Weighted Symbolic Automata with Data Storage

Luisa Herrmann(✉) and Heiko Vogler

Faculty of Computer Science, Technische Universität Dresden,
Nöthnitzer Str. 46, 01062 Dresden, Germany
{Luisa.Herrmann,Heiko.Vogler}@tu-dresden.de

Abstract. We introduce weighted symbolic automata with data storage, which combine and generalize the concepts of automata with storage types, weighted automata, and symbolic automata. By defining two particular data storages, we show that this combination is rich enough to capture symbolic visibly pushdown automata and weighted timed automata. We introduce a weighted MSO-logic and prove a Büchi-Elgot-Trakhtenbrot theorem, i.e., the new logic and the new automaton model are expressively equivalent.

1 Introduction

Finite-state (string) automata have been generalized in at least three directions.

Due to the introduction of a wealth of new automata models, like pushdown automata, stack automata, nested stack automata, and counter automata, Scott proposed a homogeneous framework [16]. Using the notions of [9], such an *automaton with storage* consists of an automaton and a storage type; in each transition, the automaton can test the current storage configuration by a predicate (like: top $= \gamma$?) and transform it by an instruction (like: push(δ) or pop).

In a second generalization, each transition of a finite-state automaton was equipped with a weight taken from some semiring in order to analyse quantitative aspects of the recognition process. This led to the concept of *weighted automata* and its well investigated theory, cf. e.g. [3,8,13,15]. In recent work, unital valuation monoids were used as weight algebras [4] in order to calculate along a run of an automaton also with non-sequential operations, like average. In the literature, combinations of the first two generalizations were investigated: weighted pushdown automata over semirings [13] and unital valuation monoids [6], and weighted automata over arbitrary storage types and unital valuation monoids [11,20].

In a third generalization, finite-state automata were allowed to process input strings over an arbitrary, not necessarily finite set. This extension is relevant, e.g., when dealing with XML-documents involving data. An example of such automata are symbolic automata [17,19] in which each transition τ involves a unary predicate π; τ is applicable to the current input symbol if it satisfies π.

In this paper, we introduce a new automaton model, called *weighted symbolic automata with data storage*. It captures all three mentioned generalizations and it

L. Herrmann—Supported by DFG Graduiertenkolleg 1763 (QuantLA).

S. Brlek and C. Reutenauer (Eds.): DLT 2016, LNCS 9840, pp. 203–215, 2016.
DOI: 10.1007/978-3-662-53132-7_17

is defined in the same modular style as automata with storage introduced in [9,16]. As weight structure we choose unital valuation monoids. We extend the concept of storage type to that of a *data storage type*. There, predicates and instructions do not only depend on the current storage configuration, but also on storage inputs. In each transition, a predicate checks a property of the current data symbol of the input string (as in symbolic automata); via an encoding function specified in the automaton, this data symbol is transformed into a storage input. In this sense, predicates and instructions become 'sensitive' to the input string.

It turns out that our combination of symbolicalness of input strings and of sensitivity of predicates and instructions is rich enough to capture two recently introduced classes of automata which can process words over infinite sets. We define the data storage types $\mathrm{VP}(N)$ and $\mathrm{TIME}(\mathcal{C})$ and show that weighted automata over $\mathrm{VP}(N)$ and $\mathrm{TIME}(\mathcal{C})$ are exactly the (weighted version of) symbolic visibly pushdown automata [1] and weighted timed automata [7], respectively.

Moreover, we introduce a *weighted MSO-logic over data storage types* extending [20] by employing an infinite set of input symbols and a data storage type. Each formula of this logic has the form $\sum_B^\eta e$ where η is an encoding of input symbols into storage inputs, and $\sum_B$ represents the weighted version of a second-order existential quantification over the second-order behavior variable B; it ranges over behaviors of the underlying data storage type. Intuitively, a behavior is an executable string $(p_1, f_1) \dots (p_n, f_n)$ of pairs of predicates p_i and instructions f_i. The subformula e is an expression as defined in [20] (also cf. [10, Definition 3.1]); we note that, for semirings, such expressions are equivalent to the fragment of restricted weighted MSO-logic introduced in [2] (cf. [10, Proposition 5.14]).

We prove a Büchi-Elgot-Trakhtenbrot (BET) theorem (cf. Theorems 13 and 16) stating that weighted symbolic automata over data storage types are expressively equivalent to weighted MSO-logic over data storage types. In particular, we obtain the BET theorem for weighted symbolic visibly pushdown automata (which is new) and for weighted timed automata (which is an alternative to [14, Theorem 41]). As a consequence of our BET theorem we obtain that, for each bounded lattice, the satisfiability problem of weighted MSO-logic over $\mathrm{VP}(N)$ is decidable.

2 Preliminaries

Notations and Notions. We denote the set of natural numbers including zero by $\mathbb{N}$. For $n \in \mathbb{N}$ we let $[n]$ denote the set $\{i \in \mathbb{N} \mid 1 \leq i \leq n\}$. Thus $[0] = \emptyset$. In the following let A, $A_1, \dots, A_n$, and B be sets. The set of all words over A is denoted by A^*. For each $w \in A^*$, $|w|$ is the length of w, $\mathrm{pos}(w) = \{1, \dots, |w|\}$ is the set of *positions of* w, and w_i is the label at the i-th position of w. The *empty word* (of length 0) is denoted by ε. We let $A^+ = A^* \backslash \{\varepsilon\}$. A *relabeling* is a mapping $\rho\colon A \to \mathcal{P}(B)$. We denote the unique extension of ρ to the morphism from the free monoid $(A^*, \cdot, \varepsilon)$ to the monoid $(\mathcal{P}(B^*), \circ, \{\varepsilon\})$, where $\circ$ denotes language concatenation, also by ρ. For each $L \subseteq A^*$ we define $\rho(L) = \bigcup_{w \in L} \rho(w)$. For each $i \in [n]$ we define the *i-th projection* as function $(.)_i\colon A_1 \times \dots \times A_n \to A_i$ such that for each $(a_1, \dots, a_n) \in A_1 \times \dots \times A_n$ we have $(a_1, \dots, a_n)_i = a_i$. We require that each function which we consider in this work is computable.

