A Link between Multioperator and Tree Valuation Automata and Logics

Markus Teichmann    Johannes Osterholzer

Faculty of Computer Science, Technische Universität Dresden, Nöthnitzer Straße 46, 01062 Dresden, Germany

May 22, 2015

Weighted tree languages over semirings lack the expressive power to model computations like taking the average or the discounting of weights in a straightforward manner. This limitation was overcome by weighted tree automata and logics using (a) tree valuation monoids and (b) multioperator monoids. We compare the expressive power of these two solutions and show that a weighted tree language recognizable (resp. definable) over a tree valuation monoid is also recognizable (resp. definable) using a multioperator monoid. For this, we provide direct, semantic-preserving transformations between the automata models and between the respective logics.

1 Introduction

The fundamental result of Büchi, Elgot, and Trakhtenbrot [1, 2, 8, 20] states the equivalence of recognizability by classical finite-state string automata and definability by means of formulas of monadic second-order (mso) logic. This equivalence result has been extended, in particular, into two directions: (1) from string automata to finite-state tree automata [19, 3] and (2) from the unweighted case to the semiring weighted case [4]. These two generalizations were merged, resulting in the concept of semiring weighted tree languages [7]. The use of a semiring as weight structure restricts the weight computation. All local weights of a run on a tree are accumulated using solely one binary operation, the multiplication of the semiring. This restriction prohibits modeling global weight features, as for example, the average of all local weights [5], in a straightforward manner. To overcome this limitation two weight structures have been introduced: multioperator monoids (m-monoid), together with the associated multioperator weighted tree automata (m-wta) [13, 14, 18, 10], and tree valuation monoids (tv-monoids), together with tree valuation weighted tree automata (tv-wta) [5]. Let us briefly recall these devices.
An m-monoid is a commutative monoid equipped with a set of (arbitrary) operations on its carrier. Each transition of an m-wta is equipped with such an operation, such that the arity of the operation coincides with the rank of the transition. For an m-wta, the weight of a run on a tree is obtained by evaluating the operations in a bottom-up manner, according to their occurrences in that run. The weight which an m-wta assigns to a tree is the m-monoid-sum over the weights of all its accepting runs on that tree.

In [10] languages recognizable by m-wta were characterized, in a Büchi-like result, by languages definable by multioperator expressions (m-expressions). Note that if the m-monoid is distributive, tree languages recognizable by m-wta over this m-monoid are also recognizable by semiring weighted tree automata [14]. The m-monoids treated in this work, however, are in general not distributive.

A tv-monoid is a commutative monoid equipped with a mapping Val which maps each unranked tree over elements of the carrier set $D$ to an element of $D$. For instance, Val can compute the average of all the elements that occur in an unranked tree. The transitions of tv-wta are weighted with values from a given tv-monoid. In order to compute the weight of a tv-wta’s run on a tree, the valuation function is applied to the unranked tree that is generated by replacing every node of the input tree by the local weight of the run’s respective transition. The tv-wta computes the weight of a tree by forming the tv-monoid-sum of the weights of all its accepting runs on that tree.

In [5] a Büchi-like characterization has been proved for tree languages recognizable by tv-wta using a tree valuation weighted monadic second-order (tv-mso) logic which is defined using an extension of tv-monoids with multiplication (product tv-monoids).

In this paper we compare the expressive power of these two generalizations of the semiring weighted case. Given a tv-monoid $D$ or product tv-monoid $D$ we construct the m-monoid $A_D$ or $A_D$ (cf. Construction 13) and compare the following four classes of weighted tree languages over some ranked input alphabet $\Sigma$: the class of weighted tree languages

(a) recognizable by tv-wta over a tv-monoid $D$ (for short: Rec($\Sigma, D$)),

(b) recognizable by m-wta over the m-monoid $A_D$ (for short: Rec($\Sigma, A_D$)),

(c) definable by syntactically restricted tv-mso formulas over a product tv-monoid $D$ (for short: Def($\Sigma, D$)), and

(d) definable by m-expressions over the m-monoid $A_D$ (for short: Def($\Sigma, A_D$)).

Our results are the following (cf. Figure 1):

(1) For every tv-monoid $D$, $\text{Rec}(\Sigma, D) = \pi_1(\text{Rec}(\Sigma, A_D))$ (cf. Theorem 22).

(2) For every regular product tv-monoid $D$, $\text{Def}(\Sigma, D) = \pi_1(\text{Def}(\Sigma, A_D))$ (cf. Theorem 28).

The carrier sets of the constructed m-monoids $A_D$ and $A_D$ are Cartesian products, where $\pi_1$ denotes the projection of a tuple to the first component. Our results relate the formally very different approaches. Note that (2) follows from (1) together with the existing Büchi-like characterizations. However, this approach can yield a nonelementary blow-up of the size of the formula. Hence we improve this result by presenting direct transformations between
tv-mso and m-expressions with a double exponential bound on the length of the constructed formulas.

The characterizations in (1) and (2) are “up to projection”. In Section 4, we show that the class of weighted tree languages recognizable by m-wta is closed under post-composition with monoid homomorphisms. As the projection is a monoid-homomorphism, it can therefore be omitted by using a different m-monoid (cf. Theorem 32). This approach results in a possibly larger class of recognizable tree languages, changing the equality in results (1) and (2) to set inclusions; i.e., for every tv-monoid $D$, there is an m-monoid $A$ such that $\text{Rec}(\Sigma, D) \subseteq \text{Rec}(\Sigma, A)$. A similar result holds for definability.

A reciprocal result is suggested in [5, Remark 3.5]. It is shown that for each absorptive m-monoid $A$, there is a tv-monoid $D_A$ such that for every m-wta $M$, a tv-wta $N_M$ is constructible whose semantics over $D_A$ equals the semantics of $M$ over $A$; hence $\text{Rec}(\Sigma, A) \subseteq \text{Rec}(\Sigma, D_A)$. From this inclusion, one obtains the inclusion for definability, i.e., $\text{Def}(\Sigma, A) \subseteq \text{Def}(\Sigma, D_A)$, similar to Figure 1, using [10, Theorem 4.1] and [5, Theorem 5.5]. Note that the latter theorem requires $D_A$ to be regular, but this is actually not needed for the direction $\text{Rec}(\Sigma, D_A) \subseteq \text{Def}(\Sigma, D_A)$.

Our results rely on notation from multiple sources. We recall important definitions in Section 2. Readers familiar with these concepts can move on to Sections 3 and 4 and consult the preliminaries if needed.

The running example in this work is centered around the concept of discounting. Discounting plays an important role in mathematical economics and game theory, and is applied to both infinite and finite structures. In our examples, given a binary input tree, we count the number of occurrences of a certain pattern, discounted according to their position in the tree. This manner of discounting is an instance of a more general notion of discounting, as described in, e.g., [15].

### 2 Preliminaries

We begin with laying out some basic notions and notation. In the following subsections, we recall the two algebraic structures which are of interest in this work: multioperator monoids and tree valuation monoids, along with their respective weighted tree automaton models, and the associated weighted logics. The definitions for multioperator monoids are based on
The set of all nonnegative integers is denoted by $\mathbb{N}$ and the set of all nonnegative real numbers by $\mathbb{R}_{\geq 0}$. Let $\mathbb{N}_+ = \mathbb{N} \setminus \{0\}$ and $\bar{\mathbb{R}} = \mathbb{R}_{\geq 0} \cup \{-\infty\}$. For every $k \in \mathbb{N}$, denote by $[k]$ the set $\{1, \ldots, k\}$. Thus $[0] = \emptyset$, the empty set. Let $A$ and $B$ be sets. We denote the power set of $A$ by $\mathcal{P}(A)$ and the cardinality of $A$ by $|A|$. For $i \in [2]$, the projection from $A \times B$ to the $i$-th component is denoted by $\pi_i$. The image of a function $f : A \to B$ is denoted by $\text{im}(f)$.

Let $k \in \mathbb{N}$. Then set $\text{Ops}^{(k)}(A) = \{\omega : A^k \to A\}$ and $\text{Ops}(A) = \bigcup_{k \in \mathbb{N}} \text{Ops}^{(k)}(A)$. We say that $\omega \in \text{Ops}(A)$ is an operation on $A$, and $\omega$ is $k$-ary if $\omega \in \text{Ops}^{(k)}(A)$. For every $\Omega \subseteq \text{Ops}(A)$, let $\text{\Omega}^{(k)}$ be the set of $k$-ary operations in $\Omega$.

An alphabet is a finite nonempty set of symbols. Let $\Sigma$ be an alphabet. The set of finite words over $\Sigma$ is denoted by $\Sigma^{*}$, the empty word by $\epsilon$, and the length of a word $w \in \Sigma^{*}$ by $|w|$. A ranked alphabet is an alphabet $\Sigma$ that is equipped with a function $\text{rk} : \Sigma \to \mathbb{N}$ which maps every symbol to its rank. For every $k \in \mathbb{N}$, let $\Sigma^{(k)} = \text{rk}^{-1}(k)$. The maximal rank of $\Sigma$ is defined as $\max_{\text{rk}}(\Sigma) = \max\{\text{rk}(\sigma) \mid \sigma \in \Sigma\}$.

In what follows, let $\Sigma$ be an arbitrary ranked alphabet, unless specified otherwise. We will tacitly assume that $\Sigma^{(0)} \neq \emptyset$.

Let $A$ be a set, and $\Omega \subseteq \text{Ops}(A)$. A $\Sigma$-family of operations in $\Omega$ is a $\Sigma$-family $\omega = (\omega_{\sigma} \mid \sigma \in \Sigma)$ such that for every $k \in \mathbb{N}$ and $\sigma \in \Sigma^{(k)}$, we have $\omega_{\sigma} \in \Omega^{(k)}$. We call the tuple $(A, \omega)$ a $\Sigma$-algebra (induced by $\omega$) if $\omega$ is a $\Sigma$-family of operations in $\text{Ops}(A)$.

