Compositions of Bottom-Up Tree Series Transformations

Andreas Maletti*

Technische Universität Dresden
Fakultät Informatik
D–01062 Dresden, Germany
e-mail: maletti@tcs.inf.tu-dresden.de

Abstract

Tree series transformations computed by bottom-up tree series transducers are called bottom-up tree series transformations. (Functional) compositions of such transformations are investigated. It turns out that bottom-up tree series transformations over commutative and $\aleph_0$-complete semirings are closed under left-composition with linear bottom-up tree series transformations and right-composition with boolean deterministic bottom-up tree series transformations.

1 Introduction

Tree series transducers [17, 9, 12] were introduced as the transducing devices corresponding to weighted tree automata [1, 16, 3]. So far, the latter are applied in code selection and tree pattern matching [11, 2]. Weighted transducers on strings are applied in image manipulation [see, *e. g.*, 7], where the images are coded as weighted string automata, and speech processing [see, *e. g.*, 19]. Since natural language processing features many transformations on parse trees, which come equipped with a degree of certainty, it seems natural to consider finite-state devices capable of transforming weighted trees. For natural language processing, the potential of tree series transducers over the semiring of the positive reals was recently discovered [14].

Since tree series transducers generalize tree transducers [21, 20, 8] by adding a cost component, we obtain top-down tree series transducers [17, 9, 12], where the input tree is processed from the root towards the leaves, and bottom-up tree series transducers [9, 12], where the input is processed from the leaves towards the root. In this paper, we deal with bottom-up tree series transducers. Moreover, four notions of substitution on tree series are known. These are

*Financially supported by the German Research Foundation (DFG, GK 334/3).

Z. Ésik and Z. Fülöp (Eds.): AFL 2005, pp. 187–199, 2005.

pure IO-substitution [5, 9], o-IO-substitution [12], [IO]-substitution [6], and OI-substitution [4, 17]. Here we deal with pure IO-substitution, since it seems to be the most appropriate choice for bottom-up tree series transducers.

Roughly speaking, a bottom-up tree series transducer is a bottom-up tree transducer [21, 20] in which the transitions carry a weight; a weight is an element of some semiring [15, 13]. The rewrite semantics works as follows. Along a successful computation on some input tree, the weights of the involved transitions are combined by means of the semiring multiplication; if there is more than one successful computation for some pair of input and output trees, then the weights of these computations are combined by means of the semiring addition.

In the unweighted case, bottom-up tree transformations are closed under left-composition with linear bottom-up tree transformations [8, Theorem 4.5] and right-composition with deterministic bottom-up tree transformations [8, Theorem 4.6]. In this paper we try to extend these results to bottom-up tree series transformations. The first result was already generalized to bottom-up tree series transformations [17, 9]. Essentially the authors obtain that, for arbitrary commutative and $\aleph_0$-complete semirings [15], bottom-up tree series transformations are closed under left-composition with nondeleting, linear bottom-up tree series transformations. We generalize this further by showing that the mentioned class of bottom-up tree series transformations is even closed under left-composition with linear bottom-up tree series transformations. The construction required to show this statement is mostly standard (*i. e.*, the transitions of the linear transducer are translated with the help of the second transducer) with one notable exception.

For commutative and $\aleph_0$-complete semirings, the class of bottom-up tree series transformations is closed under right-composition with boolean homomorphism bottom-up tree series transformations [9]. Using an adaptation of the standard construction, we also show that this class of bottom-up tree series transformations is actually closed under right-composition with boolean deterministic bottom-up tree series transformations.

2 Preliminaries

We use $\mathbb{N}$ to represent the set of nonnegative integers $\{0, 1, 2, \dots\}$, and we also use $\mathbb{N}_+ = \mathbb{N} \setminus \{0\}$. In the sequel, let $k, n \in \mathbb{N}$. We abbreviate $\{ i \in \mathbb{N} \mid 1 \leqslant i \leqslant k \}$ simply by $[k]$. Given sets A and I, we write A^I for the set of all mappings $f \colon I \longrightarrow A$. Occasionally, we use the family notation $(\, f(i)\,)_{i \in I}$ for f, and moreover, if $I = [k]$, then we generally write $\big(f(1), \dots, f(k)\big)$ or just $f(1) \cdots f(k)$. A set Σ which is nonempty and finite is also called *alphabet*, and the elements thereof are called *symbols*. A *ranked alphabet* is an alphabet Σ together with a mapping $\mathrm{rk}_\Sigma \colon \Sigma \longrightarrow \mathbb{N}$ associating to each symbol its *rank*. We use the denotation Σ_k to represent the set of symbols (of Σ) having rank k. Furthermore, we use the set $\mathrm{X} = \{ \mathrm{x}_i \mid i \in \mathbb{N}_+ \}$ of *(formal) variables* and the finite subset $\mathrm{X}_k = \{ \mathrm{x}_i \mid i \in [k] \}$. Given a ranked alphabet Σ and $V \subseteq \mathrm{X}$, the set of Σ-*trees indexed by* V, denoted by $T_\Sigma(V)$, is inductively defined to be the smallest set T

such that (i) $V \subseteq T$ and (ii) for every $k \in \mathbb{N}$, $\sigma \in \Sigma_k$, and $t_1, \ldots, t_k \in T$ also $\sigma(t_1, \ldots, t_k) \in T$. Since we generally assume that $\Sigma \cap \mathrm{X} = \emptyset$, we write α instead of $\alpha()$ whenever $\alpha \in \Sigma_0$. Moreover, we also write T_Σ to denote $T_\Sigma(\emptyset)$.

For every $t \in T_\Sigma(\mathrm{X})$, we denote by $|t|_x$ the number of occurrences of $x \in \mathrm{X}$ in t, and in addition, we use $\mathrm{var}(t) = \{ i \in \mathbb{N}_+ \mid |t|_{\mathrm{x}_i} \geqslant 1 \}$. Moreover, for every finite $I \subseteq \mathbb{N}_+$ and family $(t_i)_{i \in I}$ of $t_i \in T_\Sigma(\mathrm{X})$, the expression $t[t_i]_{i \in I}$ denotes the result of substituting in t every x_i by t_i for every $i \in I$. If $I = [n]$, then we simply write $t[t_1, \ldots, t_n]$. Let $V \subseteq \mathrm{X}$ be finite. We say that $t \in T_\Sigma(\mathrm{X})$ is *linear in* V (respectively, *nondeleting in* V), if every $x \in V$ occurs at most once (respectively, at least once) in t.

