

Weighted Regular Tree Grammars with Storage

Zoltán Fülöp^{1*}Luisa Herrmann^{2†}Heiko Vogler²¹ Department of Foundations of Computer Science, University of Szeged, Hungary² Faculty of Computer Science, Technische Universität Dresden, Germanyreceived 19th May 2017, revised 11th June 2018, accepted 22nd June 2018.

We introduce weighted regular tree grammars with storage as combination of (a) regular tree grammars with storage and (b) weighted tree automata over multioperator monoids. Each weighted regular tree grammar with storage generates a weighted tree language, which is a mapping from the set of trees to the multioperator monoid. We prove that, for multioperator monoids canonically associated to particular strong bimonoids, the support of the generated weighted tree languages can be generated by (unweighted) regular tree grammars with storage. We characterize the class of all generated weighted tree languages by the composition of three basic concepts. Moreover, we prove results on the elimination of chain rules and of finite storage types, and we characterize weighted regular tree grammars with storage by a new weighted MSO-logic.

Keywords: regular tree grammars, weighted tree automata, multioperator monoids, storage types, weighted MSO-logic

1 Introduction

In automata theory, weighted string automata with storage are a very recent topic of research [HV15, HV16, VDH16]. This model generalizes finite-state string automata in two directions. On the one hand, it is based on the concept of “sequential program + machine”, which Scott [Sco67] proposed in order to harmonize the wealth of upcoming automaton models. Each of them has a finite control (sequential program) and allows to control computations by means of some storage (machine), like pushdown, stack, nested-stack, or counter. The storage contains configurations which can be tested by predicates and transformed by instructions. On the other hand, finite-state automata have been considered in a weighted setting in order to investigate quantitative aspects of formal languages [Eil74, SS78, KS86, BR88, Sak09, DKV09]. Weighted automata have been defined over several weight structures such as semirings [Eil74], strong bimonoids [DSV10, CDIV10, DV12], and valuation monoids [DM10, DM11, DM12]. In [DV13, DV14] weighted pushdown automata were investigated where the weights are taken from a unital valuation monoid. The automata studied in [HV15, VDH16] are weighted string automata with arbitrary storage

*Supported by the NKFI grant no K 108448

†Supported by DFG Graduiertenkolleg 1763 (QuantLA)

in which the weights are taken from unital valuation monoids. In [HV16] weighted symbolic automata with data storage were introduced, which cover weighted string automata with storage and, e.g., weighted visibly pushdown automata and weighted timed automata.

Finite-state tree automata and regular tree grammars generalize finite-state string automata and regular grammars, respectively (cf. [Eng75, GS84, GS97] for surveys⁽ⁱ⁾). Also for tree automata and tree grammars there is a rich tradition of adding storages, like pushdown tree automata [Gue83] and tree pushdown automata [SG85]. In [Eng86] the general concept of regular tree S grammar was introduced, where S is an arbitrary storage type, e.g., iterated pushdown [Eng86, EV86, EV88]. Moreover, tree automata and tree grammars have also been considered in a weighted setting, in particular, for commutative semirings [AB87], for continuous semirings [ÉK03], for complete distributive lattices [ÉL07], for fields [BR82], for tree valuation monoids [DGMM11], and multiplier monoids [Kui99, Mal04, SVF09]. For a survey on weighted tree automata we refer to [FV09].

In this paper we investigate the combination of both generalizations of regular tree grammars, and we introduce *weighted regular tree grammars with storage*. We consider an arbitrary storage type S with configurations, predicates, and instructions. The generated trees are built up over some ranked alphabet Σ . The weights are taken from a complete multiplier monoid K , which is a commutative monoid $(K, +, 0)$ with a set of operations on K . We call such a device an (S, Σ, K) -regular tree grammar, for short: (S, Σ, K) -rtg. Each rule of an (S, Σ, K) -rtg \mathcal{G} has one of the following two forms:

$$A(p) \rightarrow \sigma(A_1(f_1), \dots, A_k(f_k)) \quad (1)$$

$$A(p) \rightarrow B(f) \quad (2)$$

where A, A_1, \dots, A_k , and B are nonterminals, σ is a terminal in Σ , p is a predicate of S , and f, f_1, \dots, f_k are instructions of S . We call rules of type (2) chain rules. Each rule is equipped with an operation on K , of which the arity is the rank of the symbol σ (for rules of type (1)) or one (otherwise).

The semantics of \mathcal{G} is based on the concept of *derivation tree*. A derivation tree d is a parse tree for derivations of \mathcal{G} in the sense of [GTWW77, Sect. 3.1] (where we view \mathcal{G} as a context-free grammar with extra symbols for parentheses and comma; cf., e.g., [Eng75, Def. 3.18]). Figure 1 without the grey shaded part shows an example of a derivation tree. Each position of d is labeled by a rule of \mathcal{G} , and the nonterminals occurring in the right-hand side match with the nonterminals in the left-hand sides of the children. In addition, there is a requirement which refers to the storage type. To each position of d a storage configuration is associated (cf. the grey part of Figure 1): the root is associated with the initial configuration c_0 of S , and the other configurations are computed successively by applying the corresponding instructions f_i . Moreover, at each position, the predicate of the local rule has to be satisfied. From the viewpoint of attribute grammars [Knu68], one can consider \mathcal{G} as an attribute grammar with one inherited attribute (cf. [Eng86, Sect. 1.2] for a discussion of this viewpoint). Each derivation tree represents a derivation of a terminal tree where this latter is obtained by reading off the terminal symbols from the rules of type (1) and disregarding rules of type (2). For instance, the derivation tree in Figure 1 represents a derivation of the terminal tree $\sigma(\gamma(\alpha), \beta)$. By composing the operations which are associated to the rules we obtain an element in K and we call it the weight of that derivation tree. Finally, by summing up, in the monoid $(K, +, 0)$ the weights of all derivation trees of a terminal tree, we obtain the weight of that terminal tree. We call the mapping, which takes a terminal tree to its weight, the *weighted tree language generated by \mathcal{G}* .

⁽ⁱ⁾ We use the numbering in the arXiv-version of [GS84] published in 2015.

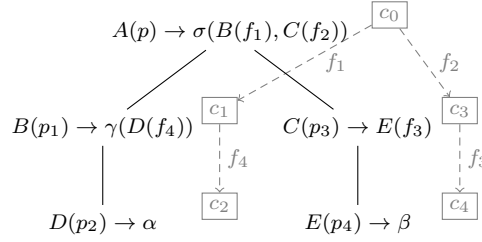


Fig. 1: An example of a derivation tree, where $c_1 = f_1(c_0)$, $c_2 = f_4(f_1(c_0))$, $c_3 = f_2(c_0)$, and $c_4 = f_3(f_2(c_0))$.

Two special cases of weighted regular tree grammars with storage are obtained by (i) choosing S to be the trivial storage type TRIV and (ii) choosing K to be the Boolean multioperator monoid \mathbb{B} . The trivial storage type contains exactly one configuration and, hence, essentially no information can be stored. Thus, for every multioperator monoid M_K associated to a semiring K , $(\text{TRIV}, \Sigma, M_K)$ -rtgs are extensions of K -weighted regular tree grammars of [AB87] (because in [AB87] there are no chain rules). The Boolean multioperator monoid \mathbb{B} is an extension of the monoid $(\{0, 1\}, \vee, 0)$ of truth values with disjunction as binary operation. The extension amounts to equip this monoid with an n -ary conjunction for each $n \in \mathbb{N}$. If $K = \mathbb{B}$, then such grammars are essentially (unweighted) regular tree grammars with storage [Eng86].

The reader might wonder why each rule of an (S, Σ, K) -rtg has either zero or one terminal, and not an arbitrary number of terminals (as it is usual for regular tree grammars). On the one hand, there is no technical problem to define such generalized (S, Σ, K) -rtgs. On the other hand, the proofs of most of our results are based on the restricted form of rules (as in case of regular tree grammars [Bra69, GS84, GS97]). To achieve a normal form lemma which takes a generalized (S, Σ, K) -rtg and transforms it into an equivalent (S, Σ, K) -rtg requires rather technical conditions (in fact restrictions) on the M-monoid (the operation of the original rule has to be decomposed into appropriate operations). Thus we refrain from dealing with generalized (S, Σ, K) -rtgs. Indeed, our (S, Σ, K) -rtgs might also be called weighted tree automata with ε -moves (and storage).

In this paper we start to develop a theory of the class of weighted tree languages generated by (S, Σ, K) -rtg. In Section 3, after introducing our new grammar model, we show that (S, Σ, \mathbb{B}) -rtgs have the same power as $(\text{TRIV}, \Sigma, \mathbb{B})$ -rtgs assuming that S is a finite storage type containing the always true predicate and the identity instruction.

In Section 4 we prove that the supports of (S, Σ, K) -regular tree languages are (S, Σ, \mathbb{B}) -regular provided that K is associated to a complete, zero-sum free, and commutative strong bimonoid. For the proof we employ the approach and the technique of [Kir11].

In Section 5 we deal with two decompositions of the weighted tree language generated by an (S, Σ, K) -rtg. First, we represent every (S, Σ, K) -regular tree language as the composition of some particular mapping \mathcal{B}_Δ and some (TRIV, Θ, K) -regular tree language. The mapping \mathcal{B}_Δ is based on the concept of storage behaviour of S : roughly speaking, it enriches each terminal tree with a possible behaviour of S . This result was inspired by the decomposition of CFT(S)-transducers into an approximation and a macro tree transducer in [EV86, Thm. 3.26]. Our second result is based on [DV13, Lm. 3 and Lm. 4] and shows that the weights of an (S, Σ, K) -rtg can be encoded into an alphabetic mapping, which is applied to an unambiguous, chain-free (S, Θ, \mathbb{B}) -rtg. As a consequence of this and the fact that finite storage can be eliminated from (S, Σ, \mathbb{B}) -rtgs, we can prove that one can drop finite storage types from (S, Σ, K) -

rtgs (for arbitrary K) without loosing power. Finally, we combine the two decomposition results and obtain a characterization of the class of (S, Σ, K) -regular tree languages in terms of three simple and basic components: the mapping \mathcal{B}_Δ , an unambiguous and chain-free $(\text{TRIV}, \Theta, \mathbb{B})$ -rtg, and an alphabetic mapping. We illustrate the decomposition results by showing an example in detail.

In Section 6 we show that, even for the Boolean multioperator monoid, chain rules increase the power of (weighted) regular tree grammars with storage. Moreover, we study under which restrictions on the multioperator monoid K chain rules can be eliminated from (S, Σ, K) -rtgs.

In Section 7 we prove a characterization of (S, Σ, K) -regular tree languages in terms of weighted monadic second order logic (MSO). For this, we introduce a weighted MSO-logic with storage behaviour based on M-expressions of [FSV12, FV18] and on the MSO-logic introduced in [VDH16]. However, our new logic generalizes the weighted MSO-logic with storage behaviour of [VDH16] by considering trees as models and by allowing chain rules.

Apart from Section 7 we have tried to write the paper in such a way that it is self-contained. In Section 7 we have given detailed references to the literature where the reader can find the relevant definitions. Readers who are familiar with algebraic structures (like semirings and multioperator monoids) and concepts concerning trees (like tree transformations, weighted tree languages, and term rewriting systems) can skip Sections 2.2 and 2.3 on first reading and consult them later if necessary.

2 Preliminaries

2.1 Notations

We denote the set $\{0, 1, 2, \dots\}$ of natural numbers by \mathbb{N} and the set $\{1, 2, \dots\}$ by \mathbb{N}_+ . For every $n \in \mathbb{N}$ we define $[n] = \{1, \dots, n\}$.

Let A_1, A_2 be two sets and let $a = (a_1, a_2) \in A_1 \times A_2$. Then, for each $i \in [2]$, the *i-th projection of a* , denoted by $\text{pr}_i(a)$, is defined by $\text{pr}_i(a) = a_i$.

Let A be a set. Then A^* denotes the set of all finite sequences over A including the empty sequence denoted by ε . We denote the set of all subsets of A by $\mathcal{P}(A)$. Moreover, we denote by id_A the identity mapping over A . If A contains exactly one element, then sometimes A is identified with that element. For every $B, C \subseteq A^*$ we let $BC = \{bc \mid b \in B, c \in C\}$.

Let $u, w \in A^*$. We say that *u is a prefix of w* , denoted by $u \preceq w$, if there is a $v \in A^*$ with $uv = w$. Moreover, for every $a \in A$ we denote by $|w|_a$ the number of occurrences of a in w .

A set is *countable* if its cardinality coincides with that of a subset of the natural numbers.

2.2 Algebraic structures

We recall the concept of strong bimonoids and semirings from [DSV10, CDIV10] and [HW98, Gol99, Eil74], respectively, and that of multioperator monoids from [Kui99].

A monoid $(K, +, 0)$ is *commutative* if $a + b = b + a$, *zero-sum free* if $a + b = 0$ implies $a = b = 0$, and *idempotent* if $a + a = a$ for every $a, b \in K$. Moreover, K is *complete* if it has a sum operation $\sum_I: K^I \rightarrow K$ for each countable index set I which coincides with $+$ when I is finite (for the axioms cf. [Eil74, p. 124]).

A *strong bimonoid* [DSV10, CDIV10] is an algebra $(K, +, \cdot, 0, 1)$, where $(K, +, 0)$ is a commutative monoid, $(K, \cdot, 1)$ is a monoid, and $0 \cdot a = a \cdot 0 = 0$ for each $a \in K$. We assume that $0 \neq 1$. We call K *commutative* if $(K, \cdot, 1)$ is commutative. The strong bimonoid K is called a *semiring* if $a \cdot (b + c) =$

$a \cdot b + a \cdot c$ and $(b + c) \cdot a = b \cdot a + c \cdot a$ for every $a, b, c \in K$. We refer to [DSV10, Ex. 1] for a number of examples of strong bimonoids, e.g., each bounded lattice is a strong bimonoid.

A semiring $(K, +, \cdot, 0, 1)$ is *complete* if the monoid $(K, +, 0)$ is complete and the generalized distributivity law holds for infinite sums (cf. [Eil74, p. 124]). The semiring $(\{0, 1\}, \vee, \wedge, 0, 1)$ is called the *Boolean semiring*, where 0 and 1 stand for ‘false’ and ‘true’; and \vee and \wedge are the usual disjunction and conjunction of truth values, respectively. The Boolean semiring and the semiring $(\mathbb{N}_\infty, +, \cdot, 0, 1)$ with $\mathbb{N}_\infty = \mathbb{N} \cup \{\infty\}$ and the natural extension of $+$ to \mathbb{N}_∞ are complete (cf. [ÉL02, Example 1],[Eil74, Chap. VI, Sec. 2]).

Let $k \in \mathbb{N}$ and K be a set. We denote the set of all k -ary operations (all operations) on K by $\text{Ops}^{(k)}(K)$ (resp. $\text{Ops}(K)$). For every $k \in \mathbb{N}$, the k -ary constant zero function $0_k \in \text{Ops}^{(k)}(K)$ is defined by $0_k(a_1, \dots, a_k) = 0$ for every $a_1, \dots, a_k \in K$. For a set $\Omega \subseteq \text{Ops}(K)$ we define $\Omega^{(k)} = \Omega \cap \text{Ops}^{(k)}(K)$.

A *multioperator monoid* (for short: *M-monoid*) [Kui99, Mal04, SVF09] is a tuple $(K, +, 0, \Omega)$ such that $(K, +, 0)$ is a commutative monoid and $\{0_k \mid k \in \mathbb{N}\} \subseteq \Omega \subseteq \text{Ops}(K)$. Moreover, we require that for every $k \in \mathbb{N}$, each operation $\omega \in \Omega^{(k)}$ is absorptive, which means that for every $a_1, \dots, a_k \in K$, and $i \in [k]$, the equality $a_i = 0$ implies $\omega(a_1, \dots, a_k) = 0$. We note that in [SVF09, FSV12] such an M-monoid was called absorptive. An M-monoid $(K, +, 0, \Omega)$ is called *complete* if $(K, +, 0)$ has this property. For instance, the structure $(\mathbb{N} \cup \{\infty\}, +, 0, \Omega)$ is a complete M-monoid, where $+$ is extended to sum over countable index sets in the obvious way, $\Omega = \{0_k \mid k \in \mathbb{N}\} \cup \{\min_k \mid k \in \mathbb{N}\}$, and \min_k is the k -ary minimum function (both extended to $\mathbb{N} \cup \{\infty\}$).

The M-monoid $(\{0, 1\}, \vee, 0, \Omega)$ is called the *Boolean M-monoid*, denoted by \mathbb{B} , where for every $k \in \mathbb{N}$, we have $\Omega^{(k)} = \{0_k, \text{all}_{k,1}\}$ and for every $a_1, \dots, a_k \in \{0, 1\}$, we have $\text{all}_{k,1}(a_1, \dots, a_k) = 1$ if and only if $a_1 = \dots = a_k = 1$. Then \mathbb{B} , equipped with the the operations $(\bigvee_I: \{0, 1\}^I \rightarrow \{0, 1\} \mid I \text{ is a countable index set})$, is a complete M-monoid, where for every I and $f \in \{0, 1\}^I$, we have $\bigvee_I(f) = 1$ if there is an $i \in I$ with $f(i) = 1$, and 0 otherwise.

To every strong bimonoid $(K, +, \cdot, 0, 1)$ we associate the M-monoid $(M_K, +, 0, \Omega)$, where $M_K = K$ and $\Omega^{(k)} = \{\text{mul}_{k,a} \mid a \in K\}$ and $\text{mul}_{k,a}(a_1, \dots, a_k) = a_1 \cdot \dots \cdot a_k \cdot a$ for every $a_1, \dots, a_k \in K$. (Note that $\text{mul}_{k,0} = 0_k$ for every $k \geq 0$.) The corresponding construction for semirings can be found in [FMV09, Def. 8.5] and [FSV12, page 261]. Note that the Boolean M-monoid is equal to the M-monoid associated to the Boolean semiring.

For the remainder of the paper, if K is left unspecified, then it stands for an arbitrary complete M-monoid $(K, +, 0, \Omega)$.

2.3 Trees, tree transformations, weighted tree languages, and term rewriting systems

By an alphabet we mean a finite, non-empty set. A *ranked alphabet* is an alphabet Σ together with a mapping $\text{rk}_\Sigma: \Sigma \rightarrow \mathbb{N}$; the natural number $\text{rk}_\Sigma(\sigma)$ is called the *rank* of σ . For every $k \in \mathbb{N}$ we let $\Sigma^{(k)} = \text{rk}_\Sigma^{-1}(k)$. To avoid obvious cases, we assume that $\Sigma^{(0)} \neq \emptyset$. If $\sigma \in \Sigma^{(k)}$, i.e., the rank of σ is k , then we write briefly $\sigma^{(k)}$. We denote the maximal rank which occurs in Σ , i.e., the number $\max\{k \mid \Sigma^{(k)} \neq \emptyset\}$, by $\text{maxrk}(\Sigma)$.

Let Σ be a ranked alphabet and H be a set. The *set of trees over Σ indexed by H* , denoted by $T_\Sigma(H)$, is the smallest set T such that (i) $H \subseteq T$ and (ii) for every $k \in \mathbb{N}$, $\sigma \in \Sigma^{(k)}$, and $\xi_1, \dots, \xi_k \in T$ also $\sigma(\xi_1, \dots, \xi_k) \in T$. We abbreviate $T_\Sigma(\emptyset)$ by T_Σ . Also, for $\sigma \in \Sigma^{(1)}$ and $\xi \in T_\Sigma$ we abbreviate

$\sigma(\dots\sigma(\xi)\dots)$ with n occurrences of σ by $\sigma^n(\xi)$. For each $\sigma \in \Sigma^{(0)}$ we abbreviate $\sigma()$ by σ . We note that each $\xi \in T_\Sigma$ has the form $\sigma(\xi_1, \dots, \xi_k)$ for some $k \in \mathbb{N}$, $\sigma \in \Sigma^{(k)}$, and $\xi_1, \dots, \xi_k \in T_\Sigma$.

We define the mappings $\text{pos} : T_\Sigma \rightarrow \mathcal{P}(\mathbb{N}^*)$, $\text{path} : T_\Sigma \rightarrow \mathcal{P}(\Sigma^*)$, and $\text{height} : T_\Sigma \rightarrow \mathbb{N}$ by induction on the structure of their argument. For this, let $\xi = \sigma(\xi_1, \dots, \xi_k)$ in T_Σ . If $k = 0$, then we let $\text{pos}(\xi) = \{\varepsilon\}$, $\text{path}(\xi) = \{\sigma\}$, and $\text{height}(\xi) = 1$. If $k \geq 1$, then we let $\text{pos}(\xi) = \{\varepsilon\} \cup \{iw \mid i \in [k], w \in \text{pos}(\xi_i)\}$, $\text{path}(\xi) = \bigcup_{i \in [k]} \{\sigma w \mid w \in \text{path}(\xi_i)\}$, and $\text{height}(\xi) = 1 + \max\{\text{height}(\xi_i) \mid i \in [k]\}$. We call $\text{pos}(\xi)$ the *set of positions in ξ* and denote the cardinality of $\text{pos}(\xi)$ by $\text{size}(\xi)$. Moreover, we can define the *lexicographic order* \leq_{lex} on $\text{pos}(\xi)$ as usual.

Let $\xi \in T_\Sigma$. For every $w \in \text{pos}(\xi)$, we define the *label* $\xi(w) \in \Sigma$ of ξ at *position* w and the *subtree* $\xi|_w \in T_\Sigma$ of ξ at *position* w in the usual way (cf. e.g. [FSV12]). Moreover, we abbreviate $\text{rk}_\Sigma(\xi(w))$ by $\text{rk}_\xi(w)$. For every $\Theta \subseteq \Sigma$, we let $\text{pos}_\Theta(\xi) = \{w \in \text{pos}(\xi) \mid \xi(w) \in \Theta\}$.

Let Σ and Δ be two ranked alphabets. A *tree transformation from Σ to Δ* is a mapping $\tau : T_\Sigma \rightarrow \mathcal{P}(T_\Delta)$. Tree relabelings are particular tree transformations. For their definition, let $\tau : \Sigma \rightarrow \mathcal{P}(\Delta)$ be a mapping such that $\tau(\sigma) \subseteq \Delta^{(k)}$ for every $k \geq 0$ and $\sigma \in \Sigma^{(k)}$. This mapping is extended to a mapping $\tau' : T_\Sigma \rightarrow \mathcal{P}(T_\Delta)$ by defining inductively

$$\tau'(\sigma(\xi_1, \dots, \xi_k)) = \{\gamma(\zeta_1, \dots, \zeta_k) \mid \gamma \in \tau(\sigma), \zeta_1 \in \tau'(\xi_1), \dots, \zeta_k \in \tau'(\xi_k)\}$$

for every $k \geq 0$, $\sigma \in \Sigma^{(k)}$, and $\xi_1, \dots, \xi_k \in T_\Sigma$. Obviously, $\tau'(\sigma(\xi_1, \dots, \xi_k))$ is a finite set. Next we extend τ' to a mapping $\tau'' : \mathcal{P}(T_\Sigma) \rightarrow \mathcal{P}(T_\Delta)$ by defining $\tau''(L) = \{\zeta \in T_\Delta \mid \zeta \in \tau'(\xi), \xi \in L\}$ for every $L \in \mathcal{P}(T_\Sigma)$. We call τ' and τ'' the *tree relabeling induced by τ* . In the sequel we will drop the primes from τ' and τ'' .

For the remainder of the paper, if Σ is unspecified, then it stands for an arbitrary ranked alphabet.

We call each mapping $s : T_\Sigma \rightarrow K$ a *weighted tree language*. The *support* of s is defined by the formula $\text{supp}(s) = \{\xi \in T_\Sigma \mid s(\xi) \neq 0\}$. A weighted tree language s is called a *monome* if $\text{supp}(s)$ is the empty set or a singleton. If $\text{supp}(s) \subseteq \{\xi\}$ for some $\xi \in T_\Sigma$, then we also write $s(\xi).\xi$ instead of s . In particular, for each $\xi \in T_\Sigma$ the expression $0.\xi$ denotes the monome s with $\text{supp}(s) = \emptyset$. We denote the set of all monomes of type $T_\Sigma \rightarrow K$ by $K[T_\Sigma]$.

Let $(s_i \mid i \in I)$ be a family of weighted tree languages of type $T_\Sigma \rightarrow K$ with a countable index set I . The *sum* of $(s_i \mid i \in I)$, denoted by $\sum_{i \in I} s_i$, is the weighted tree language $(\sum_{i \in I} s_i) : T_\Sigma \rightarrow K$ defined for each $\xi \in T_\Sigma$ by

$$\left(\sum_{i \in I} s_i\right)(\xi) = \sum_{i \in I} s_i(\xi) .$$

Note that this sum is well defined because K is complete. We write $s_1 + s_2$ for $\sum_{i \in \{1,2\}} s_i$.

Let $\tau : T_\Sigma \rightarrow \mathcal{P}(T_\Delta)$ be a tree transformation and $s : T_\Delta \rightarrow K$ be a weighted tree language. We define the *composition of τ and s* , denoted by $(\tau; s)$, to be the weighted tree language $(\tau; s) : T_\Sigma \rightarrow K$ defined for each $\xi \in T_\Sigma$ by

$$(\tau; s)(\xi) = \sum_{\zeta \in \tau(\xi)} s(\zeta) .$$

Note that $\tau(\xi)$ can be infinite; however, the sum is well defined because K is complete. Whenever there is no confusion we write $\tau; s$ instead of $(\tau; s)$.