2.1 Trees and Tree Languages

A tree domain is a finite, nonempty set $\mathbb{W} \subseteq (\mathbb{N}_+)^*$ such that for every $w \in (\mathbb{N}_+)^*$ and $i \in \mathbb{N}_+$, if $wi \in \mathbb{W}$, then $w, w1, \ldots, w(i - 1) \in \mathbb{W}$. For every $w \in \mathbb{W}$, the rank of $w$ in $\mathbb{W}$, denoted by $\text{rk}^\mathbb{W}(w)$, is defined as $\text{rk}^\mathbb{W}(w) = |\{i \in \mathbb{N}_+ \mid wi \in \mathbb{W}\}|$. Assume a nonempty set $\Delta$. A function $\xi : \mathbb{W} \to \Delta$ is an unranked tree over $\Delta$ if $\mathbb{W}$ is a tree domain. The set of all unranked trees over $\Delta$ is denoted by $T^\Delta_{\mathbb{W}}$. Let $\xi : \mathbb{W} \to \Delta$ be an unranked tree. We denote the set $\mathbb{W}$ by $\text{pos}(\xi)$; its elements are the positions of $\xi$. For every $w \in \text{pos}(\xi)$, we call $\xi(w)$ the label of $\xi$ at position $w$ and denote the subtree of $\xi$ at $w$ by $\xi|_w$. Let $\delta \in \Delta$. We say that $\xi$ is $\delta$-free if $\delta$ is not in $\text{im}(\xi)$.

Assume now a ranked alphabet $\Sigma$. A (ranked) tree over $\Sigma$ is an unranked tree $\xi \in T^\Sigma_{\mathbb{W}}$ such that $\text{rk}^\text{pos}(\xi)(w) = \text{rk}(\xi(w))$ for each $w \in \text{pos}(\xi)$. The set of all ranked trees over $\Sigma$ is denoted by $T^\Sigma_{\mathbb{W}}$. Note that $T^n_{\mathbb{W}} \subseteq T^\mathbb{W}_{\mathbb{W}}$ and, by abuse of notation, $\Delta \subseteq T^\mathbb{W}_{\mathbb{W}}$.

A tree language over $\Sigma$ is a subset $L \subseteq T_{\mathbb{W}}$. Suppose $A$ is a nonempty set, whose elements we call weights. A weighted tree language over $\Sigma$ and $A$ is then a function $L : T_{\mathbb{W}} \to A$. Let $a \in A$, let $B$ be a set, and let $L_1$ and $L_2$ be weighted tree languages over $\Sigma$ and $A$. Assume a binary operation $\odot$ on $A$, as well as a function $f : A \to B$. The weighted tree languages $L_1 \odot L_2$ and $a \odot L_1$ over $\Sigma$ and $A$, and $f(L_1)$ over $\Sigma$ and $B$ are defined by $(L_1 \odot L_2)(\xi) = L_1(\xi) \odot L_2(\xi)$, $(a \odot L_1)(\xi) = a \odot L_1(\xi)$, and $(f(L_1))(\xi) = f(L_1(\xi))$, respectively, for every $\xi \in T_{\mathbb{W}}$. For a thorough introduction to tree languages compare [12], and for the weighted case see [11].
2.2 Monoids

Let $A$ be a set, $\circ$ be a binary operation on $A$, and $e \in A$. The tuple $(A, \circ, e)$ is a monoid if for every $a, b, c \in A$, we have $(a \circ b) \circ c = a \circ (b \circ c)$ and $a \circ e = e \circ a = a$. If moreover $a \circ b = b \circ a$ for every $a, b \in A$, then the monoid is commutative.

2.2.1 Multioperator Monoids

**Definition 1.** A multioperator monoid (m-monoid) is defined as a quadruple $A = (A,+ ,0,\Omega)$ such that $(A,+,0)$ is a commutative monoid, $\Omega \subseteq \text{Ops}(A)$, and $0^{(k)} \in \Omega$ for every $k \in \mathbb{N}$, where $0^{(k)}$ is the $k$-ary operation on $A$ with image $\{0\}$. An m-monoid is called absorptive if for every $k \in \mathbb{N}$, $\omega \in \Omega^{(k)}$, and $a_1,\ldots,a_k \in A$, whenever $0 \in \{a_1,\ldots,a_k\}$, then also $\omega(a_1,\ldots,a_k) = 0$.

Note that every m-monoid can be extended to an absorptive one [10, p. 245].

For the remainder of this work, let $A = (A,+ ,0,\Omega)$ denote an arbitrary absorptive m-monoid.

**Example 2.** For our running example, we will construct an m-monoid whose operations allow for discounting. Its carrier is the set $\mathbb{R}$ of nonnegative real numbers with negative infinity. The addition $+$ and multiplication $\cdot$ on reals are extended to negative infinity in the usual manner; in particular, $-\infty \cdot 0 = 0 = -\infty = -\infty$. Every operation $\omega$ of the m-monoid is of the form

$$
\omega: \mathbb{R}^k \to \mathbb{R}, \quad (r_1,\ldots,r_k) \mapsto a + \lambda_1 \cdot r_1 + \ldots + \lambda_k \cdot r_k
$$

for some $k \in \mathbb{N}$, an arbitrary value $a \in \mathbb{R}$, and predescribed discount factors $\lambda_1,\ldots,\lambda_k \in \mathbb{R}_{\geq 0}$. These factors allow to control the weight of the contribution of each argument to the result of an operation. Later on, in the m-wta of Example 6, we will use this facility to compute the discounted number of occurrences of a certain pattern in a tree.

Formally, assume an $\mathbb{N}_+$-family $\Lambda = (\lambda_i \in \mathbb{R}_{\geq 0})_{i \in \mathbb{N}_+}$. The m-monoid $A_{\text{disc}}^\Lambda$ is given by $A_{\text{disc}}^\Lambda = (\mathbb{R}, \max, -\infty, \Omega_{\text{disc}}^\Lambda)$, where $\Omega_{\text{disc}}^\Lambda = \{\theta_{a,\Lambda}^{(k)} | k \in \mathbb{N}, a \in \mathbb{R}\}$. For every $k \in \mathbb{N}$ and $a, r_1,\ldots,r_k \in \mathbb{R}$, we let

$$
\theta_{a,\Lambda}^{(k)}(r_1,\ldots,r_k) = a + \lambda_1 \cdot r_1 + \ldots + \lambda_k \cdot r_k.
$$

Note that, for every $k \in \mathbb{N}$, $\theta_{-\infty,\Lambda}^{(k)}$ is equal to the function $0^{(k)}$ specified in the definition above, and thus $0^{(k)} \in \Omega^{(k)}$. It can be verified that the m-monoid $A_{\text{disc}}$ is absorptive.

2.2.2 (Product) Tree Valuation Monoids

**Definition 3.** A tree valuation monoid (tv-monoid) is a tuple $D = (D,+,0,\text{Val})$ such that $(D,+,0)$ is a commutative monoid and $\text{Val}: T_D^0 \to D$ is a function where (i) $\text{Val}(d) = d$ for every $d \in D$, and (ii) for every $\xi \in T_D^0$, if $\xi$ is not 0-free, then $\text{Val}(\xi) = 0$. A product tree valuation monoid (ptv-monoid) is a tuple $D = (D,+,\circ,0,1,\text{Val})$, where $(D,+,0,\text{Val})$ is a tv-monoid, $\circ$ is a binary operation on $D$, and $1 \in D$ such that (i) $0 \circ d = d \circ 0 = 0$ and $1 \circ d = d \circ 1 = d$ for every $d \in D$, and (ii) $\text{Val}(\xi) = 1$ for every $\xi \in T_D^0$ with $\text{im}(\xi) = \{1\}$. 

5
Given a ptv-monoid \((D, +, \diamond, 0, 1, \text{Val})\), we call \((D, +, 0, \text{Val})\) its underlying tv-monoid.

For the remainder of this work, let \(D = (D, +, 0, \text{Val})\) denote an arbitrary tv-monoid and \(D = (D, +, \diamond, 0, 1, \text{Val})\) an arbitrary ptv-monoid, unless specified otherwise.

Example 4. We show a tv-monoid for discounting similar to [5, Example 2.5]. Applied to some unranked tree \(\xi\) labeled by monoid elements, its valuation function computes the sum of these values, where each value is discounted with respect to the position it appears at in \(\xi\). To each positive natural number \(i\), we associate a discount factor \(\lambda_i \in \mathbb{R}_{\geq 0}\). Whenever a value appears in the \(i\)-th subtree of some node in \(\xi\), it is to be discounted by \(\lambda_i\). Hence, a value \(a\) at position \(w = j_1 \ldots j_\ell\) in \(\xi\) is discounted with respect to \(w\) by multiplying \(a\) with the respective discount factors \(\lambda_{j_1}, \ldots, \lambda_{j_\ell}\).

Formally, presume the family \(\Lambda = (\lambda_i \in \mathbb{R}_{\geq 0})_{i \in \mathbb{N}^+}\). We define the tv-monoid \(D^\Lambda_{\text{disc}} = (\mathbb{R}, \text{max}, -\infty, \text{Val}^\Lambda_{\text{disc}})\). For every \(\xi \in T^u_{\mathbb{R}}, a \in \mathbb{R}, \text{ and } j_1, \ldots, j_\ell \in \mathbb{N}^+\), where \(\ell \in \mathbb{N}\), let

\[
\text{Val}^\Lambda_{\text{disc}}(\xi) = \sum_{w \in \text{pos}(\xi)} \text{disc}^\Lambda_{w}(\xi(w)) \quad \text{and} \quad \text{disc}^\Lambda_{j_1 \ldots j_\ell}(a) = \left(\prod_{i=1}^{\ell} \lambda_{j_i}\right) \cdot a .
\]

Again sum and product are the usual ones on \(\mathbb{R}_{\geq 0}\) extended by \(-\infty\). Note that if \(\ell = 0\), then \(j_1 \ldots j_\ell = e\) and \(\text{disc}^\Lambda_e(a) = a\). It is easy to verify that this valuation function satisfies all requirements and \(D_{\text{disc}}\) is indeed a tv-monoid. The tv-monoid \(D^\Lambda_{\text{disc}}\) can be extended to the ptv-monoid

\[
D^\Lambda_{\text{disc}} = (\mathbb{R}, \text{max}, +, -\infty, 0, \text{Val}^\Lambda_{\text{disc}}) .
\]

The additional requirements follow directly from the definition.

