Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Weighted register automata and weighted logic on data words[☆]

Parvaneh Babari, Manfred Droste[*], Vitaly Perevoshchikov[*]

Universität Leipzig, Institut für Informatik, 04109 Leipzig, Germany

A R T I C L E I N F O

Article history:
Received 10 April 2017
Received in revised form 2 January 2018
Accepted 8 January 2018
Available online xxxx

Keywords:
Weighted register automata
Data languages
Data series
Weighted MSO logic
Determinizable class of register automata

A B S T R A C T

Data words are sequences of pairs where the first element is taken from a finite alphabet and the second element is taken from an infinite data domain. Register automata provide a widely studied model for reasoning on data words. In this paper, we investigate automata models for quantitative aspects of systems with infinite data domains, e.g., the costs of storing data on a remote server or the consumption of resources (e.g., memory, energy, time) during a data analysis. We introduce weighted register automata on data words over commutative data semirings equipped with a collection of binary data functions, and we investigate their closure properties. Unlike the other models considered in the literature, we allow data comparison by means of an arbitrary collection of binary data relations. This enables us to incorporate timed automata and weighted timed automata into our framework. In our main result, we give a logical characterization of weighted register automata by means of weighted existential monadic second-order logic; for the proof we employ a new class of determinizable visibly register automata.

1. Introduction

In the areas of static analysis of databases and software verification, there is much interest in processing information taken from an infinite domain. The notion of data words is a well-known concept for the modelling of these situations. A data word can be considered as a sequence of pairs where the first element is taken from a finite alphabet (as in classical words) and the second element is taken from an infinite data domain. Register automata introduced by Kaminski and Francez [27] provide a widely studied model for reasoning on data words. These automata can be considered as classical nondeterministic finite automata equipped with a finite set of registers which are used to store data in order to compare it with some data in the future. This enables them to handle parameters like user names, passwords, identifiers of connections, etc., in a fashion similar to, and slightly more expressive than, the class of *data-independent* systems. This model served as a basis for the study of various automata models and logics on data words and trees [8,11,17,25,26,28]. Classical *timed automata* of Alur and Dill [2] for real-time systems are another example of automata on data (timed) words.

For many Computer Science applications, quantitative properties of systems such as costs, probabilities, vagueness and uncertainty of a statement, consumption of memory and energy are of significant importance. Weighted automata (cf. [19] for surveys) form a well-known model for quantitative aspects which has been extended to various different settings (cf.,

☆ Work supported by DFG Graduiertenkolleg 1763 (QuantLA).

* Corresponding authors.
E-mail addresses: babari@informatik.uni-leipzig.de (P. Babari), droste@informatik.uni-leipzig.de (M. Droste), perev@informatik.uni-leipzig.de (V. Perevoshchikov).

https://doi.org/10.1016/j.tcs.2018.01.004
0304-3975/© 2018 Elsevier B.V. All rights reserved.

e.g., weighted timed automata [5,30] for the quantitative analysis of real-time systems). In this paper, we introduce *weighted register automata* for quantitative reasoning on data words. Motivated by the seminal Büchi–Elgot–Trakhtenbrot theorem [15] about the expressive equivalence of finite automata and monadic second-order (MSO) logic and by the weighted MSO logic of Droste and Gastin [18], we introduce *weighted MSO logic on data words* and give a logical characterization of weighted register automata.

To the best of our knowledge, quantitative extensions of register automata have not been studied yet. However, there is a plenty of quantitative models which are quite natural in the context of register automata. We give several examples. For instance, since register automata process data taken from an infinite domain, there arises a natural question about the efficiency of the processing of big data which can invoke big costs for data storing and data comparison. As another example, one may ask about the maximal size of data which will be stored in registers along a computation of a register automaton. Thus, our goal is to develop a model which will reflect this kind of data-dependent costs. Now we give a summary of our results.

We introduce weighted register automata over *commutative data semirings* equipped with a collection of binary data functions in the spirit of the classical theory of weighted automata [19]. Whereas in the models of register automata known from the literature data are usually compared with respect to equality or a linear order, here we allow data comparison by means of an arbitrary collection of binary data relations. This approach permits easily to incorporate timed automata [2] and weighted timed automata [5,30] into our framework. Moreover, this approach gives rise to the further investigations of the models for data processing, e.g., data comparison with an approximation error.

We introduce semiring-weighted existential MSO logic on data words equipped with binary data functions. Recall that weighted MSO logic on classical words is introduced and investigated in [18]. In order to model data comparison, we use the data predicates in the spirit of relative distance predicates of Wilke [31]. Our goal is to prove the expressive equivalence of this new logic with weighted register automata. To reach this goal, we faced the following difficulties.

- The unrestricted use of binary data functions and weighted universal quantifiers goes beyond recognizability even for very simple formulas.
- Register automata are neither determinizable nor closed under complement.

In order to overcome these problems, we obtain the suitable fragment of our weighted existential MSO logic by restricting the use of the weighted universal quantifier to formulas without weighted quantifiers and by restricting the use of data functions to an intuitively defined logical operator. In our main result, we state that this restricted weighted EMSO-logic is equivalent to weighted register automata. The existing proof techniques and ideas for weighted logic cannot be applied in our setting, since register automata are not closed under complement and we need a new construction to deal with binary data functions. For this purpose, we introduce a determinizable class of unweighted register automata, called visibly register automata. This determinizable class of register automata could be also of independent interest. Moreover, we introduce a new normalization technique for the binary data functions, in order to provide a translation of formulas with weighted universal quantifiers into weighted register automata. With this, we achieve our goal; moreover, our construction of weighted register automata equivalent to a given weighted MSO formula is effective.

In the unweighted setting, an interesting tight relationship between the two models of register and timed automata was established in [26]. In the model of register automata of [26], data are compared with respect to equality and inequality (whereas we use a collection of binary relations). The translations from timed automata to register automata and conversely are quite involved. In [14], timed languages have been reduced to data languages by a transformation from timed automata to data automata. Note that in this transformation the number of registers in the constructed data automaton is considerably larger than the number of clocks of the original timed automaton. Conversely, this data automaton device allows the recognition of a much larger class of languages. Surprisingly, in our model this transformation is obtained in a much easier way, and it enables us to extend this relation also to the weighted setting.

For future research, it could be interesting to extend our results to the setting of infinite data words and data trees and to investigate in the setting of data words the cases where the weight measure cannot be modelled using semirings (e.g., average or discounted costs, energy problems and weighted register automata with multiple cost parameters). Note that these nonclassical weight measures have been extensively studied in the setting of weighted timed automata. It could be also interesting to compare the expressive power of our register automata model with the data automata model of [11]. We believe that they are incomparable. An extension of class register automata and the logic captured by them [9], where data words have been considered as behavioral models of concurrent systems, to the weighted setting could be attractive, as well.

A preliminary version of the results of this article appeared in [6]. In contrast to the preliminary version, in this article we discuss in more detail how we incorporate weighted register and timed automata into one framework. In addition, we give full proofs of the closure properties for the class of recognizable data series. We also provide the reader with all the proofs and constructions related to the visibly register automata.

The rest of this article is outlined as follows. In Section 2, we introduce (unweighted) register automata over arbitrary data structures and provide several examples. In Section 3, we introduce a data structure for register automata which permits to simulate the behavior of timed automata. In Section 4, we lift our model of register automata to the weighted setting. In Section 5, we compare weighted timed automata with weighted register automata. In Section 6, we investigate closure properties for weighted register automata. In Section 7, we introduce weighted MSO logic for data words. In Section 8, we give several counterexamples showing that unrestricted weighted MSO logic cannot be captured by weighted register automata. Here we also introduce restricted weighted MSO logic and state our main result. In Section 9, for the proof of our main result, we introduce (unweighted) visibly register automata and study their closure properties and determinization. In Section 10, we prove our main result. In Section 11, we summarize the results of this article and discuss several open problems.

2. Register automata: definition and examples

Register automata are nondeterministic finite state automata on data words equipped with a finite set of registers for storing data (from an infinite data domain) and comparing new data instances with the already stored data. Whereas in the original definition of register automata [27] it is only possible to check the equality of data, the later models augment a data domain with a linear order and allow to compare data with respect to this linear order. However, more complicated situations of data comparison are reasonable as well (cf., e.g., timed automata [2] or Example 2.2). In this paper, we consider the model of register automata where we augment a data domain with some arbitrary binary relations which will be used for data comparison. Notice that our register automata also incorporate timed automata [2].

For a set X, let $\mathcal{P}(X) = \{Y \mid Y \subseteq X\}$, the *power set* of X. A *data structure* is a pair $\mathbb{D} = \langle D, \mathcal{R}\rangle$ where D is an arbitrary set called a *data domain* and $\mathcal{R}$ is a set of binary relations on D called *data relations*. For the convenience of presentation, we assume throughout all of this paper that D contains a designated initial data value $\perp \in D$ which will be the initial data value of all registers.

An *alphabet* is a non-empty finite set. Let Σ be an alphabet and $\mathbb{D} = \langle D, \mathcal{R}\rangle$ a data structure. A *data word* w over Σ and $\mathbb{D}$ is a finite sequence of pairs, denoted by $w = (a_1, d_1)...(a_n, d_n)$, where $n \geq 1$, $a_1, ..., a_n \in \Sigma$ and $d_1, ..., d_n \in D$. Let $\mathbb{D}\Sigma^+ = (\Sigma \times D)^+$ denote the set of all data words over Σ and $\mathbb{D}$. Note that we exclude the empty data word from consideration. Any subset $\mathcal{L} \subseteq \mathbb{D}\Sigma^+$ is called a *data language*.

Let Reg be a finite set of *registers* which take values from the data domain D of the data structure $\mathbb{D}$. A *register valuation* over Reg and $\mathbb{D}$ is any mapping $\vartheta : \mathsf{Reg} \to D$. Let $\mathsf{Val}(\mathsf{Reg}, \mathbb{D})$ denote the collection of all register valuations over Reg and $\mathbb{D}$. For a register valuation $\vartheta \in \mathsf{Val}(\mathsf{Reg}, \mathbb{D})$, a subset $\Lambda \subseteq \mathsf{Reg}$ and a data value $d \in D$, the *update* $\vartheta[\Lambda := d]$ is the register valuation in $\mathsf{Val}(\mathsf{Reg}, \mathbb{D})$ defined for all registers r by $\vartheta[\Lambda := d](r) = d$ if $r \in \Lambda$ and $\vartheta[\Lambda := d](r) = \vartheta(r)$ otherwise.

The set $\mathsf{Guard}(\mathsf{Reg}, \mathbb{D})$ of *register guards* over Reg and $\mathbb{D}$ is defined by the grammar

$$\phi \ ::= \mathsf{True} \mid Rr \mid \phi \wedge \phi \mid \neg\phi$$

where $r \in \mathsf{Reg}$ and R is a data relation in D. Given a register valuation $\vartheta \in \mathsf{Val}(\mathsf{Reg}, \mathbb{D})$, a data value $d \in D$ and a register guard $\phi \in \mathsf{Guard}(\mathsf{Reg}, \mathbb{D})$, the *satisfaction relation* $(\vartheta, d) \models \phi$ is defined inductively on the structure of ϕ as follows: $(\vartheta, d) \models \mathsf{True}$ always holds; $(\vartheta, d) \models Rr$ iff $(\vartheta(r), d) \in R$; $(\vartheta, d) \models \phi_1 \wedge \phi_2$ iff $(\vartheta, d) \models \phi_1$ and $(\vartheta, d) \models \phi_2$; $(\vartheta, d) \models \neg\phi$ iff $(\vartheta, d) \models \phi$ does not hold.

Definition 2.1. Let Σ be an alphabet and $\mathbb{D}$ a data structure. A *register automaton* over Σ and $\mathbb{D}$ is a tuple $\mathcal{A} = (Q, \mathsf{Reg}, I, T, F)$ where Q is a finite set of *states*, Reg is a finite set of *registers*, $I, F \subseteq Q$ are sets of *initial* resp. *final* states, and $T \subseteq Q \times \Sigma \times \mathsf{Guard}(\mathsf{Reg}, \mathbb{D}) \times 2^{\mathsf{Reg}} \times Q$ is a finite set of *transitions*.