A *semiring* is an algebraic structure $\mathcal{A} = (A, +, \cdot, 0, 1)$ consisting of a commutative monoid $(A, +, 0)$ and a monoid $(A, \cdot, 1)$ such that $\cdot$ distributes over $+$ and 0 is absorbing with respect to $\cdot$. The semiring is called commutative, if $\cdot$ is commutative. As usual we use $\sum_{i \in I} a_i$ (respectively, $\prod_{i \in I} a_i$ for $I \subseteq \mathbb{N}$) for sums (respectively, products) of families $(a_i)_{i \in I}$ of $a_i \in A$ where for only finitely many $i \in I$ we have $a_i \neq 0$ (respectively, $a_i \neq 1$). For products the order of the factors is given by the order $0 < 1 < \cdots$ on the index set I. We say that $\mathcal{A}$ is $\aleph_0$-*complete*, whenever it is possible to define an infinitary sum operation $\sum_I$ for each countable index set I (*i. e.*, $\mathrm{card}(I) \leqslant \aleph_0$) such that for every family $(a_i)_{i \in I}$ of $a_i \in A$ the following three conditions are satisfied.

(i) $\sum_I (a_i)_{i \in I} = a_j$, if $I = \{j\}$, and $\sum_I (a_i)_{i \in I} = a_{j_1} + a_{j_2}$, if $I = \{j_1, j_2\}$ with $j_1 \neq j_2$.

(ii) $\sum_I (a_i)_{i \in I} = \sum_J \big(\sum_{I_j} (a_i)_{i \in I_j} \big)_{j \in J}$, whenever for some countable J we have $I = \bigcup_{j \in J} I_j$ and $I_{j_1} \cap I_{j_2} = \emptyset$ for all $j_1 \neq j_2$.

(iii) $\sum_I (a \cdot a_i \cdot a')_{i \in I} = a \cdot \big(\sum_I (a_i)_{i \in I} \big) \cdot a'$ for all $a, a' \in A$.

In the sequel, we simply write the accustomed $\sum_{i \in I} a_i$ instead of the cumbersome $\sum_I (a_i)_{i \in I}$, and when speaking about an $\aleph_0$-complete semiring, we implicitly assume $\sum_I$ to be given. Well-known $\aleph_0$-complete semirings are the Boolean semiring $\mathbb{B} = (\{\bot, \top\}, \vee, \wedge, \bot, \top)$ with disjunction and conjunction and the semiring of the nonnegative reals $\mathbb{R}_+ = (\mathbb{R}_+ \cup \{\infty\}, +, \cdot, 0, 1)$.

Let S be a set and $\mathcal{A} = (A, +, \cdot, 0, 1)$ be a semiring. A *(formal) power series* φ is a mapping $\varphi \colon S \longrightarrow A$. Given $s \in S$, we denote $\varphi(s)$ also by (φ, s) and write the series as $\sum_{s \in S} (\varphi, s)\, s$. The *support* of φ is $\mathrm{supp}(\varphi) = \{ s \in S \mid (\varphi, s) \neq 0 \}$. Power series with finite support are called *polynomials*, and power series with at most one support element are also called *singletons*. We denote the set of all power series $\varphi \colon S \longrightarrow A$ by $A\langle\!\langle S \rangle\!\rangle$. We call $\varphi \in A\langle\!\langle S \rangle\!\rangle$ *boolean*, if $(\varphi, s) = 1$ for every $s \in \mathrm{supp}(\varphi)$. The boolean singleton with empty support is denoted by $\widetilde{0}$. Power series $\varphi, \varphi' \in A\langle\!\langle S \rangle\!\rangle$ are summed componentwise; *i. e.*, $(\varphi + \varphi', s) = (\varphi, s) + (\varphi', s)$ for every $s \in S$. Finally, we also multiply the power series φ with a coefficient $a \in A$ componentwise; *i. e.*, $(a \cdot \varphi, s) = a \cdot (\varphi, s)$ for every $s \in S$.

In this paper, we only consider power series in which the set S is a set of trees. Such power series are also called *tree series*. A tree series $\varphi \in A\langle\!\langle T_\Sigma(\mathrm{X}) \rangle\!\rangle$

is said to be *linear* (respectively, *nondeleting*) *in* $V \subseteq \mathrm{X}$, if every $t \in \mathrm{supp}(\varphi)$ is linear (respectively, nondeleting) in V. Let $\mathcal{A}$ be an $\aleph_0$-complete semiring, $\varphi \in A\langle\langle T_\Sigma(\mathrm{X})\rangle\rangle$, $I \subseteq \mathbb{N}_+$ be finite, and $(\psi_i)_{i\in I}$ be a family of $\psi_i \in A\langle\langle T_\Sigma(\mathrm{X})\rangle\rangle$. The *pure IO tree series substitution* (for short: IO-substitution) *(of* $(\psi_i)_{i\in I}$ *into* φ*)* [5, 9], denoted by $\varphi \longleftarrow (\psi_i)_{i\in I}$, is defined by

$$\varphi \longleftarrow (\psi_i)_{i\in I} = \sum_{\substack{t\in T_\Sigma(\mathrm{X}),\\ (\forall i\in I):\, t_i\in T_\Sigma(\mathrm{X})}} (\varphi, t) \cdot \prod_{i\in I} (\psi_i, t_i)\, t[t_i]_{i\in I} \enspace .$$

Let $\mathcal{A} = (A, +, \cdot, 0, 1)$ be a semiring, Q be an alphabet, and Σ and Δ be ranked alphabets. A *bottom-up tree representation* μ *(over* Q*,* Σ*,* Δ*, and* $\mathcal{A}$*)* [17, 9] is a family $(\mu_k(\sigma))_{k\in\mathbb{N},\sigma\in\Sigma_k}$ of matrices $\mu_k(\sigma) \in A\langle\langle T_\Delta(\mathrm{X}_k)\rangle\rangle^{Q\times Q^k}$. A tree representation μ is said to be

- *polynomial* (respectively, *boolean*), if for every $k \in \mathbb{N}$, $\sigma \in \Sigma_k$, $q \in Q$, and $w \in Q^k$ the tree series $\mu_k(\sigma)_{q,w}$ is polynomial (respectively, boolean),
- *nondeleting* (respectively, *linear*), if for every $k \in \mathbb{N}$, $\sigma \in \Sigma_k$, $q \in Q$, and $w \in Q^k$ the entry $\mu_k(\sigma)_{q,w}$ is nondeleting (respectively, linear) in X_k,
- *deterministic* (respectively, *total*), if for every $k \in \mathbb{N}$, $\sigma \in \Sigma_k$, and $w \in Q^k$, there exists at most one (respectively, at least one) $(q, t) \in Q \times T_\Delta(\mathrm{X}_k)$ such that $t \in \mathrm{supp}(\mu_k(\sigma)_{q,w})$.

