capability to recognize regular tree properties and deterministic bottom-up tree transducers are generalized by allowing states with arbitrary finite rank, then the two devices, now called deterministic top-down tree transducers with regular look-ahead and deterministic multi bottom-up tree transducers, respectively, become equivalent [Z. Fülöp, A. Kühnemann, H. Vogler, A bottom-up characterization of deterministic top-down tree transducers with regular look-ahead, *Inform. Process. Lett.* 91 (2004) 57–67].

In this paper we focus on the class *ld-MBOT* of tree transformations which are computed by linear deterministic multi bottom-up tree transducers. We investigate the relationship among *ld-MBOT* and the classes of tree transformations computed by (restricted) deterministic bottom-up tree transducers and by (restricted) deterministic top-down tree transducers with regular look-ahead. In fact, we show the inclusion diagram of nine such classes.

© 2005 Elsevier B.V. All rights reserved.

Keywords: Tree automata; Tree transducers

[☆] This research was supported by the Hungarian Scientific Foundation (OTKA) under Grant T 046686, the Exchange Programme of the DAAD and MÖB (project “Weighted Tree Automata”, Grant No. 36) and the Herbert-Quandt-Foundation.

* Corresponding author. Tel.: +36 62 544 302; fax: +36 62 546 397.

E-mail addresses: fulop@inf.u-szeged.hu (Z. Fülöp), kuehne@tcs.inf.tu-dresden.de (A. Kühnemann), vogler@tcs.inf.tu-dresden.de (H. Vogler).

0304-3975/\$ - see front matter © 2005 Elsevier B.V. All rights reserved.

doi:10.1016/j.tcs.2005.07.025

1. Introduction

In the theory of tree automata and tree transducers bottom-up (or frontier-to-root) devices and top-down (or root-to-frontier) devices are considered: in the former, the processing starts at the leaves of an input tree s and ends up at the root of s , and in the latter, the processing starts at the root of s and proceeds towards the leaves of s (cf., e.g., [2,11,12,15,16]). A number of investigations have been done in order to compare the accepting or transformational power of the two alternative processing modes of a device. For instance, the class of tree languages accepted by deterministic top-down tree automata is a proper subclass of the class of tree languages accepted by deterministic bottom-up tree automata (cf., e.g. [11, Example 2.11 in Chapter II]); or the classes of tree transformations computed by bottom-up tree transducers and by top-down tree transducers are incomparable (cf. [2, Theorem 2.3]); or: the classes of tree transformations computed by bottom-up tree-to-graph transducers and by top-down tree-to-graph transducers are equal (cf. [8, Theorem 7.1]).

In the case that the two classes are not equal, features have been added to the devices in order to remedy the deficiencies of the respective models. For instance, linear top-down tree transducers have a strictly smaller transformational power than linear bottom-up tree transducers (cf. [2, Theorem 2.8]); however, if linear top-down tree transducers are equipped with regular look-ahead, then they are as powerful as linear bottom-up tree transducers (cf. [3, Theorem 2.8]). Another example is the following: the classes of tree transformations computed by deterministic bottom-up tree transducers and by deterministic top-down tree transducers are incomparable (cf. [3, Section 3]); however, if deterministic bottom-up tree transducers are generalized by allowing the states to have an arbitrary finite rank and the deterministic top-down tree transducers are equipped with regular look-ahead, then the classes of tree transformations become equal (cf. [10, Theorem 4.4]). Such generalized bottom-up tree transducers are called *multi bottom-up tree transducers*.

In this paper we deal with deterministic multi bottom-up tree transducers. Roughly speaking, a deterministic multi bottom-up tree transducer M is a term rewriting system which is based on the three ranked alphabets of states, input symbols, and output symbols. There are rules processing input symbols. These rules have the form

$$\sigma(q_1(x_{1,1}, \dots, x_{1,n_1}), \dots, q_k(x_{k,1}, \dots, x_{k,n_k})) \rightarrow q_0(t_1, \dots, t_n),$$

where $k \geq 0$, σ is a k -ary input symbol, q_0, q_1, \dots, q_k are states of rank n, n_1, \dots, n_k , respectively, and the t_1, \dots, t_n are trees over the ranked alphabet of output symbols and the variables $x_{1,1}, \dots, x_{1,n_1}, \dots, x_{k,1}, \dots, x_{k,n_k}$. Moreover, in order to guarantee determinism, we require that there are no different rules with the same left-hand side. Now, the transformation $\tau_M(s)$ of an input tree s is computed as follows: first, a special unary symbol *root* is put on top of s ; then, using the above rules, the transducer computes a tree of the form $root(q(u_1, \dots, u_m))$ where q is an m -ary state and u_1, \dots, u_m are output trees; finally, a rule of the form

$$root(q(x_1, \dots, x_m)) \rightarrow q_f(t)$$

is applied, where q_f is a special unary symbol called final state and t is a tree over the ranked alphabet of output symbols and the variables x_1, \dots, x_m . Again, there are no different

root-rules with the same left-hand side. Now, if a tree of the form $q_f(u)$ is computed by M where u is an output tree, then we define $\tau_M(s) = u$.