Unital Valuation Monoids. The concept of valuation monoid was introduced in [4] and extended in [6] to unital valuation monoid. A *unital valuation monoid* is a tuple $(K, +, \mathrm{val}, 0, 1)$ where $(K, +, 0)$ is a commutative monoid and $\mathrm{val}\colon K^* \to K$ is a mapping such that (i) $\mathrm{val}(k) = k$ for each $k \in K$, (ii) $\mathrm{val}(k) = 0$ for each $k \in K^*$ whenever $k_i = 0$ for some $i \in [|k|]$, (iii) $\mathrm{val}(k1k') = \mathrm{val}(kk')$ for every $k, k' \in K^*$, and (iv) $\mathrm{val}(\varepsilon) = 1$. Moreover, K is called *zero-sum-free*, if $k + k' = 0$ implies $k = k' = 0$. *In the rest of this paper, we let K denote an arbitrary unital valuation monoid $(K, +, \mathrm{val}, 0, 1)$ unless specified otherwise.*

Example 1. Recall that a strong bimonoid [5] is a structure $(K, +, \cdot, 0, 1)$, where $(K, +, 0)$ is a commutative monoid, $(K, \cdot, 1)$ is a monoid, and $a \cdot 0 = 0 \cdot a = 0$ for every $a \in K$. A *semiring* is a strong bimonoid in which $\cdot$ distributes over $+$ from both sides. A particular semiring is the Boolean semiring $(\mathbb{B}, \vee, \wedge, 0, 1)$ with $\mathbb{B} = \{0, 1\}$. A *bounded lattice* is a strong bimonoid in which both operations are idempotent (and other laws are satisfied). It is clear that each strong bimonoid is a unital valuation monoid $(K, +, \mathrm{val}, 0, 1)$, where for every $n \in \mathbb{N}$ and $k_1, \ldots, k_n \in K$ we let $\mathrm{val}(k_1 \ldots k_n) = k_1 \cdot \ldots \cdot k_n$ (see [6]). Unital valuation monoids can be used to compute averages. For this consider $K_{\mathrm{avg}} = (\mathbb{R} \cup \{\infty, -\infty\}, \sup, \mathrm{avg}, -\infty, \infty)$ with $\mathrm{avg}(a_1 \ldots a_n) = \frac{1}{n} \cdot \sum_{1 \leq i \leq n} a_i$ for every $a_1, \ldots, a_n \in \mathbb{R}$.

Weighted Languages. Let D be a non-empty set. A *K-weighted language (over D)* is a mapping of the form $r\colon D^* \to K$. We denote the set of all such mappings by $K\langle\langle D^* \rangle\rangle$. Let $r \in K\langle\langle D^* \rangle\rangle$. We denote the set $\{w \in D^* \mid r(w) \neq 0\}$ by $\mathrm{supp}(r)$ (*support of r*). Moreover, let $L \subseteq D^*$ and $w \in D^*$. We define the weighted language $(r \cap L) \in K\langle\langle D^* \rangle\rangle$ by $(r \cap L)(w) = r(w)$ if $w \in L$, and 0 otherwise. Now let $r' \in K\langle\langle D^* \rangle\rangle$. We define the *sum of r and r'* as the weighted language $r + r' \in K\langle\langle D^* \rangle\rangle$ by $(r + r')(w) = r(w) + r'(w)$.

Label Structure. Let D be a non-empty set. A *predicate over D* is a mapping $\pi\colon D \to \{0, 1\}$ and we identify π with $\{a \in D \mid \pi(a) = 1\}$. We say that π *is decidable* if it is decidable whether $\pi \neq \emptyset$. We denote by $\mathrm{Pred}(D)$ the set of all decidable predicates over D. For every $\Pi \subseteq \mathrm{Pred}(D)$ we define the Boolean closure $\mathrm{BC}(\Pi)$ as usual and we denote the always true predicate by T. Obviously, if Π is recursively enumerable, then so is $\mathrm{BC}(\Pi)$. A *label structure (over D)* is a tuple (D, Π), where $\Pi \subseteq \mathrm{Pred}(D)$ is a recursively enumerable set of predicates such that $\mathrm{BC}(\Pi) = \Pi$. If D is clear from the context, then we only write Π instead of (D, Π).

3 Weighted Symbolic Automata with Data Storage

Data Storage Types and Behavior. We extend the notion of storage type [9,16] in such a way that the predicates and instructions do not only depend on the current configuration, but also on storage inputs (which, in their turn, are encodings of the input of an automaton).

A *data storage type* is a tuple $S = (C, M, P, F, c_0)$ where C is a set (*configurations*), M is a set (*storage inputs*), P is a set of functions each of the type

$p\colon C \times M \to \{\text{true}, \text{false}\}$ (*predicates*), F is a set of partial functions each of the type $f\colon C \times M \to C$ (*instructions*), and $c_0 \in C$ (*initial configuration*).

If M is a singleton, then we reobtain the concept of storage type as introduced in [11,20]. *Throughout this paper we let S denote an arbitrary data storage type (C, M, P, F, c_0) unless specified otherwise.*

Example 2. (1) For some fixed elements c and m we define the *trivial storage type* as the data storage type $\textsc{Triv} = (\{\mathrm{c}\}, \{\mathrm{m}\}, \{\mathrm{p}_{\mathrm{true}}\}, \{\mathrm{f}_{\mathrm{id}}\}, \mathrm{c})$ where $\mathrm{p}_{\mathrm{true}}(\mathrm{c}, \mathrm{m}) = \text{true}$ and $\mathrm{f}_{\mathrm{id}}(\mathrm{c}, \mathrm{m}) = \mathrm{c}$.

(2) Let Count be the data storage type $(\mathbb{N}, \mathbb{N}, \{\mathsf{T?}, 0?\}, \{+, -\}, 0)$ where for every $c, d \in \mathbb{N}$ we let $\mathsf{T?}(c, d) = \text{true}$, $0?(c, d) = \text{true}$ iff $c = 0$, $+(c, d) = c + d$, and $-(c, d) = c - d$ if $c \geq d$ and undefined otherwise.

A central notion of our MSO-logic is the concept of storage behavior. Let Ω be a finite subset of $P \times F$. Also, let $n \in \mathbb{N}$, $m_1, \ldots, m_n \in M$, and $b = (p_1, f_1) \ldots (p_n, f_n) \in \Omega^*$. We call b an *$m_1 \ldots m_n$-behavior (over Ω)* if for every $i \in [n]$ we have $p_i(c', m_i) = \text{true}$ and $f_i(c', m_i)$ is defined where $c' = f_{i-1}(\ldots f_1(c_0, m_1) \ldots, m_{i-1})$. Note that $c' = c_0$ for $i = 1$. We denote the set of all $m_1 \ldots m_n$-behaviors over Ω by $\mathrm{B}(\Omega, m_1 \ldots m_n)$.

Example 3. Consider Count and $w = 284770 \in \mathbb{N}^*$ with $|w| = 6$. Then the word $(\mathsf{T?}, +)^3(\mathsf{T?}, -)^2(0?, +)$ is a w-behavior over $\{\mathsf{T?}, 0?\} \times \{+, -\}$.

Weighted Symbolic Automata with Data Storage. Let D be a set. A *K-weighted symbolic automaton with data storage type S and input D* (short: *(S, D, K)-automaton*) is a tuple $\mathcal{A} = (Q, \Pi, Q_0, Q_f, T, \mathrm{wt}, \eta)$ where Q is a finite set (*states*), Π is a label structure over D, $Q_0 \subseteq Q$ (*initial states*), $Q_f \subseteq Q$ (*final states*), $T \subseteq Q \times \Pi \times P \times Q \times F$ is a finite set (*transitions*), $\mathrm{wt}\colon T \times D \to K$ is a function (*weight assignment*), and $\eta\colon D \to M$ is a relabeling (*storage encoding*). We call $\mathcal{A}$ *projective* if η is a projection. Moreover, we call $\mathcal{A}$ *homogeneous* if for each transition $\tau \in T$ and for every $d_1, d_2 \in (\tau)_2$ we have $\mathrm{wt}(\tau, d_1) = \mathrm{wt}(\tau, d_2)$. In this case we view wt as function of type $T \to K$.