Usually when we specify a tree representation μ, we just specify some entries of $\mu_k(\sigma)$ and implicitly assume the remaining entries to be $\widetilde{0}$. A *bottom-up tree series transducer* [9, 12] is a sextuple $M = (Q, \Sigma, \Delta, \mathcal{A}, F, \mu)$ consisting of

- an alphabet Q of *states*,
- ranked alphabets Σ and Δ, also called *input* and *output ranked alphabet*,
- an $\aleph_0$-complete semiring $\mathcal{A} = (A, +, \cdot, 0, 1)$,
- a vector $F \in A\langle\langle T_\Delta(\mathrm{X}_1)\rangle\rangle^Q$ of nondeleting and linear tree series representing *final outputs*, and
- a bottom-up tree representation μ over Q, Σ, Δ, and $\mathcal{A}$.

Bottom-up tree series transducers inherit the properties from their tree representation; *e. g.*, a bottom-up tree series transducer with a polynomial bottom-up tree representation would be called polynomial bottom-up tree series transducer. Additionally, we say that M is a *homomorphism* bottom-up tree series transducer, if $Q = \{\star\}$, $F_\star = 1\, \mathrm{x}_1$, and μ is deterministic and total.

Let $M = (Q, \Sigma, \Delta, \mathcal{A}, F, \mu)$ be a bottom-up tree series transducer over the $\aleph_0$-complete semiring $\mathcal{A} = (A, +, \cdot, 0, 1)$. Then M induces a transformation $\|M\| \colon A\langle\langle T_\Sigma\rangle\rangle \longrightarrow A\langle\langle T_\Delta\rangle\rangle$ defined as follows. For every $k \in \mathbb{N}$, $\sigma \in \Sigma_k$, and

$t_1, \dots, t_k \in T_\Sigma$ we define the mapping $h_\mu \colon T_\Sigma \longrightarrow A\langle\langle T_\Delta \rangle\rangle^Q$ componentwise for every $q \in Q$ by

$$h_\mu\big(\sigma(t_1, \dots, t_k)\big)_q = \sum_{q_1,\dots,q_k \in Q} \mu_k(\sigma)_{q,q_1\cdots q_k} \longleftarrow (\, h_\mu(t_i)_{q_i}\,)_{i\in[k]} \enspace .$$

Moreover, $h_\mu(\varphi)_q = \sum_{t\in T_\Sigma}(\varphi, t) \cdot h_\mu(t)_q$ for every $\varphi \in A\langle\langle T_\Sigma\rangle\rangle$. Then for every $\varphi \in A\langle\langle T_\Sigma\rangle\rangle$ the *(IO) tree series transformation computed by* M is

$$\|M\|(\varphi) = \sum_{q\in Q} F_q \longleftarrow (h_\mu(\varphi)_q) \enspace .$$

By $\mathrm{BOT}(\mathcal{A})$ we denote the class of tree series transformations computable by bottom-up tree series transducers over the semiring $\mathcal{A}$. Similarly, we also use p–BOT($\mathcal{A}$) (respectively, b–BOT($\mathcal{A}$), l–BOT($\mathcal{A}$), n–BOT($\mathcal{A}$), d–BOT($\mathcal{A}$), and h–BOT($\mathcal{A}$)) for the class of tree series transformations computable by polynomial (respectively, boolean, linear, nondeleting, deterministic, and homomorphism) bottom-up tree series transducers over the semiring $\mathcal{A}$. Combinations of restrictions are handled in the usual manner; *i. e.*, let x–BOT($\mathcal{A}$) and y–BOT($\mathcal{A}$) be two classes of tree series transformations, then

$$xy\text{–BOT}(\mathcal{A}) = x\text{–BOT}(\mathcal{A}) \cap y\text{–BOT}(\mathcal{A}) \enspace .$$

According to custom, we write $\circ$ for function composition; so given two tree series transformations $\tau_1 \colon A\langle\langle T_\Sigma\rangle\rangle \longrightarrow A\langle\langle T_\Delta\rangle\rangle$ and $\tau_2 \colon A\langle\langle T_\Delta\rangle\rangle \longrightarrow A\langle\langle T_\Gamma\rangle\rangle$, then for every $\varphi \in A\langle\langle T_\Sigma\rangle\rangle$ we have that $(\tau_1 \circ \tau_2)(\varphi) = \tau_2(\tau_1(\varphi))$. This composition is extended to classes of transformations in the usual manner.

3 Compositions

First let us review what is known about compositions of bottom-up tree series transformations. Bottom-up tree transformations (*i. e.*, polynomial bottom-up tree series transformations over the Boolean semiring, [see 9, Section 4]) are closed under left-composition with linear bottom-up tree transformations [8, Theorem 4.5]; *i. e.*, lp–BOT($\mathbb{B}$) $\circ$ p–BOT($\mathbb{B}$) = p–BOT($\mathbb{B}$). This result was generalized to bottom-up tree series transformations over commutative, $\aleph_0$-complete semirings $\mathcal{A}$ [18, 9]. More precisely, it is shown [18, Theorem 2.4] that

$$\text{nl–BOT}(\mathcal{A}) \circ \text{nl–BOT}(\mathcal{A}) = \text{nl–BOT}(\mathcal{A}) \enspace .$$

In fact it is shown for nondeleting, linear top-down tree series transducers [9], but nondeleting, linear top-down tree series transducers and nondeleting, linear bottom-up tree series transducers are equally powerful [see 9, Theorem 5.24]. Moreover, nl–BOT($\mathcal{A}$) $\circ$ h–BOT($\mathcal{A}$) $\subseteq$ BOT($\mathcal{A}$) [9, Corollary 5.5]. So taking those results together and a decomposition [9, Lemma 5.6], we obtain the following result.