As an example let us consider the deterministic multi bottom-up tree transducer M_{copy} with the following rules:

- (1) $\alpha \quad \rightarrow \quad q(\alpha, \alpha),$
- (2) $\gamma(q(x_1, x_2)) \quad \rightarrow \quad q(\gamma(x_1), \gamma(x_2)),$
- (3) $\text{root}(q(x_1, x_2)) \quad \rightarrow \quad q_f(\sigma(x_1, x_2)),$

where q is a state of rank 2, γ and α are input symbols of rank 1 and 0, respectively, and σ , γ , and α are output symbols of rank 2, 1, and 0, respectively. Let us compute the transformation of the input tree $s = \gamma(\gamma(\alpha))$:

$$\begin{aligned} & \text{root}(\gamma(\gamma(\alpha))) \\ & \Rightarrow \text{root}(\gamma(\gamma(q(\alpha, \alpha)))) \quad (1) \\ & \Rightarrow \text{root}(\gamma(q(\gamma(\alpha), \gamma(\alpha)))) \quad (2) \\ & \Rightarrow \text{root}(q(\gamma(\gamma(\alpha)), \gamma(\gamma(\alpha)))) \quad (2) \\ & \Rightarrow q_f(\sigma(\gamma(\gamma(\alpha)), \gamma(\gamma(\alpha)))) \quad (3) \end{aligned}$$

Thus $\tau_{M_{\text{copy}}}(\gamma(\gamma(\alpha))) = \sigma(\gamma(\gamma(\alpha)), \gamma(\gamma(\alpha)))$. In general, for every $n \geq 0$, we have that $\tau_{M_{\text{copy}}}(\gamma^n \alpha) = \sigma(\gamma^n \alpha, \gamma^n \alpha)$. We note that M_{copy} is a *linear* transducer, because every variable which occurs in the left-hand side of a rule occurs at most once in its right-hand side.

As already mentioned, in Theorem 4.4 of [10] we proved the equality $d\text{-MBOT} = d\text{-TOP}^R$, where $d\text{-MBOT}$ and $d\text{-TOP}^R$ are the classes of tree transformations computed by deterministic multi bottom-up tree transducers and by deterministic top-down tree transducers with regular look-ahead, respectively. We proved this in the way that, for every deterministic multi bottom-up tree transducer, we constructed a semantically equivalent deterministic top-down tree transducer with regular look-ahead (cf. [10, Lemma 4.1]), and vice versa (cf. [10, Lemma 4.2]).

Linearity is an interesting and important property of tree transducers. As mentioned, while the computation power of bottom-up tree transducers and of top-down tree transducers are incomparable, linear top-down tree transducers are strictly less powerful than linear bottom-up tree transducers. Another example is that, while neither bottom-up nor top-down tree transducers preserve recognizability of tree languages, their linear versions do so [2]. Therefore, after having introduced the general model in [10], in this paper we are interested in linear multi bottom-up tree transducers.

It is easy to observe that none of the constructions applied in the proofs of Lemmas 4.1 and 4.2 of [10] preserves linearity of tree transducers. This raises the question how the classes $ld\text{-MBOT}$ and $ld\text{-TOP}^R$ are related to each other, where the prefix l refers to the classes of tree transformations computed by the linear versions of the corresponding tree transducers.

In this paper we will answer this question. Actually, we will relate the class $ld\text{-MBOT}$ not only to the class $ld\text{-TOP}^R$ but also to all the classes X in Table 1 which are computed by tree transducers of type Y .

We do this by presenting the inclusion diagram of $ld\text{-MBOT}$ and all the classes of Table 1 (cf. Theorem 4.1 and Fig. 1). An inclusion diagram is a Hasse-diagram in which the nodes

Table 1

X	Y
$d\text{-TOP}^R$	Deterministic top-down tree transducers with regular look-ahead
$d\text{-BOT}$	Deterministic bottom-up tree transducers
$d\text{-TOP}$	Deterministic top-down tree transducers
$ld\text{-TOP}^R$	Linear deterministic top-down tree transducers with regular look-ahead
$ld\text{-BOT}$	Linear deterministic bottom-up tree transducers
HOM	Tree homomorphisms
$ld\text{-TOP}$	Linear deterministic top-down tree transducers
$l\text{-HOM}$	Linear tree homomorphisms

are the classes of tree transformations, and the partial order is set inclusion \subseteq . In particular, we obtain that the two classes $ld\text{-MBOT}$ and $ld\text{-TOP}^R$ are incomparable.

The part of our inclusion diagram which contains only the classes of Table 1, can be extracted from the inclusion diagram given in Fig. 1 of [13] (actually, [13, Fig. 1] contains all classes of tree transformations obtained by composition of the classes of Table 1).

The structure of our paper is as follows. Apart from this introduction, the paper contains four further sections: Section 2 introduces basic notions and notations. In Section 3 deterministic multi bottom-up tree transducers are recalled from the literature. The inclusion diagram for subclasses of tree transformations computed by deterministic multi bottom-up tree transducers is presented and proved in Section 4. Finally, Section 5 concludes and provides further research topics.

2. Preliminaries

For an integer $n \geq 0$, we let $[n] = \{1, \dots, n\}$. The cardinality of a set A is denoted by $|A|$. Let \Rightarrow be a binary relation on a set A . Then, \Rightarrow^* denotes the reflexive, transitive closure of \Rightarrow . We will need the set $X_\omega = \{x_1, x_2, \dots\}$ and, for every $m \geq 0$, the set $X_m = \{x_1, \dots, x_m\}$ of *variable symbols*. Note that $X_0 = \emptyset$. Moreover, we also need the set $X_{\omega, \omega} = \{x_{i,j} \mid i, j \geq 1\}$ of (*doubly indexed*) *variable symbols*.

