Input Products for Weighted Extended Top-down Tree Transducers

Andreas Maletti*

Universitat Rovira i Virgili, Departament de Filologies Romàniques
Avinguda de Catalunya 35, 43002 Tarragona, Spain
email: andreas.maletti@urv.cat

Abstract. A weighted tree transformation is a function $\tau\colon T_\Sigma \times T_\Delta \to A$ where T_Σ and T_Δ are the sets of trees over the ranked alphabets Σ and Δ, respectively, and A is the domain of a semiring. The input and output product of τ with tree series $\varphi\colon T_\Sigma \to A$ and $\psi\colon T_\Delta \to A$ are the weighted tree transformations $\varphi \triangleleft \tau$ and $\tau \triangleright \psi$, respectively, which are defined by $(\varphi \triangleleft \tau)(t,u) = \varphi(t) \cdot \tau(t,u)$ and $(\tau \triangleright \psi)(t,u) = \tau(t,u) \cdot \psi(u)$ for every $t \in T_\Sigma$ and $u \in T_\Delta$. In this contribution, input and output products of weighted tree transformations computed by weighted extended top-down tree transducers (wxtt) with recognizable tree series are considered. The classical approach is presented and used to solve the simple cases. It is shown that input products can be computed in three successively more difficult scenarios: nondeleting wxtt, wxtt over idempotent semirings, and weighted top-down tree transducers over rings.

1 Introduction

Top-down tree series transducers [1–3] are a weighted version of top-down tree transducers [4–8]. Here we consider weighted extended top-down tree transducers (wxtt) [9–11], which are a generalization of top-down tree series transducers to allow several symbols in the left-hand side of rules. The framework Tiburon [12] implements wxtt over various weight semirings like the Boolean semiring $(\{\bot, \top\}, \vee, \wedge, \bot, \top)$, the arctic semiring $(\mathbb{N} \cup \{-\infty\}, \max, +, -\infty, 0)$, and the probability semiring $(\mathbb{R}_{\geq 0}, +, \cdot, 0, 1)$, where elements outside the interval $[0,1]$ are included because a semiring is closed under addition.

In this contribution, we will consider two standard operations for wxtt: input and output product. Given a wxtt M and a (suitable) recognizable tree series φ [13–16, 3], their input product[1] should be a wxtt that computes the weight $(\varphi, t) \cdot \tau_M(t,u)$ for every input tree t and output tree u, where τ_M is the weighted tree transformation computed by M. In other words, the obtained wxtt shall scale the weight $\tau_M(t,u)$ assigned by M with the weight (φ, t) assigned by φ to the input tree. The output product is defined analogously. For

* Financially supported by *Ministerio de Educación y Ciencia* grants JDCI-2007-760 and MTM-2007-63422.

[1] Formally, the input product is a weighted tree transformation that might or might not be computable by another wxtt. In applications the former case is very desirable.

completeness' sake, we note that those products are partial; i.e., there exist wxtt and recognizable tree series whose product cannot be computed by a wxtt.

Input and output product have several applications. First, they offer the possibility to integrate a stand-alone parser into a wxtt M. We can encode the parser (e.g., the COLLINS parser [17]) as a recognizable tree series and then perform the input product. The obtained wxtt will multiply the parse weight of the input tree to the weight of each tree pair in M. Another common usage is restriction, in which we want to limit the input (or output) trees to be of a given (recognizable) shape. For every recognizable tree language L [18,19] we can obtain a recognizable tree series 1_L that assigns the weight 1 (neutral element of the multiplication) to each tree of L and weight 0 (neutral element of the addition) to each remaining tree. The input product with 1_L then restricts the weighted tree transformation to input trees of L. More precisely, the weight of any tree pair (t, u) will be 0 if $t \notin L$, whereas the weight of the tree pair remains $\tau_M(t, u)$ if $t \in L$. This particular use of the input product is also known as (generalized) BAR-HILLEL construction, which originally restricts (or intersects) a context-free grammar with a regular language [20]. If L is a singleton, then the BAR-HILLEL construction essentially yields a representation of the parses of the element of L. Consequently, the BAR-HILLEL construction can be understood as a parser. This use is explained in detail in [21]. Finally, the input product is equivalent to recognizable look-ahead [8, 11], so that devices with such look-ahead can be simulated by an input product. This can be used to prove that certain devices with recognizable look-ahead are as powerful as without. For example, Theorem 2 of Sect. 4 easily yields a generalization to the weighted case of [11, Theorem 4.4], which shows that nondeleting wxtt have recognizable look-ahead in the unweighted case (i.e., over the BOOLEAN semiring).

The output product can easily be obtained with existing composition constructions [22, 7, 2, 23]. The same applies to the input product if the wxtt is linear and nondeleting [2, 11]. The main results of this paper concern nonlinear wxtt and we obtain input product constructions for:

- nondeleting wxtt over commutative semirings,
- some-copy nondeleting wxtt over idempotent commutative semirings, and
- some-copy nondeleting wxtt over commutative rings.

The first construction is rather standard. Some-copy nondeleting wxtt are defined such they fully explore at least one copy of each input subtree. In contrast, in a nondeleting wxtt every copy of an input subtree is fully explored. Whenever the wxtt copies a subtree, the second construction nondeterministically guesses a copy that fully explores the subtree. The idempotence of the semiring guarantees that the correct weight is obtained even if several copies are fully explored (i.e., several guesses are successful). The final construction for rings (note that no nontrivial idempotent commutative semiring is a commutative ring) is more involved because a scheme needs to be developed such that several successful explorations cancel each other out in a systematic way. The main problem is that it cannot be enforced with the state behavior alone that only one guess is successful.