We will denote a transition $t = (q, a, \phi, \Lambda, q') \in T$ by $q \xrightarrow[\phi,\Lambda]{a} q'$. Let $\mathsf{label}(t) = a \in \Sigma$, the *label* of t. A *configuration* of $\mathcal{A}$ is a pair $c = \langle q, \vartheta\rangle$ consisting of a state $q \in Q$ and a register valuation $\vartheta \in \mathsf{Val}(\mathsf{Reg}, \mathbb{D})$. We say that c is *initial* if $q \in I$ and $\vartheta(r) = \perp$ for all $r \in \mathsf{Reg}$. We call c *final* if $q \in F$. Let $c = \langle q, \vartheta\rangle$ and $c' = \langle q', \vartheta'\rangle$ be configurations of $\mathcal{A}$, $t \in T$ a transition of the form $p \xrightarrow[\phi,\Lambda]{a} p'$, and $d \in D$ a data value. We say that $c \vdash_{t,d} c'$ is a *switch* from c to c' via the transition t and the data value d if $p = q$, $p' = q'$, $(\vartheta, d) \models \phi$ and $\vartheta' = \vartheta[\Lambda := d]$. Note that c' is uniquely determined by c, t and d. A *run* ρ of $\mathcal{A}$ is a non-empty sequence of switches between configurations starting in an initial configuration and ending in a final configuration. Formally, ρ is a sequence of the form $c_0 \vdash_{t_1,d_1} c_1 \vdash_{t_2,d_2} ... \vdash_{t_n,d_n} c_n$ where $n \geq 1$, c_0 is an initial configuration, $c_1, ..., c_{n-1}$ are configurations, c_n is a final configuration, $t_1, ..., t_n \in T$ and $d_1, ..., d_n \in D$. Let $\mathsf{label}(\rho) = (\mathsf{label}(t_1), d_1)...(\mathsf{label}(t_n), d_n) \in \mathbb{D}\Sigma^+$, the *label* of ρ. For any data word $w \in \mathbb{D}\Sigma^+$, let $\mathsf{Run}_{\mathcal{A}}(w)$ denote the set of all runs of $\mathcal{A}$ with label w. Let $\mathcal{L}(\mathcal{A}) = \{w \in \mathbb{D}\Sigma^+ \mid \mathsf{Run}_{\mathcal{A}}(w) \neq \emptyset\}$, the data language *recognized* by $\mathcal{A}$.

Example 2.2. Now we give some examples of data structures for register automata. For any data domain D, let $(=_D) \subseteq D \times D$ denote the data relation $\{(d, d) \mid d \in D\}$.

(a) Let D be any non-empty set. Then, $\mathbb{D} = (D, \{=_D\})$ is a data structure which corresponds to the original model of register automata [27]. Note that the register automata of [27] are also equipped with an initial vector of data values. In order to model this feature in our setting, we can extend the set of data relations of $\mathbb{D}$ with the set $\{R_d \mid d \in D\}$ where $R_d = \{(d, d') \mid d' \in D\}$ for all $d \in D$.

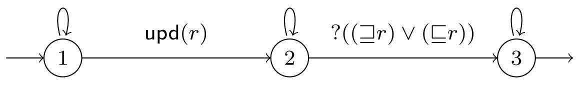

Fig. 1. The register automaton $\mathcal{A}$ of Example 2.3.

(b) Let $(D, <)$ be a linear order with a non-empty set D. Then, $(D, \{=_D, <_D\})$ is a data structure for the register automata considered in [26].

(c) Data are often represented in the form of finite strings. Here we consider the model of register automata which permits to check whether one data string is contained in another. Let Γ be an alphabet and $D = \Gamma^*$ be the data domain. Let $u, v \in \Gamma^*$. We say that u is *contained* in v, written $u \sqsubseteq v$, if $v = \alpha u \beta$ with $\alpha, \beta \in \Gamma^*$. Based on this definition, we can define the data relations $\sqsubseteq$ and $\sqsupseteq$ on D. Then, $(D, \{\sqsubseteq, \sqsupseteq\})$ is a data structure.

(d) In various situations, exact data values are not known and we deal with their approximated values, e.g., obtained from some experiments. Therefore, it can be reasonable to compare data values with respect to a given approximation error. For instance, let the data domain D be the set of all rational numbers. For any nonnegative rational number ε, let

$$R_\varepsilon = \{(q, q') \mid q, q' \in D \text{ and } |q - q'| \leq \varepsilon\}.$$

Note that R_0 is equal to $=_D$. Then, $(D, \{R_\varepsilon \mid \varepsilon \geq 0\})$ is a data structure for register automata with approximated values.

Example 2.3. Consider the data structure $\mathbb{D} = (D, \{\sqsubseteq, \sqsupseteq\})$ of Example 2.2 (c) with $D = \Gamma^*$. Let $\Sigma = \{a\}$ be a singleton alphabet. We say that a word $w = (a, d_1)...(a, d_n) \in \mathbb{D}\Sigma^+$ is *redundant* if $d_i \sqsubseteq d_j$ for some $i, j \in \{1, .., n\}$ with $i \neq j$. Let $\mathcal{L} \subseteq \mathbb{D}\Sigma^+$ be the data language of all redundant data words. The data language $\mathcal{L}$ is recognized by a register automaton $\mathcal{A}$ over $\mathbb{D}$ with one register r depicted in Fig. 1. Here, we omit the transition label a, register guard True and the empty register update; upd(r) means that the register r is updated.

3. Relation to timed automata

We show that *timed automata* [2] are also included in our model of register automata. Recall that timed automata are nondeterministic finite automata equipped with a finite set of clocks ranging over $\mathbb{R}_{\geq 0}$ and measuring the time of stay in the states. The transitions of a timed automaton are also augmented with a constraint on clock values and a subset of clocks which must be reset after taking this transition. Now we turn to a formal definition of a timed automaton. Recall that, given a finite set of clocks C, a *clock constraint* over C is generated by the grammar

$$\phi \ ::= \text{True} \mid x \bowtie k \mid \phi \wedge \phi \mid \neg\phi$$

where $x \in C$ and $k \in \mathbb{N}$. A *clock valuation* over C is a mapping $\nu : C \to \mathbb{R}_{\geq 0}$. Let $\mathbb{R}^C_{\geq 0}$ denote the set of all clock valuations. The definition that ν *satisfies* a clock constraint ϕ, written $\nu \models \phi$ is given by induction on the structure of ϕ as usual. Given a clock valuation $\nu \in \mathbb{R}^C_{\geq 0}$, a delay $d \in \mathbb{R}_{\geq 0}$ and a set $\Lambda \subseteq C$ of clocks to be reset, the clock valuations $\nu + d \in \mathbb{R}^C_{\geq 0}$ and $\nu[\Lambda := 0] \in \mathbb{R}^C_{\geq 0}$ are defined for all clocks $x \in C$ by $(\nu + d)(x) = \nu(x) + d$, $\nu[\Lambda := 0](x)$ is equal to 0 if $x \in \Lambda$ and is $\nu(x)$ otherwise.

Definition 3.1. A *timed automaton* over an alphabet Σ is a tuple $\mathcal{A} = (Q, C, I, E, F)$ where Q is a finite set of states, $I, F \subseteq Q$ are sets of initial resp. final states, C is a finite set of clocks and T is a finite set of transitions t of the form $q \xrightarrow[\phi,\Lambda]{a} q'$ where $q, q' \in Q$, $a \in \Sigma$, ϕ is a clock constraint over C and $\Lambda \subseteq C$ is a set of clocks to be reset.

Let label$(t) = a$ denote the *label* of t. A *configuration* of $\mathcal{A}$ is a pair $c = \langle q, \nu\rangle \in Q \times \mathbb{R}^C_{\geq 0}$. Given configurations $c = \langle q, \nu\rangle$, $c' = \langle q', \nu'\rangle$ of $\mathcal{A}$, a transition t of $\mathcal{A}$ of the form $p \xrightarrow[\phi,\Lambda]{a} p'$ and a time delay $d \in \mathbb{R}_{\geq 0}$, we say that $c \vdash_{t,d} c'$ is a *switch* from c to c' after the time delay d via t if $p = q$, $p' = q'$, $(\nu + d) \models \phi$ and $\nu' = (\nu + d)[\Lambda := 0]$. Then, as for register automata, we define a run ρ of $\mathcal{A}$ as a non-empty sequence of switches between configurations of the form

$$c_0 \vdash_{t_1,d_1} c_1 \vdash_{t_2,d_2} \ldots \vdash_{t_n,d_n} c_n$$

where $c_0 \in I \times \{0\}^C$ and $c_n \in F \times \mathbb{R}^C_{\geq 0}$. Then, the *label* of ρ is the word label$(\rho) = ($label$(t_1), d_1)($label$(t_2), d_1 + d_2)\ldots($label$(t_n), \sum_{i=1}^n d_i) \in (\Sigma \times \mathbb{R}_{\geq 0})^+$.

A *timed word* over Σ is a word $w = (a_1, \tau_1)...(a_n, \tau_n) \in (\Sigma \times \mathbb{R}_{\geq 0})^+$ such that $\tau_1 \leq \tau_2 \leq ... \leq \tau_n$. Let $\mathbb{T}\Sigma^+$ denote that set of all timed words over Σ. Note that, given a timed automaton and a run of it, the label of this run is necessarily a timed word. Any set of timed words is called a *timed language*. The timed language *recognized* by $\mathcal{A}$ is defined to be the set of all timed words w over Σ such that there exists a run ρ of $\mathcal{A}$ with label$(\rho) = w$.

In order to simulate timed automata by register automata, we consider the data structure $\mathbb{D}^{\mathsf{Timed}} = (D, \{R_{\bowtie k} \mid \bowtie \in \{<, =, >\} \text{ and } k \in \mathbb{N}\})$ where $D = \mathbb{R}_{\geq 0}$ with the initial data value $\bot = 0$ and, for $\bowtie \in \{<, =, >\}$ and $k \in \mathbb{N}$, $R_{\bowtie k} = \{(\tau, \tau') \mid \tau, \tau' \in D \text{ and } \tau' - \tau \bowtie k\}$. Note that $R_{=0}$ is equal to $=_D$.

Lemma 3.2. *Let Σ be an alphabet. Then every recognizable timed language over Σ is recognizable by a register automaton over Σ and $\mathbb{D}^{\mathsf{Timed}}$.*

Proof. Let $\mathcal{A} = (Q, C, I, T, F)$ be a timed automaton over Σ. We show that there exists a register automaton $\mathcal{A}^{\mathsf{data}}$ over Σ and $\mathbb{D}^{\mathsf{Timed}}$ such that $\mathcal{L}(\mathcal{A}^{\mathsf{data}}) = \mathcal{L}(\mathcal{A})$. We let $\mathcal{A}^{\mathsf{data}} = (Q, \mathsf{Reg}, I, T', F)$ where:

- $\mathsf{Reg} = C \cup \{\tilde{r}\}$ where the register $\tilde{r} \notin C$ will be used to check monotonicity of the timed part of a timed word;
- every transition $t \in T$ of the form $q \xrightarrow[\phi, \Lambda]{a} q'$ is simulated in $\mathcal{A}^{\mathsf{data}}$ by the transition $\Phi(t) = \left(q \xrightarrow[\phi', \Lambda \cup \{\tilde{r}\}]{a} q'\right)$ where $\phi' = \tilde{\phi} \wedge (R_{\geq 0}\tilde{r})$ and $\tilde{\phi} \in \mathsf{Guard}(\mathsf{Reg}, \mathbb{D}^{\mathsf{Timed}})$ is obtained from ϕ by replacing every constraint $x \bowtie k$ by $(R_{\bowtie k})x$. In other words, we let $T' = \Phi(T)$.