For a ranked alphabet Σ and a set A , we denote by $T_\Sigma(A)$ the set of *terms* (or *trees*) over Σ indexed by A (i.e. elements of A are considered as additional symbols of rank 0). Moreover, $T_\Sigma(\emptyset)$ is denoted by T_Σ . For every $t \in T_\Sigma(A)$, $\delta \in \Sigma \cup A$, and $\Delta \subseteq \Sigma \cup A$ let $|t|_\delta$ and $|t|_\Delta$ denote the number of occurrences of δ and of symbols of Δ , respectively, in t . If $t \in T_\Sigma$, then we abbreviate $|t|_\Sigma$ by $|t|$.

We denote by $\Sigma^{(k)}$ the set of all symbols of Σ having rank k and by $\sigma^{(k)}$ the fact that $\sigma \in \Sigma^{(k)}$. In case $\sigma \in \Sigma^{(0)}$, we identify $\sigma(\cdot)$ with σ . In case $\sigma \in \Sigma^{(1)}$ and $t \in T_\Sigma(A)$, we identify $\sigma(t)$ with σt , i.e. we write “monadic parts” of trees as strings. For a string w of monadic symbols and $n \geq 0$, w^n denotes the n -fold concatenation of w . We assume that $\Sigma^{(0)} \neq \emptyset$ for every ranked alphabet Σ appearing as input or output ranked alphabet of some tree transducer in this paper.

We now define the concept of *tree substitution*. Therefore, let t be a tree and let $\langle u_1, \dots, u_n \rangle$ be a sequence of pairwise different variable symbols (from $X_\omega \cup X_{\omega, \omega}$). Moreover, let

$\langle v_1, \dots, v_n \rangle$ be a sequence of trees. Then we denote by $t[u_1 \leftarrow v_1, \dots, u_n \leftarrow v_n]$, shortly also by $t[u_i \leftarrow v_i \mid i \in [n]]$, the tree which is obtained from t by replacing every occurrence of u_i by v_i for every $i \in [n]$. In the special case when $t \in T_\Sigma(X_n)$ and $u_i = x_i$ for every $i \in [n]$, $t[u_i \leftarrow v_i \mid i \in [n]]$ is also denoted by $t[v_1, \dots, v_n]$.

A *tree language* is a set $L \subseteq T_\Sigma$, where Σ is a ranked alphabet. A *tree transformation* is a relation $\tau \subseteq T_\Sigma \times T_\Delta$, where Σ and Δ are ranked alphabets. We define the *image of L under τ* to be the tree language $\tau(L) = \{t \in T_\Delta \mid \text{there is an } s \in L \text{ with } (s, t) \in \tau\}$. Moreover, for a class C of tree transformations and a class \mathcal{L} of tree languages, we put $C(\mathcal{L}) = \{\tau(L) \mid \tau \in C \text{ and } L \in \mathcal{L}\}$.

We assume the reader to be familiar with recognizable tree languages, with top-down tree transducers (with and without regular look-ahead), and with bottom-up tree transducers, cf., e.g., [2,3,11,12]. Note that in [11,12] the notions root-to-frontier and frontier-to-root are used for top-down and bottom-up, respectively. We denote the class of recognizable tree languages by *REC*, and the classes of tree transformations computed by deterministic top-down tree transducers with regular look-ahead, linear deterministic top-down tree transducers with regular look-ahead, deterministic top-down tree transducers, and linear deterministic top-down tree transducers by *d-TOP^R*, *ld-TOP^R*, *d-TOP*, and *ld-TOP*, respectively.

3. Deterministic multi bottom-up tree transducers

Here we recall the definition of the deterministic multi bottom-up tree transducer from [10] and show some examples which will be needed in Section 4.

Definition 3.1. A *deterministic multi bottom-up tree transducer* (for short *d-mbutt*) is a tuple $M = (Q, \Sigma, \Delta, \text{root}, q_f, R)$, where

- Q is a ranked alphabet with $Q^{(0)} = \emptyset$, called the *set of states*,
 - Σ and Δ are ranked alphabets, called the *input and output ranked alphabets*, respectively,
 - root is a unary symbol, called the *root symbol*,
 - q_f is a unary symbol, called the *final state*,
- such that the sets Q , $\Sigma \cup \Delta$, $\{\text{root}\}$, and $\{q_f\}$ are pairwise disjoint,
- R is a finite set of *rules*, such that

- for every $k \geq 0$, $\sigma \in \Sigma^{(k)}$, $i \in [k]$, $n_i \geq 1$, $q_i \in Q^{(n_i)}$ there is at most one rule

$$\sigma(q_1(x_{1,1}, \dots, x_{1,n_1}), \dots, q_k(x_{k,1}, \dots, x_{k,n_k})) \rightarrow q(t_1, \dots, t_n),$$

with $n \geq 1$, $q \in Q^{(n)}$, $t_j \in T_\Delta(\{x_{i,j} \mid i \in [k], j \in [n_i]\})$ for $j \in [n]$ and

- for every $n \geq 1$, $q \in Q^{(n)}$ there is at most one rule

$$\text{root}(q(x_1, \dots, x_n)) \rightarrow q_f(t),$$

with $t \in T_\Delta(X_n)$.

The above rules are called $(\sigma, q_1, \dots, q_k)$ -*rule* and (root, q) -*rule*, respectively. For simplicity, in the $(\sigma, q_1, \dots, q_k)$ -rules of examples we will use also variables from X_ω .

Definition 3.2. Let $M = (Q, \Sigma, \Delta, \text{root}, q_f, R)$ be a d-mbutt.