We fix a countable set $X = \{x_1, x_2, \dots\}$ of *variables* and let $X_k = \{x_1, \dots, x_k\}$ for each $k \in \mathbb{N}$. We assume that X is disjoint from each ranked alphabet considered in this paper.

A *term rewriting system* [BN99] is a tuple $\mathcal{R} = (\Sigma, R)$ where Σ is a ranked alphabet and R is a finite set. Each element of R is called a *rule* and it has the form $l \rightarrow r$ where $l, r \in T_\Sigma(X_k)$ for some $k \in \mathbb{N}$ such that $l \notin X$, and each variable which occurs in r also occurs in l . The *rewrite relation induced by \mathcal{R}* , denoted by $\Rightarrow_{\mathcal{R}}$, is the binary relation $\Rightarrow_{\mathcal{R}} \subseteq T_\Sigma \times T_\Sigma$ defined for each $\xi_1, \xi_2 \in T_\Sigma$ as follows: $\xi_1 \Rightarrow_{\mathcal{R}} \xi_2$ if

- there is a $\theta \in T_\Sigma(X_1)$ in which x_1 occurs exactly once,
- there is a rule $l \rightarrow r$ in R with $l, r \in T_\Sigma(X_k)$ for some $k \in \mathbb{N}$, and
- there are $\theta_1, \dots, \theta_k \in T_\Sigma$

such that ξ_1 is obtained from θ by replacing x_1 by l' , and l' is obtained from l by replacing each occurrence of x_i by θ_i (for each $i \in [k]$); ξ_2 is obtained from θ by replacing x_1 by r' , and r' is obtained from r in the same way as l' is obtained from l . As usual for binary relations, we denote the reflexive, transitive closure of $\Rightarrow_{\mathcal{R}}$ by $\Rightarrow_{\mathcal{R}}^*$.

2.4 Storage types and behaviours

We recall the concept of storage type from [Eng86] with a slight modification (cf. [HV15]).

A *storage type* is a tuple $S = (C, P, F, c_0)$, where C is a set (*configurations*), $c_0 \in C$ (*initial configuration*), P is a non-empty set of total functions each having the type $p : C \rightarrow \{0, 1\}$ (*predicates*), and F is a non-empty set of partial functions $f : C \rightarrow C$ (*instructions*). A storage type is *finite* if C is a finite set.

The *identity instruction* is the total function id_C . The *always true predicate*, denoted by true_C , is the predicate such that $\text{true}_C(c) = 1$ for each $c \in C$. For the storage type $S = (C, P, F, c_0)$, we denote by $S_{\text{true}, \text{id}}$ the storage type $(C, P \cup \{\text{true}_C\}, F \cup \{\text{id}_C\}, c_0)$. Thus, S contains true_C and id_C if and only if $S_{\text{true}, \text{id}} = S$.

We note that our definition of storage type is a special case of the one in [Eng86] (and also in [EV86, EV88]) in the sense that there the initial configuration c_0 is replaced by a set I of inputs, a set E of encoding symbols, and a meaning function m . Each encoding symbol e is interpreted as a partial function $m(e) : I \rightarrow C$ and allows to define machines with *input and output*. Thus, our storage type (C, P, F, c_0) is the storage type (C, P', F', I, E, m) in the sense of [Eng86] with $I = \{i\}$, $E = \{e\}$, $m(e)(i) = c_0$, $P' = \{p' \mid p \in P\}$, and $F' = \{f' \in F \mid f \in F\}$ are sets of names for elements in P and F , respectively, and $m(p') = p$ and $m(f') = f$.

For the remainder of the paper, if S is unspecified, then it stands for an arbitrary storage type $S = (C, P, F, c_0)$.

Particular storage types Here we recall three particular storage types from [Eng86, EV86]: the trivial storage type, the pushdown storage type, and the counter storage type.

The *trivial storage type*, denoted by TRIV, is the storage type $(\{c\}, \{\text{true}_{\{c\}}\}, \{\text{id}_{\{c\}}\}, c)$ for some arbitrary but fixed symbol c .

Let Γ be a fixed infinite set (*pushdown symbols*) and let $\gamma_0 \in \Gamma$ be a fixed symbol. The *pushdown of S* is the storage type $P(S) = (C', P', F', c'_0)$ where

- $C' = (\Gamma \times C)^+$ (there is no empty pushdown),
- $c'_0 = (\gamma_0, c_0)$,

- $P' = \{\text{bottom}\} \cup \{(\text{top} = \gamma) \mid \gamma \in \Gamma\} \cup \{\text{test}(p) \mid p \in P\}$ such that for every $(\delta, c) \in \Gamma \times C$ and $\alpha \in (\Gamma \times C)^*$ we have

$$\begin{aligned} \text{bottom}((\delta, c)\alpha) &= 1 \text{ iff } \alpha = \varepsilon \\ (\text{top} = \gamma)((\delta, c)\alpha) &= 1 \text{ iff } \gamma = \delta \\ (\text{test}(p))((\delta, c)\alpha) &= p(c) , \end{aligned}$$

- $F' = \{\text{pop}\} \cup \{\text{stay}(\gamma) \mid \gamma \in \Gamma\} \cup \{\text{push}(\gamma, f) \mid \gamma \in \Gamma, f \in F\} \cup \{\text{id}_{C'}\}$ such that for every $(\delta, c) \in \Gamma \times C$ and $\alpha \in (\Gamma \times C)^*$ we have

$$\begin{aligned} \text{pop}((\delta, c)\alpha) &= \alpha \text{ if } \alpha \neq \varepsilon \\ \text{push}(\gamma, f)((\delta, c)\alpha) &= (\gamma, f(c))(\delta, c)\alpha \text{ if } f(c) \text{ is defined} \end{aligned}$$

and undefined in all other situations. Moreover,

$$\text{stay}(\gamma)((\delta, c)\alpha) = (\gamma, c)\alpha .$$

Recall that $\text{id}_{C'}$ is the identity instruction on C' .

For each $n \geq 0$ we define the storage type $P^n(S)$ inductively as follows: $P^0(S) = S$ and $P^n(S) = P(P^{n-1}(S))$ if $n \geq 1$. The n -iterated pushdown storage, denoted by P^n , is the storage type $P^n(\text{TRIV})$. We note that P^n contains the always true predicate $\underbrace{\text{test}(\dots \text{test}(\text{true}_{\{c\}}) \dots)}_n$ and the identity instruction.

Thus, $(P^n)_{\text{true}, \text{id}} = P^n$. Moreover, for the 1-iterated pushdown storage, we write Γ^+ for $(\Gamma \times \{c\})^+$ and abbreviate instructions by ignoring the part of the trivial storage type, e.g., we write $\text{push}(\gamma)$ instead of $\text{push}(\gamma, \text{id}_{\{c\}})$. In this case we abbreviate the predicate $\text{test}(\text{true}_{\{c\}})$ by true .

The storage type *counter*, denoted by COUNT , is defined by $(\mathbb{N}, \{\text{true}_{\mathbb{N}}, \text{zero}\}, \{\text{id}_{\mathbb{N}}, \text{inc}, \text{dec}\}, 0)$ where for each $c \in \mathbb{N}$

- $\text{zero}(c) = 1$ iff $c = 0$,
- $\text{inc}(c) = c + 1$, and $\text{dec}(c) = c - 1$ if $c \geq 1$ and undefined otherwise.

Behaviour Let $P' \subseteq P$ and $F' \subseteq F$ be finite non-empty subsets. Moreover, let $n \in \mathbb{N}$. We define the ranked alphabet $\Delta = \bigcup_{0 \leq k \leq n} \Delta^{(k)}$ with $\Delta^{(k)} = P' \times (F')^k$. We call Δ the *ranked alphabet n -corresponding to P' and F'* . We write elements (p, f_1, \dots, f_k) of Δ in the shorter form $(p, f_1 \dots f_k)$.

The concept of behaviour is inspired by and closely related to the concept of approximation [EV86, Def. 3.23]. Formally, let $c \in C$, $n \in \mathbb{N}$, and Δ be the ranked alphabet n -corresponding to some finite sets $P' \subseteq P$ and $F' \subseteq F$. Then a tree $b \in T_\Delta$ is a (Δ, c) -behaviour if there is a family $(c_w \in C \mid w \in \text{pos}(b))$ of configurations such that $c_\varepsilon = c$ and for every $w \in \text{pos}(b)$: if $b(w) = (p, f_1 \dots f_k)$, then

- $p(c_w) = 1$ and
- for every $1 \leq i \leq k$, the configuration $f_i(c_w)$ is defined and $c_{wi} = f_i(c_w)$.

If b is a (Δ, c) -behaviour, then we call $(c_w \in C \mid w \in \text{pos}(b))$ the *family of configurations determined by b and c* . In Fig. 2 we illustrate these concepts for the storage type P^1 and for $n = 2$.

A Δ -behaviour is a (Δ, c_0) -behaviour. We denote the set of all (Δ, c) -behaviours by $\mathcal{B}(\Delta, c)$, and the set of all Δ -behaviours by $\mathcal{B}(\Delta)$.

We will use the concept of corresponding ranked alphabet only in particular scenarios, for which we will now define a more convenient notion. Let Σ be a ranked alphabet, $P' \subseteq P$ and $F' \subseteq F$ be finite subsets. Then we call the ranked alphabet $\text{maxrk}(\Sigma)$ -corresponding to P' and F' the *ranked alphabet corresponding to Σ , P' , and F'* .

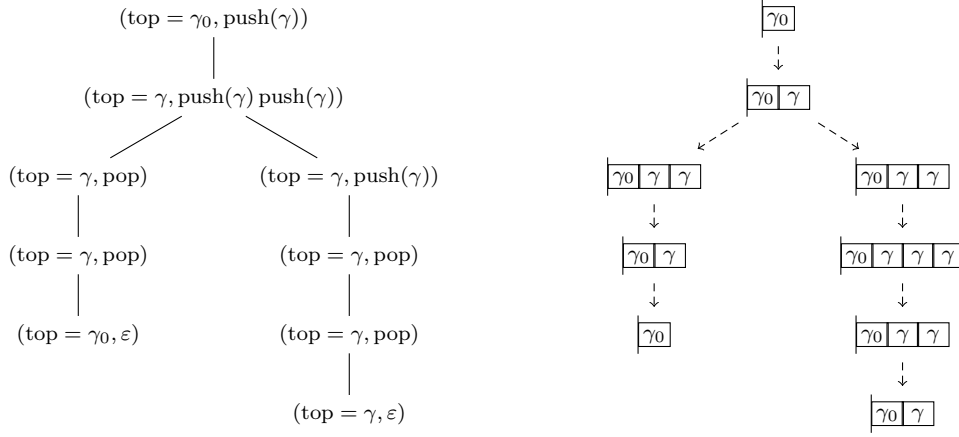


Fig. 2: On the left-hand side, a (Δ, γ_0) -behaviour b for Δ 2-corresponding to $P' = \{\text{top} = \gamma_0, \text{top} = \gamma\}$ and $F' = \{\text{push}(\gamma), \text{pop}\}$, and on the right-hand side, the family $(c_w \mid w \in \text{pos}(b))$ of configurations determined by b and γ_0 .

3 Weighted regular tree grammars with storage

In this section we combine the concept of regular tree grammar with storage [Eng86] with the weighting technique using multioperator monoids [Kui99, Mal05, SVF09, FSV12].

3.1 Definition of the concept, examples, and special cases

We recall that K is a complete M-monoid $(K, +, 0, \Omega)$, $S = (C, P, F, c_0)$ is a storage type, and Σ is a ranked alphabet.

A *regular tree grammar over Σ with storage S and weights in K* (for short: (S, Σ, K) -rtg) is a tuple $\mathcal{G} = (N, Z, R, \text{wt})$, where

- N is a finite set (*nonterminals*) such that $N \cap \Sigma = \emptyset$,
- $Z \subseteq N$ (set of *initial nonterminals*),
- R is a finite and non-empty set of *rules*; each rule has one of the following forms:

$$A(p) \rightarrow \sigma(A_1(f_1), \dots, A_k(f_k)) \quad (1)$$

$$A(p) \rightarrow B(f) \quad (2)$$

where $k \geq 0$, $A, A_1, \dots, A_k, B \in N$, $p \in P$, $\sigma \in \Sigma^{(k)}$, and $f_1, \dots, f_k, f \in F$, and

- $\text{wt} : R \rightarrow \Omega$ is the *weight function* such that each rule of the form (1) is mapped to an element in $\Omega^{(k)}$ and each rule of the form (2) is mapped to an element in $\Omega^{(1)}$.

If r is a rule of the form (1), then we denote its parts A and A_ℓ (with $1 \leq \ell \leq k$) by $\text{lhs}_N(r)$ and $\text{rhs}_{N,\ell}(r)$, respectively; if r is a rule of the form (2), then we denote A and B by $\text{lhs}_N(r)$ and $\text{rhs}_{N,1}(r)$, respectively. Rules of type (2) are called *chain rules*. If \mathcal{G} does not contain chain rules, then we call it *chain-free*.

In [Eng86, EV86, EV88] a binary derivation relation was defined for (unweighted) regular tree grammars with storage. Intuitively, the application of a rule follows the principle of context-free rewriting. In

each sentential form each occurrence of a nonterminal keeps a configuration c of S and a rule may only be applied if its predicate holds on c . Each instruction f occurring in the right-hand side of the rule is replaced by the configuration $f(c)$ if this is defined.

Here we formalize derivations in a different, but equivalent way using derivation trees (cf. Figure 1). Let $\mathcal{G} = (N, Z, R, \text{wt})$ be an (S, Σ, K) -rtg and $P_{\mathcal{G}} \subseteq P$ and $F_{\mathcal{G}} \subseteq F$ be the finite sets of predicates and instructions, respectively, which occur in R . Moreover, let $\Delta_{\mathcal{G}}$ be the ranked alphabet corresponding to Σ , $P_{\mathcal{G}}$, and $F_{\mathcal{G}}$. We note that $\Delta_{\mathcal{G}}$ is non-empty because we required that $R \neq \emptyset$. (In particular, if R contains one rule of the form (1) with $k = 0$ and no rule of the form (2), then $P_{\mathcal{G}} = \{p\}$ and $F_{\mathcal{G}} = \emptyset$. Hence $\Delta_{\mathcal{G}} = \Delta_{\mathcal{G}}^{(0)} = \{(p, \varepsilon)\}$.)

In the following we will consider R as a ranked alphabet by associating rank k with a rule if its right-hand side contains k nonterminals. Let $d \in T_R$. We can retrieve from d a tree in T_{Σ} by using the mapping $\pi : T_R \rightarrow T_{\Sigma}$ defined by induction on its argument such that for each $d = r(d_1, \dots, d_k)$ in T_R we have

$$\pi(r(d_1, \dots, d_k)) = \begin{cases} \sigma(\pi(d_1), \dots, \pi(d_k)) & \text{if } r \text{ is of type (1)} \\ \pi(d_1) & \text{if } r \text{ is of type (2)} \end{cases} .$$

Also we can retrieve from d a tree over $\Delta_{\mathcal{G}}$ by using the tree relabeling β . This tree relabeling is induced by the mapping $\beta : R \rightarrow \Delta_{\mathcal{G}}$ defined by $\beta(r) = (p, f_1 \dots f_k)$ if r has the form (1), and by $\beta(r) = (p, f)$ if r has the form (2).

Let $\xi \in T_{\Sigma}$, $N' \subseteq N$, and $c \in C$. An (N', c) -derivation tree of \mathcal{G} for ξ is a tree $d \in T_R$ such that

- $\text{lhs}_N(d(\varepsilon)) \in N'$,
- $\pi(d) = \xi$,
- for each $w \in \text{pos}(d)$ and each $\ell \in [\text{rk}_d(w)]$ we have $\text{rhs}_{N, \ell}(d(w)) = \text{lhs}_N(d(w\ell))$, and
- the tree $\beta(d)$ is in $\mathcal{B}(\Delta_{\mathcal{G}}, c)$.

We denote the set of all such trees by $D_{\mathcal{G}}(N', \xi, c)$ and we abbreviate $D_{\mathcal{G}}(Z, \xi, c_0)$ by $D_{\mathcal{G}}(\xi)$. We note that, if \mathcal{G} is chain-free, then $\text{pos}(d) = \text{pos}(\xi)$ for each $d \in D_{\mathcal{G}}(\xi)$, and hence, $D_{\mathcal{G}}(\xi)$ is finite for each $\xi \in T_{\Sigma}$. Finally, a derivation tree of \mathcal{G} for a $\xi \in T_{\Sigma}$ is an (N', c) -derivation tree of \mathcal{G} for ξ for some N' and c .

Let $d \in D_{\mathcal{G}}(N', \xi, c)$. We define $\text{wt}'(d) \in K$ by $\text{wt}'(d) = \text{wt}''(d, \varepsilon)$ and, in its turn, for each $w \in \text{pos}(d)$ we define the value $\text{wt}''(d, w) \in K$ inductively on w as follows:

$$\text{wt}''(d, w) = \text{wt}(d(w)) \left(\text{wt}''(d, w1), \dots, \text{wt}''(d, w \text{rk}_d(w)) \right).$$

We note that $\text{wt}(d(w))$ is an operation in Ω of arity $\text{rk}_d(w)$. For notational convenience we will drop in the sequel the primes from wt' and wt'' and simply write wt .

Then the weighted tree language generated by \mathcal{G} is the mapping $\llbracket \mathcal{G} \rrbracket : T_{\Sigma} \rightarrow K$ defined for each $\xi \in T_{\Sigma}$ by

$$\llbracket \mathcal{G} \rrbracket(\xi) = \sum_{d \in D_{\mathcal{G}}(\xi)} \text{wt}(d).$$

Since \mathcal{G} may contain chain rules, the set $D_{\mathcal{G}}(\xi)$ can be infinite. However, this sum is well defined because K is complete.

A weighted tree language $s : T_{\Sigma} \rightarrow K$ is called (S, Σ, K) -regular if there is an (S, Σ, K) -rtg \mathcal{G} such that $s = \llbracket \mathcal{G} \rrbracket$, and we call s chain-free (S, Σ, K) -regular if additionally \mathcal{G} is chain-free. The class of all

(S, Σ, K) -regular tree languages and of all chain-free (S, Σ, K) -regular tree languages are denoted by $\text{Reg}(S, \Sigma, K)$ and $\text{Reg}_{\text{nc}}(S, \Sigma, K)$, respectively.⁽ⁱⁱ⁾

Example 3.1. Let $(M_{\mathbb{N}_\infty}, +, 0, \Omega)$ be the complete M-monoid associated to the complete semiring $(\mathbb{N}_\infty, +, \cdot, 0, 1)$ of natural numbers. Moreover, let $\Sigma = \{\alpha^{(0)}, \delta^{(1)}, \sigma^{(2)}\}$. We consider the weighted tree language $s: T_\Sigma \rightarrow M_{\mathbb{N}_\infty}$ with

$$s(\xi) = \begin{cases} 8^n & \text{if } \xi = \sigma(\delta^n(\alpha), \delta^n(\alpha)) \text{ for some } n \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

for each $\xi \in T_\Sigma$.

Then s can be generated by a $(P^1, \Sigma, M_{\mathbb{N}_\infty})$ -rtg \mathcal{G} . Intuitively, \mathcal{G} first generates, using chain rules, a pushdown configuration of length n , then it generates σ and makes two copies of this configuration, and finally it turns each copy of each pushdown cell into a δ . For this, recall that Γ is the infinite set of pushdown symbols and that $\gamma_0 \in \Gamma$ is the initial pushdown symbol. Let $\gamma \in \Gamma$ with $\gamma \neq \gamma_0$.

We construct the $(P^1, \Sigma, M_{\mathbb{N}_\infty})$ -rtg $\mathcal{G} = (N, Z, R, \text{wt})$, where $N = \{Z, A\}$ and R consists of the rules

$$\begin{aligned} r_1 : Z(\text{true}) &\rightarrow Z(\text{push}(\gamma)) & r_2 : Z(\text{true}) &\rightarrow \sigma(A(\text{id}_{\Gamma^+}), A(\text{id}_{\Gamma^+})) \\ r_3 : A(\text{top} = \gamma) &\rightarrow \delta(A(\text{pop})) & r_4 : A(\text{top} = \gamma_0) &\rightarrow \alpha \end{aligned}$$

where $\text{wt}(r_1) = \text{wt}(r_3) = \text{mul}_{1,2}$, $\text{wt}(r_2) = \text{mul}_{2,1}$, and $\text{wt}(r_4) = \text{mul}_{0,1}$.

For each $n \geq 0$ and $\xi = \sigma(\delta^n(\alpha), \delta^n(\alpha))$, we have $D_{\mathcal{G}}(\xi) = \{d_n\}$ where $d_n = r_1^n r_2(r_3^n r_4, r_3^n r_4)$. Figure 3 shows d_2 for the tree $\xi = \sigma(\delta^2(\alpha), \delta^2(\alpha))$, the $\Delta_{\mathcal{G}}$ -behaviour $b = \beta(d_2)$, and the family of configurations determined by b and γ_0 .

Then

$$\text{wt}(d_n) = \underbrace{\text{mul}_{1,2}(\dots \text{mul}_{1,2}(\text{mul}_{2,1}(u, u)) \dots)}_n$$

with $u = \underbrace{\text{mul}_{1,2}(\dots \text{mul}_{1,2}(\text{mul}_{0,1}) \dots)}_n$ for each $n \geq 0$. Since $u = 2^n$, we have $\text{wt}(d_n) = 8^n$. Thus,

for each $\xi = \sigma(\delta^n(\alpha), \delta^n(\alpha))$, we have $\llbracket \mathcal{G} \rrbracket(\xi) = \text{wt}(d_n) = 8^n$. Since for each tree $\xi \notin \text{supp}(s)$ we have $D_{\mathcal{G}}(\xi) = \emptyset$ and hence $\llbracket \mathcal{G} \rrbracket(\xi) = \sum_{d \in \emptyset} \text{wt}(d) = 0$, we finally obtain that \mathcal{G} generates the weighted tree language s , i.e., $\llbracket \mathcal{G} \rrbracket = s$.

In the next lemma we will prove that the restriction to exactly one initial nonterminal has no effect on the generating power of (S, Σ, K) -rtg. This kind of initial state (or nonterminal) normal form has been proved for semiring-weighted tree automata (cf., e.g., [Bor04, p. 517] or [FV09, Thm.3.6]) using the following classical idea: take a new initial nonterminal and derive nondeterministically each right-hand side of an original rule that has an initial nonterminal at its left-hand side, and the weight of this new initial rule is the sum of the weights of the original initial rules. Generalizing this idea to the case of (S, Σ, K) -grammars requires to sum up operations and thus, to assume that the set of operations of K is closed under summation. We avoid this by employing a slightly more complicated construction.

⁽ⁱⁱ⁾ By “nc” we would like to express that there are “no chain rules”.

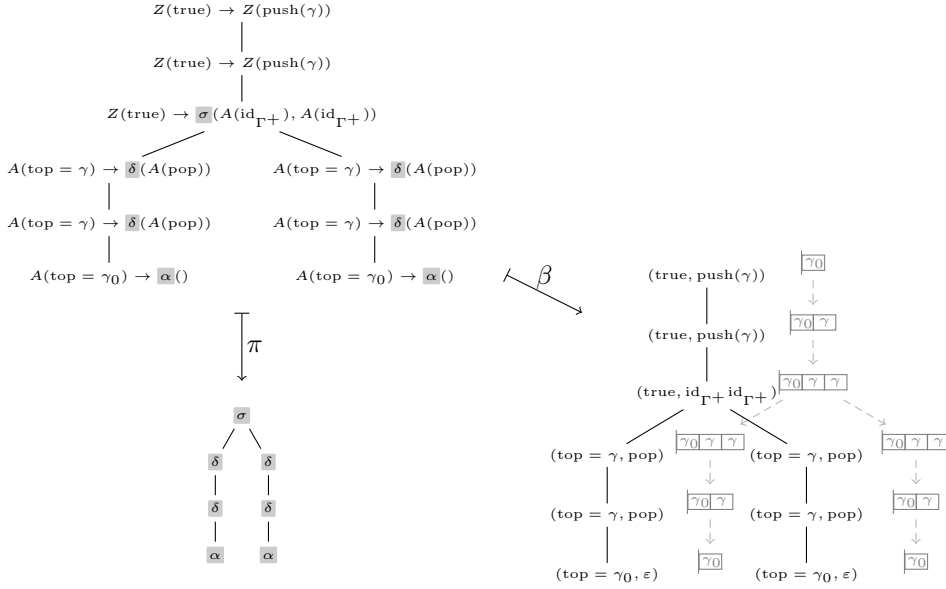


Fig. 3: The derivation tree $d_2 \in D_{\mathcal{G}}(\xi)$, the tree $\xi = \pi(d_2) = \sigma(\delta^2(\alpha), \delta^2(\alpha))$, and the $\Delta_{\mathcal{G}}$ -behaviour $b = \beta(d_2)$ together with the family $(c_w \mid w \in \text{pos}(b))$ of configurations determined by b and γ_0 .