2 Preliminaries

The set $\mathbb{N}$ is the set of nonnegative integers. We let $[k] = \{i \mid 1 \leq i \leq k\}$ for every $k \in \mathbb{N}$. Note that $[0] = \emptyset$. The set of all strings over a set Q is denoted by Q^*, of which the empty string is ε. The length $|w|$ of a string $w \in Q^*$ is the number of occurrences of symbols. The ith symbol in w is denoted by w_i. We use a fixed set $X = \{x_i \mid i \in \mathbb{N}\}$ of *formal variables* and its finite subsets $X_k = \{x_i \mid i \in [k]\}$ for every $k \in \mathbb{N}$. Consequently, $X_0 = \emptyset$.

A *ranked alphabet* (Σ, rk) is a finite set of symbols Σ together with a rank mapping $\mathrm{rk}\colon \Sigma \to \mathbb{N}$, which associates a rank to each symbol. We often just write Σ instead of (Σ, rk) and write Σ_k for the set of all symbols in Σ that have rank k. The set of *Σ-trees* indexed by a set V, which is denoted by $T_\Sigma(V)$, is the smallest set such that (i) $V \subseteq T_\Sigma(V)$ and (ii) $\sigma(t_1, \ldots, t_k) \in T_\Sigma(V)$ for every $\sigma \in \Sigma_k$ and $t_1, \ldots, t_k \in T_\Sigma(V)$. We write α for $\alpha()$ with $\alpha \in \Sigma_0$. In addition, we write T_Σ for $T_\Sigma(\emptyset)$. Let $t \in T_\Sigma(V)$ be a tree. The set $\mathrm{pos}(t)$ of *positions* (or *nodes*) in t is inductively defined by $\mathrm{pos}(v) = \{\varepsilon\}$ for every $v \in V$ and

$$\mathrm{pos}(\sigma(t_1, \ldots, t_k)) = \{\varepsilon\} \cup \{iw \mid i \in [k], w \in \mathrm{pos}(t_i)\}$$

for every $\sigma \in \Sigma_k$ and $t_1, \ldots, t_k \in T_\Sigma(V)$. We write $t(w)$ for the label of t at position w. Moreover, $\mathrm{pos}_\ell(t) = \{w \in \mathrm{pos}(t) \mid t(w) = \ell\}$ for every $\ell \in \Sigma \cup V$. The tree t is *linear* (respectively, *nondeleting*) *in* V if $|\mathrm{pos}_v(t)| \leq 1$ (respectively, $|\mathrm{pos}_v(t)| \geq 1$) for every $v \in V$. We write $C_\Sigma(X_k)$ for the subset of all trees of $T_\Sigma(X)$ that are linear and nondeleting in X_k.

Clearly, the positions $\mathrm{pos}(t) \subseteq \mathbb{N}^*$ are lexicographically ordered. Thus, the occurrences $\mathrm{pos}_v(t)$ of a variable $v \in V$ in t are also ordered. Given n pairwise distinct variables $v_1, \ldots, v_n \in V$ and $t_{v_i 1}, \ldots, t_{v_i m_i} \in T_\Sigma(V)$ for every $i \in [n]$ with $m_i = |\mathrm{pos}_{v_i}(t)|$, the substitution

$$t[v_1 \leftarrow (t_{v_1 1}, \ldots, t_{v_1 m_1}), \ldots, v_n \leftarrow (t_{v_n 1}, \ldots, t_{v_n m_n})]$$

or just $t[v_i \leftarrow (t_{v_i 1}, \ldots, t_{v_i m_i}) \mid i \in [n]]$ denotes the tree obtained by replacing, for every $i \in [n]$, the m_i occurrences of v_i in t by $(t_{v_i 1}, \ldots, t_{v_i m_i})$ in order; i.e., the leftmost occurrence of v_i is replaced by $t_{v_i 1}$ and the rightmost occurrence is replaced by $t_{v_i m_i}$. If $t \in C_\Sigma(X_n)$ and $X_n = \{v_1, \ldots, v_n\}$, then we just write $t[t_{x_1 1}, \ldots, t_{x_n 1}]$ instead of the cumbersome $t[x_1 \leftarrow (t_{x_1 1}), \ldots, x_n \leftarrow (t_{x_n 1})]$. Moreover, for every $t \in T_\Sigma$, let $\mathrm{match}(t)$ be the finite set

$$\mathrm{match}(t) = \{(l, t_1, \ldots, t_k) \mid l \in C_\Sigma(X_k), t_1, \ldots, t_k \in T_\Sigma, l[t_1, \ldots, t_k] = t\} \ .$$

A *(commutative) semiring* $\mathcal{A} = (A, +, \cdot, 0, 1)$ is an algebraic structure such that $(A, +, 0)$ and $(A, \cdot, 1)$ are commutative monoids, $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$ for every $a, b, c \in A$, and $a \cdot 0 = 0 = 0 \cdot a$ for every $a \in A$. It is *idempotent* if $1 + 1 = 1$. In an idempotent semiring $\mathcal{A}$ the *natural order* $\leq \subseteq A \times A$, which is given by $a \leq b$ if and only if $a + b = b$ for every $a, b \in A$, is a partial order for which the operations $+$ and $\cdot$ are monotone. Finally, the semiring $\mathcal{A}$ is a *ring* if there exists an element $(-1) \in A$ such that $1 + (-1) = 0$. For the rest of the paper, let $\mathcal{A} = (A, +, \cdot, 0, 1)$ be a commutative semiring.