The set of *$\mathcal{A}$-configurations* is the set $Q \times D^* \times C$. For each transition $\tau = (q, \pi, p, q', f)$ in T we define the binary relation $\vdash^\tau$ on the set of $\mathcal{A}$-configurations as follows: for every $d \in D$, $w \in D^*$, and $c \in C$, we let $(q, dw, c) \vdash^\tau (q', w, f(c, \eta(d)))$ if $\pi(d)$ is true, $p(c, \eta(d))$ is true, and $f(c, \eta(d))$ is defined. The *computation relation of $\mathcal{A}$* is the binary relation $\vdash = \bigcup_{\tau \in T} \vdash^\tau$.

A *computation* is a sequence $\xi_0 \vdash^{\tau_1} \xi_1 \cdots \vdash^{\tau_n} \xi_n$ such that $n \in \mathbb{N}$, $\tau_1, \ldots, \tau_n$ are transitions, $\xi_0, \ldots, \xi_n$ are $\mathcal{A}$-configurations, and $\xi_{i-1} \vdash^{\tau_i} \xi_i$ for each $i \in [n]$. Sometimes we abbreviate this computation by $\xi_0 \vdash^{\tau_1 \ldots \tau_n} \xi_n$. Let $w = d_1 \ldots d_n \in D^*$ with $d_i \in D$. A *successful computation on w* is a computation $\theta = ((q_0, w, c_0) \vdash^{\tau_1 \ldots \tau_n} (q_f, \varepsilon, c'))$ for some $q_0 \in Q_0$, $q_f \in Q_f$, $c' \in C$, and $\tau_1, \ldots, \tau_n \in T$. The *weight of θ* is the element in K defined by

$$\mathrm{wt}(\theta) = \mathrm{val}(\mathrm{wt}(\tau_1, d_1) \ldots \mathrm{wt}(\tau_n, d_n))$$

and we denote the set of all successful computations on w by $\Theta_{\mathcal{A}}(w)$.

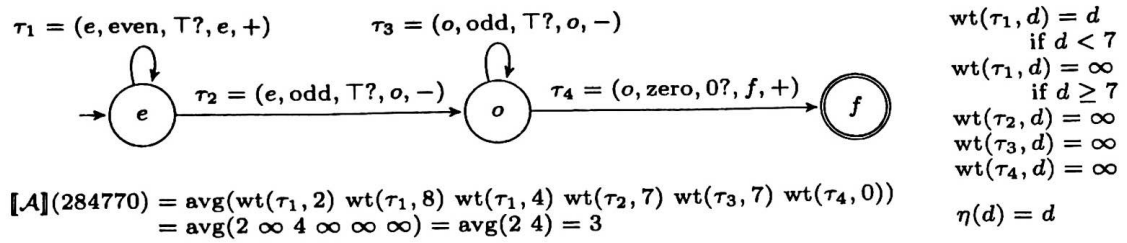

Fig. 1. The projective (COUNT, $\mathbb{N}$, K_{avg})-automaton $\mathcal{A}$ recognizing r.

The *weighted language recognized by* $\mathcal{A}$ is the K-weighted language $[\![\mathcal{A}]\!]\colon D^* \to K$ defined for every $w \in D^*$ by

$$[\![\mathcal{A}]\!](w) = \sum_{\theta \in \Theta_{\mathcal{A}}(w)} \mathrm{wt}(\theta).$$

A weighted language $r\colon D^* \to K$ is (S, D, K)-recognizable if there is an (S, D, K)-automaton $\mathcal{A}$ such that $r = [\![\mathcal{A}]\!]$. In the obvious way, we define projectively (S, D, K)-recognizable and homogeneously (S, D, K)-recognizable.

Example 4. Consider the language $L \subseteq \mathbb{N}^*$ consisting of words $u_1 \dots u_n v_1 \dots v_m 0$, $m, n \geq 1$, such that u_i is even and v_j is odd for each $i \in [n]$, $j \in [m]$, and $u_1 + \dots + u_n = v_1 + \dots + v_m$. We define the weighted language $r\colon \mathbb{N}^* \to K_{\text{avg}}$ with $\mathrm{supp}(r) = L$; each word in L is mapped to the average value of all even symbols in $u_1 \dots u_n$ which are smaller than 7. The projective, non-homogeneous (COUNT, $\mathbb{N}$, K_{avg})-automaton $\mathcal{A} = (\{e, o, f\}, \Pi, \{e\}, \{f\}, T, \mathrm{wt}, \eta)$ shown in Fig. 1 recognizes r, where $\Pi = \mathrm{BC}(\{\text{even}, \text{odd}, \text{zero}\})$ with the intuitive interpretations.

Lemma 5. *For each $(S, D, \mathbb{B})$-automaton $\mathcal{A}$ there is a homogeneous $(S, D, \mathbb{B})$-automaton $\mathcal{B}$ with $[\![\mathcal{B}]\!] = [\![\mathcal{A}]\!]$.*

Special Cases. (1) Let D be a finite set. Then it is easy to see that each weighted automaton with storage (in the manner of [20]) is a homogeneous (S, D, K)-automaton (for some data storage type S).

(2) Let $K = \mathbb{B}$. Since we can assume each $(S, D, \mathbb{B})$-automaton $\mathcal{A}$ to be homogeneous and, therefore, the weight assignment wt does not depend on its second argument we can presume that the set of transitions of $\mathcal{A}$ consists of those transitions which are mapped to 1. Thus, we can specify an $(S, D, \mathbb{B})$-automaton by a tuple $\mathcal{A} = (Q, \Pi, Q_0, Q_f, T, \eta)$ and define the *language recognized by* $\mathcal{A}$ as the set $L(\mathcal{A}) = \mathrm{supp}([\![\mathcal{A}]\!])$.

(3) Let $S = \text{TRIV}$. Then we drop all references to S from the concepts introduced for (S, D, K)-automata. Thus $T \subseteq Q \times \Pi \times Q$ and we speak about (D, K)-*automata* and (D, K)-*recognizable*. Note that homogeneous (D, K)-automata can be seen as a K-weighted version of symbolic automata.

(4) Let $S = \text{TRIV}$ and $K = \mathbb{B}$. Then we use both conventions mentioned above and speak about D-*automata* and D-*recognizable*. Moreover, we say that

a D-automaton $\mathcal{A} = (Q, \Pi, Q_0, Q_f, T)$ is *deterministic* if $|Q_0| = 1$ and for every two transitions (q, π_1, q_1) and (q, π_2, q_2) in T with $\pi_1 \cap \pi_2 \neq \emptyset$ we have $q_1 = q_2$, and *total* if for each $q \in Q$, $d \in D$ there is a transition $(q, \pi, q') \in T$ with $d \in \pi$.