This construction relies on the fact that the value of a clock at a time stamp τ is the difference between τ and the time stamp of the previous clock reset before τ. Then, whenever the timed automaton $\mathcal{A}$ resets a clock x, the register automaton $\mathcal{A}^{\mathsf{data}}$ stores in the register x the time stamp of this reset. Although negative time delays are admitted by register automata over $\mathbb{D}^{\mathsf{Timed}}$, they are not possible in $\mathcal{A}^{\mathsf{data}}$ since it checks using the register $\tilde{r}$ that all time delays are nonnegative. Then $\mathcal{L}(\mathcal{A}^{\mathsf{data}}) = \mathcal{L}(\mathcal{A})$ and hence the claim follows. □

Note that register automata over $\mathbb{D}^{\mathsf{Timed}}$ are more expressive than timed automata since they can accept non-monotonic data words. Nevertheless, the following holds true:

Lemma 3.3. *Let Σ be an alphabet and $\mathcal{L} \subseteq \mathbb{T}\Sigma^+$. Then, $\mathcal{L}$ is recognizable by a register automaton over Σ and $\mathbb{D}^{\mathsf{Timed}}$ iff $\mathcal{L}$ is recognizable by a timed automaton over Σ.*

Proof. The implication $\Leftarrow$ follows immediately from Lemma 3.2. We show the converse implication. Let $\mathcal{A}^{\mathsf{data}} = (Q, \mathsf{Reg}, I, T, F)$ be a register automaton over Σ and $\mathbb{D}^{\mathsf{Timed}}$ such that $\mathcal{L}(\mathcal{A}^{\mathsf{data}}) = \mathcal{L}$. Consider the timed automaton $\mathcal{A}^{\mathsf{time}} = (Q, C, I, T', F)$ over Σ where $C = \mathsf{Reg}$ and T' is defined as follows. Every transition $t = \left(q \xrightarrow[\phi, \Lambda]{a} q'\right) \in T$ is simulated in $\mathcal{A}^{\mathsf{time}}$ by the transition $t' = \left(q \xrightarrow[\tilde{\phi}, \Lambda]{a} q'\right) \in T'$ where the clock constraint $\tilde{\phi}$ over C is obtained from ϕ by replacing every register guard $(R_{\bowtie k})x$ by the atomic clock constant $x \bowtie k$. Then, the fact that $\mathcal{L}(\mathcal{A}^{\mathsf{data}}) \subseteq \mathbb{T}\Sigma^+$ guarantees that $\mathcal{L}(\mathcal{A}^{\mathsf{time}}) = \mathcal{L}(\mathcal{A}^{\mathsf{data}}) = \mathcal{L}$. Hence the claim follows. □

4. Weighted register automata: definition and examples

In this section, we introduce *weighted register automata* as a quantitative model for reasoning about data words. They extend the qualitative register automata of the previous section with weights and reflect the following quantitative information.

- As in classical weighted automata [19], transitions of our new weighted register automata also carry weights which do not depend on data.
- As opposed to weighted automata, weighted register automata must be able to process data taken from an infinite data domain. Therefore, the size of data can be very large and processing of such data can be expensive. Our model of weighted register automata takes into account the costs of data processing, namely, the costs of storing data in registers and the cost of comparing a new datum with old data stored in registers.

Note that our weighted register automata model is different from the cost register automata model of Alur et al. [1], because cost register automata run on words over a finite alphabet and the registers are used to compute the weight of a run and can be updated depending on the previous register values. In contrast, in our model we deal with data words over an infinite alphabet and the registers can be updated only depending on the current input data value.

In order to be able to reflect various quantitative settings, we will consider a general structure for weighted register automata. For this purpose, we adopt the structure of semirings (as in the classical weighted automata [19]) to our new setting of data words.

For any sets X, Y, let Y^X denote the collection of all mappings $f : X \to Y$. For $y \in Y$, let y^X denote the mapping $y^X : X \to \{y\}$. A *data semiring* over a data structure $\mathbb{D} = (D, \mathcal{R})$ is a pair $\mathbb{S} = \langle \mathcal{S}, \mathcal{F} \rangle$ where $\mathcal{S} = (S, +, \cdot, \mathbf{0}, \mathbf{1})$ is a semiring and

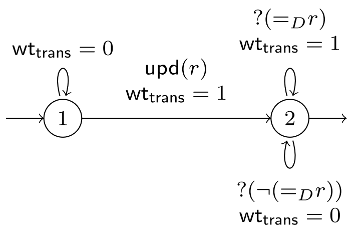
$\mathsf{wt}_{\mathsf{trans}} = 0$
$\mathsf{upd}(r)$
$\mathsf{wt}_{\mathsf{trans}} = 1$
$?(=_D r)$
$\mathsf{wt}_{\mathsf{trans}} = 1$
1
2
$?(\neg(=_D r))$
$\mathsf{wt}_{\mathsf{trans}} = 0$

Fig. 2. The WRA $\mathcal{A}$ of Example 4.3.

$\mathcal{F} \subseteq S^{D\times D}$ such that $\mathbf{1}^{D\times D} \in \mathcal{F}$. Let $\mathrm{dom}(\mathbb{S}) = S$, the *domain* of $\mathbb{S}$. We call $\mathbb{S}$ *commutative* if $\mathcal{S}$ is a commutative semiring, i.e., $s \cdot s' = s' \cdot s$ for all $s, s' \in S$.

Definition 4.1. Let Σ be an alphabet, $\mathbb{D}$ a data structure and $\mathbb{S} = \langle (S, +, \cdot, \mathbf{0}, \mathbf{1}), \mathcal{F}\rangle$ a data semiring over $\mathbb{D}$. A *weighted register automaton (WRA)* over Σ, $\mathbb{D}$ and $\mathbb{S}$ is a tuple $\mathcal{A} = (Q, \mathsf{Reg}, I, T, F, \mathsf{wt})$ where $(Q, \mathsf{Reg}, I, T, F)$ is a register automaton over Σ and $\mathbb{D}$, and $\mathsf{wt} = \langle \mathsf{wt}_{\mathsf{trans}}, \mathsf{wt}_{\mathsf{data}}\rangle$ is a pair of weight functions such that $\mathsf{wt}_{\mathsf{trans}} : T \to S$ and $\mathsf{wt}_{\mathsf{data}} : (T \times \mathsf{Reg}) \to \mathcal{F}$.

Note that $\mathsf{wt}_{\mathsf{trans}} : T \to S$ can be considered as a weight function of the classical weighted automata [19] and describes data-independent costs for transitions. The weight function $\mathsf{wt}_{\mathsf{data}}$ assigns to every transition $t \in T$, every register $r \in \mathsf{Reg}$, every data value d stored in r and every new data value $d' \in D$ the cost $\mathsf{wt}_{\mathsf{data}}(t,r)(d,d') \in S$ of data processing in the register r which includes the cost of comparing d with d' and, if necessary, the cost of storing d' in the register r.

We fix an enumeration $(r_j)_{1\le j\le m}$ of Reg. Let $c = \langle q, \vartheta\rangle$ and $c' = \langle q', \vartheta'\rangle$ be configurations of $\mathcal{A}$ and $c \vdash_{t,d} c'$ a switch between them. Then, its *weight* is defined as the product of the costs of data processing in the registers (defined by $\mathsf{wt}_{\mathsf{data}}$) and the transition cost of $\mathsf{wt}_{\mathsf{trans}}(t)$. Formally, we let

$$\mathsf{wt}(c \vdash_{t,d} c') = \prod_{j=1}^{m} \mathsf{wt}_{\mathsf{data}}(t, r_j)(\vartheta(r_j), d) \cdot \mathsf{wt}_{\mathsf{trans}}(t). \tag{1}$$

Now let $\rho = \big(c_0 \vdash_{t_1,d_1} c_1 \vdash_{t_2,d_2} \dots \vdash_{t_n,d_n} c_n\big)$ be a run of $\mathcal{A}$. Then, the *weight* of ρ is defined as the product of the weights of all switches of ρ. Formally, we let $\mathsf{wt}(\rho) = \prod_{i=1}^{n} \mathsf{wt}(c_{i-1} \vdash_{t_i,d_i} c_i)$. Then, the *behavior* of $\mathcal{A}$ is the mapping $[\![\mathcal{A}]\!] : \mathbb{D}\Sigma^+ \to S$ defined for all $w \in \mathbb{D}\Sigma^+$ by

$$[\![\mathcal{A}]\!](w) = \sum \big(\mathsf{wt}(\rho) \mid \rho \in \mathsf{Run}_{\mathcal{A}}(w)\big).$$

We will call any mapping $\mathbb{L} : \mathbb{D}\Sigma^+ \to S$ a *data series* over Σ, $\mathbb{D}$ and $\mathbb{S}$. Let $\mathbb{S}\langle\!\langle \mathbb{D}\Sigma^+\rangle\!\rangle$ denote the collection of all data series over Σ, $\mathbb{D}$ and $\mathbb{S}$. We say that $\mathbb{L} \in \mathbb{S}\langle\!\langle \mathbb{D}\Sigma^+\rangle\!\rangle$ is *recognizable* if there exists a WRA $\mathcal{A}$ over Σ, $\mathbb{D}$ and $\mathbb{S}$ such that $[\![\mathcal{A}]\!] = \mathbb{L}$.

Example 4.2.

(a) Consider the *Boolean semiring* $\mathbb{B} = (\{0,1\}, \vee, \wedge, \mathbf{0}, \mathbf{1})$ where $\mathbf{1} \vee \mathbf{1} = \mathbf{1}$. Let $\mathbb{D} = (D, \mathcal{R})$ be a data structure, and let $\mathcal{F} = \{\mathbf{1}^{D\times D}\}$. Then $\langle \mathbb{B}, \mathcal{F}\rangle$ is a *data boolean semiring*, and a data language $\mathcal{L} \subseteq \mathbb{D}\Sigma^+$ is recognizable if and only if its characteristic function $\mathbf{1}_{\mathcal{L}} : \mathbb{D}\Sigma^+ \to \mathbb{B}$ (defined by $\mathbf{1}_{\mathcal{L}}(w) = \mathbf{1}$ iff $w \in \mathcal{L}$) is a recognizable data series.

(b) Consider the *arctic semiring* $\mathsf{Arc} = (\mathbb{N} \cup \{-\infty\}, \max, +, -\infty, 0)$ of natural numbers. Let $\mathbb{D} = (D, \mathcal{R})$ be a data structure augmented with a *size function* $\mathsf{size} : D \to \mathbb{N}$ (e.g., length of a data string or number of bits of an integer). Consider the following collections of functions $\mathcal{F}_1$ and $\mathcal{F}_2$ defined as follows.
   - Let $\mathcal{F}_1$ be the collection of functions $f_c : D \times D \to \mathbb{N}$ with $c \in \mathbb{N}$ and $f_c(d,d') = c \cdot \mathsf{size}(d')$ for all $d, d' \in D$. The collection $\mathcal{F}_1$ can be useful, e.g., for the cases where we need to estimate the costs of checking the equality of data or to update a register.
   - Let $\mathcal{F}_2$ be the collection of functions $g_{k,l} : D \times D \to \mathbb{N}$ with $k, l \in \mathbb{N}$ and $g_{k,l}(d,d') = k \cdot \mathsf{size}(d) + l \cdot \mathsf{size}(d')$ for all $d, d' \in D$. The collection $\mathcal{F}_2$ can be useful, e.g., for the cases where we need to estimate the costs of finding a pattern in a data string (e.g., using the well-known Knuth–Morris–Pratt algorithm).

   Then $\langle \mathsf{Arc}, \mathcal{F}_1\rangle$ and $\langle \mathsf{Arc}, \mathcal{F}_2\rangle$ are data semirings.

Example 4.3. Consider the data structure $\langle D, \{=_D\}\rangle$ of Example 2.2 (a) and a singleton alphabet $\Sigma = \{a\}$. For a data word $w \in \mathbb{D}\Sigma^+$ and $d \in D$, let $|w|_d \in \mathbb{N}$ be the number of d's in w. Consider the data series $\mathbb{L} : \mathbb{D}\Sigma^+ \to \mathbb{N}$ defined by $\mathbb{L}(w) = \max_{d\in D} |w|_d$. Consider the data semiring $\mathbb{S} = \langle \mathsf{Arc}, \{\mathbf{1}^{D\times D}\}\rangle$ and the WRA $\mathcal{A}$ over Σ, $\mathbb{D}$ and $\mathbb{S}$ depicted in Fig. 2 which has the only register r. Moreover, for all transitions t of $\mathcal{A}$, we put $\mathsf{wt}_{\mathsf{data}}(t,r) = \mathbf{1}^{D\times D}$. Then $[\![\mathcal{A}]\!] = \mathbb{L}$.