The *derivation relation induced by M* is a binary relation \Rightarrow_M over $T_{\Sigma \cup \Delta \cup Q \cup \{\text{root}, q_f\}}$ defined in the following way. For every $\varphi, \psi \in T_{\Sigma \cup \Delta \cup Q \cup \{\text{root}, q_f\}}$, $\varphi \Rightarrow_M \psi$ iff there is a tree $\beta \in T_{\Sigma \cup \Delta \cup Q \cup \{\text{root}, q_f\}}(X_1)$ such that x_1 occurs exactly once in β and either

- there is a rule $\sigma(q_1(x_{1,1}, \dots, x_{1,n_1}), \dots, q_k(x_{k,1}, \dots, x_{k,n_k})) \rightarrow r$ in R , and
- for every $i \in [k]$ and $j \in [n_i]$ there is a tree $t_{i,j} \in T_\Delta$ such that $\varphi = \beta[\sigma(q_1(t_{1,1}, \dots, t_{1,n_1}), \dots, q_k(t_{k,1}, \dots, t_{k,n_k}))]$ and $\psi = \beta[r[x_{i,j} \leftarrow t_{i,j} \mid i \in [k], j \in [n_i]]]$ or
- there is a rule $\text{root}(q(x_1, \dots, x_n)) \rightarrow q_f(t)$ in R , and
- for every $i \in [n]$ there is a tree $t_i \in T_\Delta$ such that $\varphi = \beta[\text{root}(q(t_1, \dots, t_n))]$ and $\psi = \beta[q_f(t[t_1, \dots, t_n])]$.

The *tree transformation computed by M* is the (partial) function¹ $\tau_M : T_\Sigma \rightarrow T_\Delta$ defined by $\tau_M(s) = t$ iff $\text{root}(s) \Rightarrow_M^* q_f(t)$.

Example 3.3. Consider the d-mbutt $M_{\text{full}} = (Q, \Sigma, \Delta, \text{root}, q_f, R)$ with $Q = \{q^{(1)}\}$, $\Sigma = \{\gamma^{(1)}, \alpha^{(0)}\}$, $\Delta = \{\sigma^{(2)}, \alpha^{(0)}\}$, and with R containing the rules

$$\begin{aligned} \alpha &\rightarrow q(\alpha), \\ \gamma(q(x_1)) &\rightarrow q(\sigma(x_1, x_1)), \\ \text{root}(q(x_1)) &\rightarrow q_f(x_1). \end{aligned}$$

It is easy to see that M_{full} translates, for every $n \geq 0$, the monadic tree of height n over Σ into the full binary tree of height n over Δ .

Example 3.4. Consider the d-mbutt $M_{\text{non-cf}} = (Q, \Sigma, \Delta, \text{root}, q_f, R)$ with $Q = \{q^{(3)}\}$, $\Sigma = \{\gamma^{(1)}, \alpha^{(0)}\}$, $\Delta = \{\delta^{(3)}, \sigma^{(2)}, a^{(0)}, b^{(0)}, c^{(0)}\}$, and with R containing the rules

$$\begin{aligned} \alpha &\rightarrow q(a, b, c), \\ \gamma(q(x_1, x_2, x_3)) &\rightarrow q(\sigma(a, x_1), \sigma(b, x_2), \sigma(c, x_3)), \\ \text{root}(q(x_1, x_2, x_3)) &\rightarrow q_f(\delta(x_1, x_2, x_3)), \end{aligned}$$

which works as follows. In its three arguments, q accumulates trees of the form $\sigma(a, \dots \sigma(a, a) \dots)$, $\sigma(b, \dots \sigma(b, b) \dots)$, and $\sigma(c, \dots \sigma(c, c) \dots)$, respectively, where the number of occurrences of a , b , and c are equal. Thus, for every $n \geq 0$, $\text{yield}(\tau_{M_{\text{non-cf}}}(\gamma^n \alpha)) = a^{n+1} b^{n+1} c^{n+1}$ where, as usual, *yield* maps a tree to the concatenation of its leaves from left to right.

Example 3.5. Consider the d-mbutt $M_{\text{change}} = (Q, \Sigma, \Delta, \text{root}, q_f, R)$ with $Q = \{q^{(2)}\}$, $\Sigma = \Delta = \{\gamma^{(1)}, \delta^{(1)}, \alpha^{(0)}\}$, and with R containing the rules

$$\begin{aligned} \alpha &\rightarrow q(\alpha, \alpha), \\ \gamma(q(x_1, x_2)) &\rightarrow q(\gamma(x_1), \delta(x_2)), \\ \delta(q(x_1, x_2)) &\rightarrow q(\delta(x_2), \delta(x_2)), \\ \text{root}(q(x_1, x_2)) &\rightarrow q_f(x_1). \end{aligned}$$

¹ It is obvious that \Rightarrow_M is terminating and confluent, thus τ_M is well-defined.

The tree transducer M_{change} changes every occurrence of γ in an input tree that occurs below the topmost occurrence of δ into δ . Thus, for every $i \geq 0$ and $n_0, \dots, n_i \geq 0$,

$$\tau_{M_{\text{change}}}(\gamma^{n_i} \delta \dots \gamma^{n_1} \delta \gamma^{n_0} \alpha) = \gamma^{n_i} \delta^{i+n_{i-1}+\dots+n_0} \alpha.$$

For this purpose, reading a symbol γ in the input, M_{change} prepares in the first argument (in the second argument, respectively) of q for the situation that no more symbol δ (another symbol δ , respectively) will be found on top of γ . If later another occurrence of δ occurs, then the first argument, which became irrelevant, is deleted and two copies of the second argument are produced. Otherwise, the first argument is delivered as output.