Closure Properties. In [18, Theorem 1] it was proved that symbolic tree automata can be made total and deterministic. As a special case we easily obtain:

Lemma 6 (cf. [18, Theorem 1]). *For every D-automaton $\mathcal{A}$ there is a total and deterministic D-automaton $\mathcal{B}$ such that $L(\mathcal{A}) = L(\mathcal{B})$.*

By slightly modifying usual constructions we obtain the following two results:

Lemma 7. Let r_1, r_2 be (S, D, K)-recognizable weighted languages and let L_1, L_2 be D-recognizable languages. Then the weighted languages $r_1 + r_2$ and $r_1 \cap L_1$ are (S, D, K)-recognizable. Moreover, $L_1 \setminus L_2$ is D-recognizable (cf. [19]).

Lemma 8. Let D, D' be sets, L a D-recognizable language, and $\rho\colon D \to \mathcal{P}(D')$ a relabeling. Then $\rho(L)$ is D'-recognizable.

4 Data Storage for Symbolic Visibly Pushdown Automata

A *nested set* is a set $N = N_i \cup N_c \cup N_r$, where N_i (*internal symbols*), N_c (*call symbols*), and N_r (*return symbols*) are pairwise disjoint sets. Let $\mathcal{M} = (Q, Q_0, \Gamma, \delta_i, \delta_c, \delta_r, \delta_b, Q_f)$ be a symbolic visibly pushdown automaton (svpda) as defined in [1] (with pushdown alphabet Γ). As explained there, $\mathcal{M}$ uses binary predicates over matching positions. A pair (i, j) of positions of an input word is *matching* if the pushdown cell pushed at i is popped at j. We introduce the data storage type $\mathrm{VP}(N)$ which simulates the pushdown part of an svpda and encodes these binary predicates as parameters of storage instructions.

Let N be a nested set. We define the data storage type $\mathrm{VP}(N) = (C, N, P, F, \varepsilon)$ where $C = (\Lambda \times N_c)^*$ and Λ is an infinite set of pushdown symbols, $P = \{\mathrm{true}\}$, and $F = \{\mathrm{push}_\gamma \mid \gamma \in \Lambda\} \cup \{\mathrm{pop}_{\gamma,\pi} \mid \gamma \in \Lambda, \pi \subseteq N_c \times N_r \text{ decidable}\} \cup \{\mathrm{stay}_i, \mathrm{stay}_r\}$ such that for each $\gamma \in \Lambda$, $\pi \subseteq N_c \times N_r$, $c \in C$, and $d \in N$ we have

- $\mathrm{push}_\gamma(c, d) = (\gamma, d)c$ if $d \in N_c$,
- $\mathrm{pop}_{\gamma,\pi}(c, d) = c'$ if $d \in N_r$, $c = (\gamma, a)c'$ for some $a \in N_c$, and $(a, d) \in \pi$,
- $\mathrm{stay}_i(c, d) = c$ if $d \in N_i$ and $\mathrm{stay}_r(c, d) = c$ if $d \in N_r$ and $c = \varepsilon$,

and undefined otherwise.

Theorem 9. *Let N be a nested set and $L \subseteq N^*$. Then L is recognizable by a symbolic visibly pushdown automaton with decidable label theory (introduced in [1]) if and only if L is projectively $(\mathrm{VP}(N), N, \mathbb{B})$-recognizable.*

Table 1. The svpda $\mathcal{M}$ (left) and the $(\mathrm{VP}(N), N)$-automaton $\mathcal{A}$ (right), both recognizing L.

$\mathcal{M} = (Q, Q_0, \Gamma, \delta_i, \delta_c, \delta_r, \delta_b, Q_f)$	$\mathcal{A} = (Q, \Pi, Q_0, Q_f, T, \eta)$
$Q = Q_0 = Q_f = \{q\}$	$Q = Q_0 = Q_f = \{q\}$
$\Gamma = \{e, o\}$	
$\delta_i = \{(q, \mathsf{T}, q)\}, \delta_b = \{(q, \mathsf{T}, q)\}$	$\{(q, \mathsf{T}, \mathrm{true}, q, \mathrm{stay}_i), (q, \mathsf{T}, \mathrm{true}, q, \mathrm{stay}_r)\} \subseteq T$
$\delta_c = \{(q, \mathrm{even}, q, e), (q, \mathrm{odd}, q, o)\}$	$\{(q, \mathrm{even}, \mathrm{true}, q, \mathrm{push}_e), (q, \mathrm{odd}, \mathrm{true}, q, \mathrm{push}_o)\} \subseteq T$
$\delta_r = \{(q, \sim, e, q), (q, \mathsf{T}, o, q)\}$	$\{(q, \mathsf{T}, \mathrm{true}, q, \mathrm{pop}_{e,\sim}), (q, \mathsf{T}, \mathrm{true}, q, \mathrm{pop}_{o,\mathsf{T}})\} \subseteq T$

Instead of a formal proof we demonstrate our construction by an example. For this let $N_i = \mathbb{N}$, $N_c = \{\langle x \mid x \in \mathbb{N}\}$, and $N_r = \{x\rangle \mid x \in \mathbb{N}\}$. We consider the language $L \subseteq N^*$ which consists of all words w such that for every two symbols $\langle x$ and $y\rangle$ at matching positions of w we have $x = y$ if x is even. For an example consider $w = 3\rangle\langle 2\ 4\ \langle 3\ 5\rangle\ 2\rangle \in L$ with $|w| = 6$ and matching positions $(2, 6)$ and $(4, 5)$. Clearly, L can be recognized by the svpda $\mathcal{M}$ shown in Table 1(left). Note that $\mathcal{M}$ uses a label theory (cf. [1]), which can be seen as the Boolean closure of the unary predicates even and odd with their intuitive interpretations, and of the binary predicate $\sim$ where $(\langle x, y\rangle) \in \sim$ iff $x = y$. Then we construct the projective $(\mathrm{VP}(N), N, \mathbb{B})$-automaton $\mathcal{A}$ as shown in Table 1(right), which uses the label structure $\Pi = \mathrm{BC}(\{\mathrm{even}, \mathrm{odd}\})$. Clearly, $L(\mathcal{A}) = L$.

Now Theorem 9 opens the possibility of considering weighted svpda. For example we can easily construct a $(\mathrm{VP}(N), N, K_{\mathrm{avg}})$-automaton $\mathcal{A}'$ which maps each word in L to the average value of all its even call symbols.

5 Data Storage for Weighted Timed Automata

Our definition of timed words and weighted timed automata closely resembles the one in [14]. The only difference is that in our definition of timed words, each symbol stores the time difference to its predecessor as in [7], while in [14] the corresponding point in time is recorded. Clearly, both views are isomorphic. In the course of this section let Σ be a finite set and K a semiring.