Theorem 3.1 For every commutative and $\aleph_0$-complete semiring $\mathcal{A}$

$$\text{nlp–BOT}(\mathcal{A}) \circ \text{p–BOT}(\mathcal{A}) = \text{p–BOT}(\mathcal{A}) \ . \tag{1}$$

Proof: The direction p–BOT($\mathcal{A}$) $\subseteq$ nlp–BOT($\mathcal{A}$) $\circ$ p–BOT($\mathcal{A}$) is trivial, so it remains to prove nlp–BOT($\mathcal{A}$) $\circ$ p–BOT($\mathcal{A}$) $\subseteq$ p–BOT($\mathcal{A}$).

$$\begin{aligned}
&\text{nlp–BOT}(\mathcal{A}) \circ \text{p–BOT}(\mathcal{A}) && \\
\subseteq\ &\text{nlp–BOT}(\mathcal{A}) \circ \text{nlp–BOT}(\mathcal{A}) \circ \text{h–BOT}(\mathcal{A}) && \text{[9, Lemma 5.6]} \\
\subseteq\ &\text{nlp–BOT}(\mathcal{A}) \circ \text{h–BOT}(\mathcal{A}) && \text{[18, Theorem 2.4]} \\
\subseteq\ &\text{p–BOT}(\mathcal{A}) && \text{[9, Corollary 5.5]}
\end{aligned}$$

□

We should like to obtain a result like l–BOT($\mathcal{A}$) $\circ$ BOT($\mathcal{A}$) = BOT($\mathcal{A}$) for all commutative and $\aleph_0$-complete semirings $\mathcal{A}$. We try to follow the classical (unweighted) construction, so we first extend h_μ such that it can treat variables (of X). We extend h_μ to $T_\Sigma(\mathrm{X})$ by supplying, for some $V \subseteq \mathbb{N}_+$, a mapping $\overline{q} \in Q^V$, which associates a state $\overline{q}(v)$, often written as $\overline{q}_v$, to the variable x_v for $v \in V$. Intuitively speaking, the state $\overline{q}_v$ represents the initial state, with which the computation should be started at the leaves labeled x_v in the input tree. For all states $q \in Q$ different from $\overline{q}_v$ it should not be possible to start a (meaningful) computation at x_v (*i. e.*, $h_\mu^{\overline{q}}(\mathrm{x}_v)_q = \widetilde{0}$). This mapping is then extended to $T_\Sigma(\mathrm{X})$ in a manner analogous to h_μ.

Definition 3.2 (Extension of h_μ) Let $(Q, \Sigma, \Delta, \mathcal{A}, F, \mu)$ be a bottom-up tree series transducer. For every finite $V \subseteq \mathbb{N}_+$ and $\overline{q} \in Q^V$ we define the mapping $h_\mu^{\overline{q}} \colon T_\Sigma(\mathrm{X}) \longrightarrow A\langle\langle T_\Delta(\mathrm{X})\rangle\rangle^Q$ componentwise for every $q \in Q$ as follows. For every $v \in V$, $n \in \mathbb{N}_+ \setminus V$, $k \in \mathbb{N}$, $\sigma \in \Sigma_k$, and $t_1, \dots, t_k \in T_\Sigma(\mathrm{X})$

$$h_\mu^{\overline{q}}(\mathrm{x}_n)_q = 1\ \mathrm{x}_n \tag{2}$$

$$h_\mu^{\overline{q}}(\mathrm{x}_v)_q = \begin{cases} 1\ \mathrm{x}_v & \text{if } q = \overline{q}_v \ , \\ \widetilde{0} & \text{otherwise} \end{cases} \tag{3}$$

$$h_\mu^{\overline{q}}(\sigma(t_1, \dots, t_k))_q = \sum_{q_1, \dots, q_k \in Q} \mu_k(\sigma)_{q, q_1 \cdots q_k} \longleftarrow (\, h_\mu^{\overline{q}}(t_i)_{q_i}\,)_{i \in [k]} \ . \tag{4}$$

The mapping $h_\mu^{\overline{q}} \colon A\langle\langle T_\Sigma(\mathrm{X})\rangle\rangle \longrightarrow A\langle\langle T_\Delta(\mathrm{X})\rangle\rangle^Q$ is given for every $\varphi \in A\langle\langle T_\Sigma(\mathrm{X})\rangle\rangle$ by

$$h_\mu^{\overline{q}}(\varphi)_q = \sum_{t \in T_\Sigma(\mathrm{X})} (\varphi, t) \cdot h_\mu^{\overline{q}}(t)_q \ .$$

□

Next we define the composition of two bottom-up tree series transducers. Let $M_1 = (Q_1, \Sigma, \Gamma, \mathcal{A}, F_1, \mu_1)$ and $M_2 = (Q_2, \Gamma, \Delta, \mathcal{A}, F_2, \mu_2)$ be bottom-up tree series transducers, which are eligible for composition; *i. e.*, they are defined over the same semiring, and the output ranked alphabet of M_1 is the input ranked alphabet of M_2. Then, similar to the (unweighted) product construction of bottom-up tree transducers, we translate the transitions of M_1 with the

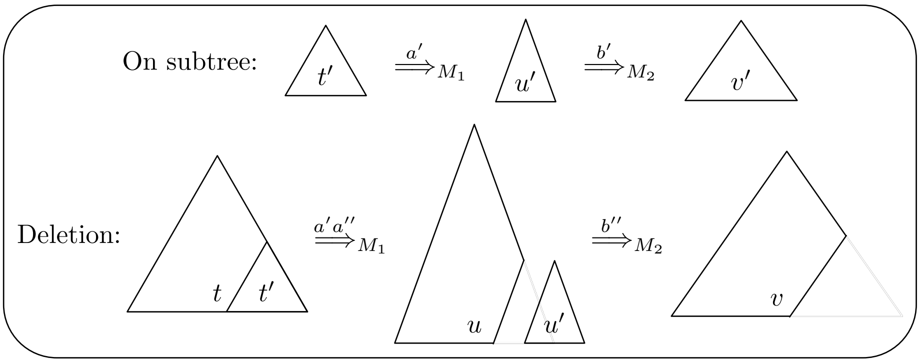

Figure 1: Computation of M_1 followed by M_2.

help of the transitions of M_2. Let $k \in \mathbb{N}$, $\sigma \in \Sigma_k$, $p, p_1, \ldots, p_k \in Q_1$, and $q, q_1, \ldots, q_k \in Q_2$. Roughly, we obtain the entry $\mu_k(\sigma)_{(p,q),(p_1,q_1)\cdots(p_k,q_k)}$ in the tree representation μ of the composition of M_1 and M_2 by applying the extended mapping $h_{\mu_2}^{q_1\cdots q_k}$ to the entry $(\mu_1)_k(\sigma)_{p,p_1\cdots p_k}$. Thereby, we process the output trees present in $\operatorname{supp}((\mu_1)_k(\sigma)_{p,p_1\cdots p_k})$ with the help of M_2 starting the computation at the variables $\mathrm{x}_1, \ldots, \mathrm{x}_k$ in states $q_1, \ldots, q_k$.