Lemma 3.2. *For each (S, Σ, K) -rtg \mathcal{G} there is an (S, Σ, K) -rtg \mathcal{G}' such that $\llbracket \mathcal{G} \rrbracket = \llbracket \mathcal{G}' \rrbracket$ and \mathcal{G}' has exactly one initial nonterminal.*

Proof: Let $\mathcal{G} = (N, Z, R, \text{wt})$ be an (S, Σ, K) -rtg. We may assume that $Z \neq \emptyset$ since otherwise $\text{supp}(\llbracket \mathcal{G} \rrbracket) = \emptyset$ and thus our statement is obvious. We construct an (S, Σ, K) -rtg \mathcal{G}' that uses only one initial nonterminal Z_0 . Intuitively, \mathcal{G}' encodes the initial nonterminals of \mathcal{G} as second component in its nonterminals. Then any (Z_0, c_0) -derivation tree d' of \mathcal{G}' encodes an (A_0, c_0) -derivation tree d of \mathcal{G} for some initial nonterminal A_0 by keeping A_0 in the nodes of d' . In this way, for each tree ξ the sets $D_{\mathcal{G}}(\xi)$ and $D_{\mathcal{G}'}(\xi)$ are in a weight preserving one-to-one correspondence.

So let Z_0 be a symbol not in $(N \times Z) \cup \Sigma$. We construct the (S, Σ, K) -rtg $\mathcal{G}' = (N', Z_0, R', \text{wt}')$ such that $\llbracket \mathcal{G} \rrbracket = \llbracket \mathcal{G}' \rrbracket$ as follows. We let $N' = (N \times Z) \cup \{Z_0\}$. Moreover, for each $A_0 \in Z$,

- if $r = (A_0(p) \rightarrow \sigma(A_1(f_1), \dots, A_k(f_k))) \in R$,
then $r' = (Z_0(p) \rightarrow \sigma((A_1, A_0)(f_1), \dots, (A_k, A_0)(f_k))) \in R'$,
- if $r = (A_0(p) \rightarrow B(f)) \in R$, then $r' = (Z_0(p) \rightarrow (B, A_0)(f)) \in R'$,
- if $r = (A(p) \rightarrow \sigma(A_1(f_1), \dots, A_k(f_k))) \in R$,
then $r' = ((A, A_0)(p) \rightarrow \sigma((A_1, A_0)(f_1), \dots, (A_k, A_0)(f_k))) \in R'$,
- if $r = (A(p) \rightarrow B(f)) \in R$, then $r' = ((A, A_0)(p) \rightarrow (B, A_0)(f)) \in R'$,

and in each case we let $\text{wt}'(r') = \text{wt}(r)$.

We can prove that $\llbracket \mathcal{G} \rrbracket = \llbracket \mathcal{G}' \rrbracket$ as follows. For every $\xi \in T_{\Sigma}$ we define the mapping $g : D_{\mathcal{G}}(\xi) \rightarrow D_{\mathcal{G}'}(\xi)$ (note that $D_{\mathcal{G}'}(\xi) = D_{\mathcal{G}'}(Z_0, \xi, c_0)$). For this let $\xi = \sigma(\xi_1, \dots, \xi_k)$ for some $k \geq 0$ and $d \in D_{\mathcal{G}}(A_0, \xi, c_0)$ for some $A_0 \in Z$.

Then either

- (1) $d = r(d_1, \dots, d_k)$ for some rule $r = (A_0(p) \rightarrow \sigma(A_1(f_1), \dots, A_k(f_k)))$ and $d_i \in D_{\mathcal{G}}(A_i, \xi_i, f_i(c_0))$ for each $i \in [k]$ or
- (2) $d = r(d_1)$ for some rule $r = (A_0(p) \rightarrow B(f))$ and $d_1 \in D_{\mathcal{G}}(B, \xi, f(c_0))$.

In case (1) we let $g(d) = r'(d'_1, \dots, d'_k)$, where

- $r' = (Z_0(p) \rightarrow \sigma((A_1, A_0)(f_1), \dots, (A_k, A_0)(f_k)))$ and
- we obtain d'_i from d_i by replacing every nonterminal A by (A, A_0) .

In case (2) let $g(d) = r'(d'_1)$, where

- $r' = (Z_0(p) \rightarrow (B, A_0)(f))$ and
- we obtain d'_1 from d_1 as in case (1).

By the construction of \mathcal{G}' it should be clear that g is a bijection and that $\text{wt}(d) = \text{wt}'(g(d))$ for each $d \in D_{\mathcal{G}}(\xi)$. Then it is easy to see that $\llbracket \mathcal{G} \rrbracket = \llbracket \mathcal{G}' \rrbracket$. \square

Special cases. Here we define and analyze four special cases of (S, Σ, K) -regular weighted tree languages: (i) the unweighted case, (ii) the storage-free case, (iii) the unweighted and storage-free case, and (iv) the string case.

(i) $K = \mathbb{B}$: A *regular tree grammar over Σ with storage S* (for short: (S, Σ) -rtg) is an (S, Σ, \mathbb{B}) -rtg. In the specification of an (S, Σ) -rtg we assume w.l.o.g. that each k -ary rule is mapped to $\text{all}_{k,1}$, and hence we drop the weight function wt .

Let \mathcal{G} be an (S, Σ) -rtg. Since $\text{wt}(d) = 1$ for each $d \in D_{\mathcal{G}}(\xi)$, we have that $\text{supp}(\llbracket \mathcal{G} \rrbracket) = \{\xi \in T_{\Sigma} \mid D_{\mathcal{G}}(\xi) \neq \emptyset\}$. We call this set the *tree language generated by \mathcal{G}* and denote it by $\mathcal{L}(\mathcal{G})$. Moreover, we say that \mathcal{G} is *unambiguous* if for each $\xi \in T_{\Sigma}$ we have $|D_{\mathcal{G}}(\xi)| \leq 1$.

A tree language $L \subseteq T_{\Sigma}$ is called (S, Σ) -*regular* if there is an (S, Σ) -rtg \mathcal{G} such that $\mathcal{L}(\mathcal{G}) = L$. We denote the class of all (S, Σ) -regular tree languages by $\text{Reg}(S, \Sigma)$.

We note that each (S, Σ) -rtg \mathcal{G} is an $\text{RT}(S)$ -transducer $\mathcal{M}_{\mathcal{G}}$ as defined in [EV86, Def. 3.3] and in [EV88, Def. 3.3] where RT means the class of regular tree grammars [Bra69, GS84, GS97]; the range of the translation induced by $\mathcal{M}_{\mathcal{G}}$ is $\mathcal{L}(\mathcal{G})$. Vice versa, each $\text{RT}(S)$ -transducer \mathcal{M} (with the definition of storage type of the present paper) is an (S, Σ) -rtg if the Boolean expressions occurring in the rules of \mathcal{M} are predicates (and not arbitrary Boolean expressions over P).

Moreover, by [EV88, Thm. 6.15], we have that $\text{Reg}(P^n, \Sigma)$ is the class of level- n OI-tree languages for each $n \geq 0$ (denoted by n - T in [EV88]; also cf. [DG81]). This hierarchy has been intensively investigated in [Dam82, DG81]. The classes for $n = 0$ and $n = 1$ are the class of regular tree languages and of OI context-free tree languages, respectively (cf. [EV88, Prop. 4.4]).

We recall from [Dam82, Thm. 7.8] that the emptiness problem for level- n OI-tree languages is decidable (for each $n \geq 0$).

(ii) $S = \text{TRIV}$: A K -*weighted regular tree grammar over Σ* (for short: (Σ, K) -rtg) is a (TRIV, Σ, K) -rtg. In the rules of a (Σ, K) -rtg we drop the always true predicate from the left-hand side and the identity instruction from the right-hand side. A weighted tree language $s : T_{\Sigma} \rightarrow K$ is called (Σ, K) -*regular* if there is a (Σ, K) -rtg \mathcal{G} such that $s = \llbracket \mathcal{G} \rrbracket$. We denote the class of all (Σ, K) -regular tree languages by $\text{Reg}(\Sigma, K)$.

In [FSV12, Sec. 2.6], weighted tree automata were defined over M -monoids which need not be absorptive and complete. However, it is obvious that each chain-free (Σ, K) -rtg corresponds to a weighted tree

automata over Σ and K (in the sense of [FSV12, Sec. 2.6]), and vice versa. Thus we obtain the following characterization.

Observation 3.3. $\text{Reg}_{\text{nc}}(\Sigma, K)$ is the class of recognizable tree series over Σ and K defined in [FSV12].

Now let, additionally, K be an arbitrary complete semiring and M_K be the M-monoid associated with K . Then each chain-free (Σ, M_K) -rtg is essentially a semiring-weighted tree automaton in the sense of [FV09], and vice versa.

(iii) $K = \mathbb{B}$ and $S = \text{TRIV}$: A *regular tree grammar* over Σ (for short: Σ -rtg) is a $(\text{TRIV}, \Sigma, \mathbb{B})$ -rtg. A tree language $L \subseteq T_\Sigma$ is called Σ -regular if there is a Σ -rtg \mathcal{G} such that $\mathcal{L}(\mathcal{G}) = L$. We denote the class of all Σ -regular tree languages by $\text{Reg}(\Sigma)$.

Clearly, the class $\text{Reg}(\Sigma)$ coincides with the class of recognizable (or: regular) tree languages over Σ [GS84]. Since finite-state tree automata can be determinized [GS84, Thm. 2.2.6], we can always assume that a Σ -rtg \mathcal{G} is unambiguous.

(iv) String case: For semirings as weight structures, it is demonstrated in [FV09, p. 324] that, roughly speaking, (a) weighted string automata (without storage) are the same as (b) weighted tree automata over a monadic ranked alphabet. We can generalize this relation to (a') weighted string automata (including ε -transitions) with arbitrary storage and (b') weighted regular tree grammars over a monadic ranked alphabet (with chain rules) with arbitrary storage. Moreover, in the tree case we can use the M-monoid associated to the semiring as weight structure.

3.2 Elimination of finite storage types in the Boolean case

It is easy to see that each (S, Σ) -rtg can be simulated by a Σ -rtg assuming that S is a finite storage type.

Lemma 3.4. *Let S be finite. For each (S, Σ) -rtg \mathcal{G} there is a Σ -rtg \mathcal{G}' such that $\mathcal{L}(\mathcal{G}) = \mathcal{L}(\mathcal{G}')$. Moreover, if \mathcal{G} is chain-free (resp., unambiguous), then so is \mathcal{G}' .*

Proof: Let $\mathcal{G} = (N, Z, R)$ be an (S, Σ) -rtg. For the construction of a Σ -rtg \mathcal{G}' , we simply encode the finitely many configurations into the new set of nonterminals. Formally, we construct $\mathcal{G}' = (N \times C, Z', R')$ such that $Z' = Z \times \{c_0\}$ and

- if $r = (A(p) \rightarrow \sigma(A_1(f_1), \dots, A_k(f_k)))$ is a rule in R , then for each $c \in C$ such that $p(c) = 1$ and $f_1(c), \dots, f_k(c)$ are defined, the rule

$$r' = ((A, c) \rightarrow \sigma((A_1, f_1(c)), \dots, (A_k, f_k(c))))$$

is in R' ,

- if $r = (A(p) \rightarrow B(f))$ is in R , then for each $c \in C$ such that $p(c) = 1$ and $f(c)$ is defined, the rule

$$r' = ((A, c) \rightarrow (B, f(c)))$$

is in R' .

It is clear that \mathcal{G}' is chain-free (resp., unambiguous) if \mathcal{G} is so.

For each $\xi \in T_\Sigma$, we define the mapping $\theta : D_{\mathcal{G}}(\xi) \rightarrow D_{\mathcal{G}'}(\xi)$ as follows. Let $d \in D_{\mathcal{G}}(\xi)$ and $(c_w \mid w \in \text{pos}(\beta(d)))$ be the family of configurations determined by the $(\Delta_{\mathcal{G}}, c_0)$ -behaviour $\beta(d)$ and c_0 . We recall that β is a tree relabeling and hence $\text{pos}(\beta(d)) = \text{pos}(d)$. Then we define $\text{pos}(\theta(d)) = \text{pos}(d)$. Moreover, for each $w \in \text{pos}(d)$,

- if $d(w) = (A(p) \rightarrow \sigma(A_1(f_1), \dots, A_k(f_k)))$, then we define

$$(\theta(d)(w)) = ((A, c_w) \rightarrow \sigma((A_1, c_{w1}), \dots, (A_k, c_{wk}))) ,$$

- if $d(w) = (A(p) \rightarrow B(f))$, then we define

$$(\theta(d)(w)) = ((A, c_w) \rightarrow (B, c_{w1})) .$$

Due to the construction, it is clear that $\theta(d) \in D_{\mathcal{G}'}(\xi)$. Moreover, θ is surjective. However, it is not necessarily injective, as can be seen if, e.g., two rules $A(p) \rightarrow B(f_1)$ and $A(p) \rightarrow B(f_2)$ are in R and there exists a $c \in C$ with $p(c) = 1$ and $f_1(c) = f_2(c)$.

Then it follows that $D_{\mathcal{G}}(\xi) \neq \emptyset$ iff $D_{\mathcal{G}'}(\xi) \neq \emptyset$ (where the surjectivity of θ is needed in the “if”-implication). Thus $\mathcal{L}(\mathcal{G}) = \mathcal{L}(\mathcal{G}')$. \square

4 Support

In this section we prove that, for each complete, zero-sum free, and commutative strong bimonoid K (cf. Section 2.2), the supports of (S, Σ, M_K) -regular weighted tree languages are (S, Σ) -regular tree languages. We follow the same approach as in [Kir11], where the support theorem was proved for weighted string automata without ε -transitions. We generalize this approach in a straightforward way to the case of weighted regular tree grammars with storage (including chain rules, which correspond to ε -transitions in string automata).

First we recall some definitions. Let $(K, \cdot, 1)$ be a monoid. An element $0 \in K$ with $0 \neq 1$ is called a *zero* if $a \cdot 0 = 0 \cdot a = 0$ for every $a \in K$. For every $a_1, \dots, a_n \in K$, we let $\langle a_1, \dots, a_n \rangle$ denote the smallest submonoid of K containing a_1, \dots, a_n . For every $a \in K$ and $A \subseteq K$, we let $a \cdot A = \{a \cdot a' \mid a' \in A\}$.

As defined by Kirsten [Kir11], the *zero generation problem (ZGP)* for a monoid $(K, \cdot, 1)$ with zero 0 is, given two integers $m, n \in \mathbb{N}$ and elements $a_1, \dots, a_m, a'_1, \dots, a'_n \in K$, the question whether $0 \in a_1 \cdot \dots \cdot a_m \cdot \langle a'_1, \dots, a'_n \rangle$. For instance, if K is idempotent and commutative, then it has a decidable ZGP, because in this case the set $a_1 \cdot \dots \cdot a_m \cdot \langle a'_1, \dots, a'_n \rangle$ is finite.

For every tuples $\bar{z} = (z_1, \dots, z_n) \in \mathbb{N}^n$ and $\bar{y} = (y_1, \dots, y_n) \in \mathbb{N}^n$, we define $\bar{z} \leq \bar{y}$ if $z_i \leq y_i$ for all $i = 1, \dots, n$. Obviously, \leq is a partial order on \mathbb{N}^n . For a subset $M \subseteq \mathbb{N}^n$, an element $\bar{z} \in M$ is *minimal in M* if $\bar{y} \leq \bar{z}$ implies $\bar{y} = \bar{z}$ for every $\bar{y} \in M$. We denote by $\min(M)$ the set of all minimal elements in M . By Dickson’s lemma [Dic13], $\min(M)$ is finite (cf. [Kir11, Lm. 2.1] and [KR08]). For every $\bar{z} \in \mathbb{N}^n$ and $k \in \mathbb{N}$, we define the *cut of \bar{z} and k* , denoted by $\lfloor \bar{z} \rfloor_k$, to be the vector $\lfloor \bar{z} \rfloor_k \in \mathbb{N}^n$ with

$$(\lfloor \bar{z} \rfloor_k)_i = \min\{z_i, k\}$$

as i -th component for each $i = 1, \dots, n$.

Now let $(K, \cdot, 1)$ be a commutative monoid with a zero 0 . For every $a \in K$ and $z \in \mathbb{N}$, we let a^z be the product of z many a s. In particular, $a^0 = 1$.

Fix a tuple $\bar{a} = (a_1, \dots, a_n) \in K^n$. We define the mapping

$$\llbracket \cdot \rrbracket_{\bar{a}} : \mathbb{N}^n \rightarrow K \text{ by letting } \llbracket \bar{z} \rrbracket_{\bar{a}} = a_1^{z_1} \cdot \dots \cdot a_n^{z_n}$$

for each $\bar{z} = (z_1, \dots, z_n) \in \mathbb{N}^n$. We put $\llbracket 0 \rrbracket_{\bar{a}}^{-1} = \{\bar{z} \in \mathbb{N}^n \mid \llbracket \bar{z} \rrbracket_{\bar{a}} = 0\}$. Note that $(0, \dots, 0) \notin \llbracket 0 \rrbracket_{\bar{a}}^{-1}$, because $a_1^0 \cdot \dots \cdot a_n^0 = 1$. Clearly, if $\bar{z}, \bar{y} \in \mathbb{N}^n$ with $\llbracket \bar{z} \rrbracket_{\bar{a}} = 0$ and $\bar{z} \leq \bar{y}$, then also $\llbracket \bar{y} \rrbracket_{\bar{a}} = 0$. Since

$\min(\llbracket 0 \rrbracket_{\bar{a}}^{-1})$ is finite, there is a smallest number $m \in \mathbb{N}$ satisfying $\min(\llbracket 0 \rrbracket_{\bar{a}}^{-1}) \subseteq \{0, \dots, m\}^n$, and we define $\text{dg}(\bar{a}) = m$.

We state the following obvious connection between the defined concepts.

Observation 4.1. Let $\bar{a} = (a_1, \dots, a_n)$ be an element of K^n . Then the following three statements are equivalent:

1. $\llbracket 0 \rrbracket_{\bar{a}}^{-1} = \emptyset$.
2. $\text{dg}(\bar{a}) = 0$.
3. The submonoid $(\langle a_1, \dots, a_n \rangle, \cdot, 1)$ is zero-divisor free, i.e., for each $a, b \in \langle a_1, \dots, a_n \rangle$ we have that $a \cdot b = 0$ implies that $a = 0$ or $b = 0$.

Moreover, we recall from [Kir11] the following statements.

Lemma 4.2. ([Kir11, Lm. 4.1]) For each $\bar{z} \in \mathbb{N}^n$, we have $\llbracket \bar{z} \rrbracket_{\bar{a}} = 0$ iff $\llbracket \lfloor \bar{z} \rfloor_{\text{dg}(\bar{a})} \rrbracket_{\bar{a}} = 0$.

Lemma 4.3. ([Kir11, Lm. 4.2]) Let $(K, \cdot, 1)$ be a commutative monoid with a zero, $n \in \mathbb{N}$, and $\bar{a} \in K^n$. If the ZGP for K is decidable, then $\text{dg}(\bar{a})$ is effectively computable.

Now we can prove the main theorem of this section. We follow the proof and the construction of the corresponding results [Kir11, Thm. 3.1] for weighted automata over semirings and [Göt16, Thm. 4.6] for weighted tree automata over tv-monoids (also cf. [DH15] for a similar result for weighted unranked tree automata over bimonoids). Also we provide the correctness proof of the construction.

Theorem 4.4. Let K be a complete, zero-sum free, and commutative strong bimonoid and M_K be the M -monoid associated with K .

1. (a) For every (S, Σ, M_K) -rtg \mathcal{G} , there is an (S, Σ) -rtg \mathcal{G}' such that $L(\mathcal{G}') = \text{supp}(\llbracket \mathcal{G} \rrbracket)$. (b) Moreover, if $(K, \cdot, 1)$ has a decidable ZGP, then \mathcal{G}' can be constructed effectively.
2. Assume that $|\Sigma^{(1)}| \geq 2$. If there is an effective construction of a Σ -rtg which generates $\text{supp}(\llbracket \mathcal{G} \rrbracket)$ from any given (Σ, M_K) -rtg \mathcal{G} , then $(K, \cdot, 1)$ has a decidable ZGP.

Proof: First we prove 1(a). Let $\mathcal{G} = (N, Z, R, \text{wt})$ be an (S, Σ, M_K) -rtg. By Lemma 3.2 we may assume that $Z \in N$. Since the weight of each rule is an operation $\text{mul}_{k,a}$ for some $k \in \mathbb{N}$ and $a \in K$, the weight of each derivation tree is the (bimonoid) multiplication of the a 's appearing in the rules in that tree. Since K is commutative, this amounts to counting how many times each such a occurs.

Formally, we define the mapping $\text{wt}_K: R \rightarrow K$ such that, for each $r \in R$, we let $\text{wt}_K(r) = b$ if $\text{wt}(r) = \text{mul}_{k,b}$ for some $k \in \mathbb{N}$. We let $W = \text{wt}_K(R)$ be the set comprising all elements of K each of which corresponds to the weight of some rule of \mathcal{G} . Let $\bar{a} = (a_1, \dots, a_n) \in K^n$ be an enumeration of W . Then the weight of each derivation tree can be written in the form $\llbracket \bar{y} \rrbracket_{\bar{a}}$ for some $\bar{y} \in \mathbb{N}^n$. Moreover, let $T = \{0, \dots, \text{dg}(\bar{a})\}$. We define the mapping

$$\begin{aligned} \oplus: T^n \times W &\rightarrow T^n && \text{by letting} \\ \bar{z} \oplus a_i &= \lfloor \bar{y} \rfloor_{\text{dg}(\bar{a})} && \text{for } \bar{y} = (y_1, \dots, y_n) \in \mathbb{N}^n \text{ with } y_i = z_i + 1 \text{ and} \\ &&& y_j = z_j \text{ for each } j \neq i, \end{aligned}$$

and the mapping

$$\begin{aligned} \bar{\oplus}: T^n \times T^n &\rightarrow T^n && \text{by letting} \\ \bar{z} \bar{\oplus} \bar{z}' &= \lfloor \bar{y} \rfloor_{\text{dg}(\bar{a})} && \text{for } \bar{y} = (y_1, \dots, y_n) \in \mathbb{N}^n \text{ with } y_i = z_i + z'_i \text{ for all } i \in [n]. \end{aligned}$$

With these mappings, we will be able to count, up to the threshold of $\text{dg}(\bar{a})$, the number of occurrences of the a_i 's in the derivation trees of \mathcal{G} .

Now we define an (unweighted) (S, Σ) -rtg \mathcal{G}' which simulates the derivations of \mathcal{G} and counts the elements of K corresponding to the weights which occur in derivation trees as follows. Let $\mathcal{G}' = (N', Z', R')$ such that

- $N' = N \times T^n$,
- $Z' = \{(Z, \bar{z}) \mid \bar{z} \in T^n, \llbracket \bar{z} \rrbracket_{\bar{a}} \neq 0\}$,
- R' is defined as follows:
 - Let $r = (A(p) \rightarrow \alpha)$ be a rule in R . Then $((A, \bar{z})(p) \rightarrow \alpha)$ is in R' where $\bar{z} = (0, \dots, 0) \oplus \text{wt}_K(r)$.
 - Let $r = (A(p) \rightarrow \sigma(A_1(f_1), \dots, A_k(f_k)))$ be a rule in R . Moreover, let $\bar{z}_1, \dots, \bar{z}_k \in T^n$. Then $((A, \bar{z})(p) \rightarrow \sigma((A_1, \bar{z}_1)(f_1), \dots, (A_k, \bar{z}_k)(f_k)))$ is in R' where $\bar{z} = (\bar{z}_1 \oplus \dots \oplus \bar{z}_k) \oplus \text{wt}_K(r)$.
 - Let $r = (A(p) \rightarrow B(f))$ be a rule in R . Moreover, let $\bar{z}_1 \in T^n$. Then $((A, \bar{z})(p) \rightarrow (B, \bar{z}_1)(f)) \in R'$ where $\bar{z} = \bar{z}_1 \oplus \text{wt}_K(r)$.

Next we prove the equality $L(\mathcal{G}') = \text{supp}(\llbracket \mathcal{G} \rrbracket)$.

First, we prove that $\text{supp}(\llbracket \mathcal{G} \rrbracket) \subseteq L(\mathcal{G}')$. For this, we show the following Statement:

(*) For each $l \in \mathbb{N}_+$, $\xi \in T_\Sigma$, $A \in N$, $c \in C$, and $d \in D_{\mathcal{G}}(A, \xi, c)$: if $\text{size}(d) = l$, then there are $\bar{y} \in \mathbb{N}^n$ and $d' \in D_{\mathcal{G}'}((A, \llbracket \bar{y} \rrbracket_{\text{dg}(\bar{a})}), \xi, c)$ such that $\text{size}(d') = l$ and $\llbracket \bar{y} \rrbracket_{\bar{a}} = \text{wt}(d)$.

We prove Statement (*) by strong induction on l .