Let Σ be a ranked alphabet. Every mapping $\varphi\colon T_\Sigma \to A$ is a *tree series*, which is also expressed by $\varphi \in A\langle\langle T_\Sigma \rangle\rangle$. For every $t \in T_\Sigma$ the value $\varphi(t)$ of t in φ is usually written as (φ, t). Next, we recall recognizable tree series [13, 24, 3]. A *weighted tree automaton* (wta) [24] is a system $N = (P, \Sigma, F, \mu)$ where (i) P is a finite set of *states*, (ii) Σ is a ranked alphabet of *input symbols*, (iii) $F\colon P \to A$ is a *final weight vector*, and (iv) $\mu = (\mu_k)_{k\in\mathbb{N}}$ is a family of *weighted transitions* with $\mu_k\colon P^k \times \Sigma_k \times P \to A$ for every $k \in \mathbb{N}$. We generalize our wta to work on trees of $T_\Sigma(X_n)$ with $n \in \mathbb{N}$. Note that $T_\Sigma = C_\Sigma(X_0) \subseteq T_\Sigma(X_n)$. For all $p_1, \ldots, p_n \in P$ we extend μ to a mapping $h_\mu^{p_1\cdots p_n} : T_\Sigma(X_n) \to A^P$ by

$$h_\mu^{p_1\cdots p_n}(x_i)_p = \begin{cases} 1 & \text{if } p = p_i \\ 0 & \text{otherwise} \end{cases}$$

$$h_\mu^{p_1\cdots p_n}(\sigma(t_1, \ldots, t_k))_p = \sum_{q_1,\ldots,q_k \in P} \mu_k(q_1 \cdots q_k, \sigma, p) \cdot \prod_{i=1}^{k} h_\mu^{p_1\cdots p_n}(t_i)_{q_i}$$

for every $i \in [n]$, $\sigma \in \Sigma_k$, $t_1, \ldots, t_k \in T_\Sigma(X_n)$, and $p \in P$. The wta N *recognizes* the tree series $\|N\| \in A\langle\langle T_\Sigma \rangle\rangle$, which is given by $(\|N\|, t) = \sum_{p\in P} F(p) \cdot h_\mu(t)_p$ for every $t \in T_\Sigma$. Recall that $\mathcal{A}$ is commutative, so $F(p) \cdot h_\mu(t)_p = h_\mu(t)_p \cdot F(p)$. A tree series that is recognized by some wta is *recognizable*.

Next, we define our main tree transducer model: the weighted extended top-down tree transducer [9–11]. Our definition will be slightly non-standard, but the particular syntax will prove useful for our constructions. We assure the reader that our semantics will be equivalent to the existing definitions [9–11, 25]. A *weighted extended top-down tree transducer* (wxtt) is a system $(Q, \Sigma, \Delta, I, R)$ where (i) Q is a finite set of *states*, (ii) Σ and Δ are ranked alphabets of *input* and *output symbols*, respectively, (iii) $I\colon Q \to A$ is an *initial weight vector*, and (iv) R is a finite set of *rules* of the form $(q, l) \xrightarrow{a} (w, r)$ with $q \in Q$, $l \in C_\Sigma(X_k)$, $a \in A$, $w \in (Q^*)^k$, and $r \in T_\Delta(X_k)$ such that $|w_i| = |\mathrm{pos}_{x_i}(r)|$ for every $i \in [k]$. Intuitively speaking, a rule $(q, l) \xrightarrow{a} (w, r)$ consists of a *state* q, a *left-hand side* l, a *weight* $a \in A$, a *control word* w, and a *right-hand side* r. The control word consists of k words $w_1, \ldots, w_k$ of states. For each $i \in [k]$, the ith word records the states (in order) that are associated to the occurrences (in lexicographic order) of the variable x_i in r. A more classical rule shape, which we use in graphical representations, assumes that the states have rank 1 and presents the rule $(q, l) \xrightarrow{a} (w, r)$ as $q(l) \xrightarrow{a} r[x_i \leftarrow ((w_i)_1(x_i), \ldots, (w_i)_{|w_i|}(x_i)) \mid 1 \le i \le |w|]$. An example is displayed in Fig. 1 (left). The rule $(q, l) \xrightarrow{a} (w, r) \in R$ is *linear* (respectively, *nondeleting*) if $|w_i| \le 1$ (respectively, $|w_i| \ge 1$) for every $1 \le i \le |w|$. The wxtt M is *linear* (respectively, *nondeleting*) if every rule $\rho \in R$ is so. It is a *weighted top-down tree transducer* (wtt) if for every $(q, l) \xrightarrow{a} (w, r) \in R$ there exists $\sigma \in \Sigma_k$ such that $l = \sigma(x_1, \ldots, x_k)$. The example rule of Fig. 1 (left) is nondeleting, but not linear. Any wxtt with that rule is not a wtt.

In the following, let $M = (Q, \Sigma, \Delta, I, R)$ be a wxtt. To simplify the development, we assume henceforth that $l \neq x_1$ for every rule $(q, l) \xrightarrow{a} (w, r) \in R$.

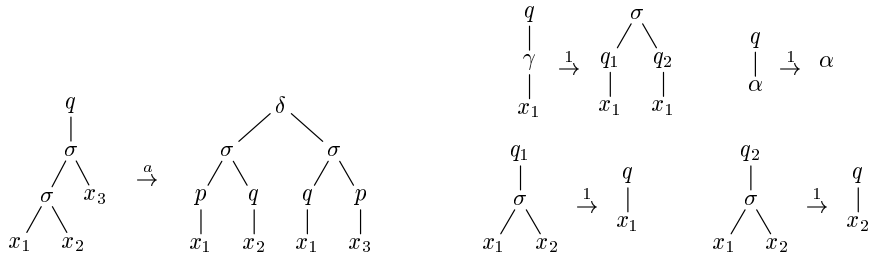

Fig. 1. Left: Graphical representation of the example rule $(q,l) \xrightarrow{a} ((pq,q,p),r)$ where $l = \sigma(\sigma(x_1,x_2),x_3)$ and $r = \delta(\sigma(x_1,x_2),\sigma(x_1,x_3))$. Right: Example of deleting rules.