However, there is a small problem which does not arise in the unweighted case. We depict the problem in Figures 1 and 2. Let us suppose that M_1 translates an input tree $t \in T_\Sigma$ into an output tree $u \in T_\Gamma$ with weight $a \in A$. During the translation, M_1 decides to delete the translation $u' \in T_\Gamma$ with weight $a' \in A$ of an input subtree $t' \in T_\Sigma$. Then due to the definition of IO-substitution the weight a' of u' contributes to the weight a of u, whereas u' does not contribute to u. Furthermore, let us suppose that M_2 would transform u into $v \in T_\Delta$ at weight $b \in A$ and u' into $v' \in T_\Delta$ at weight $b' \in A$. Since M_2 does not process u', the weight b' does not contribute to b. However, the composition of M_1 and M_2, when processing the input subtree t', transforms t' into u' at weight a' using the rules of M_1 and immediately also transforms u' into v' at weight b' using the rules of M_2. If the composition tree series transducer now deletes the translation v' of t', then a' and b' still contribute to the weight of the overall transformation. This contrasts the situation encountered when M_1 and M_2 run separately, because there only a' contributed to the weight of the overall transformation. In the classical case of tree transducers, b' could only be 0 or 1, so that one just had to avoid that $b' = 0$. In principle, this is achieved by requiring M_2 to be total (however, by adjoining a dummy state, each bottom-up tree transducer can be turned into a total one computing the same tree transformation). The construction we propose here is similar, but has the major disadvantage that, for example, determinism is not preserved.

Specifically, we address the aforementioned problem by manipulating the second transducer M_2 such that it has a state $\perp$ which transforms each input tree into some output tree $\alpha \in \Delta_0$ at weight 1. Let us call the resulting bottom-up tree series transducer M_2'. Then we compose M_1 and M_2' by processing those

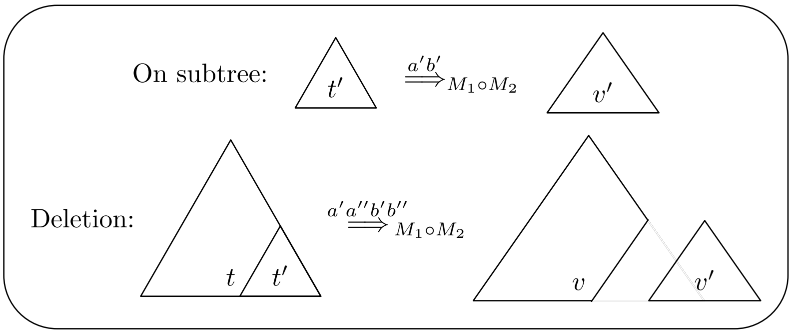

Figure 2: Computation of $M_1 \circ M_2$.

subtrees, which M_1 decided to delete, in state $\perp$.

Definition 3.3 (Composition) Let $\mathcal{A} = (A, +, \cdot, 0, 1)$ be an $\aleph_0$-complete semiring. Moreover, let $M_1 = (Q_1, \Sigma, \Gamma, \mathcal{A}, F_1, \mu_1)$ and $M_2 = (Q_2, \Gamma, \Delta, \mathcal{A}, F_2, \mu_2)$ be two bottom-up tree series transducers over $\mathcal{A}$. Let $\perp \notin Q_2$, $Q_2' = Q_2 \cup \{\perp\}$, and $\alpha \in \Delta_0$. We first construct $M_2' = (Q_2', \Gamma, \Delta, \mathcal{A}, F_2', \mu_2')$ with $(F_2')_q = (F_2)_q$ for every $q \in Q_2$ and $(F_2')_\perp = \widetilde{0}$. The tree representation μ_2' is defined for every $k \in \mathbb{N}$, $\gamma \in \Gamma_k$, and $q, q_1, \ldots, q_k \in Q_2$ by

$$(\mu_2')_k(\gamma)_{q,q_1\cdots q_k} = (\mu_2)_k(\gamma)_{q,q_1\cdots q_k} \tag{5}$$

$$(\mu_2')_k(\gamma)_{\perp,\perp\cdots\perp} = 1\;\alpha\ \ . \tag{6}$$

The *composition of* M_1 *and* M_2, denoted by $M_1 \circ M_2$, is defined to be the bottom-up tree series transducer $(M_1 \circ M_2) = (Q_1 \times Q_2', \Sigma, \Delta, \mathcal{A}, F, \mu)$ with

$$F_{(p,q)} = \sum_{q' \in Q_2'} (F_2')_{q'} \longleftarrow \left(h^q_{\mu_2'}\big((F_1)_p\big)_{q'}\right) \tag{7}$$

$$\mu_k(\sigma)_{(p,q),(p_1,q_1)\cdots(p_k,q_k)} = h^{q_1\cdots q_k}_{\mu_2'}\Bigg(\sum_{\substack{t \in T_\Gamma(X_k),\\ (\forall i \in [k]):\, i \notin \mathrm{var}(t) \iff q_i = \perp}} \big((\mu_1)_k(\sigma)_{p,p_1\cdots p_k}, t\big)\ t\Bigg)_q \tag{8}$$

$$\mu_k(\sigma)_{(p,\perp),(p_1,\perp)\cdots(p_k,\perp)} = h^{\perp\cdots\perp}_{\mu_2'}\big((\mu_1)_k(\sigma)_{p,p_1\cdots p_k}\big)_\perp \tag{9}$$

for every $k \in \mathbb{N}$, $\sigma \in \Sigma_k$, $p, p_1, \ldots, p_k \in Q_1$, $q \in Q_2$, and $q_1, \ldots, q_k \in Q_2'$. All the remaining entries in F and μ are $\widetilde{0}$. □

It is quite clear that the composition $M_1 \circ M_2$ does not always compute $\|M_1\| \circ \|M_2\|$, because already for bottom-up tree transducers (*i. e.*, polynomial bottom-up tree series transducers over $\mathbb{B}$) it can be shown that the computed transformations are not closed with respect to composition. However, we have already mentioned that p–BOT($\mathbb{B}$) is closed under left-composition with lp–BOT($\mathbb{B}$). This is why we assume M_1 to be linear in the next lemma, which shows that in this case we obtain $\|M_1 \circ M_2\| = \|M_1\| \circ \|M_2\|$.