2.3 Weighted Tree Automata

The concept of runs applies equally to automata over m-monoids and tv-monoids. A run on a tree \(\xi\) is an arbitrary relabeling of \(\xi\) with states from some set \(Q\). More formally, given a tree \(\xi \in T_{\Sigma}\) and a finite set \(Q\), the set of runs on \(\xi\), denoted by \(R_Q(\xi)\), is the set of unranked trees \(R_Q(\xi) = \{r \mid r : \text{pos}(\xi) \to Q\}\).

2.3.1 Multioperator Weighted Tree Automata

Definition 5. A multioperator weighted tree automaton over \(\Sigma\) and \(A\) (m-wta) is a triple \(M = (Q, \delta, F)\), where \(Q\) is a finite, nonempty set (of states), \(F \subseteq Q\) (the set of final states), and \(\delta\) is a \(\Sigma\)-family

\[
\delta = (\delta_\sigma \mid \sigma \in \Sigma) \quad \text{of mappings} \quad \delta_\sigma : Q^k \times Q \to \Omega^{(k)}
\]

for every \(\sigma \in \Sigma^{(k)}\) and \(k \in \mathbb{N}\) (the transition family).

In the sequel, let \(M = (Q, \delta, F)\) be an arbitrary m-wta over \(\Sigma\) and \(A\).
Let \( \xi \in T_\Sigma \) and \( r \in R_0(\xi) \). For every \( w \in \text{pos}(\xi) \), we define \( \delta(\xi, r, w) \) to be the element of \( A \) obtained by well-founded induction on the set of positions of \( \xi \) as follows. Define

\[
\delta(\xi, r, w) = \omega\left(\delta(\xi, r, w_1), \ldots, \delta(\xi, r, w_k)\right),
\]

where \( \omega = \delta_\sigma(r(w_1) \ldots r(w_k), r(w)) \), \( \sigma = \xi(w) \), and \( k = r(k) \). The weighted tree language recognized by \( M \) is the mapping \( \llbracket M \rrbracket : T_\Sigma \to A \) such that for every \( \xi \in T_\Sigma \),

\[
\llbracket M \rrbracket(\xi) = \sum_{r \in R_0(\xi), r(\varepsilon) \in F} \delta(\xi, r, \varepsilon).
\]

We say that a weighted tree language \( L \) is recognizable by m-wta if there is an m-wta \( M \) such that \( L = \llbracket M \rrbracket \). The class of all weighted tree languages recognized by some m-wta over \( \Sigma \) and \( A \) is denoted by \( \text{Rec}(\Sigma, A) \).

**Example 6.** Let \( \Sigma = \{\sigma^{(2)}, \gamma^{(1)}, a^{(0)}\} \). We show an m-wta which recognizes the discounted number of occurrences of patterns of the form \( \sigma(x, \sigma(y, z)) \) in a tree, where \( x, y, \) and \( z \) are placeholders ranging over \( T_\Sigma \). Recall the m-monoid \( A^\Lambda_{\text{disc}} \) from Example 2. In order to reduce the impact of pattern occurrences within second child trees, we set \( \lambda_1 = 1 \), \( \lambda_2 = 0.8 \), and \( \lambda_i = 1 \) for every \( i > 2 \). Define the m-wta \( M_{\text{disc}} = (\{q_0, q_1\}, \delta, \{q_0, q_1\}) \) over \( \Sigma \) and \( A^\Lambda_{\text{disc}} \) where for \( q \in \{q_0, q_1\} \),

\[
\delta_\sigma(\varepsilon, q_0) = \theta_0^{(0)}_\Lambda, \quad \delta_\gamma(q_0, q_0) = \theta_0^{(1)}_\Lambda, \quad \delta_\sigma(qq_0, q_1) = \theta_0^{(2)}_\Lambda, \quad \delta_\sigma(qq_1, q_1) = \theta_1^{(2)}_\Lambda,
\]

and each other entry of \( \delta \) is mapped to \( \theta_{2\infty, \Lambda} \) of the correct arity.

For every tree \( \xi \in T_\Sigma \), there is a unique run of \( M_{\text{disc}} \) on \( \xi \) whose weight is unequal to \( -\infty \). This run equips positions of \( \xi \) labeled by \( \sigma \) with the state \( q_1 \) and all other positions with state \( q_0 \). Whenever the pattern is detected (via a transition of the form \( \delta_\sigma(qq_1, q_1) \)), the corresponding local weight is 1. The discounting is handled by the underlying m-monoid according to \( \Lambda \).

It can be argued that, in contrast to the tv-monoid \( D_{\text{disc}}^\Lambda \) from Example 4, the local weights in this run of \( M_{\text{disc}} \) are not discounted individually by the m-monoid \( A^\Lambda_{\text{disc}} \). Instead, the discounted occurrence numbers are computed in a bottom-up manner for each subtree of a node, and then discounted further, according to the operation \( \theta_{a, \Lambda}^{(k)} \) applied at that node. However, both approaches amount to the same, by distributivity of \( \cdot \) over \( + \).

Consider the tree in Figure 2. The three pattern occurrences are shaded in gray, and the number next to each of them is the discounted value it contributes to the weight of the unique run on the tree. The sum of their contributions, and hence the run’s weight, is 2.8, as one occurrence is a second child of the root symbol.

### 2.3.2 Tree Valuation Weighted Tree Automata

**Definition 7.** A tree valuation weighted tree automaton over \( \Sigma \) and \( D \) (tv-wta) is a triple \( \mathcal{N} = (Q, \mu, F) \) where \( Q \) is a finite, nonempty set (of states), \( F \subseteq Q \) (the set of final states), and \( \mu \) is a \( \Sigma \)-family

\[
\mu = \{\mu_\sigma \mid \sigma \in \Sigma\}\text{ of mappings } \mu_\sigma : Q^k \times Q \to D
\]

for every \( \sigma \in \Sigma^{(k)} \) and \( k \in \mathbb{N} \) (the transition family).
In the sequel, let $\mathcal{N} = (Q, \mu, F)$ be an arbitrary tv-wta over $\Sigma$ and $D$.

Given a tree $\xi \in T_\Sigma$ and a run $r \in R_Q(\xi)$, define their induced tree of values $\mu(\xi, r)$ to be the tree in $T^i_D$ with $\text{pos}(\mu(\xi, r)) = \text{pos}(\xi)$ such that

$$\mu(\xi, r)(w) = \mu_\sigma(r(w1)\ldots r(wk), r(w))$$

for each $w \in \text{pos}(\xi)$ where $\sigma = \xi(w)$ and $k = \text{rk}(\sigma)$.

The weighted tree language recognized by $\mathcal{N}$ is the mapping $[\mathcal{N}] : T_\Sigma \rightarrow D$ defined for every $\xi \in T_\Sigma$, as

$$[\mathcal{N}](\xi) = \sum_{r \in R_Q(\xi)} \sum_{r(\epsilon) \in F} \text{Val}(\mu(\xi, r)) .$$

We say that a weighted tree language $L$ is recognizable by tv-wta if there is a tv-wta $\mathcal{N}$ such that $L = [\mathcal{N}]$. The class of all weighted tree languages recognized by some tv-wta over $\Sigma$ and $D$ is denoted by $\text{Rec}((\Sigma, D))$.

**Example 8.** We define a tv-wta which recognizes the discounted number of occurrences of patterns of the form $\sigma(x, \sigma(y, z))$, as already discussed in Example 6 for m-wta. Recall the tv-monoid $D^\Lambda_{\text{disc}}$ from Example 4 and set $\lambda_1 = 1$, $\lambda_2 = 0.8$, and $\lambda_i = 1$ for all $i > 2$. We define the tv-wta $\mathcal{N}_{\text{disc}} = (\{q_0, q_1\}, \mu, \{q_0, q_1\})$ over $\Sigma = \{\sigma^{(2)}, \gamma^{(1)}, \alpha^{(0)}\}$ and $D^\Lambda_{\text{disc}}$, where for every $q \in \{q_0, q_1\}$,

$$\mu_\sigma(e, q_0) = 0 , \quad \mu_\gamma(q_0, q_0) = 0 , \quad \mu_\sigma(qq_0, q_1) = 0 , \quad \mu_\sigma(qq_1, q_1) = 1 ,$$

and each other entry of $\mu$ is set to $-\infty$.

We reconsider the tree from Figure 2 and the run $r$ which maps every position which is labeled by $\sigma$ to $q_1$, and each other position to $q_0$. The induced tree of values $\mu(\xi, r)$ is labeled by 1 at every position where the pattern is detected and by 0 at each other position. The values are accumulated by the valuation function using discounted sum, and this yields the value 2.8. In contrast to $A_{\text{disc}}$, the values are discounted individually by $D^\Lambda_{\text{disc}}$, as discussed at the end of Example 6.
2.4 Logics

We briefly recall unweighted monadic second-order (mso) logic for trees, as it is fundamental both for multioperator expressions and for weighted logics over tree valuation monoids. Then we recapitulate the syntax of m-expressions and tree-valuation-weighted mso. Their respective semantics are defined formally in [10], resp. [5].

Fix two disjoint countably infinite sets \( \mathcal{X}_1 \) and \( \mathcal{X}_2 \) of first-order and second-order variables. The set of variables is \( \mathcal{X} = \mathcal{X}_1 \cup \mathcal{X}_2 \). We will follow the custom of denoting first-order variables by lowercase letters, e.g., \( x, y, x_1, \ldots \), and second-order variables by capital ones like \( X, Y, X_1, \ldots \).

**Definition 9.** The set of monadic second-order logic formulas over \( \Sigma \) (mso formulas) is generated by the EBNF definition

\[
\beta ::= \text{label}_\sigma(x) \mid \text{edge}_\sigma(x, y) \mid x \in X \mid \neg \beta \mid \beta \land \beta \mid \forall x.\beta \mid \forall X.\beta
\]

where \( \beta \) is the nonterminal, \( \sigma \in \Sigma \), \( i \in [\text{maxrk}(\Sigma)] \), \( x, y \in \mathcal{X}_1 \), and \( X \in \mathcal{X}_2 \). Let \( \varphi \) be an mso formula. The set of free variables in \( \varphi \) is denoted by free(\( \varphi \)). The size of \( \varphi \) is denoted by \( |\varphi| \) and defined by structural induction: the size of an atomic formula is 1, while in the cases that \( \varphi = \neg \psi \), \( \varphi = \forall x.\psi \), or \( \varphi = \forall X.\psi \), we set \( |\varphi| = |\psi| + 1 \). If \( \varphi = \psi_1 \land \psi_2 \), then \( |\varphi| = |\psi_1| + |\psi_2| + 1 \).