Let $l = 1$. Then $d \in D_{\mathcal{G}}(A, \xi, c)$ has to be of the form $r = (A(p) \rightarrow \alpha)$ for some $r \in R$ such that $p(c) = \text{true}$. Then $\alpha \in \Sigma^{(0)}$ and $\xi = \alpha$. Clearly, $\text{wt}(d) = \text{wt}_K(r)$. Now let $\bar{y} = (y_1, \dots, y_n)$ be the tuple in \mathbb{N}^n such that, for each $i \in [n]$, $y_i = 1$ if $a_i = \text{wt}_K(r)$, and $y_i = 0$ otherwise. Obviously, $\llbracket \bar{y} \rrbracket_{\bar{a}} = \text{wt}_K(r) = \text{wt}(d)$. Moreover, by construction, the rule $r' = ((A, \llbracket \bar{y} \rrbracket_{\text{dg}(\bar{a})})(p) \rightarrow \alpha)$ is in R' . Since $p(c) = \text{true}$, r' is in $D_{\mathcal{G}'}((A, \llbracket \bar{y} \rrbracket_{\text{dg}(\bar{a})}), \xi, c)$.

Now, let $l > 1$ and assume that Statement (*) holds for all $l' \in \mathbb{N}$ with $l' < l$. We consider two cases.

Case 1: There exist some $r = (A(p) \rightarrow B(f))$ in R and $d_1 \in D_{\mathcal{G}}(B, \xi, f(c))$ such that $d = r(d_1)$. Then $p(c) = \text{true}$ and $f(c)$ is defined. Moreover, $l = \text{size}(d_1) + 1$ and $\text{wt}(d) = \text{wt}(d_1) \cdot \text{wt}_K(r)$. By IH there exist $\bar{y}_1 = (y_{1,1}, \dots, y_{1,n})$ in \mathbb{N}^n and $d'_1 \in D_{\mathcal{G}'}((B, \llbracket \bar{y}_1 \rrbracket_{\text{dg}(\bar{a})}), \xi, f(c))$ such that $\text{size}(d'_1) = \text{size}(d_1)$ and $\llbracket \bar{y}_1 \rrbracket_{\bar{a}} = \text{wt}(d_1)$. Then let $\bar{y} = (y_1, \dots, y_n)$ in \mathbb{N}^n such that, for each $i \in [n]$, $y_i = y_{1,i} + 1$ if $a_i = \text{wt}_K(r)$, and $y_i = y_{1,i}$ otherwise. Obviously, $\llbracket \bar{y} \rrbracket_{\bar{a}} = \text{wt}(d)$. Moreover, by construction, there is a rule $r' = ((A, \llbracket \bar{y} \rrbracket_{\text{dg}(\bar{a})})(p) \rightarrow (B, \llbracket \bar{y}_1 \rrbracket_{\text{dg}(\bar{a})})(f))$ in R' . Since $p(c) = \text{true}$ and $f(c)$ is defined, $d' = r'(d'_1)$ is an element in $D_{\mathcal{G}'}((A, \llbracket \bar{y} \rrbracket_{\text{dg}(\bar{a})}), \xi, c)$ with $\text{size}(d') = l$.

Case 2: There exist some $k \geq 1$, $r = (A(p) \rightarrow \sigma(B_1(f_1), \dots, B_k(f_k)))$ in R , and $d_i \in D_{\mathcal{G}}(B_i, \xi_i, f_i(c))$ for each $i \in [k]$ such that $d = r(d_1, \dots, d_k)$. Hence, $p(c) = \text{true}$ and $f_i(c)$ is defined for each $i \in [k]$. Then $\xi = \sigma(\xi_1, \dots, \xi_k)$ and $l = \text{size}(d_1) + \dots + \text{size}(d_k) + 1$. Moreover, $\text{wt}(d) = \text{wt}(d_1) \cdot \dots \cdot \text{wt}(d_k) \cdot \text{wt}_K(r)$. For each $i \in [k]$, by IH, there exist $\bar{y}_i = (y_{i,1}, \dots, y_{i,n})$ in \mathbb{N}^n and $d'_i \in D_{\mathcal{G}'}((B_i, \llbracket \bar{y}_i \rrbracket_{\text{dg}(\bar{a})}), \xi_i, f_i(c))$ such that $\text{size}(d'_i) = \text{size}(d_i)$ and $\llbracket \bar{y}_i \rrbracket_{\bar{a}} = \text{wt}(d_i)$. Then let $\bar{y} = (y_1, \dots, y_n)$ in \mathbb{N}^n such that, for each $j \in [n]$, $y_j = y'_j + 1$ if $a_j = \text{wt}_K(r)$, and $y_j = y'_j$ otherwise, where $\bar{y}' = \bar{y}_1 + \dots + \bar{y}_k$ with $+$ denoting the pointwise addition of tuples. It is not hard to see that $\llbracket \bar{y} \rrbracket_{\bar{a}} = \text{wt}(d)$. Furthermore, by construction there exists a rule $r' = ((A, \llbracket \bar{y} \rrbracket_{\text{dg}(\bar{a})})(p) \rightarrow \sigma((B_1, \llbracket \bar{y}_1 \rrbracket_{\text{dg}(\bar{a})})(f_1), \dots, (B_k, \llbracket \bar{y}_k \rrbracket_{\text{dg}(\bar{a})})(f_k)))$ in R' . Since $p(c) = \text{true}$ and $f_i(c)$ is defined for each $i \in [k]$, $d' = r'(d'_1, \dots, d'_k)$ is an element in $D_{\mathcal{G}'}((A, \llbracket \bar{y} \rrbracket_{\text{dg}(\bar{a})}), \xi, c)$ with $\text{size}(d') = l$.

This finishes the proof of Statement (*).

Now let $\xi \in \text{supp}(\llbracket \mathcal{G} \rrbracket)$. Then there is a derivation tree $d \in D_{\mathcal{G}}(Z, \xi, c_0)$ with $\text{wt}(d) \neq 0$. By Statement (*) there are $\bar{y} \in \mathbb{N}^n$ and $d' \in D_{\mathcal{G}'}((Z, \lfloor \bar{y} \rfloor_{\text{dg}(\bar{a})}), \xi, c_0)$ such that $\llbracket \bar{y} \rrbracket_{\bar{a}} = \text{wt}(d)$. Since $\llbracket \bar{y} \rrbracket_{\bar{a}} \neq 0$, also $\llbracket \lfloor \bar{y} \rfloor_{\text{dg}(\bar{a})} \rrbracket_{\bar{a}} \neq 0$ (by Lemma 4.2). Thus, $(Z, \lfloor \bar{y} \rfloor_{\text{dg}(\bar{a})}) \in Z'$ and ξ is in $L(\mathcal{G}')$.

Secondly, we prove that $L(\mathcal{G}') \subseteq \text{supp}(\llbracket \mathcal{G} \rrbracket)$. For this, we show the following Statement:

(**) For each $l \in \mathbb{N}_+$, $\xi \in T_{\Sigma}$, $A \in N$, $\bar{z} \in T^n$, $c \in C$, and $d' \in D_{\mathcal{G}'}((A, \bar{z}), \xi, c)$: if $\text{size}(d') = l$, then there are $\bar{y} \in \mathbb{N}^n$ and $d \in D_{\mathcal{G}}(A, \xi, c)$ such that $\text{size}(d) = l$, $\llbracket \bar{y} \rrbracket_{\bar{a}} = \text{wt}(d)$ and $\bar{z} = \lfloor \bar{y} \rfloor_{\text{dg}(\bar{a})}$.

Statement (**) can be proved by strong induction on l . Since the proof is very similar to that of Statement (*), we drop it here.

Now let $\xi \in L(\mathcal{G}')$. Then there is a derivation tree $d' \in D_{\mathcal{G}'}((Z, \bar{z}), \xi, c_0)$ for some $\bar{z} \in T^n$ with $\llbracket \bar{z} \rrbracket_{\bar{a}} \neq 0$. By Statement (**) there are $\bar{y} \in \mathbb{N}^n$ and $d \in D_{\mathcal{G}}(Z, \xi, c_0)$ such that $\llbracket \bar{y} \rrbracket_{\bar{a}} = \text{wt}(d)$ and $\bar{z} = \lfloor \bar{y} \rfloor_{\text{dg}(\bar{a})}$. Since $\llbracket \bar{z} \rrbracket_{\bar{a}} \neq 0$ also $\llbracket \bar{y} \rrbracket_{\bar{a}} \neq 0$ (by Lemma 4.2). Thus, since K is zero-sum free, ξ is in $\text{supp}(\llbracket \mathcal{G} \rrbracket)$.

The proof of 1(b) follows from the fact that if the ZGP for $(K, \cdot, 1)$ is decidable, then by Lemma 4.3 we can compute the number $\text{dg}(\bar{a})$ and hence construct \mathcal{G}' effectively.

For the proof of 2, we note the following. Due to the conditions $|\Sigma^{(1)}| \geq 2$, the weighted finite automaton constructed in the corresponding part of the proof of [Kir11, Thm. 3.1.] can be simulated by an (Σ, M_K) -rtg, hence that proof can be adapted to this setting. \square

We note that the construction in the proof of Theorem 4.4(1) becomes very simple if K is zero-divisor free. Then, by Observation 4.1, $\text{dg}(\bar{a}) = 0$ for every \bar{a} , and hence N' is essentially N (and the same holds for Z' and Z). Thus, the rules of \mathcal{G}' are obtained from those of \mathcal{G} simply by dropping the weights.

Also we note that the first part of Theorem 4.4(1) (choosing $S = \text{TRIV}$) provides a sufficient condition for the recognizability of support tree languages, which is different from the following well-known fact: If K is a zero-sum free and zero-divisor free (not necessarily commutative) semiring, then the support of $\llbracket \mathcal{G} \rrbracket$ for any (Σ, K) -rtg \mathcal{G} is a recognizable tree language (cf., e.g., [FV09, Thm. 3.12]).

In the next example we illustrate the construction of Theorem 4.4.

Example 4.5. We consider the strong bimonoid $(K, \max, \cdot, 0, 1)$ where $K = \{i \in \mathbb{N} \mid 0 \leq i \leq 9\}$, \max is extended to maximum over countable index sets in the obvious way, and \cdot is the multiplication of natural numbers modulo 9; thus, e.g., $3 \cdot 4 = 12 \pmod{9} = 3$. Indeed, K is complete, zero-sum free, and commutative. We note that $(K, \cdot, 1)$ has a decidable ZGP.

We consider the ranked alphabet $\Sigma = \{\gamma^{(1)}, \alpha^{(0)}, \beta^{(0)}\}$ and the (Σ, M_K) -rtg $\mathcal{G} = (\{A\}, A, R, \text{wt})$ where the rules in R and the corresponding weights are:

$$\begin{aligned} r_1: A &\rightarrow \gamma(A), \text{wt}(r_1) = \text{mul}_{1,2} \\ r_2: A &\rightarrow \alpha, \text{wt}(r_2) = \text{mul}_{1,3} \\ r_3: A &\rightarrow \beta, \text{wt}(r_3) = \text{mul}_{1,6} \end{aligned}$$

By direct inspection of \mathcal{G} we can easily see that $\text{supp}(\llbracket \mathcal{G} \rrbracket) = T_{\Sigma}$.

Using the notations in the proof of Theorem 4.4, we have $W = \{2, 3, 6\}$. We let $\bar{a} = (2, 3, 6)$. Then the set $\llbracket 0 \rrbracket_{\bar{a}}^{-1}$ contains, e.g., the following three elements: $(0, 2, 0)$, $(0, 0, 2)$, and $(0, 1, 1)$. Indeed,

$$\min(\llbracket 0 \rrbracket_{\bar{a}}^{-1}) = \{(0, 2, 0), (0, 0, 2), (0, 1, 1)\}.$$

The degree of \bar{a} is $\text{dg}(\bar{a}) = 2$, because $\min(\llbracket 0 \rrbracket_{\bar{a}}^{-1}) \subseteq \{0, 1, 2\}^3$ and $\min(\llbracket 0 \rrbracket_{\bar{a}}^{-1}) \not\subseteq \{0, 1\}^3$. Hence $T = \{0, 1, 2\}$.

Applying the construction of Theorem 4.4, we obtain the following Σ -rtg $\mathcal{G}' = (N', Z', R')$ with

- $N' = \{A\} \times T^3$; we abbreviate $(A, (n_1, n_2, n_3))$ by (n_1, n_2, n_3) ;
- $Z' = \{(0, 0, 0)\} \cup \{(n, 1, 0) \mid 0 \leq n \leq 2\} \cup \{(n, 0, 1) \mid 0 \leq n \leq 2\}$;
- R' contains the following eight useful rules:

$$\begin{array}{ll}
(0, 1, 0) \rightarrow \alpha & (0, 0, 1) \rightarrow \beta \\
(1, 1, 0) \rightarrow \gamma((0, 1, 0)) & (1, 0, 1) \rightarrow \gamma((0, 0, 1)) \\
(2, 1, 0) \rightarrow \gamma((1, 1, 0)) & (2, 0, 1) \rightarrow \gamma((1, 0, 1)) \\
(2, 1, 0) \rightarrow \gamma((2, 1, 0)) & (2, 0, 1) \rightarrow \gamma((2, 0, 1)) .
\end{array}$$

Hence also $L(\mathcal{G}') = T_\Sigma$. □

From Theorem 4.4(1) and the fact, that the emptiness problem of iterated pushdown tree automata is decidable [Dam82, Thm. 7.8], we obtain the following result:

Corollary 4.6. *Let K be a complete, zero-sum free, and commutative strong bimonoid with a decidable ZGP and M_K be the complete M -monoid associated with K . Moreover, let $s: T_\Sigma \rightarrow M_K$ be $(\mathbb{P}^n, \Sigma, M_K)$ -recognizable for some $n \in \mathbb{N}$. Then it is decidable whether $\text{supp}(s) = \emptyset$.*

As particular case of Corollary 4.6, let us consider a complete lattice $(V, \vee, \wedge, 0, 1)$ with 0 and 1 as smallest and largest elements, respectively. It is clear that V is a complete, zero-sum free, commutative, and \wedge -idempotent strong bimonoid. Thus, the ZGP of the monoid $(V, \wedge, 1)$ is decidable. Then, Corollary 4.6 shows that, for every complete lattice $(V, \vee, \wedge, 0, 1)$ and $(\mathbb{P}^n, \Sigma, M_V)$ -recognizable weighted tree language s , it is decidable whether $\text{supp}(s) = \emptyset$.

Moreover, combining Theorem 4.4(1) and Lemma 3.4 we obtain the following result.

Corollary 4.7. *Let S be finite and K a complete, zero-sum free, and commutative strong bimonoid such that the ZGP of $(K, \cdot, 1)$ is decidable. For every (S, Σ, M_K) -rtg \mathcal{G} , a Σ -rtg can effectively be constructed which generates $\text{supp}(\llbracket \mathcal{G} \rrbracket)$.*

We finish this section by showing that, in general, the empty-support problem is not decidable. Let us recall that \mathbb{B} is the Boolean M -monoid.

Lemma 4.8. *There is a storage type S and a ranked alphabet Σ such that the empty-support problem for (S, Σ, \mathbb{B}) -rtg is undecidable.*

Proof: We reduce this decidability problem to the Post-correspondence problem. Let $\text{PCP} = (u_1, v_1) \dots (u_n, v_n)$ be a Post-correspondence problem [Pos46] where u_i, v_i are non-empty strings over some alphabet Δ for each $i \in [n]$. We construct the storage type $S_{\text{PCP}} = (\Delta^* \times \Delta^*, P, F, (\varepsilon, \varepsilon))$ with $P = \{\text{equal}, \text{true}_{\Delta^* \times \Delta^*}\}$ and $F = [n]$ where for each $(u, v) \in \Delta^* \times \Delta^*$ we define

- $\text{equal}((u, v)) = 1$ iff $u = v$

and for each $i \in [n]$:

- $i((u, v)) = (uu_i, vv_i)$.

We let Σ be the ranked alphabet with $\Sigma = \Sigma^{(0)} = \{\#\}$. We construct the $(S_{\text{PCP}}, \Sigma, \mathbb{B})$ -rtg $\mathcal{G} = (\{Z\}, Z, R, \text{wt})$ with the following rules: $Z(\text{true}_{\Delta^* \times \Delta^*}) \rightarrow Z(i)$ for each $i \in [n]$ and $Z(\text{equal}) \rightarrow \#$. Moreover, the weight of each rule is 1.

Then it should be clear that $\llbracket \mathcal{G} \rrbracket = 1.\#$ if PCP has a solution, and $\tilde{0}$ otherwise (where $\tilde{0}: T_\Sigma \rightarrow M_K$ is the weighted tree language which maps each tree to 0). Hence $\text{supp}(\llbracket \mathcal{G} \rrbracket) \neq \emptyset$ iff PCP has a solution. □

5 Decomposition results

In this section we will decompose the weighted tree language generated by an (S, Σ, K) -rtg in two different ways. As a combination of them, we can characterize elements of $\text{Reg}(S, \Sigma, K)$ by elementary concepts: a tree transformation, an alphabetic mapping, and a regular tree language.

First, we separate the storage and second we separate weights. For this we need the concept of behaviour on a tree $\xi \in T_\Sigma$. Since we need this concept also in Section 7, we place its definition here (and not inside Section 5.1). Intuitively, a behaviour on ξ is a tree that is obtained from ξ by adding to the label of each position w a pair $(p, f_1 \dots f_k)$ of predicate p and instructions f_1, \dots, f_k if w has k successors, and inserting an arbitrarily long, but finite sequence of unary symbols of the form $\langle (p, f), * \rangle$ above each position of ξ . Figure 4 gives a first rough impression (where the storage type is P^1). A behaviour on ξ can be seen as a trace of a derivation tree of an (S, Σ, K) -rtg for ξ in which the occurrences of nonterminals are dropped; the unary symbols $\langle (p, f), * \rangle$ represent applications of chain rules.

Formally, let Σ be a ranked alphabet and $P' \subseteq P$ and $F' \subseteq F$ be finite sets. Moreover, let Δ be the ranked alphabet corresponding to Σ , P' , and F' (cf. Section 2.4). Furthermore, let $*$ be a symbol of rank 1 such that $*$ $\notin \Sigma$. We define the Σ -extension of Δ , denoted by $\langle \Delta, \Sigma \rangle$, to be the ranked alphabet where $\langle \Delta, \Sigma \rangle^{(1)} = \Delta^{(1)} \times (\Sigma^{(1)} \cup \{*\})$ and $\langle \Delta, \Sigma \rangle^{(k)} = \Delta^{(k)} \times \Sigma^{(k)}$ for $k \neq 1$.

Additionally, let \mathcal{R} be the term rewriting system having the rules:

$$\begin{aligned} \sigma(x_1, \dots, x_k) &\rightarrow \langle (p, f), * \rangle(\sigma(x_1, \dots, x_k)), \\ &\text{for every } k \geq 0, \sigma \in \Sigma^{(k)}, \text{ and } (p, f) \in \Delta^{(1)}, \text{ and} \\ \sigma(x_1, \dots, x_k) &\rightarrow \langle (p, f_1 \dots f_k), \sigma \rangle(x_1, \dots, x_k), \\ &\text{for every } k \geq 0, \sigma \in \Sigma^{(k)}, \text{ and } (p, f_1 \dots f_k) \in \Delta^{(k)}. \end{aligned}$$

Then we define the mapping $\mathcal{B}_\Delta : T_\Sigma \rightarrow \mathcal{P}(T_{\langle \Delta, \Sigma \rangle})$ for each $\xi \in T_\Sigma$ by

$$\mathcal{B}_\Delta(\xi) = \{\zeta \in T_{\langle \Delta, \Sigma \rangle} \mid \xi \Rightarrow_{\mathcal{R}}^* \zeta \text{ and } \text{pr}_1(\zeta) \in \mathcal{B}(\Delta)\}$$

and we call the set $\mathcal{B}_\Delta(\xi)$ the set of Δ -behaviours on ξ .

Let $\Theta = \langle \Delta, \Sigma \rangle \setminus (\Delta^{(1)} \times \{*\})$. It is clear that, for each $\xi \in T_\Sigma$ and $\zeta \in \mathcal{B}_\Delta(\xi)$, there is a unique bijection $\theta : \text{pos}(\xi) \rightarrow \text{pos}_\Theta(\zeta)$ which preserves the lexicographic order, i.e., if $w_1 \leq_{\text{lex}} w_2$, then $\theta(w_1) \leq_{\text{lex}} \theta(w_2)$ for every $w_1, w_2 \in \text{pos}(\xi)$. We denote this bijection by $\theta_{\xi, \zeta}$. In Figure 4 we illustrate a Δ -behaviour ζ on some tree $\xi \in T_\Sigma$ (for the storage type P^1) and the bijection $\theta_{\xi, \zeta}$.

5.1 Separating storage

Inspired by the decomposition of CFT(S)-transducers into an approximation and a macro tree transducer (see [EV86, Thm. 3.26]), we can decompose an (S, Σ, K) -rtg into the mapping \mathcal{B}_Δ (for appropriate Δ) and a $(\langle \Delta, \Sigma \rangle, K)$ -rtg.

Definition 5.1. Let $\mathcal{G} = (N, Z, R, \text{wt})$ be an (S, Σ, K) -rtg. Moreover, let $P' \subseteq P$ and $F' \subseteq F$ be finite subsets, Δ the ranked alphabet corresponding to Σ , P' and F' , and $\mathcal{G}' = (N', Z', R', \text{wt}')$ a chain-free $(\langle \Delta, \Sigma \rangle, K)$ -rtg. We say that \mathcal{G} and \mathcal{G}' are *related* if

- $\Delta = \Delta_{\mathcal{G}}$,
- $N = N'$ and $Z = Z'$,
- there is a bijection between R and R' such that the following holds:

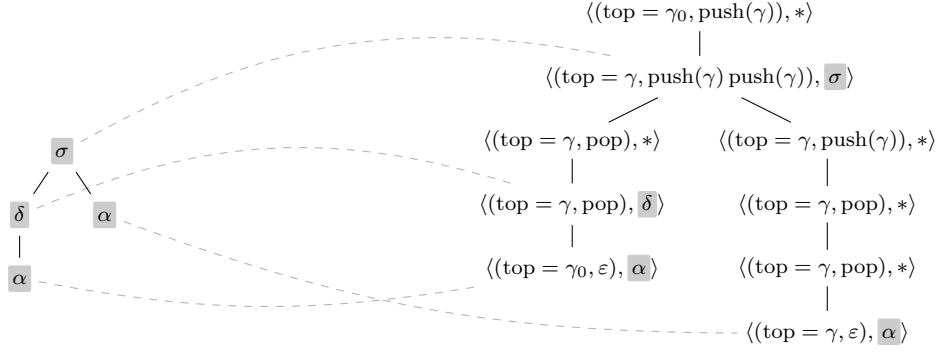


Fig. 4: A tree $\xi \in T_\Sigma$ on the left side, an element $\zeta \in \mathcal{B}_\Delta(\xi)$ on the right side for the ranked alphabet Δ corresponding to Σ , $P' = \{\text{top} = \gamma_0, \text{top} = \gamma\}$, and $F' = \{\text{push}(\gamma), \text{pop}\}$, and the bijection $\theta_{\xi, \zeta}$.

- $r = (A(p) \rightarrow \sigma(B_1(f_1), \dots, B_k(f_k)))$ in R corresponds to $r' = (A \rightarrow \langle(p, f_1 \dots f_k), \sigma\rangle(B_1, \dots, B_k))$ in R' and
- $r = (A(p) \rightarrow B(f))$ in R corresponds to $r' = (A \rightarrow \langle(p, f), *\rangle(B))$ in R' ;
- $\text{wt}'(r') = \text{wt}(r)$ for each r and r' in this bijection.

Lemma 5.2. *Let \mathcal{G} be an (S, Σ, K) -rtg. Moreover, let $P' \subseteq P$ and $F' \subseteq F$ be finite subsets and Δ the ranked alphabet corresponding to Σ , P' , and F' . Also let \mathcal{G}' be a chain-free $(\langle\Delta, \Sigma\rangle, K)$ -rtg. If \mathcal{G} and \mathcal{G}' are related, then $\llbracket \mathcal{G} \rrbracket = \mathcal{B}_\Delta; \llbracket \mathcal{G}' \rrbracket$.*

Proof: Let $\xi \in T_\Sigma$. It is obvious that there is a one-to-one correspondence between the sets $D_{\mathcal{G}}(\xi)$ and $\{(\zeta, d') \mid \zeta \in \mathcal{B}_\Delta(\xi), d' \in D_{\mathcal{G}'}(\zeta)\}$. Moreover, if d and (ζ, d') correspond to each other, then $\text{wt}(d) = \text{wt}'(d')$.