A *timed word (over Σ)* is a non-empty finite sequence $(a_1, t_1) \dots (a_n, t_n) \in (\Sigma \times \mathbb{R}_{\geq 0})^+$. The set of timed words over Σ is denoted by $T\Sigma^+$ and for some semiring K a mapping $r\colon T\Sigma^+ \to K$ is called a *timed series (over Σ and K)*.

A *clock variable* is a variable ranging over $\mathbb{R}_{\geq 0}$ and we denote the set of all clock variables by C. The set of all *clock constraints over C* is denoted by $\Phi(C)$. A *clock valuation* is a function $\nu\colon C \to \mathbb{R}_{\geq 0}$, we let $\nu_0(x) = 0$ for each $x \in C$, and for $t \in \mathbb{R}_{\geq 0}$ and $\lambda \subseteq C$ we let the clock valuations $\nu + t$ (where t is added to each clock) and $\nu[\lambda := 0]$ (where all clocks in λ are set to 0) be defined as in [14]. Moreover, the *satisfaction relation* $\models \subseteq {\mathbb{R}_{\geq 0}}^C \times \Phi(C)$ is defined as usual.

A *K-weighted timed automaton* over Σ (and C) is a tuple $\mathcal{A} = (Q, Q_i, Q_f, C, E, \mathrm{ewt}, \mathrm{dwt})$, where Q is a finite set (*states*), $Q_i \subseteq Q$ (*initial states*) and $Q_f \subseteq Q$ (*final states*), C is a finite set of clock variables,

$E \subseteq Q \times \Sigma \times \Phi(\mathcal{C}) \times 2^{\mathcal{C}} \times Q$ is a finite set (*edges*), ewt: $E \to K$ is a function (*edge weights*), and dwt: $Q \times \mathbb{R}_{\geq 0} \to K$ is a function (*delay weights*). A *run* of $\mathcal{A}$ is a finite sequence

$$\rho = (q_0, \nu_0) \xrightarrow{t_1} \xrightarrow{e_1} \cdots \xrightarrow{t_n} \xrightarrow{e_n} (q_n, \nu_n)$$

where $n \geq 1$, $q_0, \ldots, q_n \in Q$, ν_i are clock valuations, $t_i \in \mathbb{R}_{\geq 0}$, and $e_i \in E$ satisfying the following conditions: $q_0 \in Q_i$, $q_n \in Q_f$, and $e_i = (q_{i-1}, a_i, \phi_i, \lambda_i, q_i)$ such that $\nu_{i-1} + t_i \models \phi_i$ and $\nu_i = (\nu_{i-1} + t_i)[\lambda_i := 0]$. The *label* of ρ is the timed word $\mathrm{label}(\rho) = ((e_1)_2, t_1) \ldots ((e_n)_2, t_n)$, and the *running weight* $\mathrm{rwt}(\rho)$ of ρ is given by $\mathrm{rwt}(\rho) = \prod_{i \in [n]} \mathrm{dwt}(q_{i-1}, t_i) \cdot \mathrm{ewt}(e_i)$. For any timed word $w \in T\Sigma^+$ let $\mathrm{Run}_{\mathcal{A}}(w)$ denote the set of all runs ρ of $\mathcal{A}$ with $\mathrm{label}(\rho) = w$. The *timed series recognized by* $\mathcal{A}$ is the mapping $[\![\mathcal{A}]\!]: T\Sigma^+ \to K$ such that $[\![\mathcal{A}]\!](w) = \sum_{\rho \in \mathrm{Run}_{\mathcal{A}}(w)} \mathrm{rwt}(\rho)$.

Now we define a data storage type TIME($\mathcal{C}$) to simulate the clock behavior of weighted timed automata. Let $\mathcal{C}$ be a finite set of clock variables and let $\mathrm{TIME}(\mathcal{C}) = (\mathbb{R}_{\geq 0}{}^{\mathcal{C}}, \mathbb{R}_{\geq 0}, P, F, \nu_0)$ where $P = \{p_\phi \mid \phi \in \Phi(\mathcal{C})\}$, $F = \{f_\lambda \mid \lambda \subseteq \mathcal{C}\}$, and for every $\phi \in \Phi(\mathcal{C})$, $\lambda \subseteq \mathcal{C}$, $\nu \in \mathbb{R}_{\geq 0}{}^{\mathcal{C}}$, and $t \in \mathbb{R}_{\geq 0}$ we let

- $p_\phi(\nu, t) = \mathrm{true}$ iff $(\nu + t) \models \phi$, and
- $f_\lambda(\nu, t) = (\nu + t)[\lambda := 0]$.

Theorem 10. *Let K be a semiring and $r: T\Sigma^+ \to K$ a timed series. Then r is recognized by a K-weighted timed automaton over Σ and $\mathcal{C}$ if and only if r is projectively* $(\mathrm{TIME}(\mathcal{C}), \Sigma \times \mathbb{R}_{\geq 0}, K)$*-recognizable.*

The formal proof uses the following ideas. In "$\Rightarrow$" each transition results from a given edge, and the weight amounts to the product of ewt and dwt; the resulting $(\mathrm{TIME}(\mathcal{C}), \Sigma \times \mathbb{R}_{\geq 0}, K)$-automaton is (in general) not homogeneous. In "$\Leftarrow$" we code the transitions and the finite part of the input symbols into states, split up each transition, and then simulate the weight assignment with dwt.

6 Weighted Symbolic MSO-Logic with Storage Behavior

Our new logic is based on the concepts of M-expression [10, Definition 3.1] and B-expression [20, Definition 5]. Since these expressions depend on unweighted MSO formulas, we first extend unweighted MSO-logic to symbolic MSO-logic.

Symbolic MSO-Logic. As usual, we use first-order variables, like x, y, and second-order variables, like X, Y. Furthermore, we introduce one more variable B which we call *second-order behavior variable* and which ranges over behaviors of S.

Let D be a set, Π a label structure over D, and Ω a finite subset of $P \times F$. We define the set of *formulas of symbolic MSO-logic over Ω and Π*, denoted by $\mathrm{MSO}(\Omega, \Pi)$, by the following EBNF:

$$\psi ::= P_\pi(x) \mid \mathrm{next}(x, y) \mid x \in X \mid B(x) = (p, f)$$
$$\varphi ::= \psi \mid \neg\varphi \mid \varphi \wedge \varphi \mid \exists x.\varphi \mid \exists X.\varphi$$

where $\pi \in \Pi$ and $(p, f) \in \Omega$. Let $\varphi \in \mathrm{MSO}(\Omega, \Pi)$. The set of *free variables of* φ and *bound variables of* φ, denoted by $\mathrm{Free}(\varphi)$ and $\mathrm{Bound}(\varphi)$, resp., is defined as usual. In particular, we set $\mathrm{Free}(P_\pi(x)) = \{x\}$ and $\mathrm{Free}(B(x) = (p, f)) = \{x, B\}$.