It is straightforward to define macros for the connectors \( \lor \), \( \Rightarrow \), and \( \Leftrightarrow \) and for the quantifiers \( \exists x \) and \( \forall x \).

The semantics of an mso formula is defined in the conventional manner. We follow the notation of [10]. In particular, we treat tuples \((\xi, \rho)\), where \( \xi \) is a tree, and \( \rho : \text{pos}(\xi) \rightarrow \mathcal{P}(V) \) is a variable assignment over a finite set of variables \( V \), interchangeably with trees over the extended alphabet \( \Sigma_V = \Sigma \times \mathcal{P}(V) \). The set of valid trees over \( \Sigma_V \) is denoted by \( T_{\Sigma_V}^\text{v} \), and the set of all models of an mso formula \( \varphi \) in \( T_{\Sigma_V}^\text{v} \) (the tree language defined by \( \varphi \)) by \( L_V(\varphi) \). The class of tree languages definable by mso formulas is equal to the class recognizable by unweighted tree automata [19, Theorem 17].

2.4.1 Multioperator Expressions

**Definition 10.** The set of multioperator expressions over \( \Sigma \) and \( A \) (for short: m-expressions) is generated by the following EBNF definition

\[
e ::= H(\omega) \mid (e + e) \mid \sum_\sigma e \mid \sum_X e \mid (\beta \triangleright e)
\]

where \( e \) is the nonterminal, \( \omega \) is a \( \Sigma_U \)-family of operations in \( \Omega \) for some finite set of variables \( U \), \( \beta \) is the nonterminal generating mso formulas over \( \Sigma \) (cf. Definition 9), \( x \in \mathcal{X}_1 \), and \( X \in \mathcal{X}_2 \). The free variables of an m-expression \( e \), denoted by free(\( e \)), are defined straightforwardly similar to the case of mso logic. Note that all variables of \( U \) in a \( \Sigma_U \)-family of operations \( \omega \) are considered to be free in the formula \( H(\omega) \). The size of an m-expression \( e \), again denoted by \( |e| \), is defined by structural induction. If \( e = H(\omega) \) for some \( \Sigma_U \)-family \( \omega \), then we set \( |H(\omega)| = |\Sigma| \cdot 2^{|U|} \). This takes into account the cost of storing the definition of \( \omega_{(\sigma, V)} \) for each \( \sigma \in \Sigma \) and each subset \( V \) of \( U \). If \( e = \sum_\sigma e' \) or \( e = \sum_X e' \), then \( |e| = |e'| + 1 \). Moreover, \( |(e_1 + e_2)| = |e_1| + |e_2| + 1 \), and \( |(\beta \triangleright e)| = |\beta| + |e| + 1 \).
Given an m-expression $e$ and some finite set of variables $\mathcal{V} \supseteq \text{free}(e)$, the semantics of $e$ with respect to $\mathcal{V}$, denoted by $\llbracket e \rrbracket_{\mathcal{V}}$, is a weighted tree language of type $T_{2V} \rightarrow A$, following the notation and definitions in [10]. The weight of an invalid tree is 0. Assume a valid tree $\xi \in T_{2V}^\mathcal{V}$. Then $\llbracket H(\omega) \rrbracket_{\mathcal{V}}(\xi)$ amounts to evaluating $\xi$ in the $\Sigma_{\mathcal{V}}$-algebra induced by $\omega$, where variables from $\mathcal{V} \setminus \mathcal{V}'$ are ignored. Expressions of the form $e_1 + e_2$, $\Sigma x e$, and $\Sigma \xi e$ are evaluated, in the natural way, by the m-monoid’s sum. Lastly, $\llbracket B \cdot e \rrbracket_{\mathcal{V}}(\xi) = \llbracket e \rrbracket_{\mathcal{V}}(\xi)$ if the mso formula $B$ holds on $\xi$, and 0 otherwise.

A weighted tree language $L : T_{2V} \rightarrow A$ is definable by m-expressions if there is an m-expression $e$ over $\Sigma$ and $A$ such that $\mathcal{V} \supseteq \text{free}(e)$ and $\llbracket e \rrbracket_{\mathcal{V}} = L$. We denote the class of all weighted tree languages over $\Sigma$ and $A$ which are the semantics of some m-expression over $\Sigma$ and $A$ by $\text{Def}(\Sigma, A)$.

### 2.4.2 Tree Valuation Weighted mso Logic

**Definition 11.** The set of weighted monadic second-order formulas over $\Sigma$ and $\mathbb{D}$ (tv-mso formulas) is generated by the following EBNF definition

$$
\varphi ::= d \mid B \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid \exists x. \varphi \mid \forall x. \varphi \mid \exists X. \varphi,
$$

where $\varphi$ is the nonterminal, $d \in D$, and $B$ is the nonterminal generating mso formulas over $\Sigma$ (cf. Definition 9). A tv-mso formula $\varphi$ is called weighted if it cannot be generated by $B$. Otherwise it is called Boolean. The size $|\varphi|$ of a Boolean tv-mso formula $\varphi$ is already defined. If $\varphi$ is weighted, we set $|d| = 1$ for each $d \in D$, $|\varphi| = |\varphi_1| + |\varphi_2| + 1$ for $\varphi = \varphi_1 \lor \varphi_2$ or $\varphi = \varphi_1 \land \varphi_2$, and in the cases that $\varphi = \exists x. \psi$, $\varphi = \forall x. \psi$, or $\varphi = \exists X. \psi$, we have $|\varphi| = |\psi| + 1$.

The set of free variables is defined as in the mso logic case. Recall from [5] the following syntactic restrictions of a tv-mso formula: A tv-mso formula $\varphi$ is almost Boolean if it consists of finitely many conjunctions and disjunctions of Boolean formulas and values of $D$; $\forall$-restricted if for each sub-formula $\forall x. \psi$, we have that $\psi$ is almost Boolean; and strongly $\land$-restricted if each sub-formula $\psi_1 \land \psi_2$ of $\varphi$ is such that either $\psi_1$ or $\psi_2$ are Boolean, or both are almost Boolean. Furthermore, recall from [5] the definition of a regular ptv-monoid $D$, i.e., for every ranked alphabet $\Sigma$ and $d \in D$, there is a tv-wta $N_d$ over $\Sigma$ and $D$ such that $\llbracket N_d \rrbracket(\xi) = d$ for each tree $\xi \in T_{2\Sigma}^\mathcal{V}$.

Let $\varphi$ be a tv-mso formula over $\Sigma$ and $D$, and $\mathcal{V} \supseteq \text{free}(\varphi)$ be a finite set of variables. The semantics of $\varphi$ with respect to $\mathcal{V}$, denoted $\llbracket \varphi \rrbracket_{\mathcal{V}}$, is a weighted tree language of type $T_{2\mathcal{V}} \rightarrow D$. Refer to [5] for its formal definition. In a nutshell, invalid trees are mapped to 0; presuming $\xi \in T_{2\mathcal{V}}^\mathcal{V}$, $\llbracket d \rrbracket_{\mathcal{V}}(\xi)$ is $d$, formulas of kind $\varphi_1 \lor \varphi_2$, $\exists x. \varphi$, and $\exists X. \varphi$ are evaluated, in the natural manner, by the ptv-monoid’s sum, and the semantics of $\varphi_1 \land \varphi_2$ is resolved by its operation $\diamond$. One obtains $\llbracket \forall x. \varphi \rrbracket_{\mathcal{V}}(\xi)$ by computing $\text{Val}(\xi[\varphi])$, where $\xi[\varphi]$ is a tree over $D$ of the same shape as $\xi$, and the label at its position $w \in \text{pos}(\xi)$ is $\llbracket \varphi \rrbracket_{\mathcal{V}}(\xi[x \mapsto w])$. Here, $\xi[x \mapsto w]$ is the outcome of assigning the variable $x$ in $\xi$ to $w$.

A weighted tree language $L : T_{2\mathcal{V}} \rightarrow D$ is definable by tv-mso formulas if there is a $\forall$-restricted and strongly $\land$-restricted tv-mso formula $\varphi$ over $\Sigma$ and $D$ such that $\mathcal{V} \supseteq \text{free}(\varphi)$ and $\llbracket \varphi \rrbracket_{\mathcal{V}} = L$. We denote the class of all weighted tree languages over $\Sigma$ and $D$ which are the semantics of some tv-mso formula over $\Sigma$ and $D$ by $\text{Def}(\Sigma, D)$, and abbreviate $\llbracket \varphi \rrbracket_{\text{free}(\varphi)}$ by $\llbracket \varphi \rrbracket$. We stress
that, in a deviation from [5], the class \(\text{Def}(\Sigma, \mathbb{D})\) contains by definition only tree languages definable by syntactically restricted formulas. In contrast to the unweighted case, formulas which are not syntactically restricted allow the definition of weighted tree languages which are not recognizable [5, Example 5.6]. However, these are not relevant in the scope of this work.

The following construction is similar to [10, Lemma 5.8]. It is based on [5, Lemma 5.11] and the application of [5, Lemma 5.9] in a bottom-up manner.

**Observation 12.** Given an almost Boolean tv-mso formula \(\varphi\) and \(V \supseteq \text{free}(\varphi)\), we can effectively construct a sequence of pairs \((d_1, \varphi_1) \ldots (d_n, \varphi_n)\), where (i) for all \(i \in [n]\), we have \(d_i \in D\) and \(\varphi_i\) is an mso formula over \(\Sigma\), such that \(\text{free}(\varphi) = \bigcup_{i \in [n]} \text{free}(\varphi_i)\), (ii) it holds that

\[
[\varphi]_V = \sum_{i \in [n]} d_i \circ I_{L_V(\varphi_i)}
\]

and (iii) the family of tree languages \((L_V(\varphi_i) | i \in [n])\) forms a partitioning of \(T^\Sigma_{V}\). The constructed sequence \((d_1, \varphi_1) \ldots (d_n, \varphi_n)\) is called a recognizable step function and denoted by \(\text{step}(\varphi)\).