It should be clear that $\tau_{M_{\text{change}}}$ can also be computed by a nondeleting linear deterministic top-down tree transducer, even by a relabeling. \square

Note that in the previous example the nonlinearity was essential to reconstruct “useful arguments”. Later we will even prove that the same computation cannot be performed by any linear d-mbutt M (cf. Definition 3.6 for the concept of linearity). Intuitively, if M could store k arguments during its computation, then after passing k occurrences of δ it has no more useful arguments, because similarly to the previous example, M will lose at least one of it passing every occurrence of δ , but in contrast to the example, M cannot copy useful arguments.

Definition 3.6. A d-mbutt $M = (Q, \Sigma, \Delta, \text{root}, q_f, R)$ is called

- *deterministic bottom-up tree transducer*, (for short *d-butt*), if $Q = Q^{(1)}$ and for every $q \in Q$, if there is a (root, q) -rule, then it has the form $\text{root}(q(x_1)) \rightarrow q_f(x_1)$.²
- *tree homomorphism*, if Q is a singleton, i.e., $Q = \{q\}$ and M is a total d-butt (i.e. for every $k \geq 0$, $\sigma \in \Sigma^{(k)}$ there is exactly one (σ, q, \dots, q) -rule in R and there is exactly one (root, q) -rule in R).
- *linear*, if for every $\sigma(q_1(x_{1,1}, \dots, x_{1,n_1}), \dots, q_k(x_{k,1}, \dots, x_{k,n_k})) \rightarrow r$ in R , $i \in [k]$, and $j \in [n_i]$, $|r|_{x_{i,j}} \leq 1$ and if for every $\text{root}(q(x_1, \dots, x_n)) \rightarrow r$ in R and $j \in [n]$, $|r|_{x_j} \leq 1$. For a linear d-mbutt and a linear d-butt, we will use the abbreviations *ld-mbutt* and *ld-butt*, respectively.

It is easy to see that M_{copy} of the introduction and $M_{\text{non-cf}}$ of Example 3.4 are ld-mbutt which are not d-butt, M_{full} of Example 3.3 is a tree homomorphism which is not linear, and M_{change} of Example 3.5 is neither linear nor a d-butt.

The classes of tree transformations computed by d-mbutt, ld-mbutt, d-butt, ld-butt, tree homomorphisms, and linear tree homomorphisms are denoted by *d-MBOT*, *ld-MBOT*, *d-BOT*, *ld-BOT*, *HOM*, and *l-HOM*, respectively.

Finally, we state a relation between the sizes of input and output trees for tree transformations computed by ld-mbutt, more precisely, the size of an output tree is (at most) linear in that of the corresponding input tree.

²The restriction on (root, q) -rules is needed to adapt our definition of d-butt to the usual definition in the literature w.r.t. computed tree transformations.

Lemma 3.7. For every ld-mbutt $M = (Q, \Sigma, \Delta, root, q_f, R)$, there exists a constant c_M such that, for every $s \in T_\Sigma$, if $\tau_M(s)$ exists, then $|\tau_M(s)| \leq c_M |s|$.

Proof. We define $d_M = \max\{|\rho|_\Delta \mid (\lambda \rightarrow \rho) \in R\}$ and $c_M = 2d_M$.

Obviously, since M is linear, every derivation step of M contributes at most d_M symbols to the output tree. Thus, since the derivation starting with $root(s)$ consists of $|s| + 1$ derivation steps, $|\tau_M(s)| \leq d_M(|s| + 1) \leq c_M |s|$. \square

4. The inclusion diagram for subclasses of d -MBOT

In this section we relate the class ld -MBOT to the tree transformation classes shown in Table 1. In particular, the class ld -MBOT is incomparable to the classes d -BOT, ld -TOP^R, and d -TOP. Moreover, the linearity condition “splits” the classes d -MBOT and d -TOP^R which in [10] were shown to be equal.

Formally, the tree transformation classes are related by an inclusion diagram, in which there is a sequence of ascending lines from a class C_1 to a class C_2 , iff $C_1 \subset C_2$. Hence, there is no sequence of ascending lines between two classes C_1 and C_2 , iff $C_1 - C_2 \neq \emptyset$ and $C_2 - C_1 \neq \emptyset$, i.e. C_1 and C_2 are incomparable.

Theorem 4.1. The diagram in Fig. 1 is the inclusion diagram for the classes d -MBOT, ld -MBOT, d -BOT, ld -BOT, d -TOP^R, ld -TOP^R, d -TOP, ld -TOP, HOM , and l -HOM.

Proof. The equation d -MBOT = d -TOP^R is proved in Theorem 4.4 of [10]. The correctness of the diagram without the class ld -MBOT follows immediately from Theorem 5.7 of [13]. To add ld -MBOT, it suffices to prove the following four statements where the last one will be proved in a separate lemma:

- ld -MBOT – d -BOT $\neq \emptyset$
- ld -MBOT – ld -TOP^R $\neq \emptyset$
- HOM – ld -MBOT $\neq \emptyset$
- ld -TOP – ld -MBOT $\neq \emptyset$ (cf. Lemma 4.2).

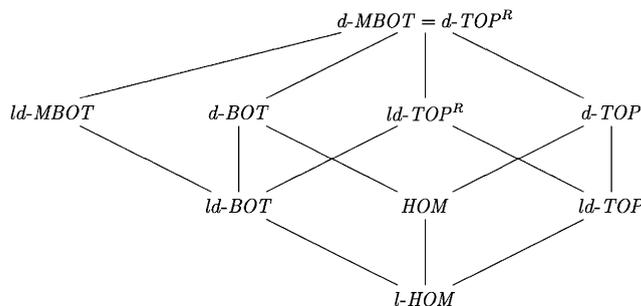


Fig. 1. Inclusion diagram for subclasses of d -MBOT.