In other words, we disallow *ε-rules*.[2] We use the symbol q with and without additional subscripts for elements of Q. Consequently, if we write $w = q_1 \cdots q_n$, then implicitly $q_i \in Q$ for every $i \in [n]$. The semantics of the wxtt M (without epsilon rules) is defined as follows: Let $h_R \colon T_\Sigma \times T_\Delta \to A^Q$ be the mapping defined for every $t \in T_\Sigma$, $u \in T_\Delta$, and $q \in Q$ by

$$h_R(t,u)_q = \sum_{\substack{(q,l) \xrightarrow{a} (q_{11}\cdots q_{1n_1},\ldots,q_{k1}\cdots q_{kn_k},r) \in R \\ (l,t_1,\ldots,t_k) \in \mathrm{match}(t), \forall i \in [k], \forall j \in [n_i] \colon u_{ij} \in T_\Delta \\ u = r[x_i \leftarrow (u_{i1},\ldots,u_{in_i}) \mid i \in [k]]}} a \cdot \prod_{\substack{i \in [k] \\ j \in [n_i]}} h_R(t_i,u_{ij})_{q_{ij}} \quad .$$

The wxtt M *computes* the tree transformation $\tau_M \colon T_\Sigma \times T_\Delta \to A$, which is given by $\tau_M(t,u) = \sum_{q \in Q} I(q) \cdot h_R(t,u)_q$ for every $t \in T_\Sigma$ and $u \in T_\Delta$.

3 Input and output product

In this section, we formally define input and output product and then discuss the standard approach to solve the associated algorithmic problems. Let $\tau \colon T_\Sigma \times T_\Delta \to A$, $\varphi \in A\langle\langle T_\Sigma \rangle\rangle$, and $\psi \in A\langle\langle T_\Delta \rangle\rangle$. The *input product* $\varphi \triangleleft \tau$ of τ with φ and the *output product* $\tau \triangleright \psi$ of τ with ψ are $(\varphi \triangleleft \tau)(t,u) = (\varphi,t) \cdot \tau(t,u)$ and $(\tau \triangleright \psi)(t,u) = \tau(t,u) \cdot (\psi,u)$ for every $t \in T_\Sigma$ and $u \in T_\Delta$.

Classical solutions to the problems of input and output products are specialized Bar-Hillel constructions [20, 21] or compositions [22, 7]. The composition approach first embeds a tree series $\varphi \in A\langle\langle T_\Sigma \rangle\rangle$ into the identity mapping $\mathrm{id}_\varphi \colon T_\Sigma \times T_\Sigma \to A$, which is defined by $\mathrm{id}_\varphi(t,t') = (\varphi,t)$ if $t = t'$ and 0 otherwise for every $t,t' \in T_\Sigma$. Given two tree transformations $\tau_1 \colon T_\Sigma \times T_\Delta \to A$ and $\tau_2 \colon T_\Delta \times T_\Gamma \to A$, the *composition* $\tau_1 \mathbin{;} \tau_2 \colon T_\Sigma \times T_\Gamma \to A$ of τ_1 and τ_2 is given by $(\tau_1 \mathbin{;} \tau_2)(t,v) = \sum_{u \in T_\Delta} \tau_1(t,u) \cdot \tau_2(u,v)$ for every $t \in T_\Sigma$ and $v \in T_\Gamma$. Note that, in general, the sum in the definition of $\tau_1 \mathbin{;} \tau_2$ may be infinite, but in our compositions $\tau_1 \mathbin{;} \tau_2$ one of the tree transformations τ_1 or τ_2 will always be an

[2] Aside from well-definedness issues, there is no additional complexity with ε-rules. All our constructions can easily be adapted to work for general wxtt with well-defined semantics, but we would like to avoid an in-depth discussion of well-definedness.

identity mapping id_φ. This yields that the sum in the definition of the composition $\tau_1 ; \tau_2$ essentially degenerates into a single summand. Thus, we can express input and output product as $\varphi \triangleleft \tau = \mathrm{id}_\varphi ; \tau$ and $\tau \triangleright \psi = \tau ; \mathrm{id}_\psi$, respectively.

In this contribution, we consider input and output products of tree transformations that are computed by wxtt with recognizable tree series. The identity mappings id_φ and id_ψ can be computed by linear and nondeleting wtt for all recognizable tree series $\varphi \in A\langle\langle T_\Sigma \rangle\rangle$ and $\psi \in A\langle\langle T_\Delta \rangle\rangle$. Thus, the main question with regard to the composition approach is whether the given compositions for input and output product can be computed by another wxtt.

First, let us discuss the question for an unweighted (i.e., BOOLEAN weighted[3]) wtt M [5, 4]. By the composition results of [22, 7], the composition approach works for: (i) the output product and (ii) the input product if M is linear and nondeleting. Further results can be obtained for special recognizable tree series such as deterministic top-down recognizable tree series [26, 18], but we will focus on general recognizable tree series in this contribution. The two mentioned results generalize to wtt over commutative semirings [2, 23] and can easily be extended to wxtt as well. If we were to discuss input and output product also for weighted extended bottom-up tree transducers [9, 2, 27], then the roles would essentially exchange. The input product would be easy with the help of the composition approach and output products could be achieved following our approaches for input products of wxtt.

4 Nondeleting transducers

From now on, let $M = (Q, \Sigma, \Delta, I, R)$ be a wxtt and $N = (P, \Sigma, F, \mu)$ be a wta. The aim of this and the following sections is to present constructions of M' such that $\tau_{M'} = \|N\| \triangleleft \tau_M$. This problem is simple if M is nondeleting. Each input subtree is visited at least once by M due to nondeletion, and we can arbitrarily select one call[4] to perform the input product. We select the first call here. As a notational convenience, we sometimes use angled brackets '$\langle$' and '$\rangle$' instead of parentheses '(' and ')'.