### 3 Characterization

In this section, we characterize languages recognizable by tv-wta (or definable by tv-mso formulas) by languages recognizable by m-wta (or definable by m-expressions, respectively). Given a tv-monoid, we construct an m-monoid which allows m-wta and m-expressions to model tv-wta and tv-mso formulas, respectively. At the same time, it is not too powerful, that is, every recognizable or definable weighted tree language over this m-monoid is also recognizable or definable by a tv-wta or tv-mso formula over the original (p)tv-monoid.

We first construct the m-monoid, followed by the examination of the automaton case. Then we investigate the logics and give transformations from one into the other and vice versa.

#### 3.1 Construction of the m-Monoid

The m-monoid simulates the internal behavior of tv-wta, i.e., building up unranked trees over the carrier set and afterwards applying the valuation function. The carrier set of the constructed m-monoid is the direct product of values from the tv-monoid and of unranked trees over these values. While solely operations from \(\Omega\) are used, the pairs’ second component models the stepwise construction of trees of values, and the first component their respective valuation. Note that the sum \(\oplus\) does not preserve this property.

**Construction 13.** Given a tv-monoid \(D\), define the operations \(\hat{\oplus} : (T^u_D)^2 \to T^u_D\) and \(\oplus : D \times (T^u_D)^2 \to D \times T^u_D\) as follows. Let \(i \in [2]\) and \(d_i \in D\), \(\xi_i \in T^u_D\). We set \(\text{pos}(\xi_1 \oplus \xi_2) = \text{pos}(\xi_1) \cup \text{pos}(\xi_2)\), and for every \(w \in \text{pos}(\xi_1 \oplus \xi_2)\),

\[
(\xi_1 \oplus \xi_2)(w) = \hat{\xi}_1(w) + \hat{\xi}_2(w) \quad \text{and} \quad (d_1, \xi_1) \oplus (d_2, \xi_2) = (d_1 + d_2, \xi_1 \oplus \xi_2),
\]
where \( \hat{\xi}_i(w) = \xi_i(w) \) if \( w \in \text{pos}(\xi_i) \), and \( \hat{\xi}_i(w) = 0 \) otherwise. Further, for every \( d \in D, k \in \mathbb{N} \), and \( (d_1, \xi_1), \ldots, (d_k, \xi_k) \in D \times T^u_{\text{disc}} \), we set \( \xi = d(\xi_1, \ldots, \xi_k) \), define
\[
\text{valtop}^d_k((d_1, \xi_1), \ldots, (d_k, \xi_k)) = \begin{cases} 
\text{Val}(\xi), \xi & \text{if } d \neq 0 \text{ and } \xi \text{ is } 0\text{-free} \\
(0, 0) & \text{otherwise},
\end{cases}
\]
and let \( \Omega = \{\text{valtop}^d_k \mid d \in D, k \in \mathbb{N}\} \). Then, the m-monoid corresponding to \( D \) is defined as \( A_D = (D \times T^u_{\text{disc}}, \Phi, (0, 0), \Omega) \).

Note that \( 0^k = \text{valtop}^0_0 \in \Omega \) for every \( k \in \mathbb{N} \) as required in the definition for m-monoids. Given a ptv-monoid \( \mathbb{D} = (D, +, \circ, 0, 1, \text{Val}) \), the m-monoid \( A_{\mathbb{D}} \) is defined as \( A_{\mathbb{D}} \), according to the construction above, where \( D \) is the underlying tv-monoid \( (D, +, 0, \text{Val}) \) of \( \mathbb{D} \).

**Lemma 14.** The structure \( A_D \) is an absorptive m-monoid.

**Proof.** It is straightforward to show that \((T^u_{\text{disc}}, \hat{\Phi}, 0)\) is a commutative monoid, and therefore, the direct product \( (D \times T^u_{\text{disc}}, \Phi, (0, 0)) \) is a commutative monoid, too. Furthermore, by close inspection of the definitions for all operations in \( \Omega \), it can be seen that these are indeed absorptive. \( \square \)

**Observation 15.** Let \( (d_1, \xi_1), (d_2, \xi_2) \in D \times T^u_{\text{disc}} \). Then by definition of \( \Phi \), we have that \( \pi_1((d_1, \xi_1) \Phi (d_2, \xi_2)) = \pi_1(d_1, \xi_1) + \pi_1(d_2, \xi_2) \).

**Example 16.** Recall the tv-monoid \( D^A_{\text{disc}} \) from Example 4 where \( \lambda_1 = 1, \lambda_2 = 0.8, \) and \( \lambda_i = 1 \) for every \( i > 2 \). The m-monoid \( A_{D^A_{\text{disc}}} = (A, \Phi, 0, \Omega) \) corresponding to \( D^A_{\text{disc}} \) is given as follows. We have \( A = \tilde{\mathbb{R}} \times T^u_{\text{disc}} \) and \( 0 = (-\infty, -\infty) \). Moreover, \( \Omega = \{\text{valtop}^k_0 \mid a \in \tilde{\mathbb{R}}, k \in \mathbb{N}\} \), and for every \( (a_1, \xi_1), (a_2, \xi_2) \in A \),
\[
(a_1, \xi_1) \Phi (a_2, \xi_2) = (\max\{a_1, a_2\}, \xi_1 \circ \xi_2).
\]
To illustrate the function \( \text{valtop}_0 \), we provide the following example calculation, which is the evaluation of the tree from Figure 2:
\[
\text{valtop}^2_1\left(\left(\frac{1}{\text{Val}(\xi_1)}, 1(0, 0, 0, 0)), \left(\frac{1}{\text{Val}(\xi_2)}, 1(0(0, 0), 0(0, 0)))\right)\right) = \left(2.8, 1(1(0, 0, 0, 0)), 1(0(0), 0(0, 0))\right)
\]

### 3.2 Relating m-wta and tv-wta

We define a relation between tv-wta over \( D \) and m-wta over \( A_D \). A tv-wta \( N \) and an m-wta \( M \) are related if they share the same states, final states, and whenever the transition weight of \( N \) is \( d \), the transition weight of \( M \) uses the operation \( \text{valtop}^k_0 \) of the correct arity \( k \). In the following, we formally define this and introduce some intermediate lemmas to show that automata which are related define the same language (up to projection).
**Definition 17.** Let $\mathcal{N} = (Q, \mu, F)$ be a tv-wta over $\Sigma$ and $D$, and let $\mathcal{M} = (Q', \delta, F')$ be an m-wta over $\Sigma$ and $A_D$. We call $\mathcal{N}$ and $\mathcal{M}$ related if

- $Q = Q'$ and $F = F'$;
- for every $k \in \mathbb{N}$, $\sigma \in \Sigma^{(k)}$, and $q, q_1, \ldots, q_k \in Q$, it holds that
  $$\mu_\sigma(q_1 \ldots q_k, q) = d \quad \text{if and only if} \quad \delta_\sigma(q_1 \ldots q_k, q) = \text{valtop}^{(k)}_d.$$  

Since given the tv-wta $\mathcal{N}$ (respectively the m-wta $\mathcal{M}$), the m-wta $\mathcal{M}$ (respectively tv-wta $\mathcal{N}$) is uniquely determined, this definition may be considered as a construction.

In the following three statements, let $\mathcal{N} = (Q, \mu, F)$ and $\mathcal{M} = (Q, \delta, F)$ be related.

**Lemma 18.** Let $\xi \in T_\Sigma$ and $r \in R_Q(\xi)$ be a run such that $\mu(\xi, r)$ is 0-free. Then we have that $\pi_2(\delta(\xi, r, \varepsilon)) = \mu(\xi, r)$.

**Proof.** We observe that for $d \in D$ and $\xi_1, \ldots, \xi_k \in \Sigma^*_D$ such that $d \neq 0$ and $\xi_1, \ldots, \xi_k$ are all 0-free, we have for every $d_1, \ldots, d_k \in D$ that

$$\pi_2(\text{valtop}^{(k)}_d)(d_1, \xi_1), \ldots, (d_k, \xi_k)) = d(\pi_2(d_1, \xi_1), \ldots, \pi_2(d_k, \xi_k)) \quad (\ast).$$

Now, for every position $w \in \text{pos}(\xi)$ with $\sigma = \xi(w)$ and $k = \text{rk}(\sigma)$, we can show by induction on the positions of $\xi$ that $\pi_2(\delta(\xi, r, w)) = \mu(\xi, r)|_w$:

$$\pi_2(\delta(\xi, r, w))$$
$$= \pi_2(\text{valtop}^{(k)}_d)((r(w_1), \ldots, r(w_k)))[\delta(\xi, r, w_1), \ldots, \delta(\xi, r, w_k)]$$
$$= \mu_\sigma(r(w_1) \ldots r(w_k), r(w))[\pi_2(\delta(\xi, r, w_1), \ldots, \pi_2(\delta(\xi, r, w_k))] \quad \text{(by } \ast \text{)}$$
$$= \mu_\sigma(r(w_1) \ldots r(w_k), r(w))[\mu(\xi, r)|_{w_1}, \ldots, \mu(\xi, r)|_{w_k}] \quad \text{(I.H.)}$$
$$= \mu(\xi, r)|_w$$

The claim follows for $w = \varepsilon$. \hfill \Box

**Lemma 19.** Let $\xi \in T_\Sigma$ and $r \in R_Q(\xi)$. Then

$$\delta(\xi, r, \varepsilon) = \begin{cases} (\text{Val}(\mu(\xi, r)), \mu(\xi, r)) & \text{if } \mu(\xi, r) \text{ is 0-free,} \\ (0, 0) & \text{otherwise.} \end{cases}$$

**Proof.** By a case analysis. If $\mu(\xi, r)$ is 0-free, Lemma 18 proves the equality $\pi_2(\delta(\xi, r, \varepsilon)) = \mu(\xi, r)$ in the second component and the first component of the tuple is just the valuation of the second one.