$ld\text{-}MBOT - d\text{-}BOT \neq \emptyset$: Recall that $\tau_{M_{\text{non-cf}}} \in ld\text{-}MBOT$ and that $yield(\tau_{M_{\text{non-cf}}}(\gamma^n \alpha)) = a^{n+1}b^{n+1}c^{n+1}$, cf. Example 3.4. On the other hand, as it is stated after Theorem 4 of [7], the string language $\{a^n b^n c^n \mid n \geq 1\}$ is not in $yield(BOT(REC))$ (where BOT is the nondeterministic version of $d\text{-}BOT$). Thus, $\tau_{M_{\text{non-cf}}} \notin d\text{-}BOT$.

$ld\text{-}MBOT - ld\text{-}TOP^R \neq \emptyset$: The ld-mbutt M_{copy} of the introduction does not preserve recognizability, because it transforms $\{\gamma^n \alpha \mid n \geq 0\} \in REC$ into $\{\sigma(\gamma^n \alpha, \gamma^n \alpha) \mid n \geq 0\} \notin REC$. On the other hand, by Theorem 2.8 of [3] and Corollary 3.11 of [2], linear top-down tree transducers with regular look-ahead do preserve recognizability, i.e., $ld\text{-}TOP^R(REC) \subseteq REC$. Hence $\tau_{M_{\text{copy}}}$ is in $ld\text{-}MBOT - ld\text{-}TOP^R$.

$HOM - ld\text{-}MBOT \neq \emptyset$: Consider $\tau_{M_{\text{full}}} \in HOM$, cf. Example 3.3. Since $|\tau_{M_{\text{full}}}(\gamma^n \alpha)| = 2^{n+1} - 1$ (the size of the fully balanced tree of height n), this translation is not in $ld\text{-}MBOT$ by Lemma 3.7. \square

In the proof of the next lemma we will use the following short hand notation for a particular tree substitution. Let $t_1 \in T_{\Sigma^{(1)}}(\{x_i\})$ with $i \geq 1$ and $t_2 \in T_{\Sigma^{(1)} \cup \Sigma^{(0)}}(X_\omega)$ be monadic trees. Then we abbreviate $t_1[x_i \leftarrow t_2]$ by $t_1 t_2$. When using this notation, the substituted variable x_i will always be clear from the context. We note that tree substitution is associative.

Lemma 4.2. $ld\text{-}TOP - ld\text{-}MBOT \neq \emptyset$.

Proof. Let $\tau_{M_{\text{change}}}$ be the tree transformation of Example 3.5. Obviously, $\tau_{M_{\text{change}}} \in ld\text{-}TOP$. We will prove $\tau_{M_{\text{change}}} \notin ld\text{-}MBOT$. For this, let us recall that, for every $i \geq 0$ and $n_0, \dots, n_i \geq 0$,

$$\tau_{M_{\text{change}}}(\gamma^{n_i} \delta \dots \gamma^{n_1} \delta \gamma^{n_0} \alpha) = \gamma^{n_i} \delta^{i+n_{i-1}+\dots+n_0} \alpha.$$

Now assume that there is an ld-mbutt $M = (Q, \Sigma, \Sigma, root, q_f, R)$ with $\Sigma = \{\gamma^{(1)}, \delta^{(1)}, \alpha^{(0)}\}$ such that $\tau_{M_{\text{change}}} = \tau_M$.

The proof by contradiction will use a kind of pumping strategy. Therefore, we search for two positions in a sufficiently large input tree, such that, roughly speaking, the transducer M arrives in the same state $q \in Q^{(k)}$ at these positions. Moreover, if by the $(root, q)$ -rule the j th argument position of the “upper” q is projected into the output, then we are searching for a repetition of q , such that also the j th argument position of the “lower” q is projected into the j th argument position of the “upper” q and thus also into the output. This repetition of a *concrete* argument position needs the linearity condition. Now, if that part of the input tree between these two occurrences of q is omitted, then still the j th argument position of the “lower” q is projected into the output. Finally this will lead to a contradiction.

First we define

$$K = \max\{|\rho|_\gamma \mid (\lambda \rightarrow \rho) \in R\} + 1.$$

Since Q is finite, there are $N_0 \geq 0, N_1 \geq 1, k \geq 1, q \in Q^{(k)}, u_1, \dots, u_k \in T_\Sigma$, and $v_1, \dots, v_k \in T_\Sigma(X_k)$, such that

- (i) $(\gamma^K \delta)^{N_0} \gamma^K \alpha \Rightarrow_M^* q(u_1, \dots, u_k)$ and
- (ii) $(\gamma^K \delta)^{N_1} q(x_1, \dots, x_k) \Rightarrow_M^* q(v_1, \dots, v_k)$.³

³ We assume that \Rightarrow_M is extended to a relation on trees with variables by treating variables as nullary symbols.

(Note that (ii) implies $(\gamma^K \delta)^{N_1} q(w_1, \dots, w_k) \Rightarrow_M^* q(v_1[w_1, \dots, w_k], \dots, v_k[w_1, \dots, w_k])$ for every $w_1, \dots, w_k \in T_\Sigma$.) Hence, for every $i \geq 0$ we have $(\gamma^K \delta)^{i \cdot N_1 + N_0} \gamma^K \alpha \Rightarrow_M^* q(\dots)$.