Definition 1. *The input product $N \triangleleft_{\mathrm{n}} M$ is the wxtt $(Q', \Sigma, \Delta, I', R \cup R')$ where*

- $Q' = Q \cup (Q \times P)$,
- $I'(q) = 0$ *and* $I'(\langle q, p\rangle) = I(q) \cdot F(p)$ *for every* $q \in Q$ *and* $p \in P$, *and*
- $(\langle q, p\rangle, l) \xrightarrow{a \cdot a'} (\langle q_{11}, p_1\rangle q_{12} \cdots q_{1n_1}, \ldots, \langle q_{k1}, p_k\rangle q_{k2} \cdots q_{kn_k}, r) \in R'$ *for every nondeleting rule* $(q, l) \xrightarrow{a} (q_{11} \cdots q_{1n_1}, \ldots, q_{k1} \cdots q_{kn_k}, r) \in R$ *and all states* $p, p_1, \ldots, p_k \in P$ *where* $a' = h_\mu^{p_1 \cdots p_k}(l)_p$.

Mind that in a nondeleting wxtt all rules are nondeleting, which allows us to arbitrarily select any call because each call will fully explore its subtree.

[3] The BOOLEAN semiring is $\mathbb{B} = (\{\bot, \top\}, \vee, \wedge, \bot, \top)$.

[4] In a rule $(q, l) \xrightarrow{a} (w, r) \in R$ any $(w_i)_j \in Q$ with $1 \le i \le |w|$ and $1 \le j \le |w_i|$ is also called '*call*' to the subtree represented by x_i. If $|w_i| \ge 2$, then there are several calls to the same subtree.

Theorem 2. *If M is nondeleting, then $\tau_{(N \lhd_{\mathrm{n}} M)} = \|N\| \lhd \tau_M$.*

Our construction can easily be generalized to settings, where based on the rule shape, the call that fully explores its subtree can be infered. The next sections will deal with cases, in which such a prediction is not possible.

5 Idempotent semirings

In this and the next section, we develop successively more complicated input-product constructions for deleting wxtt. Clearly, if M deletes a particular input subtree in all copies, then we cannot obtain a general input-product construction because the deleted subtree might be essential for the wta N. Consequently, we restrict our attention to wxtt that visit each input subtree at least once. In principle, algorithms for the input product can be imagined in this setting, but such an algorithm would require extensive book-keeping. This is due to the fact that we would need to record which subtree is visited by which call, which seems unfeasible in a general-purpose construction. Nevertheless such a construction should eventually be established for the particular wxtt that are the result of binarizing nondeleting wxtt.

Let us consider the rules of Fig. 1 (right). Each subtree in the tree $\gamma(\sigma(\alpha, \alpha))$ is visited at least once starting with state q. The first rule creates two copies of the input subtree $\sigma(\alpha, \alpha)$, of which the first call explores the node labeled 'σ' and the left subtree α (and deletes the right subtree α). The second call explores the 'σ'-node and the right subtree α. Consequently, there is no single call that explores the full subtree $\sigma(\alpha, \alpha)$, which yields that the product construction needs to keep track which parts are explored by which call. To avoid this additional book-keeping, we introduce *some-copy nondeletion*, which demands that for each input subtree there is at least one call that fully explores the whole subtree.

Definition 3. *A state $q \in Q$ is* t-nondeleting *with $t \in T_\Sigma$ if for every rule $(q, l) \xrightarrow{a} (w, r) \in R$, $(l, t_1, \ldots, t_k) \in \mathrm{match}(t)$, and $1 \leq i \leq |w|$ there exists $1 \leq j \leq |w_i|$ such that $(w_i)_j$ is t_i-non-deleting. The wxtt M is* some-copy nondeleting *if q is t-nondeleting for every $t \in T_\Sigma$ and $q \in Q$ such that $I(q) \neq 0$.*

Note that any rule fulfilling the premise in Definition 3 must be nondeleting, however, not all rules R need to be nondeleting for M to be some-copy nondeleting. Classical nondeletion might be called "all-copies nondeletion" in analogy. We further note that some-copy nondeletion is a semantic property. It is decidable (because for every state $q \in Q$ the set of all trees $t \in T_\Sigma$ such that q is t-nondeleting is a recognizable tree language [18, 19]), but it might not be a practical property. However, it encompasses several interesting forms of "nondeletion" (such as the ones mentioned at the end of the previous section), and thus allows us to present one construction (and its proof of correctness) for several "nondeletion" properties. We already argued that the state q with the rules of Fig. 1 (right) is not $\gamma(\sigma(\alpha, \alpha))$-nondeleting because neither q_1 nor q_2 are $\sigma(\alpha, \alpha)$-nondeleting. However, the wxtt with the rules of Fig. 2 is some-copy

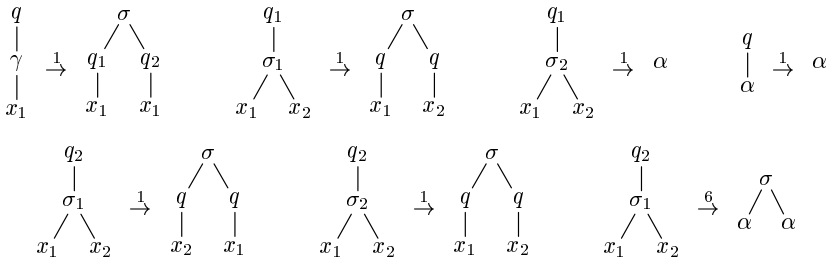

Fig. 2. Rules of a some-copy nondeleting wxtt.

nondeleting if q is the only initial state. Note that the call that fully explores a subtree here depends on the input tree. For example, the first call created in the first rule fully explores the subtree $\sigma_1(\alpha, \alpha)$ in the tree $\gamma(\sigma_1(\alpha, \alpha))$, whereas the second call fully explores $\sigma_2(\alpha, \alpha)$ in the tree $\gamma(\sigma_2(\alpha, \alpha))$. The example also demonstrates that it is not sufficient that one (non-deterministic) choice fully explores a subtree. Although the state q_2 can fully explore the subtree $\sigma_1(\alpha, \alpha)$, it is not $\sigma_1(\alpha, \alpha)$-nondeleting because of the last rule of Fig. 2.