Let $\mathcal{V}$ be a finite set of variables with $B \in \mathcal{V}$, let $\eta\colon D \to M$ be a relabeling, and let $w \in D^*$. A $(\mathcal{V}, \eta)$-*assignment for* w is a function with domain $\mathcal{V}$ which maps each first-order variable in $\mathcal{V}$ to an element of $\mathrm{pos}(w)$, each second-order variable in $\mathcal{V}$ to a subset of $\mathrm{pos}(w)$, and B to an $\eta(w)$-behavior over Ω. We let $\Phi_{(\mathcal{V},\eta),w}$ denote the set of all $(\mathcal{V}, \eta)$-assignments for w. In the usual way we define updates of $(\mathcal{V}, \eta)$-assignments. Let $\sigma \in \Phi_{(\mathcal{V},\eta),w}$ and $i \in \mathrm{pos}(w)$. By $\sigma[x \mapsto i]$ we denote the $(\mathcal{V} \cup \{x\}, \eta)$-assignment for w that agrees with σ on $\mathcal{V} \backslash \{x\}$ and that satisfies $\sigma[x \mapsto i](x) = i$. Similarly, we define the updates $\sigma[X \mapsto I]$ and $\sigma[B \mapsto b]$ for each set $I \subseteq \mathrm{pos}(w)$ and each behavior $b \in \mathrm{B}(\Omega, \eta(w))$, respectively.

Extending the usual technique we encode a pair (w, σ), where $w \in D^*$ and $\sigma \in \Phi_{(\mathcal{V},\eta),w}$, as a word over an extended set as follows. For each finite set $\mathcal{V}$ of variables with $B \in \mathcal{V}$ we let

$$D_{\mathcal{V}} = D \times \mathcal{P}(\mathrm{fo}(\mathcal{V}) \cup \mathrm{so}(\mathcal{V})) \times \Omega$$

where $\mathrm{fo}(\mathcal{V})$ and $\mathrm{so}(\mathcal{V})$ are the subsets of all first-order and second-order variables occurring in $\mathcal{V}$, respectively. Let $\zeta = \zeta_1 \dots \zeta_n \in D_{\mathcal{V}}^*$. We call ζ fo-*valid* if for each $x \in \mathrm{fo}(\mathcal{V})$ there is a unique $i \in \mathrm{pos}(\zeta)$ such that x occurs in the second component of ζ_i. We denote the set of all fo-valid words over $D_{\mathcal{V}}$ by $D_{\mathcal{V}}^{*\mathrm{fo}}$. Moreover, we call ζ η-*valid* if the word $(\zeta_1)_3 \dots (\zeta_n)_3$ is an $\eta((\zeta_1)_1 \dots (\zeta_n)_1)$-behavior over Ω. We denote the set of all η-valid words over $D_{\mathcal{V}}$ by $D_{\mathcal{V}}^{*\eta}$.

It is clear that, for each finite set $\mathcal{V}$ of variables with $B \in \mathcal{V}$ and relabeling $\eta\colon D \to M$, there is a one-to-one correspondence between the set $\{(w, \sigma) \mid w \in D^*, \sigma \in \Phi_{(\mathcal{V},\eta),w}\}$ and the set $D_{\mathcal{V}}^{*\mathrm{fo}} \cap D_{\mathcal{V}}^{*\eta}$. Thus, as usual, we will not distinguish between the pair (w, σ) and the corresponding word $\zeta \in D_{\mathcal{V}}^{*\mathrm{fo}} \cap D_{\mathcal{V}}^{*\eta}$.

Lemma 11. Let D be a set and let $\mathcal{V}$ be a finite set of variables with $B \in \mathcal{V}$. Then $D_{\mathcal{V}}^{*\mathrm{fo}}$ is $D_{\mathcal{V}}$-recognizable.

Let $\varphi \in \mathrm{MSO}(\Omega, \Pi)$ and $\mathcal{V}$ be a finite set of variables such that $\mathrm{Free}(\varphi) \subseteq \mathcal{V}$ and $B \in \mathcal{V}$. Moreover, let $\eta\colon D \to M$ be a relabeling. For every $(w, \sigma) \in D_{\mathcal{V}}^{*\mathrm{fo}} \cap D_{\mathcal{V}}^{*\eta}$ we define the relation $(w, \sigma) \models \varphi$ by extending the usual models operator of classical MSO-logic as shown for atoms in Fig. 2. Then we define the set of models of φ as the set

$$L_{\mathcal{V},\eta}(\varphi) = \{(w, \sigma) \mid w \in D^*,\ \sigma \in \Phi_{(\mathcal{V},\eta),w}, (w, \sigma) \models \varphi\}.$$

Thus, $L_{\mathcal{V},\eta}(\varphi) \subseteq D_{\mathcal{V}}^{*\mathrm{fo}} \cap D_{\mathcal{V}}^{*\eta}$.

Lemma 12. Let D be a set and Π a label structure over D, let Ω be a finite subset of $P \times F$, and let $\eta\colon D \to M$ be a relabeling. For each $\varphi \in \mathrm{MSO}(\Omega, \Pi)$ and each finite set $\mathcal{V} \supseteq \mathrm{Free}(\varphi)$ of variables with $\mathcal{V} \cap \mathrm{Bound}(\varphi) = \emptyset$ and $B \in \mathcal{V}$ there is a $D_{\mathcal{V}}$-recognizable language L such that $L_{\mathcal{V},\eta}(\varphi) = L \cap D_{\mathcal{V}}^{*\eta}$.

$$
\begin{aligned}
(w,\sigma) \models P_\pi(x) \text{ is true} &\iff w_{\sigma(x)} \in \pi \\
(w,\sigma) \models \mathrm{next}(x,y) \text{ is true} &\iff \sigma(x)+1 = \sigma(y) \\
(w,\sigma) \models (x \in X) \text{ is true} &\iff \sigma(x) \in \sigma(X) \\
(w,\sigma) \models (B(x) = (p,f)) \text{ is true} &\iff \sigma(B)_{\sigma(x)} = (p,f).
\end{aligned}
$$

Fig. 2. Models operator for atoms.

Weighted Symbolic MSO-Logic. Here we introduce our new weighted MSO-logic over data storage types. This logic extends the one in [20, Definition 5] from a finite set Σ to an arbitrary set D.

Let D be a set, Π a label structure over D, and Ω a finite subset of $P \times F$. We define the set $\mathrm{BExp}(\Omega, \Pi, K)$ of *B-expressions over* (Ω, Π, K) to be the set generated by the EBNF:

$$e ::= \mathrm{Val}_\kappa \mid (e + e) \mid (\varphi \rhd e) \mid \sum\nolimits_x e \mid \sum\nolimits_X e,$$

where $\kappa\colon D_\mathcal{U} \to K$ is a relabeling for some finite set $\mathcal{U}$ of variables with $B \in \mathcal{U}$, and $\varphi \in \mathrm{MSO}(\Omega, \Pi)$. As in the unweighted case the sets $\mathrm{Free}(e)$ and $\mathrm{Bound}(e)$ for each B-expression e are defined as usual where we set $\mathrm{Free}(\mathrm{Val}_\kappa) = \mathcal{U}$.