Example 4.4. Let $\mathbb{D} = \langle D, \mathcal{R}\rangle$ be a data structure with a size function $\mathsf{size} : D \to \mathbb{N}$. Let Σ be an alphabet and $\mathcal{A} = (Q, \mathsf{Reg}, I, T, F)$ a register automaton over Σ and $\mathbb{D}$. Let $\tilde{r} \in \mathsf{Reg}$ be a designated register. Assume that for every data word $w \in \mathbb{D}\Sigma^+$ we want to observe the maximal size of data which will be loaded into $\tilde{r}$ along any run with label w. For this, we consider the data semiring $\langle \mathsf{Arc}, \mathcal{F}_1 \rangle$ of Example 4.2 (b) and construct the WRA $\mathcal{A}'$ over Σ, $\mathbb{D}$ and $\mathbb{S}$ as follows. We take two copies $\mathcal{A}^{(1)}$ and $\mathcal{A}^{(2)}$ of $\mathcal{A}$ with the same set Reg of registers. The initial states of $\mathcal{A}'$ are the initial states of $\mathcal{A}^{(1)}$ whereas the final states of $\mathcal{A}'$ are the final states of $\mathcal{A}^{(2)}$. For all transitions t in these two copies we let $\mathsf{wt}_{\mathsf{trans}}(t) = 0$ and, for all registers $r \in \mathsf{Reg}$, we let $\mathsf{wt}_{\mathsf{data}}(t, r) = 0^{D \times D}$. Whenever there is a transition $p^{(1)} \xrightarrow[\phi,\Lambda]{a} q^{(1)}$ in $\mathcal{A}^{(1)}$ with $\tilde{r} \in \Lambda$, we add to $\mathcal{A}'$ the transition $t = \big(p^{(1)} \xrightarrow[\phi,\Lambda]{a} q^{(2)}\big)$ where $q^{(2)}$ is the state in $\mathcal{A}^{(2)}$ which corresponds to $q^{(1)}$. Let $f \in \mathcal{F}_1$ be defined as $f(d, d') = d'$. Then, we let $\mathsf{wt}_{\mathsf{trans}}(t) = 0$, $\mathsf{wt}_{\mathsf{data}}(t, \tilde{r}) = f$ and $\mathsf{wt}_{\mathsf{data}}(t, r) = 0^{D \times D}$ for all $r \in \mathsf{Reg} \setminus \{\tilde{r}\}$. Then, for all $w \in \mathbb{D}\Sigma^+$, $[\![\mathcal{A}']\!](w)$ is the maximal size of data which is loaded into $\tilde{r}$ along any run of $\mathcal{A}$ with label w. Note that if $\tilde{r}$ is not updated along any run with label w, then $[\![\mathcal{A}']\!](w) = -\infty$.

5. Relation to weighted timed automata

A semiring-based model for *weighted timed automata* (WTA) of [5,7] was investigated in [29,30]. We show that WTA are incorporated by our new weighted register automata.

A *timed semiring* is a pair $\mathbb{S} = \langle \mathcal{S}, \mathcal{F} \rangle$ where $\mathcal{S} = (S, +, \cdot, \mathbf{0}, \mathbf{1})$ is a semiring and $\mathcal{F} \subseteq S^{\mathbb{R}_{\geq 0}}$ such that $\mathbf{1}^{\mathbb{R}_{\geq 0}} \in \mathcal{F}$. In the rest of this subsection, we restrict ourselves to the case when $\mathcal{S}$ is commutative. Let Σ be an alphabet. A *weighted timed automaton* (WTA) over Σ and $\mathbb{S}$ is a tuple $\mathcal{A} = (Q, C, I, T, F, \mathsf{wt})$ where (Q, C, I, T, F) is a timed automaton over Σ (cf. Section 3) and $\mathsf{wt} = \langle \mathsf{wt}_{\mathsf{trans}}, \mathsf{wt}_{\mathsf{time}} \rangle$ is a pair of weight functions such that $\mathsf{wt}_{\mathsf{trans}} : T \to S$ and $\mathsf{wt}_{\mathsf{time}} : Q \to \mathcal{F}$. Note that, in contrast to WRA, time-dependent costs in WTA are assigned to states and these costs are modelled by functions of a single argument.

Let $\rho = \big(c_0 \vdash_{t_1,d_1} c_1 \vdash_{t_2,d_2} \ldots \vdash_{t_n,d_n} c_n\big)$ be a run of $\mathcal{A}$ where $c_i = (q_i, \nu_i)$ for all $i \in \{0, \ldots, n\}$. The *weight* of ρ is defined as

$$\mathsf{wt}(\rho) = \prod_{i=1}^{n} \mathsf{wt}_{\mathsf{time}}(q_{i-1})(d_i) \cdot \mathsf{wt}_{\mathsf{trans}}(t_i).$$

Note that $\mathsf{wt}(\rho) \in S$. The *behavior* of $\mathcal{A}$ is the mapping $[\![\mathcal{A}]\!] : \mathbb{T}\Sigma^+ \to S$ defined for all $w \in T\Sigma^+$ by

$$[\![\mathcal{A}]\!](w) = \sum \big(\mathsf{wt}(\rho) \mid \rho \text{ is a run of } \mathcal{A} \text{ with } \mathsf{label}(\rho) = w\big).$$

A mapping $\mathbb{L} : \mathbb{T}\Sigma^+ \to S$ is called a *timed series*.

For $f : \mathbb{R}_{\geq 0} \to S$, let $\Psi(f) : \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \to S$ be defined by

$$\Psi(f)(d, d') = \begin{cases} f(d' - d), & \text{if } d \leq d', \\ \mathbf{1}, & \text{otherwise.} \end{cases}$$

Now we show that WTA over the timed semiring $\mathbb{S}$ can be simulated by WRA. For this, we consider the data structure $\mathbb{D}^{\mathsf{Timed}}$ of Section 3 and the data semiring $\tilde{\mathbb{S}} = \langle \mathcal{S}, \tilde{\mathcal{F}} \rangle$ with $\tilde{\mathcal{F}} = \{\Psi(f) \mid f \in \mathcal{F}\}$.

Given a data series $\mathbb{L} : \mathbb{D}^{\mathsf{Timed}}\Sigma^+ \to S$ with $\mathbb{L}(w) = \mathbf{0}$ for all $w \in \mathbb{D}^{\mathsf{Timed}}\Sigma^+ \setminus \mathbb{T}\Sigma^+$, we identify $\mathbb{L}$ with the timed series $\tilde{\mathcal{L}} = \mathcal{L}|_{\mathbb{T}\Sigma^+}$. The following lemma extends Lemma 3.2 to the weighted setting.

Lemma 5.1. *Let $\mathbb{L} : \mathbb{T}\Sigma^+ \to S$ be a timed series recognizable by a WTA over Σ and $\mathbb{S}$. Then, $\mathbb{L}$ is recognizable by a WRA over Σ, $\mathbb{D}^{\mathsf{Timed}}$ and $\tilde{\mathbb{S}}$.*

Proof. Let $\mathcal{A}^{\mathsf{time}} = (Q, C, I, T, F, \langle \mathsf{wt}_{\mathsf{trans}}, \mathsf{wt}_{\mathsf{time}} \rangle)$ be a WTA over Σ and $\mathbb{S}$ such that $[\![\mathcal{A}^{\mathsf{time}}]\!] = \mathbb{L}$. We may assume that $T \neq \emptyset$. We define the weighted register automaton $\mathcal{A}^{\mathsf{data}} = (Q, \mathsf{Reg}, I, T', F, \langle \mathsf{wt}'_{\mathsf{trans}}, \mathsf{wt}'_{\mathsf{data}} \rangle)$ over Σ, $\mathbb{D}^{\mathsf{Timed}}$ and $\tilde{\mathbb{S}}$ as follows:

- $\mathsf{Reg} = C \cup \{\tilde{r}\}$ where $\tilde{r} \notin C$;
- T' consists of all transitions $t' = \big(q \xrightarrow[\tilde{\phi} \wedge R_{\geq 0}\tilde{r}, \Lambda \cup \{\tilde{r}\}]{a} q'\big)$ such that there exists a transition $t = \big(q \xrightarrow[\phi,\Lambda]{a} q'\big) \in T$ and $\tilde{\phi} \in \mathsf{Guard}(\mathsf{Reg}, \mathbb{D}^{\mathsf{Timed}})$ is obtained from ϕ by replacing every constraint $x \bowtie k$ by $(R_{\bowtie k})x$. Then, we let:
  - $\mathsf{wt}'_{\mathsf{trans}}(t') = \mathsf{wt}_{\mathsf{trans}}(t)$;
  - $\mathsf{wt}'_{\mathsf{data}}(t', r) = \mathbf{1}^{\mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0}}$ for all $r \in C$;
  - $\mathsf{wt}'_{\mathsf{data}}(t', \tilde{r}) = \Psi(\mathsf{wt}_{\mathsf{time}}(q))$.

It is easy to see that $[\![\mathcal{A}^{\mathsf{data}}]\!] = [\![\mathcal{A}^{\mathsf{time}}]\!]$. Hence the claim follows. □

However, in general, the statement analogous to Lemma 3.3 does not hold in the weighted setting.

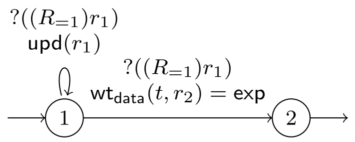

Fig. 3. The WRA $\mathcal{A}^{\mathsf{data}}$ in the proof of Lemma 5.2.

Lemma 5.2. *There exist an alphabet Σ, a commutative timed semiring $\mathbb{S}$ with domain S and a timed series $\mathbb{L} : \mathbb{T}\Sigma^+ \to S$ such that $\mathbb{L}$ is recognizable by a WRA over Σ, $\mathbb{D}^{\mathsf{Timed}}$ and $\tilde{\mathbb{S}}$, but $\mathbb{L}$ is not recognizable by a WTA over Σ and $\mathbb{S}$.*

Proof. Let $\Sigma = \{a\}$ be a singleton alphabet and $\mathbb{S} = \langle \mathcal{S}, \mathcal{F}\rangle$ where $\mathcal{S}$ is the arctic semiring of Example 4.2(b) and $\mathcal{F} = \{0^{\mathbb{R}_{\geq 0}\times\mathbb{R}_{\geq 0}}, \exp\}$ where, for all $d, d' \in \mathbb{R}_{\geq 0}$, $\exp(d, d') = e^{d'-d}$ if $d \leq d'$ and $\exp(d, d') = 0$ otherwise. For all $n \geq 1$, let $w_n = (a, 1)(a, 2)\ldots(a, n)$. Then, we define $\mathbb{L}$ for all $w \in \mathbb{T}\Sigma^+$ as

$$\mathbb{L}(w) = \begin{cases} e^n, & \text{if } w = w_n \text{ for some } n \geq 1, \\ -\infty, & \text{otherwise.} \end{cases}$$

Consider the WRA $\mathcal{A}^{\mathsf{data}}$ over Σ, $\mathbb{D}^{\mathsf{Timed}}$ and $\tilde{\mathbb{S}}$ with two registers r_1, r_2 depicted in Fig. 3. In this figure, for all transitions t we have $\mathsf{wt}_{\mathsf{trans}}(t) = 0$, and for all registers $r \in \{r_1, r_2\}$, $\mathsf{wt}_{\mathsf{data}}(t, r) = 0^{\mathbb{R}_{\geq 0}\times\mathbb{R}_{\geq 0}}$ is not specified. Note that the register r_2 is never updated and its value is always 0. Then $[\![\mathcal{A}]\!]^{\mathsf{data}} = \mathbb{L}$.