Then we can calculate as follows for each $\xi \in T_\Sigma$:

$$\begin{aligned} \llbracket \mathcal{G} \rrbracket(\xi) &= \sum_{d \in D_{\mathcal{G}}(\xi)} \text{wt}(d) = \sum_{\zeta \in \mathcal{B}_\Delta(\xi)} \sum_{d' \in D_{\mathcal{G}'}(\zeta)} \text{wt}'(d') \\ &= \sum_{\zeta \in \mathcal{B}_\Delta(\xi)} \llbracket \mathcal{G}' \rrbracket(\zeta) = (\mathcal{B}_\Delta; \llbracket \mathcal{G}' \rrbracket)(\xi) . \end{aligned}$$

□

Theorem 5.3. *For every $s: T_\Sigma \rightarrow K$ the following two statements are equivalent:*

- (i) s is (S, Σ, K) -regular.
- (ii) *There are finite sets $P' \subseteq P$ and $F' \subseteq F$ and there is a chain-free $(\langle\Delta, \Sigma\rangle, K)$ -rtg \mathcal{G} such that Δ is the ranked alphabet corresponding to Σ , P' , and F' and $s = \mathcal{B}_\Delta; \llbracket \mathcal{G} \rrbracket$.*

Proof: “(i) \Rightarrow (ii)”: Let \mathcal{G} be a (S, Σ, K) -rtg. Then we can easily construct a chain-free $(\langle\Delta_{\mathcal{G}}, \Sigma\rangle, K)$ -rtg \mathcal{G}' such that \mathcal{G} and \mathcal{G}' are related. Lemma 5.2 implies $\llbracket \mathcal{G} \rrbracket = \mathcal{B}_{\Delta_{\mathcal{G}}}; \llbracket \mathcal{G}' \rrbracket$.

“(ii) \Rightarrow (i)”: Let $P' \subseteq P$ and $F' \subseteq F$ be finite sets and Δ be the ranked alphabet corresponding to Σ , P' , and F' . Moreover, let \mathcal{G} be a chain-free $(\langle\Delta, \Sigma\rangle, K)$ -rtg. Then we can easily construct an (S, Σ, K) -rtg \mathcal{G}' such that \mathcal{G}' and \mathcal{G} are related. Lemma 5.2 implies that $\mathcal{B}_\Delta; \llbracket \mathcal{G}' \rrbracket = \llbracket \mathcal{G} \rrbracket$. □

5.2 Separating weights

Let Θ be a ranked alphabet and let $h = (h_k \mid 0 \leq k \leq \maxrk(\Theta))$ be a family of mappings such that

$$\begin{aligned} h_1 &: \Theta^{(1)} \rightarrow \Omega^{(1)} \cup (\Omega^{(1)} \times \Sigma^{(1)}) \text{ and} \\ h_k &: \Theta^{(k)} \rightarrow \Omega^{(k)} \times \Sigma^{(k)} \text{ for } k \neq 1 . \end{aligned}$$

Then the *alphabetic mapping induced by h* is $h': T_\Theta \rightarrow K[T_\Sigma]$ defined as follows: for every $k \in \mathbb{N}$, $\theta \in \Theta^{(k)}$, and $\zeta_1, \dots, \zeta_k \in T_\Theta$ we let

$$h'(\theta(\zeta_1, \dots, \zeta_k)) = \begin{cases} \omega(a_1) \cdot \xi_1 & \text{if } k = 1 \text{ and } h_1(\theta) = \omega \\ \omega(a_1, \dots, a_k) \cdot \sigma(\xi_1, \dots, \xi_k) & \text{if } h_k(\theta) = (\omega, \sigma), \end{cases}$$

where $h'(\zeta_i) = a_i \cdot \xi_i$ for each $i \in [k]$. In the sequel we identify h and h' . Now let $L \subseteq T_\Theta$. We define the weighted tree language $h(L) : T_\Sigma \rightarrow K$ by

$$h(L) = \sum_{\zeta \in L} h(\zeta) .$$

The following theorem shows how to decompose an (S, Σ, K) -rtg into an alphabetic mapping and an unambiguous and chain-free (S, Θ) -rtg. This theorem is inspired by [DV13, Lm. 3 and Lm. 4] and [HV15, Th. 6] and uses the same proof technique.

Theorem 5.4. *For every $s : T_\Sigma \rightarrow K$ the following two statements are equivalent:*

- (i) $s = \llbracket \mathcal{G} \rrbracket$ for some (S, Σ, K) -rtg \mathcal{G} .
- (ii) There are a ranked alphabet Θ , an unambiguous and chain-free (S, Θ) -rtg \mathcal{H} , and an alphabetic mapping $h : T_\Theta \rightarrow K[T_\Sigma]$ such that $s = h(\mathcal{L}(\mathcal{H}))$.

Moreover, if in (i) \mathcal{G} is chain-free, then in (ii) $h_1(\Theta^{(1)}) \subseteq \Omega^{(1)} \times \Sigma^{(1)}$, and vice versa.

Proof: (i) \Rightarrow (ii): Let $\mathcal{G} = (N, Z, R, \text{wt})$. As before we view R as ranked alphabet by associating rank k with a rule $r \in R$ if its right-hand side contains k nonterminal occurrences. We choose $\Theta = R$. Moreover, we let $\mathcal{H} = (N, Z, R')$ be the (S, R) -rtg and $h : T_R \rightarrow K[T_\Sigma]$ be the alphabetic mapping such that

- if $r = (A(p) \rightarrow \sigma(A_1(f_1), \dots, A_k(f_k)))$ is in R , then let $r' = (A(p) \rightarrow r(A_1(f_1), \dots, A_k(f_k)))$ be in R' and $h_k(r) = (\text{wt}(r), \sigma)$, and
- if $r = (A(p) \rightarrow B(f))$ is in R , then let $r' = (A(p) \rightarrow r(B(f)))$ be in R' and $h_1(r) = \text{wt}(r)$.

If \mathcal{G} is chain-free, then $h_1(R^{(1)}) \subseteq \Omega^{(1)} \times \Sigma^{(1)}$. Obviously, \mathcal{H} is chain-free. It is also easy to see that \mathcal{H} is unambiguous because the tree relabeling $\mu : T_R \rightarrow T_{R'}$ defined by the correspondence $r \mapsto r'$ is a bijection and, for every $d \in \mathcal{L}(\mathcal{H})$, the only derivation tree of \mathcal{H} for d is $\mu(d)$. Moreover,

$$\mathcal{L}(\mathcal{H}) = \bigcup_{\xi \in T_\Sigma} D_{\mathcal{G}}(\xi). \quad (\dagger)$$

In fact, each $d \in \mathcal{L}(\mathcal{H})$ is a derivation tree of \mathcal{G} for $\pi(d)$, where $\pi : T_R \rightarrow T_\Sigma$ is the mapping defined on page 10. Moreover, if $d \in D_{\mathcal{G}}(\xi)$ for some $\xi \in T_\Sigma$, then $\mu(d) \in T_{R'}$ is the (only) derivation tree of \mathcal{H} for d , hence $d \in \mathcal{L}(\mathcal{H})$. Finally, we note that for every $d \in \mathcal{L}(\mathcal{H})$ and $\xi \in T_\Sigma$ we have

$$(h(d))(\xi) = \begin{cases} \text{wt}(d) & \text{if } d \in D_{\mathcal{G}}(\xi) \\ 0 & \text{otherwise.} \end{cases} \quad (*)$$

Then we have

$$\llbracket \mathcal{G} \rrbracket(\xi) = \sum_{d \in D_{\mathcal{G}}(\xi)} \text{wt}(d) = \sum_{d \in L(\mathcal{H}): d \in D_{\mathcal{G}}(\xi)} (h(d))(\xi) = \sum_{d \in L(\mathcal{H})} (h(d))(\xi) = (h(\mathcal{L}(\mathcal{H}))) (\xi)$$

where the second equality is justified by (\dagger) and $(*)$.

(ii) \Rightarrow (i): Let $\mathcal{H} = (N', Z', R')$ be an unambiguous chain-free (S, Θ) -rtg and let $h: T_{\Theta} \rightarrow K[T_{\Sigma}]$ be an alphabetic mapping. We construct an (S, Σ, K) -rtg \mathcal{G} such that $\llbracket \mathcal{G} \rrbracket = h(\mathcal{L}(\mathcal{H}))$. The idea for the construction is to code the preimage of h in the nonterminals of \mathcal{G} as in [DV13, Lm. 4]. We let $\mathcal{G} = (N, Z, R, \text{wt})$ with $N = N' \times \Theta$, $Z = Z' \times \Theta$ and R and wt are defined as follows.

- If $A(p) \rightarrow \theta(A_1(f_1), \dots, A_k(f_k))$ is in R' with $h(\theta) = (\omega, \sigma)$, then for every $\theta_1, \dots, \theta_k \in \Theta$ the rule $r = ((A, \theta)(p) \rightarrow \sigma((A_1, \theta_1)(f_1), \dots, (A_k, \theta_k)(f_k)))$ is in R and $\text{wt}(r) = \omega$.
- If $A(p) \rightarrow \theta(A_1(f_1))$ is in R' with $h(\theta) = \omega$, then for every $\theta_1 \in \Theta$ the rule $r = ((A, \theta)(p) \rightarrow (A_1, \theta_1)(f_1))$ is in R and $\text{wt}(r) = \omega$.

If $h_1(\Theta^{(1)}) \subseteq \Omega^{(1)} \times \Sigma^{(1)}$, then \mathcal{G} is chain-free.

For every $\zeta \in \mathcal{L}(\mathcal{H})$ let us denote by $d_{\mathcal{H}}(\zeta)$ the unique derivation tree of \mathcal{H} for ζ (note that \mathcal{H} is unambiguous). Moreover, for every $\xi \in T_{\Sigma}$, let us denote by $D_{\mathcal{H}}(h^{-1}(\xi))$ the set

$$\{d_{\mathcal{H}}(\zeta) \mid \zeta \in \mathcal{L}(\mathcal{H}), h(\zeta)(\xi) \neq 0\}.$$

Let $d = d_{\mathcal{H}}(\zeta)$, for some $\zeta \in \mathcal{L}(\mathcal{H})$, and let $d' \in D_{\mathcal{G}}(\xi)$. We say that d corresponds to d' if $\text{pos}(d) = \text{pos}(d')$ and for each $w \in \text{pos}(d)$:

- if $d(w) = (A(p) \rightarrow \theta(A_1(f_1), \dots, A_k(f_k)))$ with $h(\theta) = (\omega, \sigma)$, then $d'(w) = ((A, \theta)(p) \rightarrow \sigma((A_1, \zeta(w_1))(f_1), \dots, (A_k, \zeta(w_k))(f_k)))$, and
- if $d(w) = (A(p) \rightarrow \theta(B(f)))$ with $h(\theta) = \omega$, then $d'(w) = ((A, \theta)(p) \rightarrow (B, \zeta(w_1)))$.

It is not hard to see that the above correspondence is a bijection between $D_{\mathcal{H}}(h^{-1}(\xi))$ and $\{d' \in D_{\mathcal{G}}(\xi) \mid \text{wt}(d') \neq 0\}$ for every $\xi \in T_{\Sigma}$. Moreover, if $d_{\mathcal{H}}(\zeta)$ corresponds to d' , then $\text{wt}(d') = (h(\zeta))(\xi)$. Then, for every $\xi \in T_{\Sigma}$, we have

$$\begin{aligned} (h(\mathcal{L}(\mathcal{H}))) (\xi) &= \sum_{\zeta \in \mathcal{L}(\mathcal{H})} (h(\zeta))(\xi) = \sum_{\substack{\zeta \in \mathcal{L}(\mathcal{H}): \\ h(\zeta)(\xi) \neq 0}} (h(\zeta))(\xi) \stackrel{(\dagger)}{=} \sum_{\substack{\zeta \in \mathcal{L}(\mathcal{H}): \\ d_{\mathcal{H}}(\zeta) \in D_{\mathcal{H}}(h^{-1}(\xi))}} (h(\zeta))(\xi) \\ &\stackrel{(*)}{=} \sum_{\substack{d' \in D_{\mathcal{G}}(\xi): \\ \text{wt}(d') \neq 0}} \text{wt}(d') = \sum_{d' \in D_{\mathcal{G}}(\xi)} \text{wt}(d') = \llbracket \mathcal{G} \rrbracket(\xi), \end{aligned}$$

where (\dagger) holds because for $\zeta \in \mathcal{L}(\mathcal{H})$ we have $(h(\zeta))(\xi) = 0$ if $d_{\mathcal{H}}(\zeta) \notin D_{\mathcal{H}}(h^{-1}(\xi))$; and $(*)$ holds due to the bijection described above. \square

Now we can prove that, even in the weighted case, we can eliminate finite storage types.

Corollary 5.5. *Let S be finite.*

1. $\text{Reg}(S, \Sigma, K) \subseteq \text{Reg}(\Sigma, K)$ and $\text{Reg}_{\text{nc}}(S, \Sigma, K) \subseteq \text{Reg}_{\text{nc}}(\Sigma, K)$.
2. If $S_{\text{true}, \text{id}} = S$, then $\text{Reg}(\Sigma, K) \subseteq \text{Reg}(S, \Sigma, K)$ and $\text{Reg}_{\text{nc}}(\Sigma, K) \subseteq \text{Reg}_{\text{nc}}(S, \Sigma, K)$.

Proof: First we prove 1. Let \mathcal{G} be an (S, Σ, K) -rtg. By Theorem 5.4(i) \Rightarrow (ii), there are a ranked alphabet Θ , an unambiguous and chain-free (S, Θ) -rtg \mathcal{H} , and an alphabetic mapping $h: T_{\Theta} \rightarrow K[T_{\Sigma}]$ such that

$s = h(\mathcal{L}(\mathcal{H}))$. By Lemma 3.4 there is a Θ -rtg \mathcal{H}' such that $\mathcal{L}(\mathcal{H}) = \mathcal{L}(\mathcal{H}')$. Moreover, \mathcal{H}' is unambiguous and chain-free. Hence, by Theorem 5.4(ii) \Rightarrow (i) (with $S = \text{TRIV}$), we obtain that $s \in \text{Reg}(\Sigma, K)$.

Now let, additionally, \mathcal{G} be chain-free. Then $h_1(\Theta^{(1)}) \subseteq \Omega^{(1)} \times \Sigma^{(1)}$ and thus, by the same argumentation as above, we obtain that $s \in \text{Reg}_{\text{nc}}(\Sigma, K)$.

Next we prove 2. Let $\mathcal{G} = (N, Z, R, \text{wt})$ be a (Σ, K) -rtg. Then we construct the (S, Σ, K) -rtg $\mathcal{G}' = (N, Z, R', \text{wt}')$ such that

- if $r = (A \rightarrow \sigma(A_1, \dots, A_k))$ is a rule in R ,
then $r' = (A(\text{true}_C) \rightarrow \sigma(A_1(\text{id}_C), \dots, A_k(\text{id}_C)))$ is in R' ,
- if $r = (A \rightarrow B)$ is a rule in R ,
then $r' = (A(\text{true}_C) \rightarrow B(\text{id}_C))$ is in R' , and
- in both cases we let $\text{wt}'(r') = \text{wt}(r)$.

If \mathcal{G} is chain-free, then so is \mathcal{G}' . For each $\xi \in T_\Sigma$, there is a bijection $\theta : D_{\mathcal{G}}(\xi) \rightarrow D_{\mathcal{G}'}(\xi)$ such that $\text{wt}'(\theta(d)) = \text{wt}(d)$ for each $d \in D_{\mathcal{G}}(\xi)$. This implies that $\llbracket \mathcal{G}' \rrbracket = \llbracket \mathcal{G} \rrbracket$. \square

From Corollary 5.5 we immediately obtain the following result.

Corollary 5.6. *If S is finite and $S_{\text{true}, \text{id}} = S$, then $\text{Reg}(S, \Sigma, K) = \text{Reg}(\Sigma, K)$ and $\text{Reg}_{\text{nc}}(S, \Sigma, K) = \text{Reg}_{\text{nc}}(\Sigma, K)$.*

5.3 Combination of separation results

In this section we combine the separation of storage with the separation of weights. In this way, we can characterize each element in $\text{Reg}(S, \Sigma, K)$ by elementary concepts: a tree transformation \mathcal{B}_Δ , an alphabetic mapping h , and an element in $\text{Reg}(\Theta)$.

Theorem 5.7. *For every $s : T_\Sigma \rightarrow K$ the following two statements are equivalent:*

- (i) s is (S, Σ, K) -regular.
- (ii) $s = \mathcal{B}_\Delta; h(\mathcal{L}(\mathcal{H}))$ for some finite sets $P' \subseteq P$, $F' \subseteq F$, ranked alphabet Δ corresponding to Σ , P' , and F' , ranked alphabet Θ , unambiguous and chain-free Θ -rtg \mathcal{H} , and alphabetic mapping $h : T_\Theta \rightarrow K[T_{\langle \Delta, \Sigma \rangle}]$.

Proof: (i) \Rightarrow (ii): By Theorem 5.3 there are finite sets $P' \subseteq P$ and $F' \subseteq F$ and there is a chain-free $(\langle \Delta, \Sigma \rangle, K)$ -rtg \mathcal{G} such that Δ is the ranked alphabet corresponding to Σ , P' , and F' and $s = \mathcal{B}_\Delta; \llbracket \mathcal{G} \rrbracket$.

According to Theorem 5.4 there are a ranked alphabet Θ , an unambiguous and chain-free Θ -rtg \mathcal{H} , and an alphabetic mapping $h : T_\Theta \rightarrow K[T_{\langle \Delta, \Sigma \rangle}]$ such that $\llbracket \mathcal{G} \rrbracket = h(\mathcal{L}(\mathcal{H}))$.

(ii) \Rightarrow (i): By Theorem 5.4 we have that $h(\mathcal{L}(\mathcal{H}))$ is $(\langle \Delta, \Sigma \rangle, K)$ -recognizable. Then, by Theorem 5.3, $\mathcal{B}_\Delta; h(\mathcal{L}(\mathcal{H}))$ is (S, Σ, K) -recognizable. \square

Example 5.8. Here we show a slightly more complex example of a weighted tree language s , we show a weighted regular tree grammar \mathcal{G} with storage, and we apply the decompositions inherent in the proof of Theorem 5.7 to \mathcal{G} in order to show that $s = \llbracket \mathcal{G} \rrbracket$.

First, we define the weighted tree language s . For this, we let $\Sigma = \{\sigma^{(2)}, \delta^{(2)}, \#^{(0)}\}$. Roughly speaking, the mapping s maps each tree $\xi \in T_\Sigma$ to its degree of unbalancedness [SVF09, Example 1] if ξ has the following path property, and to 0 otherwise. A tree ξ has the path property if, for each position w of ξ , the number of occurrences of σ 's above w or at w is greater than or equal to the number of occurrences of δ 's above w or at w .

The degree of unbalancedness is formalized as follows. For each tree $\xi \in T_\Sigma$ and each $w \in \text{pos}(\xi)$ we define

$$\text{ubal}(\xi, w) = \begin{cases} |\text{height}(\xi|_{w1}) - \text{height}(\xi|_{w2})| & \text{if } \xi(w) \in \{\sigma, \delta\}, \\ 0 & \text{otherwise,} \end{cases}$$

and we define $\text{ubal}(\xi) = \max(\text{ubal}(\xi, w) \mid w \in \text{pos}(\xi))$, which is the *degree of unbalancedness of ξ* . (At the end of this example we will use ubal for another ranked alphabet.)

The path property can be captured by the formal string language

$$L = \{w \in \{\sigma, \delta\}^* \{\#\} \mid \forall u \in \{\sigma, \delta\}^* \{\#\} : \text{if } u \preceq w, \text{ then } |u|_\sigma \geq |u|_\delta\} .$$

Then we define the weighted tree language $s : T_\Sigma \rightarrow \mathbb{N}$ for each $\xi \in T_\Sigma$ by

$$s(\xi) = \begin{cases} \text{ubal}(\xi) & \text{if } \text{path}(\xi) \subseteq L, \\ 0 & \text{otherwise.} \end{cases}$$

Second, we define a $(\text{COUNT}, \Sigma, K_{\max})$ -rtg \mathcal{G} (for some specific M-monoid K_{\max}) which generates s (cf. Section 2.4 for the definition of COUNT ; we abbreviate the predicate $\text{true}_{\mathbb{N}}$ by true). For this, we define the complete M-monoid

$$K_{\max} = (\mathbb{N} \cup \{\infty\}, \max, 0, \Omega)$$

where \max is extended to maximum over countable index sets in the obvious way. Moreover, we let

$$\Omega = \{0_k \mid k \in \mathbb{N}\} \cup \{1_0\} \cup \{\pi_1^{(2)}, \pi_2^{(2)}, \text{diff}^{(2)}, \text{ht}^{(2)}\}$$

where 1_0 is the 0-ary function defined by $1_0() = 1$, and for every $a, b \in \mathbb{N} \cup \{\infty\}$ we define the binary operations by case analysis as follows.

Case 1: If $a = 0$ or $b = 0$, then $\pi_1(a, b) = \pi_2(a, b) = \text{diff}(a, b) = \text{ht}(a, b) = 0$. (These definitions guarantee that each binary operation of Ω is absorptive.)

Case 2: If $a \in \mathbb{N} \setminus \{0\}$ and $b \in \mathbb{N} \setminus \{0\}$, then $\pi_1(a, b) = a$, $\pi_2(a, b) = b$, $\text{diff}(a, b) = |a - b|$, and $\text{ht}(a, b) = 1 + \max(a, b)$.

Case 3: Otherwise we define $\pi_1(a, b)$, $\pi_2(a, b)$, $\text{diff}(a, b)$, and $\text{ht}(a, b)$ arbitrarily.

We note that Cases 1 and 3 will not occur in computations in our example.

Now we construct the $(\text{COUNT}, \Sigma, K_{\max})$ -rtg $\mathcal{G} = (N, U, R, \text{wt})$ with the following intuition. As in [SVF09, Example 1], we use two nonterminals, i.e., $N = \{H, U\}$, such that for each tree $\xi \in T_\Sigma$ and derivation tree d of ξ , there is exactly one position $w \in \text{pos}(d)$ such that at w and each of its predecessors a rule with left-hand side nonterminal U is applied and at each other position a rule with left-hand side nonterminal H is applied. Then the weight of this derivation tree is $\text{ubal}(\xi, w)$. Since the position w is chosen nondeterministically, the value $\max(\text{ubal}(\xi, w) \mid w \in \text{pos}(\xi))$ is computed, which is the degree of unbalancedness of the whole tree ξ . The path property is checked by using the storage COUNT : on each generation of a σ -labeled position, the counter is incremented, and on each generation of a δ -labeled

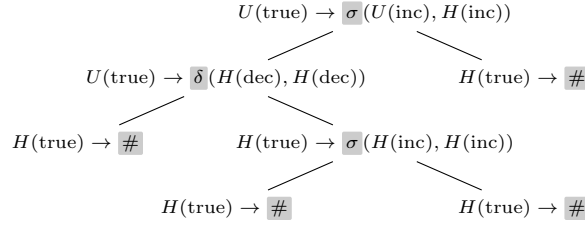


Fig. 5: A derivation tree $d \in D_{\mathcal{G}}(\xi)$ for the tree $\xi = \sigma(\delta(\#, \sigma(\#, \#)), \#)$.

position it is decremented.

$$\begin{array}{ll}
 r_1 : H(\text{true}) \rightarrow \#, 1_0 & r_2 : U(\text{true}) \rightarrow \#, 0_0 \\
 r_3 : H(\text{true}) \rightarrow \sigma(H(\text{inc}), H(\text{inc})), \text{ht} & r_4 : U(\text{true}) \rightarrow \sigma(H(\text{inc}), H(\text{inc})), \text{diff} \\
 r_5 : U(\text{true}) \rightarrow \sigma(H(\text{inc}), U(\text{inc})), \text{pr}_2 & r_6 : U(\text{true}) \rightarrow \sigma(U(\text{inc}), H(\text{inc})), \text{pr}_1 \\
 r_7 : H(\text{true}) \rightarrow \delta(H(\text{dec}), H(\text{dec})), \text{ht} & r_8 : U(\text{true}) \rightarrow \delta(H(\text{dec}), H(\text{dec})), \text{diff} \\
 r_9 : U(\text{true}) \rightarrow \delta(H(\text{dec}), U(\text{dec})), \text{pr}_2 & r_{10} : U(\text{true}) \rightarrow \delta(U(\text{dec}), H(\text{dec})), \text{pr}_1 .
 \end{array}$$

Indeed, the grammar \mathcal{G} does not use the predicate zero. In Figure 5 we show a derivation tree d of \mathcal{G} for the tree $\xi = \sigma(\delta(\#, \sigma(\#, \#)), \#)$ with $\text{wt}(d) = 1$.

Third, we apply to the $(\text{COUNT}, \Sigma, K_{\max})$ -rtg \mathcal{G} the decompositions that are inherent in the proof of Theorem 5.7 consecutively, i.e., the separation of the storage (cf. Theorem 5.3(i) \Rightarrow (ii)) and the separation of the weights (cf. Theorem 5.4(i) \Rightarrow (ii)). Since we use the storage to check the path property and the weights to calculate the degree of unbalancedness, Theorem 5.7 suggests that we can prove $s = \llbracket \mathcal{G} \rrbracket$ in two independent steps. For this, let $\xi \in T_{\Sigma}$.