Lemma 3.4 (Correctness of the composition) Let $\mathcal{A}$ be a commutative, $\aleph_0$-complete semiring, $M_1 = (Q_1, \Sigma, \Gamma, \mathcal{A}, F_1, \mu_1)$ and $M_2 = (Q_2, \Gamma, \Delta, \mathcal{A}, F_2, \mu_2)$ be bottom-up tree series transducers, of which M_1 is linear.

$$\|M_1 \circ M_2\| = \|M_1\| \circ \|M_2\| \ . \tag{10}$$

Proof: We assume the symbols of Definition 3.3.

$$\begin{aligned}
&\|M_2\|\big(\|M_1\|(\varphi)\big) \\
= &\sum_{p \in Q_1, q' \in Q_2'} (F_2')_{q'} \longleftarrow \big(h_{\mu_2'}\big((F_1)_p \longleftarrow (h_{\mu_1}(\varphi)_p)\big)_{q'}\big) \\
&\quad \text{(by the definition of } \|\cdot\|) \\
= &\sum_{p \in Q_1, q \in Q_2'} \Big(\sum_{q' \in Q_2'} (F_2')_{q'} \longleftarrow \big(h^q_{\mu_2'}\big((F_1)_p\big)_{q'}\big)\Big) \longleftarrow \big(h_{\mu_2'}(h_{\mu_1}(\varphi)_p)_q\big) \\
&\quad \text{(see [10, Lemma 6.5] and [18, Lemma 2.2])} \\
= &\sum_{p \in Q_1, q \in Q_2'} F_{(p,q)} \longleftarrow (h_\mu(\varphi)_{(p,q)}) \\
&\quad \text{(by } h_{\mu_2'}\big(h_{\mu_1}(t)_p\big)_q = h_\mu(t)_{(p,q)} \text{ and the definition of } F_{(p,q)}) \\
= &\|M\|(\varphi) \\
&\quad \text{(by the definition of } \|\cdot\|) \qquad \Box
\end{aligned}$$

In the sequel we use the notation $[y]$ where y is one of the abbreviations of restrictions (*i. e.*, $y \in \{\mathrm{p}, \mathrm{b}, \mathrm{l}, \mathrm{n}, \mathrm{d}, \mathrm{h}\}$) in equalities to mean that this restriction is optional; *i. e.*, throughout the statement $[y]$ can be substituted by the empty word or by y. For example, [l]p–BOT($\mathcal{A}$) = nlp–BOT($\mathcal{A}$) $\circ$ [l]h–BOT($\mathcal{A}$) states that the class of tree series transformations computable by polynomial (respectively, linear polynomial) bottom-up tree series transducers coincides with the composition of the class of tree series transformations computable by nondeleting and linear polynomial bottom-up tree series transducers with the class of tree series transformations computable by homomorphism (respectively, linear homomorphism) bottom-up tree series transducers.

It is easy to see that whenever M_1 and M_2 are polynomial (respectively, nondeleting, linear), then also $M_1 \circ M_2$ is polynomial (respectively, nondeleting, linear). Together with Lemma 3.4 this yields the first main theorem.

Theorem 3.5 Let $\mathcal{A}$ be a commutative and $\aleph_0$-complete semiring.

$$[\mathrm{p}][\mathrm{n}]\mathrm{l}\text{–BOT}(\mathcal{A}) \circ [\mathrm{p}][\mathrm{n}][\mathrm{l}]\text{–BOT}(\mathcal{A}) = [\mathrm{p}][\mathrm{n}][\mathrm{l}]\text{–BOT}(\mathcal{A}) \tag{11}$$

Proof: The statement follows from Lemma 3.4. $\Box$

We note that our construction does not preserve determinism [*cf.* 9, Corollary 5.5]. Further, neither the statement hl–BOT($\mathcal{A}$) $\circ$ h–BOT($\mathcal{A}$) = h–BOT($\mathcal{A}$) nor hnl–BOT($\mathcal{A}$) $\circ$ h–BOT($\mathcal{A}$) = h–BOT($\mathcal{A}$) follow from Lemma 3.4, because we introduce the state $\perp$ and thus our composition $M_1 \circ M_2$, in general, has more than one state. The correctness of the latter two statements thus remains open.

Let us consider an example. Imagine a game to be played between two players. Player I moves first and the moves of the players alternate. Each player can play one out of three potential moves (called l, m, and r), however the second player may not play the same move as the first player just played. We model this scenario by a game tree which contains three types of nodes. First there are σ-nodes indicating that one of the players should make a move. Such a node has exactly three successors, which represent the remaining game to be played in case the moving player chooses to play l, m, and r, respectively. Second, there are α- and β-nodes indicating that Player I, respectively Player II, has won the game. Third, l-, m-, and r-nodes represent that the player played this option. (Randomized) strategies for both players can now be coded as bottom-up tree series transducers (in fact, it is easier to code them as linear top-down tree series transducers, but given such we can easily obtain a semantically equivalent linear bottom-up tree series transducer [12, Theorem 5.26]). The composition of the two bottom-up tree series transducers (*i. e.*, of the two strategies) can then be applied to compute, for example, the chances of winning the game for each player.