Suppose now that there exists a WTA $\mathcal{A}^{\mathsf{time}}$ over Σ and $\mathbb{S}$ such that $[\![\mathcal{A}^{\mathsf{time}}]\!] = \mathbb{L}$. Let $M \geq 1$ be such that $\mathsf{wt}_{\mathsf{trans}}(t) \leq M$ for all transitions t of $\mathcal{A}^{\mathsf{time}}$. Let $n \geq 1$ and ρ be a run of $\mathcal{A}^{\mathsf{time}}$ with label w_n. Then ρ has the form $c_0 \vdash_{t_1,1} c_1 \vdash_{t_2,1} \ldots \vdash_{t_n,1} c_n$, i.e., the delays between any two subsequent configurations equal one. Then, $\mathsf{wt}(c_{i-1} \vdash_{t_i,1}, c_i) \leq M + e^1$ for all $i \in \{1, \ldots, n\}$ and hence

$$\mathsf{wt}(\rho) \leq \sum_{i=1}^{n} (M + e) = (M + e)n.$$

This means that $e^n = [\![\mathcal{A}^{\mathsf{time}}]\!](w_n) \leq (M + e)n$ for all $n \geq 1$. A contradiction. Then $\mathbb{L}$ is not recognizable by a WTA. □

6. Closure properties of weighted register automata

Now we establish some basic closure properties for the class of recognizable data series. We will apply them in the proof of our logic characterization result. We note that the class of timed series recognizable by weighted timed automata of [30] is not stable under the Hadamard product even in the case of commutative semirings (cf. Example 5 of [30]). Interestingly, our model of WRA extends weighted timed automata and the class of recognizable data languages is closed under the Hadamard product. Let Σ be an alphabet, $\mathbb{D} = \langle D, \mathcal{R}\rangle$ a data structure and $\mathbb{S} = \langle (S, +, \cdot, \mathbf{0}, \mathbf{1}), \mathcal{F}\rangle$ a commutative data semiring over $\mathbb{D}$.

Let $\mathbb{L}_1, \mathbb{L}_2 \in \mathbb{S}\langle\langle \mathbb{D}\Sigma^+\rangle\rangle$ be two data series. The *sum* $\mathbb{L}_1 + \mathbb{L}_2 \in \mathbb{S}\langle\langle \mathbb{D}\Sigma^+\rangle\rangle$ and the *Hadamard product* $\mathbb{L}_1 \odot \mathbb{L}_2 \in \mathbb{S}\langle\langle \mathbb{D}\Sigma^+\rangle\rangle$ are defined pointwise by $(\mathbb{L}_1 + \mathbb{L}_2)(w) = \mathbb{L}_1(w) + \mathbb{L}_2(w)$ respectively $(\mathbb{L}_1 \odot \mathbb{L}_2)(w) = \mathbb{L}_1(w) \cdot \mathbb{L}_2(w)$ for all $w \in \mathbb{D}\Sigma^+$.

Lemma 6.1. *Let Σ, Γ be alphabets, $\mathbb{D}$ a data structure and $\mathbb{S}$ a commutative data semiring over $\mathbb{D}$. If $\mathbb{L}_1, \mathbb{L}_2 \in \mathbb{S}\langle\langle \mathbb{D}\Sigma^+\rangle\rangle$ are recognizable, then so are $\mathbb{L}_1 + \mathbb{L}_2$ and $\mathbb{L}_1 \odot \mathbb{L}_2$.*

Proof. The claim for $\mathbb{L}_1 + \mathbb{L}_2$ can be easily shown by applying the standard disjoint union construction for automata. For $\mathbb{L}_1 \odot \mathbb{L}_2$, the idea of our construction goes back to the standard product construction of automata. For $i \in \{1, 2\}$, let $\mathcal{A}_i = (Q_i, \mathsf{Reg}_i, I_i, T_i, F_i, \langle \mathsf{wt}^i_{\mathsf{trans}}, \mathsf{wt}^i_{\mathsf{data}}\rangle)$ be a WRA with $[\![\mathcal{A}_i]\!] = \mathbb{L}_i$, and let $|\mathsf{Reg}_i| = m_i$ for $m_i \geq 1$. We may assume that $Q_1 \cap Q_2 = \emptyset$, $\mathsf{Reg}_1 \cap \mathsf{Reg}_2 = \emptyset$. Let $\mathsf{Reg}_1 = \{r_1, \ldots, r_{m_1}\}$ and $\mathsf{Reg}_2 = \{r_{m_1+1}, \ldots, r_{m_1+m_2}\}$. We construct the WRA $\mathcal{A} = (Q, \mathsf{Reg}, I, T, F, \langle \mathsf{wt}_{\mathsf{trans}}, \mathsf{wt}_{\mathsf{data}}\rangle)$ for $\mathbb{L}_1 \odot \mathbb{L}_2$ as follows. We let $Q = Q_1 \times Q_2$, $\mathsf{Reg} = \mathsf{Reg}_1 \cup \mathsf{Reg}_2$, $I = I_1 \times I_2$ and $F = F_1 \times F_2$. We also let T consist of all transitions $t = \big((p_1, q_1) \xrightarrow[\phi\wedge\phi', \Lambda\cup\Lambda']{a} (p_2, q_2)\big)$ where $t_1 = \big(p_1 \xrightarrow[\phi, \Lambda]{a} p_2\big)$ is a transition in T_1 and $t_2 = \big(q_1 \xrightarrow[\phi', \Lambda']{a} q_2\big)$ is a transition in T_2. For such a transition t and a register $r \in \mathsf{Reg}$, we let $\mathsf{wt}_{\mathsf{trans}}(t) = \mathsf{wt}^1_{\mathsf{trans}}(t_1) \cdot \mathsf{wt}^2_{\mathsf{trans}}(t_2)$ and $\mathsf{wt}_{\mathsf{data}}(t, r) = \mathsf{wt}^i_{\mathsf{data}}(t_i, r)$ whenever $r \in \mathsf{Reg}_i$ for $i \in \{1, 2\}$. The goal is to show that $[\![\mathcal{A}]\!] = [\![\mathcal{A}_1]\!] \odot [\![\mathcal{A}_2]\!]$. First we show that there is a weight-preserving bijection between the set of runs of $\mathcal{A}$ and the set of pairs of runs of $\mathcal{A}_1$ and $\mathcal{A}_2$. Let $w \in \mathbb{D}\Sigma^+$ with $|w| = n$ for $n \geq 1$. Assume that

$$\rho ::= c_0 \vdash_{t_1,d_1} c_1 \vdash_{t_2,d_2} \ldots \vdash_{t_n,d_n} c_n$$

is a successful run of $\mathcal{A}$ on w such that $t_i = \big((p_{i-1}, q_{i-1}) \xrightarrow[\phi_i \wedge \phi_i', \Lambda_i \cup \Lambda_i']{a} (p_i, q_i)\big) \in T$ for $i \in \{1, 2, ..., n\}$. With the given construction for $\mathcal{A}$ it can be seen that there are runs

$$\rho' ::= c_0' \vdash_{t_1', d_1} c_1' \vdash_{t_2', d_2} \ldots \vdash_{t_n', d_n} c_n'$$

of $\mathcal{A}_1$ and

$$\rho'' ::= c_0'' \vdash_{t_1'', d_1} c_1'' \vdash_{t_2'', d_2} \ldots \vdash_{t_n'', d_n} c_n''$$

of $\mathcal{A}_2$, respectively over w such that $t_i' = \big(p_{i-1} \xrightarrow[\phi_i, \Lambda_i]{a} p_i\big) \in T_1$ and $t_i'' = \big(q_{i-1} \xrightarrow[\phi_i', \Lambda_i']{a} q_i\big) \in T_2$ for $i \in \{1, 2, ..., n\}$. Now using the definition for the weight functions and commutativity of the semiring, for a data word $w \in \mathbb{D}\Sigma^+$ we have:

$$\begin{aligned}
\mathsf{wt}(\rho) &= \prod_{i=1}^{n} \mathsf{wt}(c_{i-1} \vdash_{t_i, d_i} c_i) \\
&= \prod_{i=1}^{n} \prod_{j=1}^{m_1+m_2} \mathsf{wt}_{\mathsf{data}}(t_i, r_j)(\vartheta_i(r_j), d_i) \cdot \mathsf{wt}_{\mathsf{trans}}(t_i) \\
&= \prod_{i=1}^{n} \Big(\prod_{j=1}^{m_1} \mathsf{wt}_{\mathsf{data}}(t_i', r_j)(\vartheta_i(r_j), d_i) \cdot \mathsf{wt}_{\mathsf{trans}}(t_i') \cdot \prod_{j=1}^{m_2} \mathsf{wt}_{\mathsf{data}}(t_i'', r_{m_1+j})(\vartheta_i(r_{m_1+j}), d_i) \cdot \mathsf{wt}_{\mathsf{trans}}(t_i'') \Big) \\
&= \prod_{i=1}^{n} \prod_{j=1}^{m_1} \mathsf{wt}_{\mathsf{data}}(t_i', r_j)(\vartheta_i(r_j), d_i) \cdot \mathsf{wt}_{\mathsf{trans}}(t_i') \cdot \prod_{i=1}^{n} \prod_{j=1}^{m_2} \mathsf{wt}_{\mathsf{data}}(t_i'', r_{m_1+j})(\vartheta_i(r_{m_1+j}), d_i) \cdot \mathsf{wt}_{\mathsf{trans}}(t_i'') \\
&= \prod_{i=1}^{n} \mathsf{wt}(c_{i-1}' \vdash_{t_i', d_i} c_i') \cdot \prod_{i=1}^{n} \mathsf{wt}(c_{i-1}'' \vdash_{t_i'', d_i} c_i'') = \mathsf{wt}(\rho_1) \cdot \mathsf{wt}(\rho_2).
\end{aligned}$$

For the other direction, let ρ' and ρ'' as above be runs of $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively, on w. One can see that the composition of ρ' and ρ'' gives the run ρ of $\mathcal{A}$ on w, and therefore with the same argument as above we have $\mathsf{wt}(\rho') \cdot \mathsf{wt}(\rho'') = \mathsf{wt}(\rho)$. We apply these arguments and hence for each $w \in \mathbb{D}\Sigma^+$ we have:

$$\begin{aligned}
[\![\mathcal{A}]\!](w) &= \sum \big(\mathsf{wt}(\rho) \mid \rho \in \mathsf{Run}_{\mathcal{A}}(w)\big) \\
&= \sum \big(\mathsf{wt}(\rho') \cdot \mathsf{wt}(\rho'') \mid \rho' \in \mathsf{Run}_{\mathcal{A}_1}(w) \text{ and } \rho'' \in \mathsf{Run}_{\mathcal{A}_2}(w)\big) \\
&= \sum \big(\mathsf{wt}(\rho') \mid \rho' \in \mathsf{Run}_{\mathcal{A}_1}(w)\big) \cdot \sum \big(\mathsf{wt}(\rho'') \mid \rho'' \in \mathsf{Run}_{\mathcal{A}_2}(w)\big) \\
&= ([\![\mathcal{A}_1]\!] \odot [\![\mathcal{A}_2]\!])(w) \\
&= (\mathbb{L}_1 \odot \mathbb{L}_2)(w).
\end{aligned}$$

Note that the third equality is implied by the arguments above and also by using distributivity and commutativity of the data semiring. □

Let Γ be an alphabet and $h : \Gamma \to \Sigma$ a mapping called henceforth a *renaming*. For a data word $u = (a_1, d_1)...(a_n, d_n) \in \mathbb{D}\Gamma^+$, let $h(u) = (h(a_1), d_1)...(h(a_n), d_n) \in \mathbb{D}\Sigma^+$. For a data series $\mathbb{L} \in \mathbb{S}\langle\langle \mathbb{D}\Gamma^+ \rangle\rangle$, the *renaming* $h(\mathbb{L}) \in \mathbb{S}\langle\langle \mathbb{D}\Sigma^+ \rangle\rangle$ is defined for all $w \in \mathbb{D}\Sigma^+$ by $h(\mathbb{L})(w) = \sum \big(\mathbb{L}(u) \mid u \in \mathbb{D}\Gamma^+ \text{ and } h(u) = w\big)$. For a data series $\mathbb{L} \in \mathbb{S}\langle\langle \mathbb{D}\Sigma^+ \rangle\rangle$, the *inverse renaming* $h^{-1}(\mathbb{L}) \in \mathbb{S}\langle\langle \mathbb{D}\Gamma^+ \rangle\rangle$ is defined for all $u \in \mathbb{D}\Gamma^+$ by $h^{-1}(\mathbb{L})(u) = \mathbb{L}(h(u))$.