Step 1: Applying the construction of the storage separation theorem (cf. Theorem 5.3(i) \Rightarrow (ii)), we obtain the following objects:

1. the ranked alphabet $\Delta_{\mathcal{G}}$ corresponding to \mathcal{G} with $P' = \{\text{true}\}$, $F' = \{\text{inc}, \text{dec}\}$, and

$$\Delta_{\mathcal{G}} = \{(\text{true}, \varepsilon)^{(0)}\} \cup \{(\text{true}, f)^{(1)} \mid f \in F'\} \cup \{(\text{true}, f_1 f_2)^{(2)} \mid f_1, f_2 \in F'\} \text{ and}$$

2. the chain-free $(\langle \Delta_{\mathcal{G}}, \Sigma \rangle, K_{\max})$ -rtg $\mathcal{G}' = (N, U, R', \text{wt}')$ where R' consists of rules derived from R by using enriched terminal symbols instead of storage predicates and instructions, e.g., the rule

$$r_3 : H(\text{true}) \rightarrow \sigma(H(\text{inc}), H(\text{inc}))$$

of \mathcal{G} is transformed into the rule

$$r'_3 : H \rightarrow \langle (\text{true}, \text{inc inc}), \sigma \rangle (H, H)$$

of \mathcal{G}' with $\text{wt}'(r'_3) = \text{wt}(r_3)$. It is obvious how the other rules of R' and wt' look like. It is easy to see that $\text{supp}(\llbracket \mathcal{G}' \rrbracket) \subseteq T_{\Psi}$ where $\Psi \subseteq \langle \Delta_{\mathcal{G}}, \Sigma \rangle$ is the ranked alphabet

$$\Psi = \{(\langle (\text{true}, \varepsilon), \# \rangle)^{(0)}, \langle (\text{true}, \text{inc inc}), \sigma \rangle^{(2)}, \langle (\text{true}, \text{dec dec}), \delta \rangle^{(2)}\} .$$

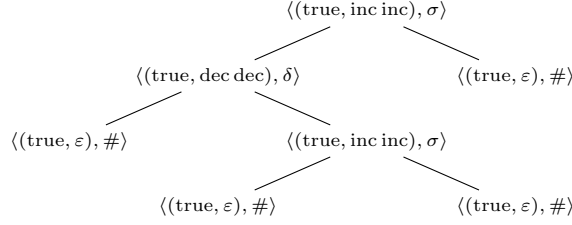


Fig. 6: The tree $\zeta_\xi \in \mathcal{B}_{\Delta_{\mathcal{G}}}(\xi) \cap T_\Psi$ for the tree $\xi = \sigma(\delta(\#, \sigma(\#, \#)), \#)$.

By Theorem 5.3(i) \Rightarrow (ii) and the fact that $\text{supp}(\llbracket \mathcal{G}' \rrbracket) \subseteq T_\Psi$, we have

$$\llbracket \mathcal{G} \rrbracket(\xi) = \max(\llbracket \mathcal{G}' \rrbracket(\zeta) \mid \zeta \in \mathcal{B}_{\Delta_{\mathcal{G}}}(\xi)) = \max(\llbracket \mathcal{G}' \rrbracket(\zeta) \mid \zeta \in \mathcal{B}_{\Delta_{\mathcal{G}}}(\xi) \cap T_\Psi) .$$

We distinguish two cases. First, let $\mathcal{B}_{\Delta_{\mathcal{G}}}(\xi) \cap T_\Psi = \emptyset$. Then $\llbracket \mathcal{G} \rrbracket(\xi) = \max(\llbracket \mathcal{G}' \rrbracket(\zeta) \mid \zeta \in \emptyset) = 0$. Moreover, there is no behaviour $b \in \mathcal{B}(\Delta_{\mathcal{G}})$ such that $\text{pos}(b) = \text{pos}(\xi)$ and b increments at σ -labeled positions of ξ and decrements at δ -labeled positions of ξ . In other words, ξ does not have the path property. Hence $s(\xi) = 0$ and thus $\llbracket \mathcal{G} \rrbracket(\xi) = s(\xi)$.

Now let $\mathcal{B}_{\Delta_{\mathcal{G}}}(\xi) \cap T_\Psi \neq \emptyset$. For each $\zeta \in \mathcal{B}_{\Delta_{\mathcal{G}}}(\xi) \cap T_\Psi$, we have that $\text{pos}(\xi) = \text{pos}(\zeta)$, because \mathcal{G} does not have chain rules. Moreover, due to the definition of the mapping $\mathcal{B}_{\Delta_{\mathcal{G}}}$, the set $\mathcal{B}_{\Delta_{\mathcal{G}}}(\xi) \cap T_\Psi$ has one element, denoted by, say ζ_ξ , that is defined for each $w \in \text{pos}(\xi)$ by

$$\zeta_\xi(w) = \begin{cases} \langle\langle \text{true}, \text{inc inc} \rangle, \sigma\rangle & \text{if } \xi(w) = \sigma \\ \langle\langle \text{true}, \text{dec dec} \rangle, \delta\rangle & \text{if } \xi(w) = \delta \\ \langle\langle \text{true}, \varepsilon \rangle, \#\rangle & \text{if } \xi(w) = \# . \end{cases}$$

Figure 6 shows ζ_ξ for the tree $\xi = \sigma(\delta(\#, \sigma(\#, \#)), \#)$. Thus,

$$\llbracket \mathcal{G} \rrbracket(\xi) = \llbracket \mathcal{G}' \rrbracket(\zeta_\xi) .$$

Since $\text{pr}_1(\zeta_\xi)$ is a $\Delta_{\mathcal{G}}$ -behaviour, we can consider the family $(c_w \mid w \in \text{pos}(\text{pr}_1(\zeta_\xi)))$ of configurations determined by $\text{pr}_1(\zeta_\xi)$ and 0 (the initial configuration of COUNT). Then, by definition, $c_w \geq 0$ for each $w \in \text{pos}(\text{pr}_1(\zeta_\xi))$. Moreover, it is easy to see that c_w is the difference between the number of occurrences of σ above w or at w and the number of occurrences of δ above w or at w . Hence, ξ has the path property.

Step 2: It remains to prove that, if ξ has the path property, then $\llbracket \mathcal{G} \rrbracket(\xi) = \text{ubal}(\xi)$. So let $\xi \in T_\Sigma$ and ξ have the path property. Applying the construction of the weight separation theorem to \mathcal{G}' (cf. Theorem 5.4(i) \Rightarrow (ii)), we obtain the following objects:

1. the ranked alphabet Θ , i.e., $\Theta = \{r_1^{(0)}, r_2^{(0)}\} \cup \{r_i^{(2)} \mid 3 \leq i \leq 10\}$,
2. the unambiguous and chain-free Θ -rtg $\mathcal{H} = (N, U, R'')$ and the alphabetic mapping $h : T_\Theta \rightarrow K[T_\Sigma]$ where R'' contains the following rules; for each rule r_i'' we have indicated the pair $h_j(r_i'')$ (for appropriate rank $j \in \{0, 2\}$) after the comma.

$$\begin{array}{ll} r_1'' : H \rightarrow r_1', (1, \langle\langle \text{true}, \varepsilon \rangle, \#\rangle) & r_2'' : U \rightarrow r_2', (1, \langle\langle \text{true}, \varepsilon \rangle, \#\rangle) \\ r_3'' : H \rightarrow r_3'(H, H), (\text{ht}, \langle\langle \text{true}, \text{inc inc} \rangle, \sigma\rangle) & r_4'' : U \rightarrow r_4'(H, H), (\text{diff}, \langle\langle \text{true}, \text{inc inc} \rangle, \sigma\rangle) \\ r_5'' : U \rightarrow r_5'(H, U), (\text{pr}_2, \langle\langle \text{true}, \text{inc inc} \rangle, \sigma\rangle) & r_6'' : U \rightarrow r_6'(U, H), (\text{pr}_1, \langle\langle \text{true}, \text{inc inc} \rangle, \sigma\rangle) \\ r_7'' : H \rightarrow r_7'(H, H), (\text{ht}, \langle\langle \text{true}, \text{dec dec} \rangle, \delta\rangle) & r_8'' : U \rightarrow r_8'(H, H), (\text{diff}, \langle\langle \text{true}, \text{dec dec} \rangle, \delta\rangle) \\ r_9'' : U \rightarrow r_9'(H, U), (\text{pr}_2, \langle\langle \text{true}, \text{dec dec} \rangle, \delta\rangle) & r_{10}'' : U \rightarrow r_{10}'(U, H), (\text{pr}_1, \langle\langle \text{true}, \text{dec dec} \rangle, \delta\rangle). \end{array}$$

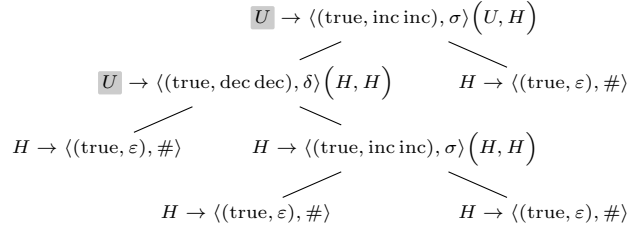


Fig. 7: The tree $d_w \in T_\Phi$ for $w = 1$ and ζ_ξ of Figure 6; w is indicated by the grey shaded nonterminals.

According to Theorem 5.4(i) \Rightarrow (ii), we have that

$$\llbracket \mathcal{G}' \rrbracket(\zeta_\xi) = \max(h(d)(\zeta_\xi) \mid d \in \mathcal{L}(\mathcal{H})) .$$

Let $d \in \mathcal{L}(\mathcal{H})$. As shown in the proof of Theorem 5.4(i) \Rightarrow (ii), the tree d is a derivation tree of \mathcal{G}' for $\pi(d)$ (where π is the mapping defined on page 10). If $\pi(d) \neq \zeta_\xi$, then $h(d)(\zeta_\xi) = 0$ by definition of h . Thus we have

$$\llbracket \mathcal{G}' \rrbracket(\zeta_\xi) = \max(h(d)(\zeta_\xi) \mid d \in \mathcal{L}(\mathcal{H}) \cap \pi^{-1}(\zeta_\xi)) .$$

We note that $\text{pos}(d) = \text{pos}(\zeta_\xi) = \text{pos}(\xi)$ for each $d \in \mathcal{L}(\mathcal{H}) \cap \pi^{-1}(\zeta_\xi)$.

Due to the behaviour of the nonterminals in \mathcal{H} , each tree in $\mathcal{L}(\mathcal{H}) \cap \pi^{-1}(\zeta_\xi)$ has a particular shape. More precisely,

$$\mathcal{L}(\mathcal{H}) \cap \pi^{-1}(\zeta_\xi) = \{d_w \mid w \in \text{pos}(\zeta_\xi)\}$$

where, for each $w \in \text{pos}(\zeta_\xi)$, we define the tree $d_w \in T_\Theta$ such that $\text{pos}(d_w) = \text{pos}(\zeta_\xi)$ and, for each $v \in \text{pos}(d_w)$, we have

$$d_w(v) = \begin{cases} U \rightarrow (\zeta_\xi(v)) \left(U, H \right) & \text{if } (\exists u \in \mathbb{N}^*) : w = v1u \\ U \rightarrow (\zeta_\xi(v)) \left(H, U \right) & \text{if } (\exists u \in \mathbb{N}^*) : w = v2u \\ U \rightarrow (\zeta_\xi(v)) \left(H, H \right) & \text{if } v = w \text{ and } v \text{ is not a leaf} \\ U \rightarrow \zeta_\xi(v) & \text{if } v = w \text{ and } v \text{ is a leaf} \\ H \rightarrow (\zeta_\xi(v)) \left(H, H \right) & \text{if } v \text{ is not a leaf and } (v \not\leq w \text{ or } \exists u \in \mathbb{N}^+ : v = wu) \\ H \rightarrow \zeta_\xi(v) & \text{if } v \text{ is a leaf and } (v \not\leq w \text{ or } \exists u \in \mathbb{N}^+ : v = wu) . \end{cases}$$

Figure 7 shows the tree $d_w \in \mathcal{L}(\mathcal{H}) \cap \pi^{-1}(\zeta_\xi)$ for $w = 1$ and ζ_ξ as shown in Figure 6.

Hence

$$\llbracket \mathcal{G}' \rrbracket(\zeta_\xi) = \max(h(d_w)(\zeta_\xi) \mid w \in \text{pos}(\zeta_\xi)) .$$

Due to the definition of h it is also obvious that, for each $w \in \text{pos}(\zeta_\xi)$,

$$\begin{aligned} h(d_w|_{w1})(\zeta_\xi|_{w1}) &= \text{height}(\zeta_\xi|_{w1}), \\ h(d_w|_{w2})(\zeta_\xi|_{w2}) &= \text{height}(\zeta_\xi|_{w2}), \\ h(d_w|_w)(\zeta_\xi|_w) &= \text{diff}(\text{height}(\zeta_\xi|_{w1}), \text{height}(\zeta_\xi|_{w2})) = \text{ubal}(\zeta_\xi, w), \\ h(d_w)(\zeta_\xi) &= \text{ubal}(\zeta_\xi, w) \end{aligned}$$

where we have used here ubal for the ranked alphabet Ψ .

$$\llbracket \mathcal{G}' \rrbracket(\zeta_\xi) = \max(\text{ubal}(\zeta_\xi, w) \mid w \in \text{pos}(\zeta_\xi)) = \text{ubal}(\zeta_\xi) .$$

Finally, we have $\llbracket \mathcal{G} \rrbracket(\xi) = \llbracket \mathcal{G}' \rrbracket(\zeta_\xi) = \text{ubal}(\zeta_\xi) = \text{ubal}(\xi) = s(\xi)$, where the last but one equation holds because $\text{pos}(\zeta_\xi) = \text{pos}(\xi)$.

6 Elimination of chain rules

It is well known that, for each (Σ, M_K) -rtg where K is a complete semiring, an equivalent chain-free (Σ, K) -rtg can be constructed [ÉK03, FMV09] by solving the corresponding classical algebraic path problem [Rot85]. This is not true if we consider non-trivial storage types as the following theorem shows.

Theorem 6.1. $\text{Reg}(\mathbb{P}^1, \Sigma) \setminus \bigcup_S \text{Reg}_{nc}(S, \Sigma) \neq \emptyset$ where S ranges over the set of all storage types.

Proof: Let $\Sigma = \{\alpha^{(0)}, \delta^{(1)}, \sigma^{(2)}\}$. The tree language

$$L = \{\sigma(\delta^n(\alpha), \delta^n(\alpha)) \mid n \geq 0\}$$

is in $\text{Reg}(\mathbb{P}^1, \Sigma)$, because it can be generated by the (\mathbb{P}^1, Σ) -grammar which we obtain from \mathcal{G} of Example 3.1 by dropping its weight structure and weight function (cf. [Gue83]).

On the other hand, we can show by contradiction that $L \notin \text{Reg}_{nc}(S, \Sigma)$ for any storage type S . For this, we assume that there is a storage type S and a chain-free (S, Σ) -rtg $\mathcal{G} = (N, Z, R)$ such that $\mathcal{L}(\mathcal{G}) = L$. By Lemma 3.2 we can assume that Z is a single nonterminal. There are finitely many σ -rules in R , i.e., rules of the form

$$r = (Z(p) \rightarrow \sigma(A(f), B(g))) . \tag{1}$$

Since L is infinite, there is a σ -rule which is the root of some $d \in D_{\mathcal{G}}(\sigma(\delta^n(\alpha), \delta^n(\alpha)))$ for infinitely many n 's. Let (1) be such a rule. Then $p(c_0) = \text{true}$ and $f(c_0)$ and $g(c_0)$ are defined (where c_0 is the initial configuration of S), and there are

- integers $n, m \in \mathbb{N}$ with $n \neq m$,
- $(A, f(c_0))$ -derivation trees d_n, d_m for $\delta^n(\alpha), \delta^m(\alpha)$, respectively,
- $(B, g(c_0))$ -derivation trees d'_n, d'_m for $\delta^n(\alpha), \delta^m(\alpha)$, respectively,

such that $r(d_n, d'_n) \in D_{\mathcal{G}}(\sigma(\delta^n(\alpha), \delta^n(\alpha)))$ and $r(d_m, d'_m) \in D_{\mathcal{G}}(\sigma(\delta^m(\alpha), \delta^m(\alpha)))$. But then $r(d_n, d'_m) \in D_{\mathcal{G}}(\sigma(\delta^n(\alpha), \delta^m(\alpha)))$ and hence $\sigma(\delta^n(\alpha), \delta^m(\alpha)) \in L$ with $m \neq n$. This is a contradiction to $L = \{\sigma(\delta^n(\alpha), \delta^n(\alpha)) \mid n \geq 0\}$. \square

Thus, in general, chain rules cannot be eliminated from (S, Σ, K) -rtg (even if $K = \mathbb{B}$). But we can eliminate chain rules for particular (S, Σ, K) -rtg, which we will call 'simple'. There is another problem which concerns the weight algebra. For (Σ, M_K) -rtg where K is a complete semiring, the elimination typically uses elements $a^* \in K$ (for some $a \in K$) to capture the weight of cycles of chain rules. Here a^* is the sum of all powers a^n of a and the powers are defined by the multiplication of the semiring. In our setting we deal with M-monoids and, instead of the binary multiplication, we have operations with different arities. Thus, we will have to guarantee that the M-monoid is closed under iterated composition of operations. Let us now formalize these requirements on the (S, Σ, K) -rtg and on the M-monoid.

We call an (S, Σ, K) -rtg *simple* if for each chain rule $A(p) \rightarrow B(f)$ we have $p = \text{true}_C$ and $f = \text{id}_C$. Let $\text{Reg}_{\text{simple}}(S, \Sigma, K)$ denote the class of all weighted tree languages generated by simple (S, Σ, K) -rtgs.

Now we introduce the necessary closure properties of K . An operation $\omega \in \text{Ops}^{(k)}(K)$ for $k \geq 1$ is *completely distributive* if

$$\omega\left(\sum_{i_1 \in I_1} a_{i_1}, \dots, \sum_{i_k \in I_k} a_{i_k}\right) = \sum_{i_1 \in I_1} \dots \sum_{i_k \in I_k} \omega(a_{i_1}, \dots, a_{i_k})$$

for all countable index sets I_j and family $(a_{i_j} \in K \mid i_j \in I_j), j \in [k]$. We say that K is *completely distributive* if each operation in Ω is completely distributive. (This concept was introduced as complete DM-monoid in [Kui98], cf. also [ÉK03].)

Let $k \geq 0$, I be a countable index set, and $(\omega_i \in \text{Ops}^{(k)}(K) \mid i \in I)$ a family of operations. We define the operation $\sum_{i \in I} \omega_i \in \text{Ops}^{(k)}(K)$ by letting

$$\left(\sum_{i \in I} \omega_i\right)(a_1, \dots, a_k) = \sum_{i \in I} \omega_i(a_1, \dots, a_k)$$

for every $a_1, \dots, a_k \in K$. We say that K is *completely 1-sum closed* if $\sum_{i \in I} \omega_i \in \Omega^{(1)}$ for every countable index set I and family $(\omega_i \in \Omega^{(1)} \mid i \in I)$.

Let $\omega \in \text{Ops}^{(1)}(K)$ and $\omega' \in \text{Ops}^{(k)}(K)$ for some $k \geq 0$. The *composition of ω and ω'* is the operation $\omega \circ \omega' \in \text{Ops}^{(k)}$ defined by $(\omega \circ \omega')(a_1, \dots, a_k) = \omega(\omega'(a_1, \dots, a_k))$ for every $a_1, \dots, a_k \in K$. We say that K is *$(1, k)$ -composition closed* if $\omega \circ \omega' \in \Omega^{(k)}$ for every $\omega \in \Omega^{(1)}$ and $\omega' \in \Omega^{(k)}$. Moreover, K is *$(1, *)$ -composition closed* if it is $(1, k)$ -composition closed for every $k \geq 0$ (cf. [FMV09, Def. 4.3]).

The following statement follows easily from the corresponding definitions.

Observation 6.2. For every countable index set I , family $(\omega_i \in \text{Ops}^{(1)}(K) \mid i \in I)$, $k \geq 0$, and $\omega \in \text{Ops}^{(k)}(K)$, we have $\sum_{i \in I} (\omega_i \circ \omega) = \left(\sum_{i \in I} \omega_i\right) \circ \omega$.

Let N be a finite set. Moreover, let V and W be $(N \times N)$ -matrices over $\text{Ops}^{(1)}(K)$. We define the *product $V \cdot W$ of V and W* by $(V \cdot W)_{A,B} = \sum_{C \in N} V_{A,C} \circ W_{C,B}$ for every $A, B \in N$. Note that, although the set N is not ordered, the expression $\sum_{C \in N} V_{A,C} \circ W_{C,B}$ is well defined, because the monoid $(\text{Ops}^{(1)}(K), +, 0)$ is commutative. Moreover, for every $n \geq 0$ we define the $(N \times N)$ -matrix W^n over $\text{Ops}^{(1)}(K)$ by induction as follows: let $W^0 = E$ and $W^n = W \cdot W^{n-1}$ for every $n \geq 1$, where E is the *unit matrix over $\text{Ops}^{(1)}(K)$* defined by $E_{A,B} = \text{id}_K$ if $A = B$ and 0_1 otherwise for every $A, B \in N$. Finally, we define $W^* = \sum_{n \geq 0} W^n$, where $\left(\sum_{n \geq 0} W^n\right)_{A,B} = \sum_{n \geq 0} W_{A,B}^n$ for every $A, B \in N$.

Finally, we say that K *has identity* if $\text{id}_K \in \Omega^{(1)}$.

We call K *compressible* if it has identity, it is $(1, *)$ -composition closed, completely 1-sum closed, and completely distributive. Each M-monoid associated with a complete semiring is compressible. The fact that such an M-monoid is completely 1-sum closed and completely distributive can be derived from the generalized distributivity law of the complete semiring. In particular, the Boolean M-monoid is compressible.

First we show that if K is compressible and S contains the always true predicate and the identity instruction, then chain rules can be eliminated from simple (S, Σ, K) -rtgs.

Similar results for the elimination of chain rules (or ε -transitions) in the weighted case have been proved in [ÉK03, Thm. 3.2] and [FMV11, Lm. 3.2]. In fact, in [ÉK03, Thm. 3.2] it was shown that ε -transitions can be eliminated from weighted tree automata over commutative and continuous semirings. The same was shown for weighted tree automata over commutative and complete semirings in [FMV11, Lm. 3.2]. The second result generalizes the first because any continuous semiring is complete [ÉK03, Prop. 2.2]. The next lemma generalizes further [FMV11, Lm. 3.2] because simple (S, Σ, K) -rtgs, where K is a compressible M-monoid, generalize weighted tree automata over commutative and complete semirings.

Theorem 6.3. *If $S_{\text{true}, \text{id}} = S$ and K is compressible, then $\text{Reg}_{\text{simple}}(S, \Sigma, K) = \text{Reg}_{\text{nc}}(S, \Sigma, K)$.*

Proof: Since each chain-free (S, Σ, K) -rtg is simple, we only have to prove $\text{Reg}_{\text{simple}}(S, \Sigma, K) \subseteq \text{Reg}_{\text{nc}}(S, \Sigma, K)$. Let $\mathcal{G} = (N, Z, R, \text{wt})$ be a simple (S, Σ, K) -rtg. Recall that $P_{\mathcal{G}}$ and $F_{\mathcal{G}}$ are the finite sets of predicates and instructions, respectively, which occur in \mathcal{G} . Without loss of generality we can assume that for each $k \in \mathbb{N}$, $\sigma \in \Sigma^{(k)}$, $A, B_1, \dots, B_k \in N$, $p \in P_{\mathcal{G}}$, and $f_1, \dots, f_k \in F_{\mathcal{G}}$, there is a rule $r = (A(p) \rightarrow \sigma(B_1(f_1), \dots, B_k(f_k)))$ in R . If there is no such rule, then we can add it to R and let $\text{wt}(r) = 0_k$. In a similar way, we can assume that for each $A, A' \in N$ there is a rule $r = (A(\text{true}_C) \rightarrow A'(\text{id}_C))$ in R .

We let W be the $(N \times N)$ -matrix over $\Omega^{(1)}$ such that

$$W_{A, A'} = \text{wt}(A(\text{true}_C) \rightarrow A'(\text{id}_C))$$

for each $A, A' \in N$.

We construct the chain-free (S, Σ, K) -rtg $\mathcal{G}' = (N, Z, R', \text{wt}')$ as follows. For each $k \in \mathbb{N}$, $\sigma \in \Sigma^{(k)}$, $A, B_1, \dots, B_k \in N$, $p \in P_{\mathcal{G}}$, and $f_1, \dots, f_k \in F_{\mathcal{G}}$ there is a rule $r' = (A(p) \rightarrow \sigma(B_1(f_1), \dots, B_k(f_k)))$ in R' and

$$\text{wt}'(r') = \sum_{A' \in N} \left((W^*)_{A, A'} \circ \text{wt}(A'(p) \rightarrow \sigma(B_1(f_1), \dots, B_k(f_k))) \right).$$

Since K has identity, it is $(1, 1)$ -composition closed, and completely 1-sum closed, each entry of the matrix W^* is in $\Omega^{(1)}$. Moreover, since K is $(1, k)$ -composition closed, the right-hand side of the above equality is an operation in $\Omega^{(k)}$. Hence, $\text{wt}'(r')$ is well defined.