We define the set $\mathrm{Exp}(\Omega, \Pi, K)$ of *MSO-expressions over* (Ω, Π, K) as the set of all expressions of the form $\sum_B^\eta e$ with $e \in \mathrm{BExp}(\Omega, \Pi, K)$, $\mathrm{Free}(e) = \{B\}$, and relabeling $\eta\colon D \to M$. An *MSO-expression over* (S, D, K) is an MSO-expression over (Ω, Π, K) for some finite $\Omega \subseteq P \times F$ and label structure Π over D.

$$
\begin{aligned}
[\![\mathrm{Val}_\kappa]\!]\nu,\eta(\zeta) &= \mathrm{val}(\kappa(\zeta_\mathcal{U})) \text{ where } \zeta_\mathcal{U} \text{ is obtained from } \zeta \text{ by replacing each} \\
&\qquad \text{symbol } (a, V, \omega) \text{ by } (a, V \cap (\mathrm{fo}(\mathcal{U}) \cup \mathrm{so}(\mathcal{U})), \omega) \\
[\![e_1 + e_2]\!]\nu,\eta(\zeta) &= [\![e_1]\!]\nu,\eta(\zeta) + [\![e_2]\!]\nu,\eta(\zeta) \\
[\![\varphi \rhd e]\!]\nu,\eta(\zeta) &= [\![e]\!]\nu,\eta(\zeta), \text{if } \zeta \in \mathcal{L}\nu,\eta(\varphi), \text{ and 0 otherwise} \\
[\![\textstyle\sum_x e]\!]\nu,\eta(\zeta) &= \sum_{i \in \mathrm{pos}(\zeta)} [\![e]\!]\nu\cup\{x\},\eta(w, \sigma[x \mapsto i]) \\
[\![\textstyle\sum_X e]\!]\nu,\eta(\zeta) &= \sum_{I \subseteq \mathrm{pos}(\zeta)} [\![e]\!]\nu\cup\{X\},\eta(w, \sigma[X \mapsto I])
\end{aligned}
$$

Fig. 3. Semantics of B-expressions (also cf. [20, Definition 6]).

Let $e \in \mathrm{BExp}(\Omega, \Pi, K)$, $\mathcal{V}$ be a finite set of variables containing $\mathrm{Free}(e)$, and $\eta\colon D \to M$ be a relabeling. The *semantics of e with respect to $\mathcal{V}$* and η is the weighted language $[\![e]\!]\nu,\eta\colon D_\mathcal{V}^* \to K$ such that $\mathrm{supp}([\![e]\!]\nu,\eta) \subseteq D_\mathcal{V}^{*\mathrm{fo}} \cap D_\mathcal{V}^{*\eta}$ and for each $\zeta = (w, \sigma) \in D_\mathcal{V}^{*\mathrm{fo}} \cap D_\mathcal{V}^{*\eta}$ we define $[\![e]\!]\nu,\eta(\zeta)$ inductively as shown in Fig. 3. Let $e = \sum_B^\eta e'$ be an MSO-expression over (Ω, Π, K). We define the weighted language $[\![e]\!]\colon D^* \to K$ for each $w \in D^*$ by:

$$[\![\textstyle\sum_B^\eta e']\!](w) = \sum_{b \in \mathrm{B}(\Omega, \eta(w))} [\![e']\!]_{\{B\},\eta}(w, [B \mapsto b]).$$

We say that a weighted language $r\colon D^* \to K$ is *definable by an MSO-expression over* (S, D, K) if there is an MSO-expression e over (S, D, K) such that $r = [\![e]\!]$.

From Automata to Logic. The proof of the claim that recognizability implies definability follows the standard construction idea and is exactly the same as the proof of Lemma 9 of [20] where D is a finite set (there denoted by Σ), except that (1) the atomic formula $P_a(x)$ (for $a \in \Sigma$) in ψ_1 has to be replaced by $P_\pi(x)$ and (2) $\kappa((d, V, \omega)) = \mathrm{wt}(\tau, d)$ if $V = \{X_\tau\}$ and 0 otherwise, where wt is the weight function of the given automaton.

Theorem 13. *Let $r\colon D^* \to K$. If r is (S, D, K)-recognizable, then r is definable by some MSO-expression over (S, D, K).*

From Logic to Automata. We can prove the following lemma by induction on the structure of the B-expression e. The proof of the cases are easy generalizations of Lemmas 11–14 of [20].

Lemma 14. Let $e \in \mathrm{BExp}(\Omega, \Pi, K)$ and $\mathcal{V} \supseteq \mathrm{Free}(e)$ a finite set of variables with $\mathcal{V} \cap \mathrm{Bound}(\varphi) = \emptyset$ and $B \in \mathcal{V}$. Moreover, let $\eta\colon D \to M$ be a relabeling. There is a $(D_\mathcal{V}, K)$-recognizable weighted language r such that $[\![e]\!]_{\mathcal{V},\eta} = r \cap D_\mathcal{V}^{*\eta}$.

Due to the symbolicness of the automata, the next lemma is slightly more complicated to prove than the corresponding Lemma 15 of [20].

Lemma 15. Let $e \in \mathrm{BExp}(\Omega, \Pi, K)$ with $\mathrm{Free}(e) = \{B\}$ and $\eta\colon D \to M$ be a relabeling. If $[\![e]\!]_{\{B\},\eta} = r \cap D_{\{B\}}^{*\eta}$ for some $(D_{\{B\}}, K)$-recognizable weighted language r, then $[\![\sum_B^\eta e]\!]$ is an (S, D, K)-recognizable weighted language.

Proof. Let $\mathcal{A} = (Q, \Pi, Q_0, Q_f, T, \mathrm{wt})$ be a $(D_{\{B\}}, K)$-automaton such that $[\![e]\!]_{\{B\},\eta} = [\![\mathcal{A}]\!] \cap D_{\{B\}}^{*\eta}$. We will construct the (S, D, K)-automaton $\mathcal{A}' = (Q', \mathrm{BC}(\Pi'), Q_0', Q_f, T', \mathrm{wt}', \eta)$ such that $[\![\mathcal{A}']\!] = [\![\sum_B^\eta e]\!]$, using the following idea. Since each predicate $\pi \subseteq D \times \{\emptyset\} \times \Omega$ occurring in T combines elements from D and Ω, we have to keep these combinations also in $\mathcal{A}'$. For this, we partition Π into a family $\Pi \times \Omega$ and keep track of elements of Ω in the state set of $\mathcal{A}'$.