If $\mu(\xi, r)$ is not 0-free, then there is a position $w \in \text{pos}(\xi)$ with $\sigma = \xi(w)$, $k = \text{rk}(\sigma)$ such that $\mu_\sigma(r(w_1) \ldots r(w_k), r(w)) = 0$, and by definition of relatedness, we have $\delta_\sigma(r(w_1) \ldots r(w_k), r(w)) = \text{valtop}^{(k)}_d$. Hence, by definition of valtop, the result is $(0, 0)$, which carries on as $A_D$ is absorptive. \hfill \Box
Lemma 20. $[\mathcal{N}] = \pi_1([\mathcal{M}])$.

Proof. For every $\xi \in T_\Sigma$,

$[\mathcal{N}](\xi) = \sum_{r \in R_q(\xi)} \sum_{r(\epsilon) \in F} \text{Val}(\mu(\xi, r))$

$= \sum_{r \in R_q(\xi)} \sum_{r(\epsilon) \in F} \pi_1\left( \begin{cases} (\text{Val}(\mu(\xi, r)), \mu(\xi, r)) & \text{if } \mu(\xi, r) \text{ is 0-free} \\ (0, 0) & \text{otherwise} \end{cases} \right)$

$= \sum_{r \in R_q(\xi)} \sum_{r(\epsilon) \in F} \pi_1(\delta(\xi, r, \epsilon))$ (Lemma 19)

$= \pi_1\left( \sum_{r \in R_q(\xi)} \sum_{r(\epsilon) \in F} \delta(\xi, r, \epsilon) \right)$ (Observation 15)

$= \pi_1([\mathcal{M}])(\xi)$. □

Example 21. Recall the tv-monoid $D_\Sigma^\Lambda$ from Example 4, the corresponding m-monoid $A_{D_\Sigma^\Lambda}$ from Example 16, and the tv-wta $N_\Sigma^\Lambda$ from Example 8. We construct the related m-wta $M_\Sigma^\Lambda' = (\{q_0, q_1\}, \delta, \{q_0, q_1\})$, where for $q \in \{q_0, q_1\}$,

$\delta_\alpha(\epsilon, q_0) = \text{valtop}^{(0)}_0$, $\delta_\sigma(q q_1, q_1) = \text{valtop}^{(2)}_1$,

$\delta_\gamma(q, q_0) = \text{valtop}^{(1)}_0$, $\delta_\sigma(q_0 q_1) = \text{valtop}^{(2)}_0$,

and every other entry of $\delta$ is set to $\text{valtop}^{(k)}_\infty$ of the correct arity $k$. Note that the state behavior does not change.

Reconsider the tree from Figure 2 together with the run that maps each position which is labeled by $\sigma$ to $q_1$, and every other position to $q_0$. The weight of this run in $M_\Sigma^\Lambda'$ is $(2.8, 1(0, 0), 0(0, 0), 0(0, 0)))$. Note that this tuple’s first component contains the weight of the run in the tv-wta $N_\Sigma^\Lambda$, and the second component contains the tree of values of this run on the input tree.

The results of this section are summed up in the following theorem.

Theorem 22. Let $D$ be a tv-monoid, $A_D$ the corresponding m-monoid, and $\Sigma$ be a ranked alphabet. The following holds:

1. For every tv-wta $\mathcal{N}$ over $\Sigma$ and $D$, there is an m-wta $\mathcal{M}$ over $\Sigma$ and $A_D$ such that $[\mathcal{N}] = \pi_1([\mathcal{M}])$.

2. For every m-wta $\mathcal{M}$ over $\Sigma$ and $A_D$, there is a tv-wta $\mathcal{N}$ over $\Sigma$ and $D$ such that $\pi_1([\mathcal{M}]) = [\mathcal{N}]$. 

14
The theorem immediately follows from Lemma 20 and Definition 17 seen as a construction. Observe that if an m-wta \( M \) is related to a tv-wta \( N \), then they have equal state sets. Hence, taking the number of an automaton’s states as its size, both \( M \) and \( N \) in items (1) and (2) above can be chosen such that their size is linear in the size of the original automaton.

The following corollary is an immediate consequence of the above theorem together with the Büchi-like characterizations in \([5, \text{Theorem 5.5}]\) and \([10, \text{Theorem 4.1}]\).

**Corollary 23.** Let \( D \) be a regular ptv-monoid, \( A_D \) the corresponding m-monoid, and \( \Sigma \) a ranked alphabet. Then the following holds:

1. For every \( \forall \)-restricted and strongly \( \land \)-restricted tv-mso formula \( \varphi \), there is an m-expression \( e \) over \( \Sigma \) and \( A_D \) such that \( \llbracket \varphi \rrbracket = \pi_1(\llbracket e \rrbracket) \).

2. For every m-expression \( e \) over \( \Sigma \) and \( A_D \), there is a \( \forall \)-restricted and strongly \( \land \)-restricted tv-mso formula \( \varphi \) over \( \Sigma \) and \( D \) such that \( \pi_1(\llbracket e \rrbracket) = \llbracket \varphi \rrbracket \).

However, both items (1) and (2) in the corollary give rise to a constructed m-expression \( e \) (resp. tv-mso formula \( \varphi \)) whose size grows nonelementary in the size of the formula \( \varphi \) (resp. of the expression \( e \)). This means that there is no elementary function \( f : \mathbb{N} \to \mathbb{N} \) such that for every formula \( \varphi \) (resp. for every m-expression \( e \)) of length \( n \in \mathbb{N} \), the length of the constructed formula \( \varphi \) (resp. of the expression \( e \)) is bounded by \( f(n) \). This is clearly evident from the facts that (i) both the proof of \([5, \text{Theorem 5.5}]\) and of \([10, \text{Theorem 4.1}]\) make use of the classical result of \([19, 3]\) in order to construct a tree automaton that accepts the tree language of a Boolean subformula; (ii) Boolean mso formulas can be easily embedded both into m-expressions and tv-mso formulas; and (iii) already in the case of mso over finite words, the size of the constructed automata can not be bounded by an elementary function, as satisfiability of this logic is not elementary-recursive \([16, 17]\), while the emptiness problem of finite string automata can be decided in linear time.

However, this blowup can be avoided by giving an explicit construction for (1), resp. (2), which preserves Boolean subformulas. We will sketch such constructions in the next section.

### 3.3 Relating m-Expressions and tv-mso Formulas

In this section, we give semantics-preserving transformation functions (a) from tv-mso formulas over \( D \) to m-expressions over \( A_D \) and (b) the other way around. Common structures like Boolean subformulas, disjunction, and existential quantification which are available in both logics are translated directly. Features which are not in the respective other logic are simulated by more complex formulas.

The semantics of tv-mso formulas allows weighting trees with arbitrary values \( d \in D \). Since a similar construct is not available for m-expressions we need to guarantee that we can obtain a value \( d \in D \) independent from the input tree. Hence, we require the ptv-monoid \( D \) to be regular (cf. page 10).

In the following, let \( D \) be regular.

First, we outline a semantics-preserving transformation from tv-mso to m-expressions. This can be done by induction on the formula.
Construction 24. Let $\varphi$ be a $\forall$-restricted and strongly $\land$-restricted tv-mso formula over $\Sigma$ and $D$. We construct the m-expression $t(\varphi)$ over $\Sigma$ and $A_D$ by induction on the structure of $\varphi$ as follows:

$\varphi = d$: Since $D$ is regular, there is a tv-wta $N_d = (Q, \mu, F)$ which assigns $d$ to every input tree $\xi \in T_\Sigma$. Our aim is to construct an m-expression which simulates the behavior of this automaton. This is done using ideas from [7, Section 5.2], the semiring case.

The automaton $N_d$ is encoded into an m-expression by representing every possible transition $\mu_\sigma(q_1 \ldots q_k, q)$ of $N_d$ by a distinct second-order variable $X_i$. For the sake of readability, $X_i$ is denoted by $X_{(q_1, \ldots, q_k, \sigma, q)}$. Denote the set of all such variables by $U$. A run of $N_d$ on $\xi$ then corresponds to a variable assignment of $\xi$ for the variables from $U$.

In [7, Definition 5.10] there is an mso formula that checks if a run of $N_d$ on $\xi$ encoded in this way is indeed a valid run (originally called ‘run formula’ and denoted by $\psi_{M, \varphi}$). We denote this formula by $\varphi_{\text{run}}$. Since the automaton in [7] has no final states, we use an mso formula $\varphi_{\text{final}}$ that additionally enforces the run to end in a final state.

We define $\omega_d$ as the $\Sigma[U]$-family of operations as follows. For every $k \in \mathbb{N}$, $\sigma \in \Sigma(k)$, and $q, q_1, \ldots, q_k \in Q$, we set $(\omega_d)^{(k)}((X_{(q_1, \ldots, q_k, \sigma, q)})) = \text{valtop}^k_{\mu_\sigma(q_1, \ldots, q_k, q)}$, and we assign $\text{valtop}_0$ of the correct arity to every other combination of symbol and variables. Finally, we construct the m-expression

$$t(d) = \sum_{X_{(q_1, \ldots, q_k, \sigma, q)}} \cdots \sum_{X_{(q_1, \ldots, q_k, \sigma, q)}} (\varphi_{\text{run}} \land \varphi_{\text{final}}) \triangleright H(\omega^d),$$

where $\sum_{X_{(q_1, \ldots, q_k, \sigma, q)}}$ quantifies over each variable from $U$, in some arbitrary but fixed order.

$\varphi = \varphi_1 \land \varphi_2$: As $\varphi$ is strongly $\land$-restricted, one of the following cases holds.

Case 1: The formula $\varphi_1 \land \varphi_2$ is almost boolean, i.e., according to Observation 12, we have $\text{step}(\varphi) = (d_1, \psi_1) \ldots (d_n, \psi_n)$ and we set

$$t(\varphi_1 \land \varphi_2) = \sum^{+}_{i \in [n]} (\psi_i \triangleright t(d_i)),$$

where $\sum^{+}$ abbreviates a finite sum of m-expressions using $\cdot$.

Case 2: Either $\varphi_1$ or $\varphi_2$ is Boolean and the respective other formula is not almost Boolean. Assume that $\varphi_1$ is Boolean, then $t(\varphi_1 \land \varphi_2) = \varphi_1 \triangleright t(\varphi_2)$.