Since we cannot infer the correct call that fully explores a subtree from the information present in a rule, we have to guess it. If M is some-copy nondeleting, then we know that at least one such guess must be correct. However, it might happen that several calls eventually fully explore the same subtree. The author is unaware of any method to exclude this behavior with the help of the states alone. Consequently, we only prove that our construction is correct in idempotent semirings $\mathcal{A}$. The idempotency yields that several equivalent (or even partial) explorations can be performed without effect on the weight.

Definition 4. *The input product $N \triangleleft_{\mathrm{i}} M$ is the wxtt $(Q', \Sigma, \Delta, I', R \cup R')$ where*

- $Q' = Q \cup (Q \times P)$,
- $I'(q) = 0$ *and* $I'(\langle q, p\rangle) = I(q) \cdot F(p)$ *for every* $q \in Q$ *and* $p \in P$, *and*
- $(\langle q, p\rangle, l) \xrightarrow{a \cdot a'} (q_{11} \cdots \langle q_{1j_1}, p_1\rangle \cdots q_{1n_1}, \ldots, q_{k1} \cdots \langle q_{kj_k}, p_k\rangle \cdots q_{kn_k}, r) \in R'$ *for every nondeleting rule* $(q, l) \xrightarrow{a} (q_{11} \cdots q_{1n_1}, \ldots, q_{k1} \cdots q_{kn_k}, r) \in R$, *all states* $p, p_1, \ldots, p_k \in P$, *and* $j_1 \in [n_1], \ldots, j_k \in [n_k]$ *where* $a' = h_\mu^{p_1 \cdots p_k}(l)_p$.

Note that only one state of each part in the control word is replaced. Let us illustrate the construction on the rules of Fig. 2.

Example 5. Suppose that $\mathcal{A} = (\mathbb{N} \cup \{-\infty\}, \max, +, -\infty, 0)$ is the arctic semiring, which is idempotent. Moreover, suppose that M is given by the rules of Fig. 2 and that N is the trivial wta with $P = \{p\}$ such that $(\|N\|, t) = 0$ for every $t \in T_\Sigma$. Clearly, $\|N\| \triangleleft \tau_M = \tau_M$, so we would not need a construction at all, but let us consider the input tree $t = \sigma_1(\alpha, \alpha)$ and the output tree $u = \sigma(\alpha, \alpha)$. A simple calculation shows that $h_R(t, u)_{q_1} = 3$ and $h_R(t, u)_{q_2} = \max(3, 6)$. Now, let $N \triangleleft_{\mathrm{i}} M = (Q', \Sigma, \Delta, I', R')$. Then $h_{R'}(t, u)_{\langle q_1, p\rangle} = 3$ and $h_{R'}(t, u)_{\langle q_2, p\rangle} = 3$, which shows that the derivation with weight 6 is not possible in R' with a paired

state because the last rule (which resulted in the weight 6) of Fig. 2 is deleting and thus no new rules (with a paired state) are created for it.

Theorem 6. *If M is some-copy nondeleting and $\mathcal{A}$ an idempotent semiring, then $\tau_{(N \lhd_{\mathrm{i}} M)} = \|N\| \lhd \tau_M$.*

6 Rings

In this final section, we will discuss the input product for wxtt over rings. From here on, we assume that $\mathcal{A}$ is a ring. To keep the presentation simple, we make the additional restriction that M is a wtt. This will make the discussion of the t-nondeletion property easier. We note that the restriction is done only for convenience. More elaborate versions of our construction could easily overcome the restriction. In the previous section we could easily allow spurious derivations because idempotence would "hide" the wrong weights corresponding to those spurious derivations. This is no longer possible in rings because no non-trivial (i.e., $0 \neq 1$) ring is idempotent. Consequently, our final construction needs to take care of those spurious derivations. To this end, we use a simple *elimination pattern*. Let $Q \times C$ be such that $(Q \times C) \cap Q = \emptyset$ for some set C. For every set Q' such that $Q \subseteq Q'$ and $w \in (Q')^*$, we let $|w|_Q = |\{i \mid w_i \in Q\}|$ be the number of Q-symbols in w. Our elimination pattern $f\colon (Q')^* \to A$ is given by $f(w) = 1$ if $|w| - |w|_Q$ is odd, and $f(w) = -1$ otherwise. Intuitively speaking, an elimination mapping is a strategy such that the derivations properly cancel to just one derivation irrespective of the set of calls that fully explore a certain subtree. This set of calls is non-empty because we will again assume some-copy nondeletion. Let us illustrate this for $Q' = Q \cup (Q \times C)$. For every $w \in (Q')^*$ let $\mathrm{base}(w)$ be the unique word such that $\mathrm{base}(\varepsilon) = \varepsilon$ and

$$\mathrm{base}(q'w') = \begin{cases} q'\,\mathrm{base}(w') & \text{if } q' \in Q \\ q\,\mathrm{base}(w') & \text{if } q' = \langle q, c\rangle \end{cases}$$

for every $q' \in Q'$ and $w' \in (Q')^*$. Let $q \in Q$ be an arbitrary state and $c \in C$. Given a nonempty set $J \subseteq [n]$ with $m = |J|$, which represents the calls that fully explore a given subtree, $w \in Q^n$, and $c_1, \ldots, c_n \in \mathcal{C}$ we have

$$\sum_{\substack{w' \in (Q')^* \setminus Q^* \\ \mathrm{base}(w') = w \\ \forall i \in [n]\colon w'_i = \langle w_i, c_i\rangle \text{ if } w'_i \notin Q \\ \forall i \in [n] \setminus J\colon w'_i \in Q}} f(w') = \sum_{w' \in \{q, \langle q, c\rangle\}^m \setminus \{q\}^m} f(w') = 1 \ ,$$

where the last step is a trivial consequence of Pascal's triangle. Consequently, for any choice of calls that fully explore a certain subtree, our elimination strategy ensures that all but one cancel.