Formally, we let $Q' = (Q \times \Omega) \cup Q_f$ and $Q_0' = Q_0 \times \Omega$. We let $\Pi' = \Pi \times \Omega$ such that for every $(\pi, (p, f)) \in \Pi'$ we have $(\pi, (p, f)) = \{d \mid (d, \emptyset, (p, f)) \in \pi\}$. If $\tau = (q, \pi, q')$ is in T, then for every $(p, f), (p', f') \in \Omega$ we let $\tau' = ((q, (p, f)), (\pi, (p, f)), p, (q', (p', f')), f)$ be in T'. Moreover, if $q' \in Q_f$, then also $\tau' = ((q, (p, f)), (\pi, (p, f)), p, q', f)$ is in T'. For every $\tau' \in T'$ and $d \in D$ we define $\mathrm{wt}'(\tau', d) = \mathrm{wt}(\tau, (d, \emptyset, (p, f)))$ if $(\tau')_2 = (\pi, (p, f))$ and $d \in (\pi, (p, f))$, and $\mathrm{wt}'(\tau', d) = 0$ otherwise (where τ is the transition from which τ' was constructed). It is not difficult to prove that $[\![\mathcal{A}']\!] = [\![\sum_B^\eta e]\!]$. □

Using Lemma 14 for $\mathcal{V} = \{B\}$ and Lemma 15 we obtain the following theorem.

Theorem 16. *Let $r\colon D^* \to K$. If r is definable by some MSO-expression over (S, D, K), then r is (S, D, K)-recognizable.*

Decidability Result. Based on the method introduced by Kirsten in [12] and the *zero generation problem* (ZGP), we can prove the following theorem (also cf. [20, Theorem 17]), where a strong bimonoid $(K, +, \cdot, 0, 1)$ is *commutative* if $\cdot$ is so.

Theorem 17 (cf. [12, Theorem 1]). *Let K be a zero-sum-free commutative strong bimonoid. Then, for each (S, D, K)-recognizable weighted language r, the support* $\mathrm{supp}(r)$ *is $(S, D, \mathbb{B})$-recognizable. Moreover, if $(K, \cdot, 1)$ has a decidable ZGP, then there is an effective construction of an $(S, D, \mathbb{B})$-automaton recognizing* $\mathrm{supp}([\![\mathcal{A}]\!])$ *from any given (S, D, K)-automaton $\mathcal{A}$.*

In particular, each bounded lattice satisfies the conditions of Theorem 17. An MSO-expression e over (S, D, K) is *satisfiable* if $\mathrm{supp}(r) \neq \emptyset$.

Corollary 18. *Let N be a nested set, K a zero-sum-free commutative strong bimonoid with a decidable ZGP, and r a projectively* $(\mathrm{VP}(N), N, K)$*-recognizable weighted language. (1) It is decidable whether* $\mathrm{supp}(r) = \emptyset$*. (2) The satisfiability problem of each MSO-expression over* $(\mathrm{VP}(N), N, K)$ *is decidable.*

Proof. (1) follows from Theorem 17, Theorem 9, and the decidability of the emptiness problem of symbolic visibly pushdown automata [1, Theorem 4]. (2) follows from Theorem 16 and (1).

References

1. D'Antoni, L., Alur, R.: Symbolic Visibly Pushdown Automata. In: Biere, A., Bloem, R. (eds.) CAV 2014. LNCS, vol. 8559, pp. 209–225. Springer, Heidelberg (2014)
2. Droste, M., Gastin, P.: Weighted Automata and Weighted Logics. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 513–525. Springer, Heidelberg (2005)
3. Droste, M., Kuich, W., Vogler, H. (eds.): Handbook of Weighted Automata. EATCS Monographs in Theoretical Computer Science. Springer, Heidelberg (2009)
4. Droste, M., Meinecke, I.: Describing Average- and Longtime-Behavior by Weighted MSO Logics. In: Hliněný, P., Kučera, A. (eds.) MFCS 2010. LNCS, vol. 6281, pp. 537–548. Springer, Heidelberg (2010)
5. Droste, M., Stüber, T., Vogler, H.: Weighted finite automata over strong bimonoids. Inf. Sci. **180**(1), 156–166 (2010)
6. Droste, M., Vogler, H.: The Chomsky-Schützenberger Theorem for Quantitative Context-Free Languages. In: Béal, M.-P., Carton, O. (eds.) DLT 2013. LNCS, vol. 7907, pp. 203–214. Springer, Heidelberg (2013)
7. Droste, M., Perevoshchikov, V.: A Nivat Theorem for Weighted Timed Automata and Weighted Relative Distance Logic. In: Esparza, J., Fraigniaud, P., Husfeldt, T., Koutsoupias, E. (eds.) ICALP 2014, Part II. LNCS, vol. 8573, pp. 171–182. Springer, Heidelberg (2014)
8. Eilenberg, S.: Automata, Languages, and Machines. Pure and Applied Mathematics, vol. 59. Academic Press, New York (1974)
9. Engelfriet, J.: Context-free grammars with storage. Technical report 86–11, University of Leiden (1986), see also: arXiv:1408.0683 [cs.FL] (2014)

10. Fülöp, Z., Stüber, T., Vogler, H.: A Büchi-like theorem for weighted tree automata over multioperator monoids. Theor. Comput. Syst. **50**(2), 241–278 (2012)
11. Herrmann, L., Vogler, H.: A Chomsky-Schützenberger theorem for weighted automata with storage. In: Maletti, A. (ed.) CAI 2015. LNCS, vol. 9270, pp. 115–127. Springer, Switzerland (2015)
12. Kirsten, D.: The support of a recognizable series over a zero-sum free, commutative semiring is recognizable. Acta Cybern. **20**(2), 211–221 (2011)
13. Kuich, W., Salomaa, A.: Semirings, Automata, Languages. EATCS Monographs on Theoretical Computer Science, vol. 5. Springer, Heidelberg (1986)
14. Quaas, K.: MSO logics for weighted timed automata. Form. Methods Syst. Des. **38**(3), 193–222 (2011)
15. Sakarovitch, J.: Elements of Automata Theory. Cambridge University Press, Cambridge (2009)
16. Scott, D.: Some definitional suggestions for automata theory. J. Comput. Syst. Sci. **1**, 187–212 (1967)
17. Veanes, M.: Applications of Symbolic Finite Automata. In: Konstantinidis, S. (ed.) CIAA 2013. LNCS, vol. 7982, pp. 16–23. Springer, Heidelberg (2013)
18. Veanes, M., Bjørner, N.: Symbolic tree automata. Inf. Process. Lett. **115**(3), 418–424 (2015)
19. Veanes, M., Bjørner, N., de Moura, L.: Symbolic Automata Constraint Solving. In: Fermüller, C.G., Voronkov, A. (eds.) LPAR-17. LNCS, vol. 6397, pp. 640–654. Springer, Heidelberg (2010)
20. Vogler, H., Droste, M., Herrmann, L.: A weighted MSO logic with storage behaviour and its Büchi-Elgot-Trakhtenbrot theorem. In: Dediu, A.H., Janoušek, J., Martín-Vide, C., Truthe, B. (eds.) LATA 2016. LNCS, vol. 9618, pp. 127–139. Springer, Switzerland (2016)