$\varphi$ is Boolean, $\varphi = \varphi_1 \lor \varphi_2$, $\varphi = \exists x. \psi$, or $\varphi = \forall X. \psi$;
We transform these cases straightforwardly using $\lor$, $\land$, $\sum^X$, or $\sum^X$, respectively.

$\varphi = \forall x. \psi$: Let free($\varphi$) = $\forall$. Since $\varphi$ is $\forall$-restricted, $\psi$ is almost Boolean and, by Observation 12, $\text{step}(\psi) = (d_1, \psi_1) \ldots (d_n, \psi_n)$. We use second order variables $X_1, \ldots, X_n$ to encode a partitioning on the positions of an input tree which corresponds to the partitioning on $T^\psi_{\Sigma[U]}$ that is induced by the formulas $\psi_1, \ldots, \psi_n$. Given this partitioning, the $\Sigma(X_1, \ldots, X_n)$-family of operations $\omega^\psi$ is defined such that $\omega^\psi((\sigma, U)) = \text{valtop}^k_{\mu_\sigma(q_1, \ldots, q_k, q)}$ whenever $U \cap \{X_1, \ldots, X_n\} = \{X_i\}$. We set

$$t(\forall x. \psi) = \sum_{X_1} \cdots \sum_{X_n} (\forall x. (\bigwedge_{i \in [n]} (x \in X_i) \leftrightarrow \psi_i)) \triangleright H(\omega^\psi).$$

The idea behind the above construction is based on [4, Lemma 4.4] and similar to the proof in [10, Lemma 5.10].
We analyze the size of the constructed m-expression $t(\varphi)$ in relation to the size of $\varphi$. Note that in the case $\varphi = d$, $|t(d)|$ depends on the size of the tv-wta $N_d$ (where the size of $N_d$ is, say, the number of its states). It is not clear whether there is a bound for the size of $N_d$ in general. Hence, to analyze the growth of $|t(d)|$ in isolation, we make the assumption that there is some constant $c$ which depends only on $D$ and bounds the size of $N_d$, for every $d \in D$.

We will call such a $D$ finitary regular. As an example, it is easy to see that $D$ is finitary regular if it is left-multiplicative, or if it is left-Val-distributive, as defined in [5].

**Lemma 25.** Let $\varphi$ be a $\forall$-restricted and strongly $\land$-restricted tv-mso formula over $\Sigma$ and $D$, and $\forall \supseteq \text{free}(\varphi)$. Then $\llbracket \varphi \rrbracket_\nu = \pi_1(\llbracket t(\varphi) \rrbracket_\nu)$. Moreover, if $D$ is finitary regular, $|t(\varphi)|$ is bounded by a double exponential function in $|\varphi|$.

**Proof.** The lemma’s first part is proved by showing, by induction on the structure of $\varphi$, that for every $\xi \in T_{\Sigma,\nu}^*$, the equation $\llbracket \varphi \rrbracket_\nu(\xi) = \pi_1(\llbracket t(\varphi) \rrbracket_\nu)(\xi)$ holds. The interesting parts of the proof are the cases $\varphi = d$, $\varphi = \exists x. \psi$, and $\varphi = \varphi_1 \land \varphi_2$.

In the case $\varphi = d$, a tree $\xi' \in L_\nu(\varphi_{\text{run}} \land \varphi_{\text{final}})$ encodes a run $r$ of $N_d$ on $\xi$, moreover $\text{Val}(\mu(\xi, r))$ is then equal to $\pi_1(\llbracket H(\omega^d) \rrbracket_\nu(\xi'))$. But then also $\llbracket d \rrbracket_\nu(\xi) = \pi_1(\llbracket t(d) \rrbracket_\nu)(\xi)$, because we sum over all possible runs on $\xi$.

For the case $\varphi = \exists x. \psi$, recall the correspondence between the recognizable step function of $\llbracket \psi \rrbracket_\nu$ and the unique partitioning of $\text{pos}(\xi)$ into $W_1, \ldots, W_n$ induced by the formulas $\psi_1, \ldots, \psi_n$, and encoded by the second-order variables $X_1, \ldots, X_n$. One can prove by well-founded induction on $w \in \text{pos}(\xi)$ that

$$\llbracket H(\omega^\psi) \rrbracket_{\forall u(X_1, \ldots, X_n)}(\xi[X_1 \mapsto W_1, \ldots, X_n \mapsto W_n]|_w) = (\text{Val}(\xi[\psi]|_w), \xi[\psi]|_w).$$

Here, $\xi[\psi]$ denotes the tree of values as defined in Section 2.4.2. As a direct consequence, $\llbracket \exists x. \psi \rrbracket_\nu(\xi) = \pi_1(\llbracket t(\exists x. \psi) \rrbracket_\nu)(\xi)$.

Finally, we treat the case $\varphi = \varphi_1 \land \varphi_2$. The subcase that $\varphi_1$ (resp. $\varphi_2$) is Boolean is trivial. In the other subcase, $\varphi$ is almost Boolean, and $\text{step}(\varphi) = (d_1, \psi_1) \ldots (d_n, \psi_n)$. Note that the languages $L_\nu(\psi_i), i \in [n]$, form a partitioning on $T_{\Sigma,\nu}^*$. It is evident that $t(\varphi_1 \land \varphi_2)$ encodes the recognizable step function, i.e., $\pi_1(\llbracket \sum_{i \in [n]} \psi_i \triangleright t(d_i) \rrbracket_\nu(\xi)) = d_i$ if $\xi \in L_\nu(\psi_i)$.

We still must substantiate the claim on the size of $t(\varphi)$. For a finitary regular ptv-monoid $D$, there are still two sources of growth: conjunctions and first-order universal quantifications in $\varphi$. In the former case $\varphi = \psi_1 \land \psi_2$, $|t(\psi_1 \land \psi_2)|$ is bounded by the number $n \in \mathbb{N}$ of partitions in the recognizable step function of $\psi_1 \land \psi_2$, and the sizes of their respective defining mso formulas. By close inspection of the construction in [10], it can be seen that $n$ may grow exponentially in the case of conjunction, and that each formula is of polynomial size in the size of the input.

A similar argument holds for the case of $\varphi = \exists x. \psi$. The size of the step function of $\psi$ is bounded exponentially and the size of the family $\omega^\psi$ is exponential in the size of the step function, since for each subset $U \subseteq [n]$, an operation from $\Omega$ must be stored. Hence, the size of the m-expression $t(\exists x. \psi)$ is double exponential in $|\psi|$, and thus in the size of $\varphi$.

We will now describe the other direction of the logical transformation. Given an m-expression over $A_D$, we construct a $\forall$-restricted and strongly $\land$-restricted tv-mso formula over the original ptv-monoid $D$. 

17
Construction 26. Given the m-monoid $A_D$ and an m-expression $e$ over $\Sigma$ and $A_{\mathbb{D}}$, we construct the tv-mso formula $t'(e)$ over $\Sigma$ and $\mathbb{D}$ by induction as follows.

Let $e = H(\omega)$, where $\omega$ is a $\Sigma_{\mathfrak{U}}$-family of operations in $\Omega$. For every symbol $\sigma \in \Sigma$ and every subset of variables $V \subseteq \mathcal{U}$, we construct a Boolean formula $\psi_{\sigma, V}(x)$ with the following property. For every $\xi \in T^\omega_{\mathfrak{U}}$, and every $w \in \text{pos}(\xi)$, we have that $[\psi_{\sigma, V}(\xi[x \rightarrow w])]$ is true iff $\xi(w) = (\sigma, V)$. We set $t'(H(\omega)) = \forall x.\psi_{H(\omega)}^V$, where $\psi_{H(\omega)}^V = \bigvee_{\sigma \in \Sigma} \bigvee_{V \subseteq \mathcal{U}} \psi_{\sigma, V}(x) \land d(\sigma, V)$ and $d(\sigma, V)$ is the $d \in D$ such that $\omega(\sigma, V) = \text{valtop}_d^{\text{rk}(\sigma)}$. In the case $e = \varphi \lor e'$, we set $t'(\varphi \lor e') = \varphi \land t'(e')$. The other cases for $e$ are straightforward.

The formula $\psi_{H(\omega)}$ consists of finite number of disjunctions and conjunctions of Boolean formulas and values from $D$, hence it is almost Boolean. As a consequence, the constructed formula $t'(e)$ is $\forall$-restricted and strongly $\land$-restricted, as required for definability of weighted tree languages by tv-mso formulas (cf. Section 2.4.2).

Lemma 27. Let $e$ be an m-expression over $\Sigma$ and $A_{\mathbb{D}}$, and $\forall \Sigma \supseteq \text{free}(e)$. Then $\pi_1([e]_V) = [t'(e)]_V$. Moreover, $[t'(e)]_V$ is in $O(|e| \cdot \log|e|)$.

Proof. Consider the case that $e = H(\omega)$, for a $\Sigma_{\mathfrak{U}}$-family of operations $\omega$ in $\Omega$ such that $\mathcal{U} \subseteq V$. By induction on $\text{pos}(\xi)$, one can prove the auxiliary property that $[H(\omega)]_V(\xi) = (\text{Val}(\xi[\psi_{H(\omega)}^V]), \xi[\psi_{H(\omega)}^V])$ for every $\xi \in T^\omega_{\mathfrak{U}}$. It follows that $\pi_1([H(\omega)]_V(\xi)) = [t'(H(\omega))_V](\xi)$. The other cases for $e$ go through easily.

Note that the size of each subformula $\psi_{\sigma, V}$ of $t'(H(\omega))$ is linear in $|U|$. Hence, $|t'(H(\omega))|$ is in $O(|\Sigma| \cdot 2^{|U|} \cdot |U|)$. Recall that $|H(\omega)| = |\Sigma| \cdot 2^{|U|}$. Therefore, $t'(H(\omega))$ is in $O(H(\omega) \cdot \log|H(\omega)|)$. The other, straightforward, cases of the transformation do not result in an essential blow-up of the size.

Summing up the results of this section allows us to strengthen Corollary 23 by giving concrete bounds for the sizes of the constructed formulas.