Lemma 6.2. *Let Σ, Γ be alphabets, $\mathbb{D}$ a data semiring and $\mathbb{S}$ a commutative data semiring over $\mathbb{D}$, and $h : \Gamma \to \Sigma$ a renaming.*

(a) *If $\mathbb{L} \in \mathbb{S}\langle\langle \mathbb{D}\Gamma^+ \rangle\rangle$ is recognizable, then so is $h(\mathbb{L})$.*
(b) *If $\mathbb{L} \in \mathbb{S}\langle\langle \mathbb{D}\Sigma^+ \rangle\rangle$ is recognizable, then so is $h^{-1}(\mathbb{L})$.*

Proof. (a) The construction of a WRA $\mathcal{A}$ for $h(\mathbb{L})$ is based on the same idea as in Lemma 1 of [21]. Let $\mathcal{A}' = (Q', \mathsf{Reg}', I', T', F', \langle \mathsf{wt}_{\mathsf{trans}}', \mathsf{wt}_{\mathsf{data}}' \rangle)$ be a WRA over the alphabet Γ with $[\![\mathcal{A}']\!] = \mathbb{L}$. We construct a WRA $\mathcal{A} = (Q, \mathsf{Reg}, I, T, F, \langle \mathsf{wt}_{\mathsf{trans}}, \mathsf{wt}_{\mathsf{data}} \rangle)$ over Σ for $h(\mathbb{L})$ as follows: We let $Q = Q' \times \Gamma$, $\mathsf{Reg} = \mathsf{Reg}'$, $I = I' \times \{a_0\}$ for a fixed $a_0 \in \Gamma$ and $F = F' \times \Gamma$. We also let T consist of all transitions $t = ((p, \gamma') \xrightarrow[\phi, \Lambda]{h(\gamma)} (q, \gamma))$ where $t' = (p \xrightarrow[\phi, \Lambda]{\gamma} q)$ is a transition in T'. For such a transition t and a register $r \in \mathsf{Reg}$ we let $\mathsf{wt}_{\mathsf{trans}}(t) = \mathsf{wt}_{\mathsf{trans}}'(t')$ and $\mathsf{wt}_{\mathsf{data}}(t, r) = \mathsf{wt}_{\mathsf{data}}'(t', r)$ for the corresponding transition $t' \in T'$.

Now we show that $[\![\mathcal{A}]\!] = h([\![\mathcal{A}']\!])$. For a data word $u = (\gamma_1, d_1)...(\gamma_n, d_n) \in \mathbb{D}\Gamma^+$, let $w = (h(\gamma_1), d_1)...(h(\gamma_n), d_n) \in \mathbb{D}\Sigma^+$. Consider the run

$$\rho = c_0 \vdash_{t_1,d_1} c_1 \vdash_{t_2,d_2} \dots \vdash_{t_n,d_n} c_n$$

of $\mathcal{A}$ on w such that $t_i = ((p_{i-1}, \gamma_{i-1}) \xrightarrow[\phi_i,\Lambda_i]{h(\gamma_i)} (p_i, \gamma_i))$ for $i \in \{1, 2, ..., n\}$. By the given construction for $\mathcal{A}$, clearly $\rho' = c_0 \vdash_{t'_1,d_1} c_1 \vdash_{t'_2,d_2} \dots \vdash_{t'_n,d_n} c_n$, where $t'_i = (p_i \xrightarrow[\phi_i,\Lambda_i]{\gamma_i} q_i)$ is a run of $\mathcal{A}'$ on $u \in h^{-1}(w)$. Now, we can define the mapping

$$\pi : \mathsf{Run}_{\mathcal{A}}(w) \to \bigcup_{u\in h^{-1}(w)} \mathsf{Run}_{\mathcal{A}'}(u)$$

by $\pi(\rho) = \rho'$. It is easy to see that π is a bijection. In addition, by the definition of the weight functions we have $\mathsf{wt}(\rho') = \mathsf{wt}(\pi(\rho))$. Therefore,

$$\begin{aligned}
[\![\mathcal{A}]\!](w) &= \sum \big(\mathsf{wt}(\rho) \mid \rho \in \mathsf{Run}_{\mathcal{A}}(w)\big) \\
&= \sum \Big(\mathsf{wt}(\pi^{-1}(\rho')) \mid \rho' \in \bigcup_{u\in h^{-1}(w)} \mathsf{Run}_{\mathcal{A}'}(u)\Big) \\
&= \sum_{u\in h^{-1}(w)} \sum \big(\mathsf{wt}(\rho') \mid \rho' \in \mathsf{Run}_{\mathcal{A}'}(u)\big) \\
&= \sum_{u\in h^{-1}(w)} [\![\mathcal{A}']\!](u) = h([\![\mathcal{A}']\!])(w) = h(\mathbb{L})(w)
\end{aligned}$$

which shows that $[\![\mathcal{A}]\!] = h(\mathbb{L})$ and hence $h(\mathbb{L})$ is recognizable.

(b) This case is straightforward: the state space is preserved and every transition of a WRA for $\mathbb{L}$ with label $a \in \Sigma$ is simulated in a WRA for $h^{-1}(\mathbb{L})$ by several transitions with labels from $h^{-1}(a)$. $\square$

7. Weighted existential MSO logic for data words

In this section we introduce weighted existential monadic second-order (wEMSO) logic over data semirings for data words augmented with binary data functions. Later we will show that a suitable fragment of our weighted logic and our weighted register automata model are expressively equivalent. Our new logic will be based on the ideas of Wilke and Bouyer [31,11] of logical characterizations of timed and data automata and on the approach of Droste and Gastin [18] to weighted logic over semirings. As in [10], in order to describe easily boolean properties, we introduce two levels of formulas: boolean and weighted. We operate with the boolean formulas as in the usual logic. On the weighted level, we add weights and binary functions from a data semiring and extend the logical operations by computations in the data semiring.

Let V_1 and V_2 be countable pairwise disjoint sets of first-order and second-order variables. Let $V = V_1 \cup V_2$. Let Σ be an alphabet, $\mathbb{D} = (D, \mathcal{R})$ a data structure with the initial data value $\perp \in D$ and $\mathbb{S} = \langle(S, +, \cdot, \mathbf{0}, \mathbf{1}), \mathcal{F}\rangle$ a data semiring over $\mathbb{D}$. *Weighted first-order logic* $\mathsf{wFO}(\Sigma, \mathbb{D}, \mathbb{S})$ over Σ, $\mathbb{D}$ and $\mathbb{S}$ is defined by the grammar

$$\begin{aligned}
\beta &::= P_a(x) \mid x \le y \mid x \in X \mid R(X, x) \mid \beta \vee \beta \mid \neg\beta \mid \exists x.\beta \\
\varphi &::= \beta \mid s \mid f(x, y) \mid f(\perp, y) \mid \varphi \oplus \varphi \mid \varphi \otimes \varphi \mid \textstyle\bigoplus x.\varphi \mid \textstyle\bigotimes x.\varphi
\end{aligned}$$

where $a \in \Sigma$, $x, y \in V_1$, $X \in V_2$, $R \in \mathcal{R}$, $s \in S$ and $f \in \mathcal{F}$. The formulas β are called *boolean* over Σ and $\mathbb{D}$. Let $\mathsf{Bool}(\Sigma, \mathbb{D})$ denote the set of all boolean formulas. Using boolean formulas in $\mathsf{Bool}(\Sigma, \mathbb{D})$, we define the boolean formulas $x < y$, $x = y$, $x \notin X$, $\beta_1 \wedge \beta_2$, $\forall x.\beta$, $\beta_1 \to \beta_2$ and $\beta_1 \leftrightarrow \beta_2$ as usual.

Weighted existential MSO logic $\mathsf{wEMSO}(\Sigma, \mathbb{D}, \mathbb{S})$ over Σ, $\mathbb{D}$ and $\mathbb{S}$ is defined to be the set of all formulas of the form $\bigoplus X_1 ... \bigoplus X_n.\varphi$ where $n \ge 0$, $X_1, ..., X_n$ are second-order variables and $\varphi \in \mathsf{wFO}(\Sigma, \mathbb{D}, \mathbb{S})$. Given a formula $\psi \in \mathsf{wEMSO}(\Sigma, \mathbb{D}, \mathbb{S})$, the set $\mathsf{Free}(\psi) \subseteq V$ of *free variables* of ψ is defined as usual. We say that ψ is a *sentence* if $\mathsf{Free}(\psi) = \emptyset$.

Let $w = (a_1, d_1)...(a_n, d_n) \in \mathbb{D}\Sigma^+$ be a data word. Let $\mathsf{dom}(w) = \{1, ..., n\}$, the *domain* of w. A *w-assignment* is a mapping $\sigma : V \to \mathsf{dom}(w) \cup \mathcal{P}(\mathsf{dom}(w))$ mapping first-order variables to elements in $\mathsf{dom}(w)$ and second-order variables to subsets of $\mathsf{dom}(w)$. For a first-order variable x and a position $i \in \mathsf{dom}(w)$, the w-assignment $\sigma[x/i]$ is defined on $V \setminus \{x\}$ as σ, and we let $\sigma[x/i](x) = i$. For a second-order variable X and $I \subseteq \mathsf{dom}(w)$, the w-assignment $\sigma[X/I]$ is defined similarly. Given a formula $\beta \in \mathsf{Bool}(\Sigma, \mathbb{D})$ and a w-assignment σ, the satisfaction relation $(w, \sigma) \models \beta$ is defined by induction on the structure of β as usual. For new formulas of the form $R(X, x)$, the semantics is defined in a similar manner to relative distance formulas of Wilke [31]: for the closest position y of the set X to the position x from the left side, the data values at the positions x and y must be in relation R (if such a position y does not exist, we take $\perp$ as the data value for y). If we assume

Table 1
The semantics of wEMSO-formulas.

$$\begin{aligned}
[\![\beta]\!]_V(w,\sigma) &= \begin{cases} \mathbf{1}, & \text{if } (w,\sigma) \models \beta, \\ \mathbf{0}, & \text{otherwise} \end{cases} \\
[\![s]\!]_V(w,\sigma) &= s \\
[\![f(x,y)]\!]_V(w,\sigma) &= f(d_{\sigma(x)}, d_{\sigma(y)}) \\
[\![f(\perp,y)]\!]_V(w,\sigma) &= f(\perp, d_{\sigma(y)}) \\
[\![\varphi_1 \oplus \varphi_2]\!]_V(w,\sigma) &= [\![\varphi_1]\!]_V(w,\sigma) + [\![\varphi_2]\!]_V(w,\sigma) \\
[\![\varphi_1 \otimes \varphi_2]\!]_V(w,\sigma) &= [\![\varphi_1]\!]_V(w,\sigma) \cdot [\![\varphi_2]\!]_V(w,\sigma) \\
[\![\bigoplus x.\varphi]\!]_V(w,\sigma) &= \sum_{i \in \mathsf{dom}(w)} [\![\varphi]\!]_{V\cup\{x\}}(w,\sigma[x/i]) \\
[\![\bigotimes x.\varphi]\!]_V(w,\sigma) &= \prod_{i \in \mathsf{dom}(w)} [\![\varphi]\!]_{V\cup\{x\}}(w,\sigma[x/i]) \\
[\![\bigoplus X.\varphi]\!]_V(w,\sigma) &= \sum_{I \subseteq \mathsf{dom}(w)} [\![\varphi]\!]_{V\cup\{X\}}(w,\sigma[X/I])
\end{aligned}$$

that X is a set of positions where some register r of a register automaton was updated, then $R(X,x)$ describes that the data value stored in the register r and the data value at the position x are in relation R. Formally, we let $(w,\sigma) \models R(X,x)$ iff, letting $d_0 = \perp$, for the greatest position $i \in \sigma(X) \cup \{0\}$ with $i < \sigma(x)$ we have $(d_i, d_{\sigma(x)}) \in R$.

Since the satisfaction relation depends only on values of free variables, we will abuse notation and also write $(w, \sigma|_U) \models \beta$ for any $U \subseteq V$ with $\mathsf{Free}(\beta) \subseteq U$.