We define the family $\text{eff} = (\text{eff}_{A, \xi, c} \mid \xi \in T_{\Sigma}, A \in N, c \in C)$ of mappings

$$\text{eff}_{A, \xi, c} : D_{\mathcal{G}}(A, \xi, c) \rightarrow D_{\mathcal{G}'}(A, \xi, c)$$

as follows. Let $\xi = \sigma(\xi_1, \dots, \xi_k)$ and $d \in D_{\mathcal{G}}(A, \xi, c)$. Then

- there are $n \geq 0$ and rules $r_1 = (A_1(\text{true}_C) \rightarrow A_2(\text{id}_C)), \dots, r_n = (A_n(\text{true}_C) \rightarrow A_{n+1}(\text{id}_C))$, and $A_{n+1}(p) \rightarrow \sigma(B_1(f_1), \dots, B_k(f_k))$ in R such that $p(c) = \text{true}$ and
- for each $i \in [k]$ the function f_i is defined on c and there is a derivation tree $d_i \in D_{\mathcal{G}}(B_i, \xi_i, f_i(c))$

such that

$$d = r_1 \dots r_n (A_{n+1}(p) \rightarrow \sigma(B_1(f_1), \dots, B_k(f_k))) (d_1, \dots, d_k) .$$

We define

$$\text{eff}_{A, \xi, c}(d) = (A_1(p) \rightarrow \sigma(B_1(f_1), \dots, B_k(f_k))) (d'_1, \dots, d'_k),$$

where $d'_i = \text{eff}_{B_i, \xi_i, f_i(c)}(d_i)$ for each $i \in [k]$. Note that $\text{lhs}_N(d'_i(\varepsilon)) = B_i$.

Then we can prove:

$$\begin{aligned}
\llbracket \mathcal{G} \rrbracket(\xi) &= \sum_{d \in D_{\mathcal{G}}(\xi)} \text{wt}(d) = \sum_{A \in Z} \sum_{d \in D_{\mathcal{G}}(A, \xi, c_0)} \text{wt}(d) \\
&= \sum_{A \in Z} \sum_{d' \in D_{\mathcal{G}'}(A, \xi, c_0)} \sum_{\substack{d \in D_{\mathcal{G}}(A, \xi, c_0): \\ \text{eff}_{A, \xi, c_0}(d) = d'}} \text{wt}(d) \\
&=^{(*)} \sum_{A \in Z} \sum_{d' \in D_{\mathcal{G}'}(A, \xi, c_0)} \text{wt}'(d') = \sum_{d' \in D_{\mathcal{G}'}(\xi)} \text{wt}'(d') = \llbracket \mathcal{G}' \rrbracket(\xi) .
\end{aligned}$$

At (*) we have used the following statement: for each $\xi \in T_{\Sigma}$, $A \in N$, $c \in C$, and $d' \in D_{\mathcal{G}'}(A, \xi, c)$:

$$\sum_{\substack{d \in D_{\mathcal{G}}(A, \xi, c): \\ \text{eff}_{A, \xi, c}(d) = d'}} \text{wt}(d) = \text{wt}'(d') .$$

We prove this statement by induction on ξ . We only show the induction step because it contains the base of the induction.

Let $\xi = \sigma(\xi_1, \dots, \xi_k)$ for some $k \in \mathbb{N}$. Let $A \in N$, $c \in C$, and $d' \in D_{\mathcal{G}'}(A, \xi, c)$. Then

- there is a rule $r' = (A(p) \rightarrow \sigma(B_1(f_1), \dots, B_k(f_k)))$ in R' such that $p(c) = \text{true}$ and $f_i(c)$ is defined and
- for each $i \in [k]$ there is a $d'_i \in D_{\mathcal{G}'}(B_i, \xi_i, f_i(c))$

such that $d' = r'(d'_1, \dots, d'_k)$. Then we can calculate as follows (by abbreviating $\text{wt}(B(\text{true}_C) \rightarrow B'(\text{id}_C))$ by $\text{wt}(B, B')$ and $\text{wt}(B(p) \rightarrow \sigma(B_1(f_1), \dots, B_k(f_k)))$ by $\text{wt}(B(p), \sigma(B_1(f_1), \dots, B_k(f_k)))$); moreover, we use $\bigcirc_{j=1}^n \omega_j$ to denote the composition $\omega_1 \circ \dots \circ \omega_n$ of n unary operations):

$$\begin{aligned}
&\sum_{\substack{d \in D_{\mathcal{G}}(A, \xi, c): \\ \text{eff}_{A, \xi, c}(d) = r'(d'_1, \dots, d'_k)}} \text{wt}(d) \\
&= \sum_{\substack{n \in \mathbb{N} \\ A_1 = A}} \sum_{A_1, \dots, A_{n+1} \in N} \sum_{\substack{d_1 \in D_{\mathcal{G}}(B_1, \xi_1, f_1(c)), \dots, d_k \in D_{\mathcal{G}}(B_k, \xi_k, f_k(c)): \\ \forall i \in [k]: \text{eff}_{B_i, \xi_i, f_i(c)}(d_i) = d'_i}} \\
&\quad \bigcirc_{j=1}^n \text{wt}(A_j, A_{j+1}) \circ \text{wt}(A_{n+1}(p), \sigma(B_1(f_1), \dots, B_k(f_k))) \left(\text{wt}(d_1), \dots, \text{wt}(d_k) \right) \\
&= \sum_{n \in \mathbb{N}} \sum_{\substack{A_1, \dots, A_{n+1} \in N: \\ A_1 = A}} \bigcirc_{j=1}^n \text{wt}(A_j, A_{j+1}) \circ \text{wt}(A_{n+1}(p), \sigma(B_1(f_1), \dots, B_k(f_k))) \\
&\quad \left(\sum_{\substack{d_1 \in D_{\mathcal{G}}(B_1, \xi_1, f_1(c)): \\ \text{eff}_{B_1, \xi_1, f_1(c)}(d_1) = d'_1}} \text{wt}(d_1), \dots, \sum_{\substack{d_k \in D_{\mathcal{G}}(B_k, \xi_k, f_k(c)): \\ \text{eff}_{B_k, \xi_k, f_k(c)}(d_k) = d'_k}} \text{wt}(d_k) \right)
\end{aligned}$$

(because K is completely distributive)

$$\begin{aligned}
&= \sum_{n \in \mathbb{N}} \sum_{\substack{A_1, \dots, A_{n+1} \in \mathcal{N}: \\ A_1 = A}} \bigcirc_{j=1}^n \text{wt}(A_j, A_{j+1}) \circ \text{wt}(A_{n+1}(p), \sigma(B_1(f_1), \dots, B_k(f_k))) \\
&\quad \left(\text{wt}'(d'_1), \dots, \text{wt}'(d'_k) \right) \\
&\quad \text{(due to I.H.)} \\
&= \sum_{A' \in \mathcal{N}} \sum_{n \in \mathbb{N}} \sum_{\substack{A_1, \dots, A_{n+1} \in \mathcal{N}: \\ A_1 = A, A_{n+1} = A'}} \bigcirc_{j=1}^n \text{wt}(A_j, A_{j+1}) \circ \text{wt}(A'(p), \sigma(B_1(f_1), \dots, B_k(f_k))) \\
&\quad \left(\text{wt}'(d'_1), \dots, \text{wt}'(d'_k) \right) \\
&\quad \text{(by renaming of } A_{n+1} \text{ by } A') \\
&= \sum_{A' \in \mathcal{N}} \left(\left(\sum_{n \in \mathbb{N}} \sum_{\substack{A_1, \dots, A_{n+1} \in \mathcal{N}: \\ A_1 = A, A_{n+1} = A'}} \bigcirc_{j=1}^n \text{wt}(A_j, A_{j+1}) \right) \circ \text{wt}(A'(p), \sigma(B_1(f_1), \dots, B_k(f_k))) \right) \\
&\quad \left(\text{wt}'(d'_1), \dots, \text{wt}'(d'_k) \right) \\
&\quad \text{(by Observation 6.2)} \\
&= \sum_{A' \in \mathcal{N}} \left(\left(\sum_{n \in \mathbb{N}} (W^n)_{A, A'} \right) \circ \text{wt}(A'(p), \sigma(B_1(f_1), \dots, B_k(f_k))) \right) \left(\text{wt}'(d'_1), \dots, \text{wt}'(d'_k) \right) \\
&\quad \text{(by definition of } W^n \text{)} \\
&= \sum_{A' \in \mathcal{N}} \left((W^*)_{A, A'} \circ \text{wt}(A'(p), \sigma(B_1(f_1), \dots, B_k(f_k))) \right) \left(\text{wt}'(d'_1), \dots, \text{wt}'(d'_k) \right) \\
&= \left(\sum_{A' \in \mathcal{N}} (W^*)_{A, A'} \circ \text{wt}(A'(p), \sigma(B_1(f_1), \dots, B_k(f_k))) \right) \left(\text{wt}'(d'_1), \dots, \text{wt}'(d'_k) \right) \\
&\quad \text{(by Observation 6.2)} \\
&= \text{wt}'(r') \circ \left(\text{wt}'(d'_1), \dots, \text{wt}'(d'_k) \right) \\
&\quad \text{(by construction of } \mathcal{G}' \text{)} \\
&= \text{wt}'(r'(d'_1, \dots, d'_k)).
\end{aligned}$$

□

We can instantiate the previous theorem to (1) the trivial storage type and (2) the Boolean M-monoid and obtain the following corollary.

Corollary 6.4.

1. If K is compressible, then $\text{Reg}(\Sigma, K) = \text{Reg}_{\text{nc}}(\Sigma, K)$.
2. If $S_{\text{true}, \text{id}} = S$, then $\text{Reg}_{\text{simple}}(S, \Sigma) = \text{Reg}_{\text{nc}}(S, \Sigma)$. In particular, for each $n \in \mathbb{N}$ we have $\text{Reg}_{\text{simple}}(\mathbb{P}^n, \Sigma) = \text{Reg}_{\text{nc}}(\mathbb{P}^n, \Sigma)$.

Proof: First we prove Statement 1. Since $\text{TRIV}_{\text{true}, \text{id}} = \text{TRIV}$ and each (Σ, K) -rtg is simple, the statement follows from Theorem 6.3.

Then we prove Statement 2 as follows. Since the Boolean M-monoid \mathbb{B} is compressible, the first statement follows from Theorem 6.3. Then the second follows from the first one, because $(\mathbb{P}^n)_{\text{true}, \text{id}} = \mathbb{P}^n$. \square

For an arbitrary compressible M-monoid, we can even go beyond the trivial storage type and prove the following chain rule elimination result for particular finite storage types.

Corollary 6.5. *If K is compressible, S is finite, and $S_{\text{true}, \text{id}} = S$, then $\text{Reg}(S, \Sigma, K) = \text{Reg}_{\text{simple}}(S, \Sigma, K) = \text{Reg}_{\text{nc}}(\Sigma, K)$.*

Proof: We have $\text{Reg}(S, \Sigma, K) = \text{Reg}(\Sigma, K) = \text{Reg}_{\text{nc}}(\Sigma, K)$ by Corollary 5.6 and Corollary 6.4(1), respectively. The inclusion $\text{Reg}_{\text{nc}}(\Sigma, K) \subseteq \text{Reg}_{\text{nc}}(S, \Sigma, K)$ follows from Corollary 5.5(2), and the inclusions $\text{Reg}_{\text{nc}}(S, \Sigma, K) \subseteq \text{Reg}_{\text{simple}}(S, \Sigma, K)$ and $\text{Reg}_{\text{simple}}(S, \Sigma, K) \subseteq \text{Reg}(S, \Sigma, K)$ are obvious. \square

7 Büchi-Elgot-Trakhtenbrot theorem

By the classical results of Büchi [Büc60, Büc62], Elgot [Elg61], and Trakhtenbrot [Tra61], recognizable languages are the same as languages definable in monadic second order logic (MSO-logic). We call this characterization Büchi-Elgot-Trakhtenbrot theorem and in this section we present a corresponding one for weighted tree languages generated by (S, Σ, K) -rtg (including chain rules). For this, we will introduce a weighted MSO-logic with storage behaviour, where the weights are taken from a complete M-monoid. Each formula, called expression, of this logic is interpreted over finite, labeled, and ordered trees. Our new weighted MSO-logic generalizes (i) the weighted MSO-logic with storage behaviour of [VDH16] by considering trees as models, and (ii) M-expressions of [FSV12, FV18] by adding storage type. Our Büchi-Elgot-Trakhtenbrot theorem states that weighted tree languages generated by (S, Σ, K) -rtg are the same as weighted tree languages definable by expressions. Thus, our result generalizes the corresponding one of [VDH16] in the sense that we allow chain rules on the rtg side (which corresponds to ε -transitions for the automata considered in [VDH16]).

We note that in [DV11, Ch. 7] an alternative weighted MSO-logic was used for the characterization of the class $\text{Reg}_{\text{nc}}(\Sigma, M_K)$ where K is an arbitrary semiring. In its turn, that logic is based on the weighted MSO-logic in [DG05, DG07, DG09] for weighted string automata. For a recent survey we refer to [GM18].

Since the formulas of our new weighted MSO-logic generalize M-expressions of [FSV12], we recall the syntax of M-expressions. For the definitions of all semantic notions, like variable assignment, correspondence between (i) pairs of a tree and a variable assignment and (ii) trees over some extended ranked alphabet, and semantics of M-expressions we refer to [FSV12]; here we only recall some of them.

First, we recall the (unweighted) MSO-logic for trees [GS97]. For this, let Θ be an arbitrary ranked alphabet. We define the *set of formulas of MSO-logic over Θ* , denoted by $\text{MSO}(\Theta)$, as the language generated by the following EBNF with nonterminals ψ and φ and with initial nonterminal φ :

$$\begin{aligned}\psi &::= \text{label}_\sigma(x) \mid \text{edge}_i(x, y) \mid (x \in X) \\ \varphi &::= \psi \mid \neg\varphi \mid (\varphi \vee \varphi) \mid (\varphi \wedge \varphi) \mid \exists x.\varphi \mid \forall x.\varphi \mid \exists X.\varphi \mid \forall X.\varphi\end{aligned}$$

where $\sigma \in \Theta$ and $i \in [\text{maxrk}_\Theta]$. For any finite set \mathcal{V} of first-order or second-order variables we define the ranked alphabet $\Theta_{\mathcal{V}}$ by $\Theta_{\mathcal{V}}^{(k)} = \Theta^{(k)} \times \mathcal{P}(\mathcal{V})$. We denote the set of all trees in $T_{\Theta_{\mathcal{V}}}$ in which each first-order variable of \mathcal{V} occurs exactly once by $T_{\Theta_{\mathcal{V}}}^v$. It is well known that such trees can be identified with pairs of a tree and an assignment to variables in \mathcal{V} . For an MSO-formula φ over Θ with free variables contained in \mathcal{V} , we let

$$\mathcal{L}_{\mathcal{V}}(\varphi) = \{ \xi \in T_{\Theta_{\mathcal{V}}}^v \mid \varphi \models \xi \} ,$$

the set of models of φ . The models operator \models is defined straightforwardly as for (classical) MSO-formulas for the string case (cf. [Str84, Ch. II.2]).

Second, we recall from [FSV12, Def. 3.1] the definition of M-expressions. Note that K is some arbitrary complete M-monoid. The atomic M-expressions have the form $H(\omega)$ where $\omega = (\omega_\sigma \mid \sigma \in \Theta_{\mathcal{U}})$ is a $\Theta_{\mathcal{U}}$ -family of operations for some finite set \mathcal{U} of variables; moreover, we require that $\omega_\sigma \in \Omega^{(k)}$ for every $k \in \mathbb{N}$ and $\sigma \in \Theta_{\mathcal{U}}^{(k)}$. The set of *M-expressions over (Θ, K)* , denoted by $\text{MExp}(\Theta, K)$, is the set of all formulas generated by the following EBNF with nonterminal E :

$$E ::= H(\omega) \mid (E + E) \mid (\varphi \triangleright E) \mid \sum_x E \mid \sum_X E ,$$

where ω is a $\Theta_{\mathcal{U}}$ -family of operations in Ω for some finite set \mathcal{U} of variables, and $\varphi \in \text{MSO}(\Theta)$. A sentence is an M-expression without free variables.

The semantics of $H(\omega)$ with $\omega = (\omega_\sigma \mid \sigma \in \Theta_{\mathcal{U}})$ is based on the following concept from universal algebra. Since (K, ω) is a $\Theta_{\mathcal{U}}$ -algebra, there is a unique $\Theta_{\mathcal{U}}$ -homomorphism from the $\Theta_{\mathcal{U}}$ -term algebra $T_{\Theta_{\mathcal{U}}}$ to (K, ω) [Wec92, Thm. 4]; we denote this homomorphism by h_ω . Then the semantics of $H(\omega)$ on a tree ξ is obtained by applying h_ω to ξ (after adaptation of ω to the set of free variables occurring in ξ , cf. [FSV12, p. 249]).

Intuitively, the semantics of $(\varphi \triangleright e)$ on a tree ξ is the semantics of e on ξ if ξ is a model of φ , otherwise it is 0. Formulas of the form $(e_1 + e_2)$, $\sum_x e$, and $\sum_X e$ are interpreted by employing the summation operation of K in the usual way (viewing $\sum_x e$ and $\sum_X e$ as weighted first-order existential quantification and weighted second-order existential quantification, respectively).

The semantics of a sentence $e \in \text{MExp}(\Theta, K)$ is a weighted tree language $\llbracket e \rrbracket : T_\Theta \rightarrow K$ defined in [FSV12, Def. 3.3]. We say that a weighted tree language $s : T_\Theta \rightarrow K$ is *M-definable* if there is a sentence e such that $\llbracket e \rrbracket = s$. We denote by $\text{M}(\Theta, K)$ the class of all weighted tree languages which are M-definable by some sentence $e \in \text{MExp}(\Theta, K)$.

We recall the main theorem of [FSV12] (using Observation 3.3).

Theorem 7.1. [FSV12, Thm. 4.1] $\text{Reg}_{\text{nc}}(\Theta, K) = \text{M}(\Theta, K)$ for each ranked alphabet Θ .

Now we define the main logic of this paper: (weighted) expressions with behaviour. We recall that $S = (C, P, F, c_0)$ denotes an arbitrary storage type and Σ is a ranked alphabet. In the sequel, we let

$P' \subseteq P$ and $F' \subseteq F$ be finite non-empty sets and we let Δ be the ranked alphabet corresponding to Σ , P' , and F' (cf. Section 2.4).

In a similar spirit as in [VDH16], an expression is an existentially quantified M-expression where the quantification runs over the set $\mathcal{B}_\Delta(\xi)$ of Δ -behaviours on the tree $\xi \in T_\Sigma$ over which the expression is interpreted. The involved M-expression is over $(\langle \Delta, \Sigma \rangle, K)$.

Definition 7.2. We define the set of *expressions over Σ and K with Δ -behaviours* (for short: (Δ, Σ, K) -expressions) to be the set of all formulas of the form

$$\sum^{\text{beh}} e$$

where $e \in \text{MExp}(\langle \Delta, \Sigma \rangle, K)$ is a sentence. The *semantics* of $\sum^{\text{beh}} e$ is the weighted tree language $\llbracket \sum^{\text{beh}} e \rrbracket : T_\Sigma \rightarrow K$ defined by

$$\llbracket \sum^{\text{beh}} e \rrbracket = \mathcal{B}_\Delta; \llbracket e \rrbracket. \quad \square$$

Let $s : T_\Sigma \rightarrow K$ be a weighted tree language. We say that s is (Δ, Σ, K) -*definable* if there is a (Δ, Σ, K) -expression $\sum^{\text{beh}} e$ with $\llbracket \sum^{\text{beh}} e \rrbracket = s$. Moreover, s is (S, Σ, K) -*definable* if there are non-empty finite subsets $P' \subseteq P$ and $F' \subseteq F$ such that s is (Δ, Σ, K) -definable where Δ is the ranked alphabet corresponding to Σ , P' , and F' . We denote the *class of all (S, Σ, K) -definable weighted tree languages* by $\text{Def}(S, \Sigma, K)$.

Now we want to compare (Δ, Σ, K) -expressions of Definition 7.2 with (Δ, Σ, K) -expressions of [VDH16, Def. 5 and 6]. (Since in [VDH16] Ω refers to a finite subset of $P \times F$, whereas in this paper Ω is used as set of operations of an M-monoid, we have replaced in the notations of [VDH16] Ω by Δ .) The following table gives an overview on this comparison.

| | [VDH16] | present paper |
|-----------------|--|---|
| structures: | string $u \in \Sigma^*$ | tree $\xi \in T_\Sigma$ |
| behaviours: | $b \in \Delta^*$ with $\text{pos}(b) = \text{pos}(u)$ | $\zeta \in \mathcal{B}_\Delta(\xi) \subseteq T_{\langle \Delta, \Sigma \rangle}$ and $\text{pos}(\xi)$ can be embedded into $\text{pos}(\zeta)$ (cf. Sect. 5) |
| weight algebra: | valuation monoid $(K, +, 0, \text{Val})$ | complete M-monoid $(K, +, 0, \Omega)$ |
| expressions: | $\sum_B^{\text{beh}} e$ where e is a B-expression over (Δ, Σ, K) | $\sum^{\text{beh}} e$ where e is an M-expression of [FSV12] over $(\langle \Delta, \Sigma \rangle, K)$ |
| semantics: | $\llbracket \sum_B^{\text{beh}} e \rrbracket(u) = \sum_{b \in \mathcal{B}(\Delta, u)} \llbracket e \rrbracket_{\{B\}}(u, [B \mapsto b])$ | $\llbracket \sum^{\text{beh}} e \rrbracket(\xi) = \sum_{\zeta \in \mathcal{B}_\Delta(\xi)} \llbracket e \rrbracket(\zeta)$ |

There are two significant differences between the two approaches:

1) The behaviour b in [VDH16] has the same shape as the structure u over which the formula is interpreted (i.e., $\text{pos}(b) = \text{pos}(u)$). In the present paper, the behaviour ζ is obtained by replacing, at each

position $w \in \text{pos}(\xi)$, the tree fragment $\xi(w)(x_1, \dots, x_k)$ by the tree fragment

$$\langle (p_1, g_1), * \rangle \left(\dots \langle (p_n, g_n), * \rangle \left(\langle (p, f_1 \dots f_k), \xi(w) \rangle (x_1, \dots, x_k) \right) \dots \right)$$

for some $n \in \mathbb{N}$, $\langle (p_i, g_i), * \rangle \in \Delta^{(1)} \times \{*\}$, and $\langle (p, f_1 \dots f_k), \xi(w) \rangle \in \Delta^{(k)} \times \Sigma^{(k)}$. Then we might say that the sequence

$$\langle (p_1, g_1), * \rangle \dots \langle (p_n, g_n), * \rangle \langle (p, f_1 \dots f_k), \xi(w) \rangle$$

is the *segment of ζ belonging to w* .

In particular, ξ can be embedded into ζ , i.e., there is a unique bijection $\theta_{\xi, \zeta}: \text{pos}(\xi) \rightarrow \text{pos}_\Theta(\zeta)$ which preserves the lexicographic order, where $\Theta = \langle \Delta, \Sigma \rangle \setminus (\Delta^{(1)} \times \{*\})$ (cf. Section 5 and Fig. 4). This extension of ξ will allow us to cope with chain rules in our Büchi-Elgot-Trakhtenbrot theorem.

2) B-expressions of [VDH16] are almost the same as M-expressions except that, in addition, there is a variable B , which is assigned to a behaviour, and in a B-expression of the form $\varphi \triangleright e$, the guard formula φ may contain atomic formulas of the form $(B(x) = (p, f))$; they allow to check whether, for a variable assignment ρ , the behaviour $\rho(B)$ carries the symbol (p, f) at position $\rho(x)$. In M-expressions such atomic formulas do not occur, and the M-expression e in the formula $\sum^{\text{beh}} e$ is a sentence, i.e., does not contain any free variable (in particular, it does not contain the variable B). The information about the behaviour is coded into the tree $\zeta \in T_{\langle \Delta, \Sigma \rangle}$ over which the M-expression e is interpreted. For instance, an atomic formula of the form $(B(x) = (p, f))$ occurring in a B-expression will be represented by the formula $\bigvee_{\sigma \in \Sigma^{(1)}} \text{label}_{\langle (p, f), \sigma \rangle}(x)$ in the M-expression (note that this replacement only makes sense if the variable x is associated with a unary position). This modular definition of syntax and semantics makes it possible to apply directly results on M-expressions known from the literature. In particular, when we will prove the Büchi-Elgot-Trakhtenbrot theorem, there is no need for a proof by induction on the structure of M-expressions to show that such formulas induce recognizability/regularity (as it was necessary for B-expression in [VDH16]); instead, we can directly apply the appropriate result from the literature.

Example 7.3. Here we show an example of our new logic. For this, recall from Example 3.1 the $(P^1, \Sigma, M_{\mathbb{N}_\infty})$ -recognizable tree language s , which maps each tree of the form $\sigma(\delta^n(\alpha), \delta^n(\alpha))$ to 8^n ($n \geq 0$) and any other tree to 0. This tree language can be defined by a $(\Delta, \Sigma, M_{\mathbb{N}_\infty})$ -expression $\sum^{\text{beh}} e$ for the ranked alphabet Δ corresponding to Σ , $P' = \{\text{true}, \text{top} = \gamma_0, \text{top} = \gamma\}$, and $F' = \{\text{push}(\gamma), \text{pop}\}$ and an appropriate sentence $e \in \text{MExp}(\langle \Delta, \Sigma \rangle, M_{\mathbb{N}_\infty})$.