Because of the definition of K and since $\tau_M((\gamma^K \delta)^{N_0} \gamma^K \alpha) = \gamma^K \delta^{N_0 \cdot (K+1)} \alpha$, the $(root, q)$ -rule contains at most $K - 1$ γ -symbols and it contains neither δ nor α . Hence there are $j \in [k]$ and $A \in \{0, \dots, K - 1\}$ such that

(iii) $(root(q(x_1, \dots, x_k)) \rightarrow q_f(\gamma^A x_j)) \in R$.

For every $i \geq 1$ define the argument position $a_i \in [k]$ of q as follows: $a_1 = j$ (i.e. we start at argument position j) and for every $i \geq 2$ the variable x_{a_i} occurs in $v_{a_{i-1}}$. This (infinite) sequence exists, which we prove by contradiction. Assume that there is $i \geq 1$ such that $v_{a_i} \in T_\Sigma$. Then, for every $b \in \{0, 1\}$,

$$\begin{aligned}
& root(\gamma^K \delta)^{(i+b) \cdot N_1 + N_0} \gamma^K \alpha \\
& \Rightarrow_M^* root(\gamma^K \delta)^{i \cdot N_1} q(\dots) \\
& \Rightarrow_M^* root(\gamma^K \delta)^{(i-1) \cdot N_1} q(\dots, v_{a_i}, \dots) \quad \text{with } v_{a_i} \in T_\Sigma \text{ at argument position } a_i \text{ of } q \\
& \Rightarrow_M^* root(\gamma^K \delta)^{(i-2) \cdot N_1} q(\dots, v_{a_{i-1}} v_{a_i}, \dots) \quad \text{with } v_{a_{i-1}} v_{a_i} \text{ at argument position } a_{i-1} \text{ of } q \\
& \Rightarrow_M^* \dots \\
& \Rightarrow_M^* root q(\dots, v_{a_1} \dots v_{a_{i-1}} v_{a_i}, \dots) \quad \text{with } v_{a_1} \dots v_{a_{i-1}} v_{a_i} \text{ at argument position} \\
& \quad \quad \quad a_1 = j \text{ of } q \\
& \Rightarrow_M q_f(\gamma^A v_{a_1} \dots v_{a_{i-1}} v_{a_i}).
\end{aligned}$$

Thus, $\tau_M((\gamma^K \delta)^{(i+1) \cdot N_1 + N_0} \gamma^K \alpha) = \gamma^A v_{a_1} \dots v_{a_{i-1}} v_{a_i} = \tau_M((\gamma^K \delta)^{i \cdot N_1 + N_0} \gamma^K \alpha)$ in contradiction to $\tau_M = \tau_{M_{\text{change}}}$ (because $N_1 \geq 1$).

Now we show that there is $L \geq 1$ such that $a_{L+1} = j$, i.e., x_j occurs in v_{a_L} (which means that we can return to argument position j):

Since k is finite, there are $i', i'' \geq 1$ with $i'' > i'$ such that $a_{i'+1} = a_{i''+1}$, i.e., the same variable occurs in $v_{a_{i'}}$ and $v_{a_{i''}}$. Since M is linear, by derivation (ii), for every $j' \in [k]$ there is at most one $j'' \in [k]$, such that $x_{j'}$ occurs in $v_{j''}$, and hence $a_{i'} = a_{i''}$. Thus, $a_{i'+1} = a_{i''+1}$ implies $a_{i'} = a_{i''}$. Choose i' and i'' such that i' is minimal. Then obviously, $i' = 1$. Now, taking $L = i'' - 1$ we get $a_{L+1} = a_{i''} = a_{i'} = a_1 = j$.

Define $t_j = v_{a_1} \dots v_{a_L}$. Note that t_j contains x_j , because v_{a_L} contains x_j . Then,

$$\begin{aligned}
& root(\gamma^K \delta)^{L \cdot N_1 + N_0} \gamma^K \alpha \\
& \Rightarrow_M^* root(\gamma^K \delta)^{L \cdot N_1} q(\dots, u_j, \dots) \quad \text{with } u_j \text{ at argument position } a_1 = j \text{ of } q \\
& \Rightarrow_M^* root(\gamma^K \delta)^{(L-1) \cdot N_1} q(\dots, v_{a_L} u_j, \dots) \quad \text{with } v_{a_L} u_j \text{ at argument position } a_L \text{ of } q \\
& \Rightarrow_M^* \dots \\
& \Rightarrow_M^* root q(\dots, v_{a_1} \dots v_{a_L} u_j, \dots) \quad \text{with } v_{a_1} \dots v_{a_L} u_j \text{ at argument position} \\
& \quad \quad \quad a_1 = j \text{ of } q \\
& \Rightarrow_M q_f(\gamma^A v_{a_1} \dots v_{a_L} u_j) \\
& = q_f(\gamma^A t_j u_j)
\end{aligned}$$

and thus, since $\tau_M = \tau_{M_{\text{change}}}$,

$$\gamma^K \delta^{(L \cdot N_1 + N_0) \cdot (K+1)} \alpha = \tau_{M_{\text{change}}}((\gamma^K \delta)^{L \cdot N_1 + N_0} \gamma^K \alpha) = \gamma^A t_j u_j. \quad (1)$$