Theorem 28. Let $\mathbb{D}$ be a regular ptv-monoid, $A_D$ the corresponding m-monoid, and $\Sigma$ a ranked alphabet. Then the following holds:

1. For every $\forall$-restricted and strongly $\land$-restricted tv-mso formula $\varphi$, there is an m-expression $e$ over $\Sigma$ and $A_{\mathbb{D}}$ such that $[\varphi] = \pi_1([e])$. Furthermore, if $\mathbb{D}$ is finitary regular, then $e$ can be chosen such that its size is at most double exponential in the size of $\varphi$.

2. For every m-expression $e$ over $\Sigma$ and $A_{\mathbb{D}}$, there is a $\forall$-restricted and strongly $\land$-restricted tv-mso formula $\varphi$ over $\Sigma$ and $\mathbb{D}$ such that $\pi_1([e]) = [\varphi]$. Furthermore, $\varphi$ can be chosen such that its size is in $O(|e| \cdot \log|e|)$.

The above theorem poses the question whether it would have been more economical just to establish the connection between tv-mso and m-expressions, and use the Büchi-like results of [5] and [10] in order to receive the according link between m- and tv-recognizability as a corollary. However, as remarked below Theorem 22, the direct construction of automata is very efficient, while the detour over logics would again introduce an unnecessary blowup.
4 Closure under Homomorphisms

In Section 3, languages recognizable by tv-wta were characterized “up to projection” by languages recognizable by m-wta. In this section we show how to avoid this projection. For this, we need a special property of m-wta based on a construction for semiring weighted tree automata from [6, Lemma 4.8].

Definition 29. An m-wta $M = (Q, \delta, F)$ is final-state normalized if for every $k \in \mathbb{N}$, $\sigma \in \Sigma^{(k)}$, and $q, q_1, \ldots, q_k \in Q$, whenever there is some $i \in [k]$ such that $q_i \in F$, it holds that $\delta(\sigma(q_1 \ldots q_k), q) = 0^{(k)}$.

Note that the normal form in [6] requires a single final state, whereas an arbitrary number of final states is admissible here. It is easy to see that every m-wta can be transformed into an equivalent final-state normalized one. One may just add dedicated final states which are disallowed from occurring below other states in a run (by setting the assigned weights of the corresponding transitions to the operation $0^{(k)}$ of appropriate arity $k$).

Observation 30. Let $M = (Q, \delta, F)$ be a final-state normalized m-wta. For every $\xi \in T_q$ and $r \in R_q(\xi)$ such that there is a $w \in \text{pos}(\xi) \setminus \{\epsilon\}$ with $r(w) \in F$, we have that $\delta(\xi, r, w) = 0$; cf. [18, Lemma 1] for a similar observation.

Lemma 31. Let $\Sigma$ be a ranked alphabet, $A = (A, +_A, 0_A, \Omega)$ be an m-monoid, $B = (B, +_B, 0_B)$ be a commutative monoid and $h : A \to B$ be a monoid homomorphism from the additive monoid $(A, +_A, 0_A)$ of $A$ to $B$. Then there is an m-monoid $C = (C, +_C, 0_C, \Theta)$ with $B \subseteq C$ such that for every $L \in \text{Rec}(\Sigma, A)$, we have $h(L) \in \text{Rec}(\Sigma, C)$.

Proof. The intuition behind this proof is to define an m-monoid $C$ whose carrier set contains $A$ and $B$, where $0_A$ and $0_B$ are identified. Without loss of generality, we may assume that $A$ and $B$ are disjoint. We construct the m-monoid $C = (C, +_C, 0_C, \Theta)$ as follows.

The carrier set is $C = B \cup A' \cup D$, where $A' = A \setminus \{0_A\}$ and $D = \{(a, b) \in A \times B \mid a \neq 0_A, b \neq 0_B\}$, and the neutral element $0_C = 0_B$.

Furthermore, for every $c_1, c_2 \in C$, we let $c_1 +_C c_2 = e(i(c_1) +_A +_{A \times B} i(c_2))$, where $+_A$ is the sum of the direct product of the monoids $(A, +_A, 0_A)$ and $B$, while $i : C \to A \times B$ and $e : A \times B \to C$ are such that $i(c) = \begin{cases} (0_A, c) & \text{if } c \in B, \\ (c, 0_B) & \text{if } c \in A', \\ c & \text{otherwise}, \end{cases}$ and $e((a, b)) = \begin{cases} b & \text{if } a = 0_A, \\ a & \text{if } b = 0_B \text{ and } a \in A', \\ (a, b) & \text{otherwise}, \end{cases}$

for every $c \in C$, and, respectively, $(a, b) \in A \times B$.

We set $\Theta = \{\text{ext}(\omega) \mid \omega \in \Omega\} \cup \{\text{ext}(h \circ \omega) \mid \omega \in \Omega\}$, where for every $k \in \mathbb{N}$ and mapping $f : A^k \to (A \cup B)$, the extension $\text{ext}(f) : C^k \to C$ of $f$ is given for every $c_1, \ldots, c_k \in C$ as $\text{ext}(f)(c_1, \ldots, c_k) = \begin{cases} f(c_1, \ldots, c_k) & \text{if } c_1, \ldots, c_k \in A' \text{ and } f(c_1, \ldots, c_k) \neq 0_A, \\ 0_C & \text{otherwise}. \end{cases}$

This concludes the construction of $C$. 
We can prove that, by this construction, Corollary 23 is lost. To show this, it suffices to define an m-wta as an m-wta which, for some input tree $\xi$, reaches in a final state, computing in $A$. Finally, when the root of the input tree is reached in a final state, $M'$ additionally applies $h$ to the result, yielding an output value in $B$. We can prove that, by this construction, $[M'] = h(L)$. 

Lemma 31 allows us to state the following theorem, in which we again relate the classes of languages recognizable (resp. definable) by m-wta and by tv-wta (resp. by m-expressions and by tv-mso formulas), but now without the projection.

**Theorem 32.**

1. For every tv-monoid $D$, there is an m-monoid $A$ such that $\text{Rec}(\Sigma, D) \subseteq \text{Rec}(\Sigma, A)$.

2. For every regular product tv-monoid $D$, there is an m-monoid $A$ such that $\text{Def}(\Sigma, D) \subseteq \text{Def}(\Sigma, A)$.

**Proof.** Statement (1): Let $L \in \text{Rec}(\Sigma, D)$ and $\mathcal{N}$ be a tv-wta over $\Sigma$ and $D$ such that $[\mathcal{N}] = L$. Then we know by Theorem 22 that there is an m-wta $M$ over $\Sigma$ and $A_D$ such that $[\mathcal{N}] = \pi_1([M])$. Since $\pi_1$ is a monoid homomorphism from the additive monoid of $A_D$ into $(D, +, 0)$, by Lemma 31, there is an m-monoid $C$ such that $\pi_1([\mathcal{N}]) \in \text{Rec}(\Sigma, C)$. Hence, we can conclude that $\text{Rec}(\Sigma, D) \subseteq \text{Rec}(\Sigma, C)$.

Statement (2): Let $L \in \text{Def}(\Sigma, D)$ and $\varphi$ be a $V$-restricted and strongly $\wedge$-restricted tv-mso formula such that $L = [\varphi]$. Then we know by Corollary 23 that there is an m-expression $e$ over $\Sigma$ and $A_D$ such that $[\varphi] = \pi_1([e])$ and hence $\pi_1([e]) \in \pi_1(\text{Def}(\Sigma, A_D))$. By [10, Theorem 4.1], we can conclude that $\pi_1(\text{Def}(\Sigma, A_D)) = \pi_1(\text{Rec}(\Sigma, A_D))$ and by Lemma 31, it follows that there is an m-monoid $C$ such that $\pi_1(\text{Rec}(\Sigma, A_D)) \subseteq \text{Rec}(\Sigma, C)$. Using [10, Theorem 4.1] again, we have that $\text{Rec}(\Sigma, C) = \text{Def}(\Sigma, C)$. Hence, it holds that $\text{Def}(\Sigma, D) \subseteq \text{Def}(\Sigma, C)$ which proves the claim. 

A picture visualizing the idea can be found in Figure 3. As remarked above, with this approach the equality of the classes of languages under consideration in Theorems 22 and Corollary 23 is lost. To show this, it suffices to define an m-wta $M'$ over $\Sigma$ and $A$ (from Theorem 32) such that there is no tv-wta recognizing the same language. Clearly, it is possible to construct an m-wta which, for some input tree $\xi \in T_D$, does not apply $\pi$ at the root of $\xi$. Hence, $[M'](\xi) \in (D \times T_D)$, which cannot be recreated by any tv-wta over $D$. 

20
5 Conclusion

In this paper we showed that the languages recognizable by tv-wta (resp. definable by tv-mso formulas) are recognizable by m-wta (definable by m-expressions), and thus gave some insight on the close relation between the two automaton models (cf. Figure 1, page 3). The automaton equivalence together with the already known Büchi-like results could be used to obtain the semantic equivalence of the definable classes, but led to a nonelementary blow-up when transforming formulas from one logic into the other. We reduced this blow-up by giving direct transformations between the logics and obtained a double exponential upper bound.

We also showed that the additional use of the projection in Theorems 22 and Corollary 23 does not lead to an increase in expressive power, by proving the closure of the m-recognizable weighted tree languages under arbitrary monoid homomorphisms.

An open question is how to simplify the logic transformations if even further restricted formulas are given or if the ptv-monoid has additional properties. For example, given a left-multiplicative tv-monoid (cf. [5, p. 43]), the construction of the automaton $N_d$ is not necessary in the transformation from tv-mso formulas to m-expressions, as arbitrary values can be created more easily. Similar simplifications could apply for other syntactic restrictions of tv-mso formulas or different properties of ptv-monoids.

6 Acknowledgments

The work of Markus Teichmann was supported by DFG Graduiertenkolleg 1763 (QuantLA). Johannes Osterholzer was partially supported by DFG Graduiertenkolleg 1763 (QuantLA).

On a final note, the authors want to express their gratitude to both anonymous reviewers. Their comments and hints, in particular on the article’s first submitted version, were valuable and led to a clearer account of the ideas contained in this work.
References