Example 3.6 Let $\Sigma = \Sigma_0 \cup \Sigma_3$ with $\Sigma_3 = \{\sigma\}$ and $\Sigma_0 = \{\alpha, \beta\}$, $\Gamma_1' = \{\mathrm{l}, \mathrm{m}, \mathrm{r}\}$, and $\Gamma = \Gamma_1' \cup \Sigma$. Moreover, let $M_1 = (\{\perp, \top\}, \Sigma, \Gamma, \mathbb{R}_+, F_1, \mu_1)$ be the bottom-up tree series transducer with $(F_1)_\top = 1\,\mathrm{x}_1$ and $(F_1)_\perp = \widetilde{0}$ and

$$
\begin{aligned}
(\mu_1)_0(\alpha)_\perp = (\mu_1)_0(\alpha)_\top &= 1\,\alpha \\
(\mu_1)_0(\beta)_\perp = (\mu_1)_0(\beta)_\top &= 1\,\beta \\
(\mu_1)_3(\sigma)_{\top,\perp\perp\perp} &= 0.1\,\mathrm{l}(\mathrm{x}_1) + 0.3\,\mathrm{m}(\mathrm{x}_2) + 0.6\,\mathrm{r}(\mathrm{x}_3) \\
(\mu_1)_3(\sigma)_{\perp,\top\top\top} &= 1\,\sigma(\mathrm{x}_1, \mathrm{x}_2, \mathrm{x}_3) \enspace .
\end{aligned}
$$

The first player's strategy is modeled by M_1, and we represent a strategy of the second player by $M_2 = (\Gamma_1' \cup \{\top\}, \Gamma, \Sigma, \mathbb{R}_+, F_2, \mu_2)$ with $(F_2)_\top = 1\,\mathrm{x}_1$, $(F_2)_\gamma = \widetilde{0}$ and for every $\gamma \in \Gamma_1'$

$$
\begin{aligned}
(\mu_2)_0(\alpha)_\gamma = (\mu_2)_0(\alpha)_\top &= 1\,\alpha \\
(\mu_2)_0(\beta)_\gamma = (\mu_2)_0(\beta)_\top &= 1\,\beta \\
(\mu_2)_1(\gamma)_{\top,\gamma} &= 1\,\mathrm{x}_1 \\
(\mu_2)_3(\sigma)_{\mathrm{l},\top\top\top} &= 0.4\,\mathrm{x}_2 + 0.6\,\mathrm{x}_3 \\
(\mu_2)_3(\sigma)_{\mathrm{m},\top\top\top} &= 0.5\,\mathrm{x}_1 + 0.5\,\mathrm{x}_3 \\
(\mu_2)_3(\sigma)_{\mathrm{r},\top\top\top} &= 0.7\,\mathrm{x}_1 + 0.3\,\mathrm{x}_2 \enspace .
\end{aligned}
$$

Now let us consider the game tree $t = \sigma\big(\sigma(\alpha, \beta, \alpha), \beta, \sigma(\alpha, \beta, \beta)\big)$. Then

$$
\begin{aligned}
\|M_1\|(1\,t) &= 0.1\,\mathrm{l}\big(\sigma(\alpha, \beta, \alpha)\big) + 0.3\,\mathrm{m}(\beta) + 0.6\,\mathrm{r}\big(\sigma(\alpha, \beta, \beta)\big) \\
\|M_2\|\big(\|M_1\|(1\,t)\big) &= 0.48\,\alpha + 0.52\,\beta \enspace ,
\end{aligned}
$$

showing that for this particular game Player II has a slightly higher chance to win the game.

Now let us compose the two bottom-up tree series transducers. The composition $M_1 \circ M_2 = (Q, \Sigma, \Sigma, \mathcal{A}, F, \mu)$ is defined by $Q = \{\bot, \top\} \times \{\bot, \top, \mathrm{l}, \mathrm{m}, \mathrm{r}\}$ and $F_{(\top,\top)} = 1\,\mathrm{x}_1$ and $F_q = \widetilde{0}$ for all $q \in Q \setminus \{(\top,\top)\}$. Finally, the tree representation μ is defined for every $p \in \{\bot, \top\}$, $q \in \Gamma'_1 \cup \{\top\}$, and $\gamma \in \Gamma'_1$ by

$$
\begin{aligned}
\mu_0(\alpha)_{(p,q)} = \mu_0(\alpha)_{(p,\bot)} = \mu_0(\beta)_{(p,\bot)} &= 1\,\alpha \\
\mu_0(\beta)_{(p,q)} &= 1\,\beta \\
\mu_3(\sigma)_{(\top,\top),(\bot,\mathrm{l})(\bot,\bot)(\bot,\bot)} &= 0.1\,\mathrm{x}_1 \\
\mu_3(\sigma)_{(\top,\top),(\bot,\bot)(\bot,\mathrm{m})(\bot,\bot)} &= 0.3\,\mathrm{x}_2 \\
\mu_3(\sigma)_{(\top,\top),(\bot,\bot)(\bot,\bot)(\bot,\mathrm{r})} &= 0.6\,\mathrm{x}_3 \\
\mu_3(\sigma)_{(\bot,\gamma),(\top,\top)(\top,\top)(\top,\top)} &= \begin{cases} 0.4\,\mathrm{x}_2 + 0.6\,\mathrm{x}_3 & \text{if } \gamma = \mathrm{l}\ , \\ 0.5\,\mathrm{x}_1 + 0.5\,\mathrm{x}_3 & \text{if } \gamma = \mathrm{m}\ , \\ 0.7\,\mathrm{x}_1 + 0.3\,\mathrm{x}_2 & \text{if } \gamma = \mathrm{r}\ , \end{cases} \\
\mu_3(\sigma)_{(\bot,\bot),(\top,\bot)(\top,\bot)(\top,\bot)} &= 1\,\alpha\ .
\end{aligned}
$$

If we compute $\|M\|(1\,t)$, it shows the expected result $0.48\,\alpha + 0.52\,\beta$. □

Finally, let us consider the second result, which states that bottom-up tree transformations are closed under right-composition with deterministic bottom-up tree transformations [8, Theorem 4.6]. This result was also generalized to $\mathrm{BOT}(\mathcal{A}) \circ \mathrm{bh–BOT}(\mathcal{A}) = \mathrm{BOT}(\mathcal{A})$ [9, Corollary 5.5]. Since we have already seen that our previous construction destroys determinism, we simplify the construction somewhat to obtain a construction which is the analogue of the construction for the unweighted case. Note that without loss of generality we may assume a bottom-up tree series transducer to be total; the construction required to show this is the usual one.