In analogy, $\text{root}(\gamma^K \delta)^{N_0} \gamma^K \alpha \Rightarrow_M^* \text{root } q(\dots, u_j, \dots) \Rightarrow_M q_f(\gamma^A u_j)$ and thus

$$\gamma^K \delta^{N_0 \cdot (K+1)} \alpha = \gamma^A u_j. \quad (2)$$

From (2) follows that $u_j = \gamma^{K-A} \delta^{N_0 \cdot (K+1)} \alpha$. Substituting this in (1) we get $\gamma^K \delta^{(L \cdot N_1 + N_0) \cdot (K+1)} \alpha = \gamma^A t_j \gamma^{K-A} \delta^{N_0 \cdot (K+1)} \alpha$ and hence $t_j = x_j$. So $\gamma^K \delta^{(L \cdot N_1 + N_0) \cdot (K+1)} \alpha = \gamma^K \delta^{N_0 \cdot (K+1)} \alpha$, contradicting $L \cdot N_1 \geq 1$. \square

5. Conclusions and future research

In this paper we have shown the inclusion diagram of the class *ld-MBOT* of tree transformations computed by linear deterministic multi bottom-up tree transducers and of the classes in Table 1.

As a further research topic one might investigate a complete description of the composition monoid (in the sense of [9]) of these classes, as it was done in [13] for the classes of tree transformations in Table 1. A first step into this direction is established by the fact that the class *ld-MBOT* is closed under composition; this can be shown by the usual composition construction (as in [2,1]). A key problem, which also arises already in the case that we consider only the composition monoid of the bottom-up classes, is to answer whether $d\text{-MBOT} - \text{ld-MBOT} \circ \text{HOM} \neq \emptyset$ holds or not.

One of our referees pointed out that there is a close connection between our class *ld-MBOT* and $d\text{-TOP}_{su}^R$, where this latter is the class of tree transformations computed by single use deterministic top-down tree transducers with regular look-ahead. (For the single use property, cf. [14,5].) Moreover, the referee guesses that $\text{ld-MBOT}(\text{REC}) = d\text{-TOP}_{su}(\text{REC})$, which would imply $\text{ld-MBOT}(\text{REC}) = d\text{-TOP}_{fc}(\text{REC})$, where *fc* refers to the finite copying property, cf. [6]. The class $d\text{-TOP}_{fc}(\text{REC})$ is well-known in tree transducer theory. This opens up the possibility to prove some of our results by using further known facts from the literature. The referee also recalled that the classes $\text{yield}(d\text{-TOP}_{fc}(\text{REC}))$ and $d\text{-TOP}_{fc}(\text{REC})$ are characterized by multiple context-free grammars and multiple regular tree grammars, respectively, cf. Section 6 of [4]. Hence these multiple grammars are closely related to our multi bottom-up tree transducers. We will consider these relationships in future research.

Acknowledgements

We would like to thank the referees for their helpful comments; in particular, one of the referees indicated how to shorten the paper considerably.

References

- [1] B.S. Baker, Composition of top-down and bottom-up tree transductions, Inform. and Control 41 (2) (1979) 186–213.

- [2] J. Engelfriet, Bottom-up and top-down tree transformations—a comparison, *Math. Systems Theory* 9 (3) (1975) 198–231.
- [3] J. Engelfriet, Top-down tree transducers with regular look-ahead, *Math. Systems Theory* 10 (1977) 289–303.
- [4] J. Engelfriet, Context-free graph grammars, in: G. Rozenberg, A. Salomaa (Eds.), *Handbook of Formal Languages—Volume 3: Beyond Words*, Springer, Berlin, 1997, pp. 125–213, (Chapter 3).
- [5] J. Engelfriet, S. Maneth, Macro tree transducers, attribute grammars, and MSO definable tree translations, *Inform. and Comput.* 154 (1999) 34–91.
- [6] J. Engelfriet, G. Rozenberg, G. Slutzki, Tree transducers, L systems, and two-way machines, *J. Comput. System Sci.* 20 (2) (1980) 150–202.
- [7] J. Engelfriet, S. Skyum, Copying theorems, *Inform. Process. Lett.* 4 (1976) 157–161.
- [8] J. Engelfriet, H. Vogler, The equivalence of bottom-up and top-down tree-to-graph transducers, *J. Comput. System Sci.* 56 (1998) 332–356.
- [9] Z. Fülöp, A complete description for a monoid of deterministic bottom-up tree transformation classes, *Theoret. Comput. Sci.* 88 (1991) 253–268.
- [10] Z. Fülöp, A. Kühnemann, H. Vogler, A bottom-up characterization of deterministic top-down tree transducers with regular look-ahead, *Inform. Process. Lett.* 91 (2004) 57–67.
- [11] F. Gécseg, M. Steinby, *Tree Automata*, Akadémiai Kiadó, Budapest, 1984.
- [12] F. Gécseg, M. Steinby, Tree languages, in: G. Rozenberg, A. Salomaa (Eds.), *Handbook of Formal Languages*, Vol. 3, Springer, Berlin, 1997, pp. 1–68, (Chapter 1).
- [13] P. Gyenizse, S. Vágvölgyi, Compositions of deterministic bottom-up, top-down, and regular look-ahead tree transformations, *Theoret. Comput. Sci.* 156 (1996) 71–97.
- [14] A. Kühnemann, Berechnungsstärken von Teilklassen primitiv-rekursiver Programmschemata, Ph.D. Thesis, Dresden University of Technology, Dresden, 1997, Shaker, Aachen.
- [15] W.C. Rounds, Mappings and grammars on trees, *Math. Systems Theory* 4 (3) (1970) 257–287.
- [16] J.W. Thatcher, Generalized² sequential machine maps, *J. Comput. System Sci.* 4 (4) (1970) 339–367.