Let $\mathbb{D}\Sigma_V^+$ denote the set of all pairs (w,σ) where $w \in \mathbb{D}\Sigma^+$ and σ is a w-assignment. Given a formula $\psi \in$ wEMSO$(\Sigma,\mathbb{D},\mathbb{S})$, the *semantics* of ψ is the mapping $[\![\psi]\!]_V : \mathbb{D}\Sigma_V^+ \to S$ defined for all $(w,\sigma) \in \mathbb{D}\Sigma_V^+$ with $w = (a_1,d_1)\ldots(a_n,d_n)$ as shown in Table 1. Note that the semantics of a formula $f(x,y)$ applies the function $f \in \mathcal{F}$ of two variables to the pair of the data values at the positions x and y. If ψ is a sentence, then we can ignore the w-assignments in the definition of the semantics and consider it as the data series $[\![\psi]\!] : \mathbb{D}\Sigma^+ \to S$. When $V = \mathsf{Free}(\psi)$, $[\![\psi]\!]_V$ is also abbreviated as $[\![\psi]\!]$.

The following lemma shows that for each formula $\psi \in$ wEMSO$(\Sigma,\mathbb{D},\mathbb{S})$, the semantics for the different sets V of variables containing $\mathsf{Free}(\psi)$ are consistent with each other. Since the proof is similar to the existing result for different weighted structures (see [19]), we do not give the proof here.

Lemma 7.1. *Let $\psi \in$ wEMSO$(\Sigma,\mathbb{D},\mathbb{S})$ and V a finite set of variables with $\mathsf{Free}(\psi) \subseteq V$. Then $[\![\psi]\!]_V(w,\sigma) = [\![\psi]\!](w, \sigma \restriction_{\mathsf{Free}(\psi)})$ for any valid $(w,\sigma) \in \mathbb{D}\Sigma_V^+$. In addition, $[\![\psi]\!]$ is recognizable if and only if $[\![\psi]\!]_V$ is recognizable.*

Example 7.2. Consider the data series $\mathbb{L}$ of Example 4.3. Note that $\mathbb{L}$ is definable by the wEMSO$(\Sigma,\mathbb{D},\mathbb{S})$-sentence

$$\bigoplus X.\bigoplus x.[(X = \{x\}) \otimes \bigotimes (y > x).([R(X,y) \otimes 1] \oplus \neg R(X,y))]$$

where $X = \{x\}$ is an abbreviation for the formula $\forall z.(z \in X \leftrightarrow z = x)$, $\bigotimes(y > x).\varphi$ abbreviates the formula $\bigotimes y.((y > x \otimes \varphi) \oplus (y \le x))$, and $R = (=_D)$.

8. Restricted wEMSO

Our goal is to study the connection between WRA and our new weighted logic on data words. Similarly to the result of [18], the unrestricted use of formulas of the form $\bigotimes x.\varphi$ leads to unrecognizable data series. Below we give a further example of unrecognizability which is specific for wEMSO on data words.

Example 8.1. Let $\Sigma = \{a\}$ be a singleton alphabet and $\mathbb{D}$ be a data structure with the data domain $\mathbb{N}$. Consider the data semiring $\mathbb{S} = \langle \mathsf{Arc}, \mathcal{F} \rangle$ where Arc is the arctic semiring of Example 4.2 (a) and $\mathcal{F} = \{0^{\mathbb{N}\times\mathbb{N}}, f\}$ where $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ is defined by $f(n,n') = n$ for all $n,n' \in \mathbb{N}$. Consider the sentence $\varphi \in$ wEMSO$(\Sigma,\mathbb{D},\mathbb{S})$ defined by $\varphi = \bigoplus x.\bigoplus y.f(x,y)$. Note that, for every $n \in \mathbb{N}$ and the data word $w_n = (a,n) \in \mathbb{D}\Sigma^+$ of length 1, we have $[\![\varphi]\!](w_n) = n$. Now suppose that there exists a WRA $\mathcal{A}$ over Σ, $\mathbb{D}$ and $\mathbb{S}$ with $[\![\mathcal{A}]\!] = [\![\varphi]\!]$. When the length of a data word is one, the weight computed by $\mathcal{A}$ cannot exceed a constant because $\mathsf{wt}_{\mathsf{data}}(t,r)(\vartheta(r),d) \le \max\{0^{\mathbb{N}\times\mathbb{N}}(\vartheta(r),d), f(\vartheta(r),d)\} \le \vartheta(r)$ for some register r by assumption on f and because $\vartheta(r)$ must be the initial value of r. In other words, there exists a constant $M \in \mathbb{N}$ such that $[\![\mathcal{A}]\!](w_n) \le M$ for all $n \in \mathbb{N}$. A contradiction.

Now we will investigate a fragment of wEMSO which is expressively equivalent to WRA. Example 8.1 shows that the use of binary functions $f(x,y)$ is not suitable for a logical characterization of WRA. We follow the approach of Wilke [31] for a logical characterization of timed automata where the use of every expressive time distance predicate $\mathsf{dist}(x,y) \bowtie k$ with $x,y \in V_1$, $\bowtie \in \{<,\le,=,\ge,>\}$ was restricted to relative time distance predicate $\mathsf{dist}(X,y) \bowtie k$ with $X \in V_2$ (note that the relative time distance predicates correspond to the formulas $R(X, y)$ in our logic $\mathsf{Bool}(\Sigma, \mathbb{D})$). We replace the formulas $f(x, y)$ by the formulas $f(X, y)$ whose semantics is defined in a similar manner as for $R(X, y)$, as follows:

For $f \in \mathcal{F}$, a first-order variable x and a second-order variable X, let $f(X, x)$ denote the $\mathsf{wFO}(\Sigma, \mathbb{S}, \mathbb{D})$-formula

$$\bigoplus y.(\beta(X, x, y) \otimes f(y, x)) \oplus (\beta'(X, x) \otimes f(\bot, x))$$

where $\beta(X, x, y)$ is the boolean formula $y \in X \wedge y < x \wedge \forall z.([y < z \wedge z < x] \rightarrow z \notin X)$ and $\beta'(X, x)$ is the boolean formula $\forall y.(y < x \rightarrow y \notin X)$. Then, for all $(w, \sigma) \in \mathbb{D}\Sigma^+$ with $w = (a_1, d_1)...(a_n, d_n)$, we have $[\![f(X, x)]\!](w, \sigma) = f(d_i, d_{\sigma(x)})$ where $i \in \sigma(X) \cup \{0\}$ is the greatest position with $i < \sigma(x)$ and $d_0 = \bot$.

The following example shows that the use of our new logical operator $f(X, x)$ in the scope of a weighted quantifier $\bigotimes y$ with $y \neq x$ also goes beyond recognizability by WRA.

Example 8.2. Let Σ and $\mathbb{D}$ be defined as in Example 8.1. Consider the data semiring $\mathbb{S} = \langle \mathsf{Arc}, \mathcal{F}\rangle$ with $\mathcal{F} = \{0^{\mathbb{N}\times\mathbb{N}}, f\}$ where f is defined for all $n, n' \in \mathbb{N}$ by $f(n, n') = n'$. Consider the sentence $\varphi \in \mathsf{wEMSO}(\Sigma, \mathbb{D}, \mathbb{S})$ defined by $\varphi = \bigoplus X.\bigoplus x.(\beta(X, x) \otimes \bigotimes y.f(X, x))$ where $y \neq x$ and the boolean formula $\beta(X, x)$ is defined as $\forall y.y \leq x \wedge \forall y.(y \in X \leftrightarrow \forall z.y \leq z)$. Note that $\beta(X, x)$ describes that x is the last position of a data word and X is the singleton set containing the first position of a data word. For any $n \geq 2$ and the data word $w_n = (a, 0)^{n-1}(a, n) \in \mathbb{D}\Sigma^+$, we have $[\![\varphi]\!](w_n) = n^2$. Suppose that there exists a WRA $\mathcal{A}$ over Σ, $\mathbb{D}$ and $\mathcal{S}$ with $[\![\mathcal{A}]\!] = [\![\varphi]\!]$. Let $\mathcal{A}$ be defined as in Definition 4.1. Let $W \in \mathbb{N}$ be such that $\mathsf{wt}_{\mathsf{trans}}(t) \leq W$ for all transitions $t \in T$. Let $R = |\mathsf{Reg}|$. Note that, for every transition $t \in T$, every register $r \in \mathsf{Reg}$, and every pair of data values $d, d' \in \mathbb{N}$: $\mathsf{wt}_{\mathsf{data}}(t, r)(d, d') \leq \max\{0^{\mathbb{N}\times\mathbb{N}}(d, d'), f(d, d')\} \leq d'$. Then, by (1), for every switch $c \vdash_{t,d} c'$ we have $\mathsf{wt}(c \vdash_{t,d} c') \leq RW \cdot d$. In particular, this means that $\mathsf{wt}(c \vdash_{t,0} c') \leq 0$. Then, for every run $\rho \in \mathsf{Run}_{\mathcal{A}}(w_n)$, the weight of ρ is not greater than the weight of the last configuration switch in ρ, i.e., $\mathsf{wt}(\rho) \leq RW \cdot n$. Hence, there exists a constant $M = RW \in \mathbb{N}$ such that $[\![\mathcal{A}]\!](w_n) \leq M \cdot n$ for all $n \geq 2$. A contradiction.

Now, based on the explanations above, we will define the desired fragment of wEMSO for WRA. Note that, similarly to [18], we must restrict the use of $\bigotimes x$ to simplified formulas without weighted quantifiers. Let x be a first-order variable. We say that a formula $\gamma \in \mathsf{wFO}(\Sigma, \mathbb{D}, \mathbb{S})$ is *almost boolean over* x if it is derived by the grammar

$$\gamma \ ::= \ \beta \mid s \mid f(X, x) \mid \gamma \oplus \gamma \mid \gamma \otimes \gamma$$

where $\beta \in \mathsf{Bool}(\Sigma, \mathbb{D})$, $s \in S$, $f \in \mathcal{F}$ and X is a second-order variable. Notice that the variable x in $f(X, x)$ is fixed. Let $\mathsf{aBool}[x](\Sigma, \mathbb{D}, \mathbb{S})$ denote the set of all almost boolean formulas over x. Then, *restricted weighted first-order logic* $\mathsf{wFO}^{\mathsf{res}}(\Sigma, \mathbb{D}, \mathbb{S}) \subseteq \mathsf{wFO}(\Sigma, \mathbb{D}, \mathbb{S})$ is defined by the grammar

$$\varphi \ ::= \ \beta \mid s \mid f(X, x) \mid \varphi \oplus \varphi \mid \varphi \otimes \varphi \mid \bigoplus x.\varphi \mid \bigotimes x.\gamma_x$$

where $\beta \in \mathsf{Bool}(\Sigma, \mathbb{D})$, $s \in S$, $f \in \mathcal{F}$, x is a first-order variable, X is a second-order variable and $\gamma_x \in \mathsf{aBool}[x](\Sigma, \mathbb{D}, \mathbb{S})$. In other words, whenever $\bigotimes x.\gamma_x$ is in $\mathsf{wFO}^{\mathsf{res}}(\Sigma, \mathbb{D}, \mathbb{S})$, each subformula $f(y, X)$ of γ_x must have $y = x$. *Restricted weighted existential MSO logic* $\mathsf{wEMSO}^{\mathsf{res}}(\Sigma, \mathbb{D}, \mathbb{S}) \subseteq \mathsf{wEMSO}(\Sigma, \mathbb{D}, \mathbb{S})$ is defined to be the set of all formulas of the form $\bigoplus X_1 ... \bigoplus X_n.\varphi$ where $n \geq 0$, $X_1, ..., X_n$ are second-order variables and $\varphi \in \mathsf{wFO}^{\mathsf{res}}(\Sigma, \mathbb{D}, \mathbb{S})$.

We say that a fragment $\mathsf{Frag} \subseteq \mathsf{wEMSO}(\Sigma, \mathbb{D}, \mathbb{S})$ is *expressively equivalent to WRA* if, for every data series $\mathbb{L} : \mathbb{D}\Sigma^+ \rightarrow S$, $\mathbb{L}$ is recognizable by a WRA over Σ, $\mathbb{D}$ and $\mathbb{S}$ iff $\mathbb{L}$ is definable by a sentence in Frag.