Definition 3.7 Let $\mathcal{A} = (A, +, \cdot, 0, 1)$ be an $\aleph_0$-complete semiring. Further, let $M_1 = (Q_1, \Sigma, \Gamma, \mathcal{A}, F_1, \mu_1)$ and $M_2 = (Q_2, \Gamma, \Delta, \mathcal{A}, F_2, \mu_2)$ be two bottom-up tree series transducers over $\mathcal{A}$. The *(simple) composition of M_1 and M_2*, denoted by $M_1 \circ_{\mathrm{s}} M_2$, is defined to be the bottom-up tree series transducer $M_1 \circ_{\mathrm{s}} M_2 = (Q_1 \times Q_2, \Sigma, \Delta, \mathcal{A}, F, \mu)$ with

$$F_{(p,q)} = \sum_{q' \in Q_2} (F_2)_{q'} \longleftarrow \big(h^{q}_{\mu_2}\big((F_1)_p\big)_{q'}\big) \tag{12}$$

$$\mu_k(\sigma)_{(p,q),(p_1,q_1)\cdots(p_k,q_k)} = h^{q_1\cdots q_k}_{\mu_2}\big((\mu_1)_k(\sigma)_{p,p_1\cdots p_k}\big)_q \tag{13}$$

for every $k \in \mathbb{N}$, $\sigma \in \Sigma_k$, $p, p_1, \ldots, p_k \in Q_1$, and $q, q_1, \ldots, q_k \in Q_2$. All the remaining entries in F and μ are $\widetilde{0}$. □

It is easily seen that $M_1 \circ_{\mathrm{s}} M_2$ is deterministic, whenever M_1 and M_2 are deterministic. Moreover, $M_1 \circ_{\mathrm{s}} M_2$ is a homomorphism, if M_1 and M_2 are homomorphisms and M_2 is boolean. Note that, in general, the restriction that M_2

is boolean is necessary in the last statement, because otherwise the composition $M_1 \circ_{\mathrm{s}} M_2$ might not be total.

Lemma 3.8 Let $\mathcal{A}$ be a commutative and $\aleph_0$-complete semiring, M_1 and M_2 be bottom-up tree series transducers eligible for composition, of which M_2 is boolean, total, and deterministic.

$$\|M_1 \circ_{\mathrm{s}} M_2\| = \|M_1\| \circ \|M_2\| \ . \tag{14}$$

Proof: The proof is similar to the proof of Lemma 3.4. □

Thus we obtain the following final theorem [see 9, Corollary 5.5].

Theorem 3.9 Let $\mathcal{A}$ be a commutative and $\aleph_0$-complete semiring.

$$\text{[p][n][l][d][h]–BOT}(\mathcal{A}) \circ \text{[p][n][l][h]bd–BOT}(\mathcal{A}) = \text{[p][n][l][d][h]–BOT}(\mathcal{A}) \tag{15}$$

Proof: The statement follows from Lemma 3.8. □

Acknowledgements: The author wishes to express his deepest gratitude to the referees of the draft version of this paper. Their comments enabled the author to substantially improve the presentation of the results.

References

[1] J. Berstel and C. Reutenauer. Recognizable formal power series on trees. *Theoret. Comput. Sci.*, 18(2):115–148, 1982.

[2] B. Borchardt. Code selection by tree series transducers. In *Proc. 9th Int. Conf. on Implementation and Application of Automata*, volume 3317 of LNCS, pages 57–67. Springer, 2004.

[3] B. Borchardt and H. Vogler. Determinization of finite state weighted tree automata. *J. Autom. Lang. Combin.*, 8(3):417–463, 2003.

[4] S. Bozapalidis. Equational elements in additive algebras. *Theory Comput. Systems*, 32(1):1–33, 1999.

[5] S. Bozapalidis. Context-free series on trees. *Inform. Comput.*, 169(2):186–229, 2001.

[6] S. Bozapalidis and G. Rahonis. On the closure of recognizable tree series under tree homomorphism. In M. Droste and H. Vogler, editors, *Weighted Automata—Theory and Applications*, page 34. Technische Universität Dresden, 2004.

[7] K. Culik II and J. Kari. Digital images and formal languages. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3 — Beyond Words, chapter 10, pages 599–616. Springer, 1997.

[8] J. Engelfriet. Bottom-up and top-down tree transformations—a comparison. *Math. Systems Theory*, 9(3):198–231, 1975.

[9] J. Engelfriet, Z. Fülöp, and H. Vogler. Bottom-up and top-down tree series transformations. *J. Autom. Lang. Combin.*, 7(1):11–70, 2002.

[10] Z. Ésik and W. Kuich. Formal tree series. *J. Autom. Lang. Combin.*, 8(2): 219–285, 2003.

[11] C. Ferdinand, H. Seidl, and R. Wilhelm. Tree automata for code selection. *Acta Inform.*, 31(8):741–760, 1994.

[12] Z. Fülöp and H. Vogler. Tree series transformations that respect copying. *Theory Comput. Systems*, 36(3):247–293, 2003.

[13] J. S. Golan. *Semirings and their Applications.* Kluwer Academic, Dordrecht, 1999.

[14] J. Graehl and K. Knight. Training tree transducers. In S. Dumais, D. Marcu, and S. Roukos, editors, *Proc. of the Human Language Technology Conf. of the North American Chapter of the ACL*, pages 105–112. Association for Computational Linguistics, 2004.

[15] U. Hebisch and H. J. Weinert. *Semirings—Algebraic Theory and Applications in Computer Science.* World Scientific, Singapore, 1998.

[16] W. Kuich. Formal power series over trees. In S. Bozapalidis, editor, *Proc. 3rd Int. Conf. on Developments in Language Theory*, pages 61–101. Aristotle University of Thessaloniki, 1997.

[17] W. Kuich. Tree transducers and formal tree series. *Acta Cybernet.*, 14(1): 135–149, 1999.

[18] W. Kuich. Full abstract families of tree series I. In J. Karhumäki, H. A. Maurer, G. Paun, and G. Rozenberg, editors, *Jewels are Forever*, pages 145–156. Springer, 1999.

[19] M. Mohri. Finite-state transducers in language and speech processing. *Comput. Linguist.*, 23(2):269–311, 1997.

[20] W. C. Rounds. Mappings and grammars on trees. *Math. Systems Theory*, 4(3):257–287, 1970.

[21] J. W. Thatcher. Generalized^2 sequential machine maps. *J. Comput. System Sci.*, 4(4):339–367, 1970.