Before we proceed, let us consider the negation of t-nondeletion for wtt. Let $t = \sigma(t_1, \ldots, t_k)$ for some $\sigma \in \Sigma_k$ and $t_1, \ldots, t_k \in T_\Sigma$. A state $q \in Q$ is *t-deleting*

if there exist a rule $(q, \sigma(x_1, \ldots, x_k)) \xrightarrow{a} (w, r) \in R$ and $i \in [k]$ such that $(w_i)_j$ is t_i-deleting for every $1 \leq j \leq |w_i|$. Consequently, a state q is $\sigma(t_1, \ldots, t_k)$-deleting for one of two reasons: (i) It has a deleting rule $(q, \sigma(x_1, \ldots, x_k)) \xrightarrow{a} (w, r) \in R$, or (ii) it has a nondeleting rule $(q, \sigma(x_1, \ldots, x_k)) \xrightarrow{a} (w, r) \in R$ such that all states called on a subtree t_i are t_i-deleting. In the first case, the subtrees $t_1, \ldots, t_k$ are obviously irrelevant. This characterization yields a simple check of t-deletion inside a wtt, which will be coded in the next construction. We need some additional notation. Let $\mathcal{C} = \mathcal{P}(\mathcal{P}(Q))$ where $\mathcal{P}(S)$ denotes the powerset of S. Moreover, for every $\sigma \in \Sigma_k$, $i \in [k]$, $S \subseteq Q$, $C \in \mathcal{C}$, and $q \in Q$, let

$$\begin{aligned}
\mathrm{next}_i(\sigma, q) &= \{\{(w_i)_1, \ldots, (w_i)_{|w_i|}\} \mid (q, \sigma(x_1, \ldots, x_k)) \xrightarrow{a} (w, r) \in R\} \\
\mathrm{next}_i(\sigma, S) &= \{\bigcup\nolimits_{q \in S} w_q \mid \forall q \in S \colon w_q \in \mathrm{next}_i(\sigma, q)\} \\
\mathrm{next}_i(q, \sigma, C) &= \bigcup_{S \in C} \mathrm{next}_i(\sigma, S) \cup \mathrm{next}_i(\sigma, q) \ .
\end{aligned}$$

Definition 7. *The input product $N \triangleleft_{\mathrm{r}} M$ is the wtt $(Q', \Sigma, \Delta, I', R'')$ where*

- *$Q' = Q \cup (Q \times \mathcal{C}) \cup (Q \times \mathcal{C} \times P)$ and $R'' = R \cup \overline{R} \cup R'$,*
- *for every $q \in Q$, $C \in \mathcal{C}$, and $p \in P$, let $I'(q) = I'(\langle q, C\rangle) = 0$ and*

$$I'(\langle q, C, p\rangle) = \begin{cases} I(q) \cdot F(p) & \text{if } C = \{\{q\}\} \\ 0 & \text{otherwise,} \end{cases}$$

- *for every $q \in Q$, $C \in \mathcal{C}$ such that $\emptyset \notin C$, $\sigma \in \Sigma_k$, and every nondeleting rule $(q, \sigma(x_1, \ldots, x_k)) \xrightarrow{a} (w, r) \in R$, let*

$$(\langle q, C\rangle, \sigma(x_1, \ldots, x_k)) \xrightarrow{a \cdot f(w'_1) \cdot \ldots \cdot f(w'_k)} (w'_1 \cdots w'_k, r) \in \overline{R}$$

  for every $w'_1, \ldots, w'_k \in (Q \cup (Q \times C))^* \setminus Q^*$ such that for every $i \in [k]$

$$w'_i = \begin{cases} w_i & \text{if } w'_i \in Q \\ (w_i, \mathrm{next}_i(q, \sigma, C)) & \text{otherwise,} \end{cases}$$

- *for every rule $(\langle q, C\rangle, \sigma(x_1, \ldots, x_k)) \xrightarrow{a} (w, r) \in \overline{R}$ and $p, p_1, \ldots, p_k \in P$ let $a' = h_\mu^{p_1 \cdots p_k}(l)_p$ and*

$$(\langle q, C, p\rangle, \sigma(x_1, \ldots, x_k)) \xrightarrow{a \cdot a'} (w'_1 \cdots w'_k, r) \in R' \ ,$$

  where, for every $i \in [k]$, the control word w'_i is obtained from w_i by replacing the first paired state $\langle q', C'\rangle$, by $\langle q', C', p_i\rangle$.

In other words, the paired states guess the calls that fully explore their subtrees. Note that if there exists any deleting rule $(q, l) \xrightarrow{a} (w, r) \in R$ with $l = \sigma(x_1, \ldots, x_k)$ and $w_i = \varepsilon$ for some $i \in [k]$, then for every $(\langle q, C\rangle, l) \xrightarrow{e} (w', r')$ in $\overline{R}$ there exists $1 \leq j \leq |w'_i|$ such that $(w'_i)_j = \langle q', C'\rangle$ or $(w'_i)_j = \langle q', C', p\rangle$ with

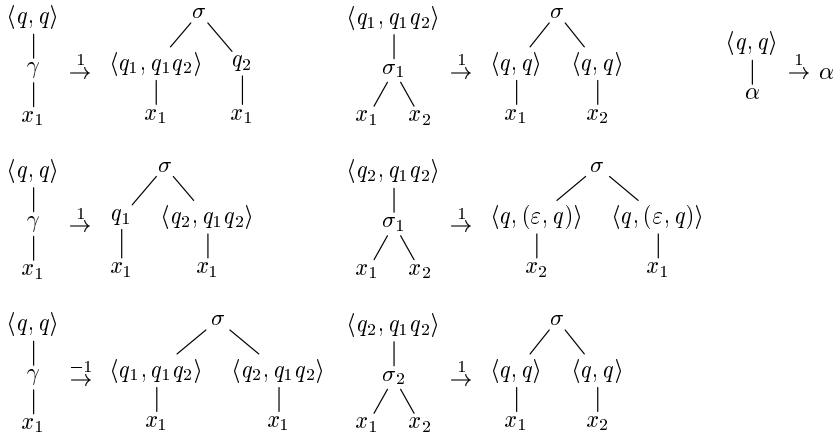

Fig. 3. Some constructed rules of $\overline{R}$ where R is the set of rules of Fig. 2. We use a string notation for the elements of $\mathcal{C}$ here.