For the specification of e we need some auxiliary formulas. We define the binary relation $\text{edge}(x, y)$ by

$$\text{edge}(x, y) = \text{edge}_1(x, y) \vee \dots \vee \text{edge}_{\max \text{rk}(\langle \Delta, \Sigma \rangle)}(x, y) .$$

Moreover, we define the binary relation $\text{edge}^+(x, y)$ to be the transitive closure of $\text{edge}(x, y)$; recall from [CE12, p. 43] that it can be defined by a formula in $\text{MSO}(\langle \Delta, \Sigma \rangle)$. Finally, we denote the formula $\neg\varphi \vee \psi$ by $\varphi \rightarrow \psi$ for every $\varphi, \psi \in \text{MSO}(\langle \Delta, \Sigma \rangle)$.

Now we construct

$$e = \varphi \triangleright H(w) \quad \text{with} \quad \varphi = \exists x. (\varphi_{\text{label}}(x) \wedge \varphi_{\text{above}}(x) \wedge \varphi_{\text{below}}(x)).$$

Intuitively, for every $\xi \in T_\Sigma$ and behaviour $\zeta \in \mathcal{B}_\Delta(\xi)$ on ξ , the formula φ determines a position $w \in \text{pos}(\zeta)$ such that the following holds (recall the definition of $\theta_{\xi, \zeta}$ from the beginning of Section 5):

- $(\zeta, [x \mapsto v]) \models \varphi_{\text{label}}(x) \wedge \varphi_{\text{above}}(x)$ iff the segment of ζ belonging to $\theta_{\xi, \zeta}^{-1}(v)$ has the form

$$\langle (\text{true}, \text{push}(\gamma)), * \rangle^n \langle (\text{true}, \text{id}_{\Gamma^+} \text{id}_{\Gamma^+}), \sigma \rangle$$

for some $n \in \mathbb{N}$ where σ is the label of ξ at $\theta_{\xi, \zeta}^{-1}(v)$

- $(\zeta, [x \mapsto v]) \models \varphi_{\text{below}}(x)$ iff for each position $v' \in \text{pos}(\zeta)$ below v , the segment of ζ belonging to $\theta_{\xi, \zeta}^{-1}(v')$ has the form

$$\langle (\text{top} = \gamma, \text{pop}), \delta \rangle \text{ or } \langle (\text{top} = \gamma_0, \varepsilon), \alpha \rangle .$$

Formally, let

- $\varphi_{\text{label}}(x) = \text{label}_{\langle (\text{true}, \text{id}_{\Gamma^+} \text{id}_{\Gamma^+}), \sigma \rangle}(x)$,
- $\varphi_{\text{above}}(x) = \forall y. \text{edge}^+(y, x) \rightarrow \text{label}_{\langle (\text{true}, \text{push}(\gamma)), * \rangle}(y)$,
- $\varphi_{\text{below}}(x) = \forall y. \text{edge}^+(x, y) \rightarrow (\text{label}_{\langle (\text{top} = \gamma, \text{pop}), \delta \rangle}(y) \vee \text{label}_{\langle (\text{top} = \gamma_0, \varepsilon), \alpha \rangle}(y))$.

Moreover, we define the $\langle \Delta, \Sigma \rangle$ -family of operations ω by letting

$$\begin{aligned} \omega_{\langle (\text{true}, \text{id}_{\Gamma^+} \text{id}_{\Gamma^+}), \sigma \rangle} &= \text{mul}_{2,1}, \\ \omega_{\langle (\text{true}, \text{push}(\gamma)), * \rangle} &= \omega_{\langle (\text{top} = \gamma, \text{pop}), \delta \rangle} = \text{mul}_{1,2}, \\ \omega_{\langle (\text{top} = \gamma_0, \varepsilon), \alpha \rangle} &= \text{mul}_{0,1}. \end{aligned}$$

This finishes the construction of $e = \varphi \triangleright \mathbf{H}(\omega)$. Next we show that $\llbracket \sum^{\text{beh}} e \rrbracket = s$. For this, let $\xi \in T_{\Sigma}$. Then:

$$\begin{aligned} \llbracket \sum^{\text{beh}} e \rrbracket(\xi) &= (\mathcal{B}_{\Delta}; \llbracket e \rrbracket)(\xi) = \sum_{\zeta \in \mathcal{B}_{\Delta}(\xi)} \llbracket e \rrbracket(\zeta) \\ &= \sum_{\zeta \in \mathcal{B}_{\Delta}(\xi)} \llbracket \varphi \triangleright \mathbf{H}(\omega) \rrbracket(\zeta) = \sum_{\zeta \in \mathcal{B}_{\Delta}(\xi) \cap \mathcal{L}_{\theta}(\varphi)} \llbracket \mathbf{H}(\omega) \rrbracket(\zeta) . \end{aligned}$$

Now we analyse the set of models of φ . If $\xi \in \text{supp}(s)$, i.e., $\xi = \sigma(\delta^n(\alpha), \delta^n(\alpha))$ for some $n \geq 0$, then the set $\mathcal{B}_{\Delta}(\xi) \cap \mathcal{L}_{\theta}(\varphi)$ consists of exactly one tree, denoted as ζ_{ξ} , that is determined as follows.

- If $n = 0$, then $\zeta_{\xi} = \langle b_1, \sigma \rangle \langle \langle b_2, \alpha \rangle, \langle b_2, \alpha \rangle \rangle$, and
- if $n \geq 1$, then $\zeta_{\xi} = \langle b_3, * \rangle^n \langle \langle b_4, \sigma \rangle (\zeta', \zeta') \rangle$

where $\zeta' = \langle b_5, \delta \rangle^n \langle \langle b_2, \alpha \rangle \rangle$ and $b_1 = (\text{true}, \text{id}_{\Gamma^+} \text{id}_{\Gamma^+})$, $b_2 = (\text{top} = \gamma_0, \varepsilon)$, $b_3 = (\text{true}, \text{push}(\gamma))$, $b_4 = (\text{true}, \text{id}_{\Gamma^+} \text{id}_{\Gamma^+})$, and $b_5 = (\text{top} = \gamma, \text{pop})$. Thus

$$\sum_{\zeta \in \mathcal{B}_{\Delta}(\xi) \cap \mathcal{L}_{\theta}(\varphi)} \llbracket \mathbf{H}(\omega) \rrbracket(\zeta) = \llbracket \mathbf{H}(\omega) \rrbracket(\zeta_{\xi}) .$$

Moreover, we have

$$\llbracket \mathbf{H}(\omega) \rrbracket(\zeta_{\xi}) = \underbrace{\text{mul}_{1,2}(\dots \text{mul}_{1,2}(\text{mul}_{2,1}(u, u)) \dots)}_n$$

with $u = \underbrace{\text{mul}_{1,2}(\dots \text{mul}_{1,2}(\text{mul}_{0,1}) \dots)}_n$ for every $n \geq 0$. Since $u = 2^n$, we obtain:

$$\llbracket \sum^{\text{beh}} e \rrbracket(\xi) = \llbracket \mathbf{H}(\omega) \rrbracket(\zeta_{\xi}) = 8^n = s(\xi) .$$

If $\xi \notin \text{supp}(s)$, then $\mathcal{B}_\Delta(\xi) \cap \mathcal{L}_\emptyset(\varphi) = \emptyset$ and thus we have

$$\llbracket \sum^{\text{beh}} e \rrbracket(\xi) = \sum_{\zeta \in \emptyset} \llbracket \mathbf{H}(\omega) \rrbracket(\zeta) = 0 .$$

Hence $\llbracket \sum^{\text{beh}} e \rrbracket = s$.

As first result we prove a Büchi-Elgot-Trakhtenbrot theorem for (S, Σ, K) -rtgs.

Theorem 7.4. $\text{Reg}(S, \Sigma, K) = \text{Def}(S, \Sigma, K)$.

Proof: First we prove $\text{Reg}(S, \Sigma, K) \subseteq \text{Def}(S, \Sigma, K)$. Let s be an (S, Σ, K) -regular weighted tree language. By Theorem 5.3 there are finite sets $P' \subseteq P$ and $F' \subseteq F$ and there is a chain-free $(\langle \Delta, \Sigma \rangle, K)$ -rtg \mathcal{G} such that Δ is the ranked alphabet corresponding to Σ , P' , and F' and $s = \mathcal{B}_\Delta; \llbracket \mathcal{G} \rrbracket$. By Theorem 7.1, there is a sentence $e \in \text{MExp}(\langle \Delta, \Sigma \rangle, K)$ such that $\llbracket \mathcal{G} \rrbracket = \llbracket e \rrbracket$. Then, by Definition 7.2, we have that s is (S, Σ, K) -definable.

By reading the above proof backwards, we obtain the proof of $\text{Def}(S, \Sigma, K) \subseteq \text{Reg}(S, \Sigma, K)$. \square

As second result we prove that expressions with behaviours generalize M-expressions as defined in [FSV12, FV18].

Theorem 7.5. *Let $s : T_\Sigma \rightarrow K$ be a weighted tree language. Then the following two statements hold.*

1. *If $s = \llbracket e \rrbracket$ for some sentence $e \in \text{MExp}(\Sigma, K)$, then s is (TRIV, Σ, K) -definable.*
2. *If K is compressible and s is (TRIV, Σ, K) -definable, then $s = \llbracket e \rrbracket$ for some sentence $e \in \text{MExp}(\Sigma, K)$.*

Proof: We abbreviate the only predicate $\text{true}_{\{e\}}$ and the only instruction $\text{id}_{\{e\}}$ of TRIV by true and id , respectively. Let Δ be the ranked alphabet corresponding to Σ , $\{\text{true}\}$, and $\{\text{id}\}$ and $\langle \Delta, \Sigma \rangle$ as defined in Subsection 2.4.

For each $\xi \in T_\Sigma$, the set $\mathcal{B}_\Delta(\xi)$ contains exactly one element ζ with $\text{pos}(\zeta) = \text{pos}(\xi)$. We denote this element by ζ_ξ . Then for each $w \in \text{pos}(\xi)$ we have $\zeta_\xi(w) = \langle (\text{true}, \text{id} \dots \text{id}), \xi(w) \rangle$ where the number of occurrences of id is the rank of $\xi(w)$.

Proof of 1: Let $e \in \text{MExp}(\Sigma, K)$ be a sentence. Let us construct the formula $\bar{e} \in \text{MExp}(\langle \Delta, \Sigma \rangle, K)$ that can be obtained from e by replacing each subformula of the form $\text{label}_\sigma(x)$ by $\text{label}_{\langle (\text{true}, \text{id} \dots \text{id}), \sigma \rangle}(x)$ where the number of occurrences of id is the rank of σ . Then $\llbracket \bar{e} \rrbracket(\zeta_\xi) = \llbracket e \rrbracket(\xi)$ for every $\xi \in T_\Sigma$. Moreover, let $e' = \varphi \triangleright \bar{e}$, where $\varphi = \neg \exists x. \text{label}_{\langle (\text{true}, \text{id}), * \rangle}(x)$. Then

$$\mathcal{L}_\emptyset(\varphi) = \{\zeta \in T_{\langle \Delta, \Sigma \rangle} \mid \neg \exists w \in \text{pos}(\zeta) : \zeta(w) = \langle (\text{true}, \text{id}), * \rangle\}.$$

Thus, for each $\xi \in T_\Sigma$ we have

$$\llbracket \sum^{\text{beh}} e' \rrbracket(\xi) = \sum_{\zeta \in \mathcal{B}_\Delta(\xi)} \llbracket e' \rrbracket(\zeta) = \sum_{\zeta \in \mathcal{B}_\Delta(\xi)} \llbracket \varphi \triangleright \bar{e} \rrbracket(\zeta) = \llbracket \bar{e} \rrbracket(\zeta_\xi) = \llbracket e \rrbracket(\xi) ,$$

where the last but one equality is due to the fact that $\mathcal{B}_\Delta(\xi) \cap \mathcal{L}_\emptyset(\varphi) = \{\zeta_\xi\}$.

Proof of 2: Let s be (TRIV, Σ, K) -definable. By Theorem 7.4 we have that s is (TRIV, Σ, K) -regular. Since K is compressible, Lemma 6.3 implies that s is even chain-free (TRIV, Σ, K) -regular. By Theorem 7.1 we obtain that there is a sentence $e \in \text{MExp}(\Sigma, K)$ such that $\llbracket e \rrbracket = s$. \square

Acknowledgments

We thank the anonymous reviewers for their helpful comments.

References

- [AB87] A. Alexandrakis and S. Bozapalidis. Weighted grammars and Kleene’s theorem. *Inform. Process. Lett.*, 24(1):1–4, 1987.
- [BN99] F. Baader and T. Nipkov. *Term Rewriting and All That*. Cambridge University Press, 1999.
- [Bor04] B. Borchardt. A pumping lemma and decidability problems for recognizable tree series. *Acta Cybernetica*, 16(4):509–544, 2004.
- [BR82] J. Berstel and C. Reutenauer. Recognizable formal power series on trees. *Theoretical Computer Science*, 18(2), 1982.
- [BR88] J. Berstel and C. Reutenauer. *Rational Series and Their Languages*, volume 12 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1988.
- [Bra69] W. S. Brainerd. Tree generating regular systems. *Inform. and Control*, 14:217–231, 1969.
- [Büc60] J.R. Büchi. Weak second-order arithmetic and finite automata. *Zeitschr. für math. Logik und Grundl. der Mathematik*, 6:66–92, 1960.
- [Büc62] J.R. Büchi. On a decision method in restricted second-order arithmetic. In *Proc. 1960 Int. Congr. for Logic, Methodology and Philosophy of Science*, pages 1–11. Stanford Univ. Press, 1962.
- [CDIV10] M. Ćirić, M. Droste, J. Ignjatović, and H. Vogler. Determinization of weighted finite automata over strong bimonoids. *Inform. Sci.*, 180(1):156–166, 2010.
- [CE12] B. Courcelle and J. Engelfriet. *Graph Structure and Monadic Second-Order Logic - A Language-Theoretic Approach*, volume 138 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, 2012.
- [Dam82] W. Damm. The IO- and OI-hierarchies. *Theoret. Comput. Sci.*, 20:95–208, 1982.
- [DG81] W. Damm and I. Guessarian. Combining T and level-N. In *Proc. of Mathematical Foundations of Computer Science 1981*, volume 118 of *LNCS*, pages 262–270. Springer-Verlag, 1981.
- [DG05] M. Droste and P. Gastin. Weighted automata and weighted logics. In L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, editors, *Automata, Languages and Programming – 32nd Int. Colloquium, ICALP 2005*, volume 3580 of *LNCS*, pages 513–525. Springer-Verlag, 2005.
- [DG07] M. Droste and P. Gastin. Weighted automata and weighted logics. *Theor. Comput. Sci.*, 380(1-2):69–86, 2007.

- [DG09] M. Droste and P. Gastin. Weighted automata and weighted logics. In W. Kuich M. Droste and H. Vogler, editors, *Handbook of Weighted Automata*, chapter 5. Springer-Verlag, 2009.
- [DGMM11] M. Droste, D. Götze, S. Märcker, and I. Meinecke. Weighted tree automata over valuation monoids and their characterization by weighted logics. In W. Kuich and G. Rahonis, editors, *Algebraic Foundations in Computer Science: Essays Dedicated to Symeon Bozapalidis on the Occasion of His Retirement*, volume 7020 of *LNCS*, pages 30–55. Springer-Verlag, 2011.
- [DH15] M. Droste and D. Heusel. The supports of weighted unranked tree automata. *Fundam. Inform.*, 136(1-2):37–58, 2015.
- [Dic13] L. E. Dickson. Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors. *American Journal of Mathematics*, 35:413–422, 1913.
- [DKV09] M. Droste, W. Kuich, and H. Vogler. *Handbook of Weighted Automata*. EATCS Monographs in Theoretical Computer Science. Springer-Verlag, 2009.
- [DM10] M. Droste and I. Meinecke. Describing average- and longtime-behavior by weighted MSO logics. In P. Hliněný and A. Kučera, editors, *35th Int. Symp. Mathematical Foundations of Computer Science (MFCS 2010)*, volume 6281 of *LNCS*, pages 537–548. Springer-Verlag, 2010.
- [DM11] M. Droste and I. Meinecke. Weighted automata and regular expressions over valuation monoids. *Intern. J. of Found. of Comp. Science*, 22(8):1829–1844, 2011.
- [DM12] M. Droste and I. Meinecke. Weighted automata and weighted MSO logics for average and long-time behaviors. *Inform. and Computation*, 220–221:44–59, 2012.
- [DSV10] M. Droste, T. Stüber, and H. Vogler. Weighted finite automata over strong bimonoids. *Information Sciences*, 180:156–166, 2010.
- [DV11] M. Droste and H. Vogler. Weighted logics for unranked tree automata. *Theory of Computing Systems*, 48(1):23–47, 2011.
- [DV12] M. Droste and H. Vogler. Weighted automata and multi-valued logics over arbitrary bounded lattices. *Theoretical Computer Science*, 418:14–36, 2012.
- [DV13] M. Droste and H. Vogler. The Chomsky-Schützenberger theorem for quantitative context-free languages. In M.-P. Béal and O. Carton, editors, *17th Int. Conf. on Developments in Language Theory (DLT 2013)*, volume 7907 of *LNCS*, pages 203–214. Springer-Verlag, 2013.
- [DV14] M. Droste and H. Vogler. The Chomsky-Schützenberger theorem for quantitative context-free languages. *Int. J. of Foundations of Comput. Sci.*, 25(8):955–969, 2014.
- [Eil74] S. Eilenberg. *Automata, Languages, and Machines – Volume A*, volume 59 of *Pure and Applied Mathematics*. Academic Press, 1974.
- [ÉK03] Z. Ésik and W. Kuich. Formal tree series. *J. Autom. Lang. Comb.*, 8(2):219–285, 2003.

- [ÉL02] Z. Ésik and H. Leiβ. Greibach normal form in algebraically complete semirings. In J.C. Bradfield, editor, *Computer Science Logic, 16th International Workshop, CSL 2002*, volume 2471 of *LNCS*, pages 135–150. Springer-Verlag, 2002.
- [ÉL07] Z. Ésik and G. Liu. Fuzzy tree automata. *Fuzzy Sets and Systems*, 158:1450–1460, 2007.
- [Elg61] C. C. Elgot. Decision problems of finite automata design and related arithmetics. *Trans. Amer. Math. Soc.*, 98:21–52, 1961.
- [Eng75] J. Engelfriet. Tree automata and tree grammars. Technical report, 1975. Technical Report DAIMI FN-10, Inst. of Mathematics, University of Aarhus, Department of Computer Science, Ny Munkegade, 8000 Aarhus C, Denmark, 1975; see also: arXiv:1510.02036v1 [cs.FL] 7 Oct 2015.
- [Eng86] J. Engelfriet. Context-free grammars with storage. Technical report, 1986. Technical Report 86-11, University of Leiden, 1986; see also: arXiv:1408.0683 [cs.FL], 2014.
- [EV86] J. Engelfriet and H. Vogler. Pushdown machines for the macro tree transducer. *Theoret. Comput. Sci.*, 42(3):251–368, 1986.
- [EV88] J. Engelfriet and H. Vogler. High level tree transducers and iterated pushdown tree transducers. *Acta Inform.*, 26:131–192, 1988.
- [FMV09] Z. Fülöp, A. Maletti, , and H. Vogler. A Kleene theorem for weighted tree automata over distributive multioperator monoids. *Theory Comput. Syst.*, 44:455–499, 2009.
- [FMV11] Z. Fülöp, A. Maletti, and H. Vogler. Weighted extended tree transducers. *Fundamenta Informaticae*, 112:1–39, 2011.
- [FSV12] Z. Fülöp, T. Stüber, and H. Vogler. A Büchi-like theorem for weighted tree automata over multioperator monoids. *Theory of Computing Systems*, 50(2):241–278, 2012. published online 28.10.2010, doi:10.1007/s00224-010-9296-1.
- [FV09] Z. Fülöp and H. Vogler. Weighted tree automata and tree transducers. In M. Droste, W. Kuich, and H. Vogler, editors, *Handbook of Weighted Automata*, chapter 9, pages 313–403. Springer-Verlag, 2009.
- [FV18] Z. Fülöp and H. Vogler. Characterizations of recognizable weighted tree languages by logic and bimorphisms. *Soft Computing*, 22:1035–1046, 2018.
- [GM18] P. Gastin and B. Monmege. A unifying survey on weighted logics and weighted automata. 22:1047–1065, 2018.
- [Gol99] J.S. Golan. *Semirings and their Applications*. Kluwer Academic Publishers, Dordrecht, 1999.
- [Göt16] D. Götze. *Weighted Unranked Tree Automata over Tree Valuation Monoids*. PhD thesis, Universität Leipzig, 2016.

- [GS84] F. Gécseg and M. Steinby. *Tree Automata*. Akadémiai Kiadó, Budapest, 1984. see also: arXiv:1509.06233v1 [cs.FL] 21 Sep 2015.
- [GS97] F. Gécseg and M. Steinby. Tree languages. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3, chapter 1, pages 1–68. Springer-Verlag, 1997.
- [GTWW77] J.A. Goguen, J.W. Thatcher, E.G. Wagner, and J.B. Wright. Initial algebra semantics and continuous algebras. *J. ACM*, 24:68–95, 1977.
- [Gue83] I. Guessarian. Pushdown tree automata. *Math. Systems Theory*, 16:237–263, 1983.
- [HV15] L. Herrmann and H. Vogler. A Chomsky-Schützenberger theorem for weighted automata with storage. In A. Maletti, editor, *6th Int. Conf. on Algebraic Informatics (CAI 2015)*, volume 9270 of *LNCS*, pages 115–127. Springer-Verlag, 2015.
- [HV16] L. Herrmann and H. Vogler. Weighted symbolic automata with data storage. volume 9840 of *LNCS*, pages 203–215. Springer-Verlag, 2016.
- [HW98] U. Hebisch and H.J. Weinert. *Semirings - Algebraic Theory and Applications in Computer Science*. World Scientific, Singapore, 1998.
- [Kir11] D. Kirsten. The support of a recognizable series over a zero-sum free, commutative semiring is recognizable. *Acta Cybern.*, 20(2):211–221, 2011.
- [Knu68] D.E. Knuth. Semantics of context-free languages. *Mathematical Systems Theory*, 2:127–145, 1968. Corrections in *Math. Systems Theory* 5 (1971), 95–96.
- [KR08] M. Kreuzer and L. Robbiano. *Computational Commutative Algebra 1*. Springer Publishing Company, Incorporated, 2008.
- [KS86] W. Kuich and A. Salomaa. *Semirings, Automata, Languages*, volume 5 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1986.
- [Kui98] W. Kuich. Formal power series over trees. In S. Bozapalidis, editor, *Proc. Developments in Language Theory (DLT)*, pages 61–101. Aristotle University of Thessaloniki, 1998.
- [Kui99] W. Kuich. Linear systems of equations and automata on distributive multioperator monoids. In *Contributions to General Algebra 12 - Proceedings of the 58th Workshop on General Algebra "58. Arbeitstagung Allgemeine Algebra"*, Vienna University of Technology. June 3-6, 1999, pages 1–10. Verlag Johannes Heyn, 1999.
- [Mal04] A. Maletti. Relating tree series transducers and weighted tree automata. In C.S. Calude, editor, *8th Int. Conf. on Developments in Language Theory (DLT 2004)*, volume 3340 of *LNCS*, pages 321–333. Springer-Verlag, 2004.
- [Mal05] A. Maletti. Relating tree series transducers and weighted tree automata. *Internat. J. Found. Comput. Sci.*, pages 723–741, 2005.
- [Pos46] E. Post. A variant of a recursively unsolvable problem. *Bull. AMS*, 52:264–268, 1946.

- [Rot85] G. Rote. A systolic array algorithm for the algebraic path problem (shortest paths; Matrix inversion). *Computing*, 34(3):191–219, 1985.
- [Sak09] J. Sakarovitch. *Elements of Automata Theory*. Cambridge University Press, 2009.
- [Sco67] D. Scott. Some definitional suggestions for automata theory. *J. Comput. System Sci.*, 1:187–212, 1967.
- [SG85] K. M. Schimpf and J. H. Gallier. Tree pushdown automata. *J. Comput. System Sci.*, 30:25–40, 1985.
- [SS78] A. Salomaa and M. Soittola. *Automata-Theoretic Aspects of Formal Power Series*. Texts and Monographs in Computer Science. Springer-Verlag, 1978.
- [Str84] H. Straubing. *Finite automata, formal logics, and circuit complexity*. Birkhäuser, 1984.
- [SVF09] T. Stüber, H. Vogler, and Z. Fülöp. Decomposition of weighted multioperator tree automata. *Int. J. Foundations of Computer Sci.*, 20(2):221–245, 2009.
- [Tra61] B. A. Trakhtenbrot. Finite automata and logic of monadic predicates. *Doklady Akademii Nauk SSSR*, 149:326–329, 1961. in Russian.
- [VDH16] H. Vogler, M. Droste, and L. Herrmann. A weighted MSO logic with storage behaviour and its Büchi-Elgot-Trakhtenbrot theorem. In A.-H. Dediu, J. Janoušek, C. Martín-Vide, and B. Truthe, editors, *10th Int. Conf. on Language and Automata Theory and Applications (LATA 2016)*, volume 9618 of *LNCS*, pages 127–139. Springer-Verlag, 2016.
- [Wec92] W. Wechler. *Universal Algebra for Computer Scientists*, volume 25 of *EATCS Monographs in Theoretical Computer Science*. Springer-Verlag, 1992.