$\emptyset \in C'$. This is due to the fact that $\emptyset \in \mathrm{next}_i(q, \sigma, C)$ because $\emptyset \in \mathrm{next}_i(\sigma, q)$. A similar statement holds for R'. In essence, this blocks the computations, in which the guess was wrong because a deletion would be possible. It is not quite obvious that (aside from one) the various correct guesses cancel each other out.Let us illustrate the construction on the example rules of Fig. 2.

Example 8. Suppose that $\mathcal{A} = (\mathbb{Z}, +, \cdot, 0, 1)$ is the ring of integers, and let M be given by the rules of Fig. 2. To keep the example short, Fig. 3 only presents some relevant rules of $\overline{R}$ that are constructed in the third item of Definition 7. For $t = \gamma(\sigma_1(\alpha, \alpha))$ and $u = \sigma(\sigma(\alpha, \alpha), \sigma(\alpha, \alpha))$ we can compute that

$$h_R(t, u)_q = 1 + 6 = h_{R \cup \overline{R}}(t, u)_{\langle q, \{\{q\}\}\rangle} \quad .$$

Now, let S be the set R without the rule with weight 6, and let $\overline{S}$ be the corresponding rule set constructed in the third item of Definition 7. Then

$$h_S(t, u)_q = 1 = 1 + 1 - 1 = h_{S \cup \overline{S}}(t, u)_{\langle q, \{\{q\}\}\rangle} \quad .$$

Theorem 9. *If M is a some-copy nondeleting wtt and $\mathcal{A}$ is a ring, then*

$$\tau_{(N \lhd_r M)} = \|N\| \lhd \tau_M \quad .$$

References

1. Kuich, W.: Tree transducers and formal tree series. Acta Cybernet. **14**(1) (1999) 135–149
2. Engelfriet, J., Fülöp, Z., Vogler, H.: Bottom-up and top-down tree series transformations. J. Autom. Lang. Combin. **7**(1) (2002) 11–70

3. Fülöp, Z., Vogler, H.: Weighted tree automata and tree transducers. In: Handbook of Weighted Automata. Springer (2009) 313–403
4. Rounds, W.C.: Mappings and grammars on trees. Math. Systems Theory **4**(3) (1970) 257–287
5. Thatcher, J.W.: Generalized^2 sequential machine maps. J. Comput. System Sci. **4**(4) (1970) 339–367
6. Thatcher, J.W.: Tree automata: An informal survey. In: Currents in the Theory of Computing. Prentice Hall (1973) 143–172
7. Engelfriet, J.: Bottom-up and top-down tree transformations: A comparison. Math. Systems Theory **9**(3) (1975) 198–231
8. Engelfriet, J.: Top-down tree transducers with regular look-ahead. Math. Systems Theory **10**(1) (1977) 289–303
9. Arnold, A., Dauchet, M.: Morphismes et bimorphismes d'arbres. Theoret. Comput. Sci. **20**(1) (1982) 33–93
10. Graehl, J., Knight, K., May, J.: Training tree transducers. Computational Linguistics **34**(3) (2008) 391–427
11. Maletti, A., Graehl, J., Hopkins, M., Knight, K.: The power of extended top-down tree transducers. SIAM J. Comput. **39**(2) (2009) 410–430
12. May, J., Knight, K.: Tiburon: A weighted tree automata toolkit. In: CIAA. Volume 4094 of LNCS., Springer (2006) 102–113
13. Berstel, J., Reutenauer, C.: Recognizable formal power series on trees. Theoret. Comput. Sci. **18**(2) (1982) 115–148
14. Bozapalidis, S.: Equational elements in additive algebras. Theory Comput. Systems **32**(1) (1999) 1–33
15. Kuich, W.: Formal power series over trees. In: DLT, Aristotle University of Thessaloniki (1998) 61–101
16. Borchardt, B., Vogler, H.: Determinization of finite state weighted tree automata. J. Autom. Lang. Combin. **8**(3) (2003) 417–463
17. Collins, M.: Head-Driven Statistical Models for Natural Language Parsing. PhD thesis, University of Pennsylvania (1999)
18. Gécseg, F., Steinby, M.: Tree Automata. Akadémiai Kiadó, Budapest (1984)
19. Gécseg, F., Steinby, M.: Tree languages. In: Handbook of Formal Languages. Volume 3. Springer (1997) 1–68
20. Bar-Hillel, Y., Perles, M., , Shamir, E.: On formal properties of simple phrase structure grammars. In: Language and Information: Selected Essays on their Theory and Application. Addison Wesley (1964) 116–150
21. Satta, G.: Translation algorithms by means of language intersection. Manuscript (2010) available at: `www.dei.unipd.it/~satta/publ/paper/inters.pdf`.
22. Baker, B.S.: Composition of top-down and bottom-up tree transductions. Inform. and Control **41**(2) (1979) 186–213
23. Maletti, A.: Compositions of tree series transformations. Theoret. Comput. Sci. **366**(3) (2006) 248–271
24. Borchardt, B.: The Theory of Recognizable Tree Series. PhD thesis, Technische Universität Dresden (2005)
25. Maletti, A.: Why synchronous tree substitution grammars? In: NAACL, Association for Computational Linguistics (2010) to appear.
26. Virágh, J.: Deterministic ascending tree automata I. Acta Cybernet. **5**(1) (1980) 33–42
27. Engelfriet, J., Lilin, E., Maletti, A.: Composition and decomposition of extended multi bottom-up tree transducers. Acta Inform. **46**(8) (2009) 561–590