Theorem 8.3. *Let Σ be an alphabet, $\mathbb{D}$ a data structure, and $\mathbb{S}$ a commutative data semiring over $\mathbb{D}$. Then $\mathsf{wEMSO}^{\mathsf{res}}(\Sigma, \mathbb{D}, \mathbb{S})$ is expressively equivalent to WRA.*

Remark 8.4. Note that a logical characterization of weighted timed automata was given in [30], Theorems 30 and 39. The structure of the syntactically restricted logical fragment of our Theorem 8.3 has the following advantages. First, we do not restrict the use of weighted conjunctions $\varphi_1 \otimes \varphi_2$ and we only need to restrict the use of $\bigotimes$-quantifier to formulas without weighted quantifiers. Second, in the syntactically restricted weighted logic of [30] over non-idempotent semirings (cf. Theorem 39 of [30]), the use of $\forall$-quantifier is restricted to formulas which define timed languages of *bounded variability*, a notion introduced by Wilke [31]. Intuitively, the variability of a timed word corresponds to the maximum number of events that may occur within one time unit. Since we deal with register automata, the same notion cannot be applied here, and therefore we need a new technique. We will prove Theorem 8.3 in Section 10.

9. Visibly register automata: a determinizable class of register automata

Two of the main difficulties of the proof of Theorem 8.3 are that register automata are neither determinizable nor closed under complement. The goal of this section is to investigate a subclass of register automata which can be applied in the proof of our logical characterization result. Our determinizable subclass could be also of independent interest. The idea is to make the register updates visible in transition labels. Note that a similar idea was applied in *event-clock automata* [3] and *visibly pushdown automata* [4]. Since the class of languages recognizable by event-clock automata is not closed under renamings of input symbols [3], this model is not suitable for the translation of logical formulas. We use a slightly different approach. Recall that in event-clock automata, the input alphabet is arbitrary and with every letter a clock is associated. In contrast, in our model we take an arbitrary set of registers and an input alphabet is defined depending on this set of registers.

Throughout all of this section, we fix an alphabet Σ, a data structure $\mathbb{D} = (D, \mathcal{R})$ and a finite set of *registers* Reg. Let $\Sigma^{\langle \mathsf{Reg} \rangle}$ denote the alphabet $\Sigma \times \{0,1\}^{\mathsf{Reg}}$.

Definition 9.1. A *visibly register automaton* over Σ, $\mathbb{D}$ and Reg is a register automaton $\mathcal{A}$ over $\Sigma^{\langle \mathsf{Reg} \rangle}$ and $\mathbb{D}$ with the set of registers Reg such that, for every transition $q \xrightarrow[\phi,\Lambda]{(a,\theta)} q'$ of $\mathcal{A}$ where q, q' are states, $a \in \Sigma$, $\theta \in \{0,1\}^{\mathsf{Reg}}$, $\Lambda \subseteq \mathsf{Reg}$ and ϕ is a register guard, we have $\Lambda = \{r \in \mathsf{Reg} \mid \theta(r) = 1\}$.

Note that $\mathcal{A}$ recognizes the language $\mathcal{L}(\mathcal{A}) \subseteq \mathbb{D}(\Sigma^{\langle \mathsf{Reg} \rangle})^+$. Note also that visibly register automata form a subclass of register automata.

We say that a register automaton $\mathcal{A}$ over Σ and $\mathbb{D}$ is *deterministic* if it has a single initial state and whenever $p \xrightarrow[\phi,\Lambda]{a} q$ and $p \xrightarrow[\phi',\Lambda']{a} q'$ are two distinct transitions of $\mathcal{A}$, ϕ and ϕ' are mutually exclusive, i.e., for all registers valuations ϑ and all data values $d \in D$, we have $(\vartheta, d) \not\models \phi \wedge \phi'$. We call $\mathcal{A}$ *complete* if for all states p of $\mathcal{A}$, all letters $a \in \Sigma$, all register valuations ϑ and all data values $d \in D$, there exists a transition $p \xrightarrow[\phi,\Lambda]{a} q$ of $\mathcal{A}$ with $(\vartheta, d) \models \phi$.

The following lemma shows that the class of visibly register automata is determinizable.

Theorem 9.2. *Let $\mathcal{A}$ be a visibly register automaton over Σ, $\mathbb{D}$ and* Reg*. Then, there exists a deterministic and complete visibly register automaton $\mathcal{A}'$ over Σ, $\mathbb{D}$ and* Reg *such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$.*

Proof. The proof follows a similar idea as the proof of Theorem 1 of [3] about determinization of event-clock automata. Let $\mathcal{A} = (Q, \mathsf{Reg}, I, T, F)$ be a visibly register automaton over the alphabet Σ recognizing the data language $\mathcal{L}(\mathcal{A})$. From $\mathcal{A}$ we construct a visibly register automaton $\mathcal{A}' = (Q', \mathsf{Reg}, I', T', F')$ with $\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A})$ as follows:

- $Q' = \mathcal{P}(Q)$, $I' = \{I\}$, $F' = \{U \subseteq Q \mid U \cap F \neq \emptyset\}$.
- T' is defined as follows. Suppose that $U \in \mathcal{P}(Q)$ and $(a, \theta) \in \Sigma^{\langle \mathsf{Reg} \rangle}$. Let $(t_i)_{i \in \{1,\dots,m\}}$ be an enumeration of the set of all transitions in T with label (a, θ) starting in a state from U. For each $i \in \{1, \dots, m\}$ let $t_i = \big(p_i \xrightarrow[\phi_i,\Lambda]{(a,\theta)} q_i\big)$ with $\Lambda = \{r \in \mathsf{Reg} \mid \theta(r) = 1\}$. For any subset $J \subseteq \{1, \dots, m\}$, we add to T' the transition $t' = \big(U \xrightarrow[\phi_J,\Lambda]{(a,\theta)} U'\big)$ where $U' = \{q_i \mid i \in J\}$ with $\phi_J = \bigwedge_{i \in J} \phi_i \wedge \bigwedge_{i \notin J} \neg \phi_i$.

We show that

(i) $\mathcal{A}'$ is deterministic and complete;
(ii) $\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A})$.

First we show (i). The completeness of the automaton $\mathcal{A}'$ is implied by the given construction. Now we show that $\mathcal{A}'$ is deterministic. Assume that $t'_1 = \big(U \xrightarrow[\phi_{J_1},\Lambda]{(a,\theta)} U'_1\big) \in T'$ and $t'_2 = \big(U \xrightarrow[\phi_{J_2},\Lambda]{(a,\theta)} U'_2\big) \in T'$ with $t'_1 \neq t'_2$. Then, for the subsets $J_1, J_2 \subseteq \{1, 2, \dots, m\}$ we have $J_1 \neq J_2$ and hence ϕ_{J_1} and ϕ_{J_2} are mutually exclusive. Therefore $\mathcal{A}'$ is deterministic.

Now we show (ii). Let $u = ((a_1, \theta_1), d_1)\dots((a_n, \theta_n), d_n) \in \mathcal{L}(\mathcal{A})$. Then there exists a run

$$\rho = \langle p_0, \vartheta_0 \rangle \vdash_{t_1,d_1} \langle p_1, \vartheta_1 \rangle \vdash_{t_2,d_2} \dots \vdash_{t_n,d_n} \langle p_n, \vartheta_n \rangle \tag{2}$$

of $\mathcal{A}$ on u where $t_i = \big(p_{i-1} \xrightarrow[\phi_i,\Lambda_i]{(a_i,\theta_i)} p_i\big)$ with $\Lambda_i = \{r \in \mathsf{Reg} \mid \theta_i(r) = 1\}$ for all $i \in \{1, \dots, n\}$. Note that $\langle p_0, \vartheta_0 \rangle$ is the initial configuration and all the register valuations $\vartheta_1, \dots, \vartheta_n$ are uniquely determined by ϑ_0 and u. Now consider the run

$$\rho' = \langle U_0, \vartheta_0 \rangle \vdash_{t'_1,d_1} \langle U_1, \vartheta_1 \rangle \vdash_{t'_2,d_2} \dots \vdash_{t'_n,d_n} \langle U_n, \vartheta_n \rangle \tag{3}$$

where $t'_i = \big(U_{i-1} \xrightarrow[\phi_{J_i},\Lambda_i]{(a_i,\theta_i)} U_i\big)$ with $\Lambda_i = \{r \in \mathsf{Reg} \mid \theta_i(r) = 1\}$ for all $i \in \{1,\dots,n\}$. Note that $U_0 = I' = \{I\}$ and each U_i is obtained based on the idea in the construction explained above. By the definition of the transitions t'_i, $i \in \{1,\dots,n\}$, we can see that $p_i \in U_i$ and therefore $t'_i \in T'$. In addition, since $p_n \in F$ we have $F \cap U_n \neq \emptyset$ and so $U_n \in F'$. Thus, ρ' is a run of $\mathcal{A}'$ on u. Hence, $u \in \mathcal{L}(\mathcal{A}')$. It implies that $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{A}')$.

Conversely, let $u = ((a_1,\theta_1),d_1)\dots((a_n,\theta_n),d_n) \in \mathcal{L}(\mathcal{A}')$. Then there exists a successful run of the form (3) on w where $t'_i = \big(U_{i-1} \xrightarrow[\phi_{J_i},\Lambda_i]{(a_i,\theta_i)} U_i\big)$ with $\Lambda_i = \{r \in \mathsf{Reg} \mid \theta_i(r) = 1\}$ for all $i \in \{1,\dots,n\}$. Then, there exist $p_0 \in I$, $p_1,\dots,p_{n-1} \in Q$ and $p_n \in F$ such that $t_i = \big(p_{i-1} \xrightarrow[\phi_i,\Lambda_i]{(a_i,\theta_i)} p_i\big) \in T$, for $i \in \{1,\dots,n\}$. Thus, we can define a run of the form (2) of $\mathcal{A}$ on u. Hence $u \in \mathcal{L}(\mathcal{A})$. Therefore, $\mathcal{L}(\mathcal{A}') \subseteq \mathcal{L}(\mathcal{A})$. Hence (ii) holds true. $\square$

Using Theorem 9.2 for the complement, it is not difficult to verify the closure properties for visibly register automata stated in the next lemma.

Lemma 9.3. *The class of data languages recognizable by visibly register automata over Σ, $\mathbb{D}$ and* Reg *is closed under union, intersection and complement.*

Proof. For union, we apply the standard automata-theoretic disjoint union construction. Now we give the proof for the complement. By Theorem 9.2, there exists a deterministic and complete visibly register automaton $\mathcal{A} = (Q, \mathsf{Reg}, I, T, F)$ over Σ such that $\mathcal{L}(\mathcal{A}) = \mathcal{L} \subseteq \mathbb{D}(\Sigma^{\langle \mathsf{Reg}\rangle})^+$. Then, we can define the visibly register automaton $\mathcal{A}' = (Q, \mathsf{Reg}, I, T, Q \setminus F)$. Since $\mathcal{A}$ is deterministic, for all $u \in \mathcal{L}$, there exists only one successful run ρ of $\mathcal{A}$ on u which ends in a state in F. It means, $u \notin \mathcal{L}(\mathcal{A}')$. In addition, since $\mathcal{A}$ is complete, for all $u \in \mathbb{D}\Sigma^+ \setminus \mathcal{L}$ there exists a run which ends in a state from $Q \setminus F$. Hence, $u \in \mathcal{L}(\mathcal{A}')$. It implies that $\mathcal{L}(\mathcal{A}') = \mathbb{D}(\Sigma^{\langle \mathsf{Reg}\rangle})^+ \setminus \mathcal{L}$. Finally, the claim for the intersection is implied by the closure under complement and union. $\square$

Let Γ, Δ be alphabets and $h : \Gamma \to \Delta$ a renaming. We extend h to the mapping $h : \mathbb{D}(\Gamma^{\langle \mathsf{Reg}\rangle})^+ \to \mathbb{D}(\Delta^{\langle \mathsf{Reg}\rangle})^+$ by applying h only to the Σ-components of data words, i.e., for a data word $u = ((a_1,\theta_1),d_1)\dots((a_n,\theta_n),d_n) \in \mathbb{D}(\Gamma^{\langle \mathsf{Reg}\rangle})^+$ we can define the data word $h(u) = ((h(a_1),\theta_1),d_1)\dots((h(a_n),\theta_n),d_n) \in \mathbb{D}(\Delta^{\langle \mathsf{Reg}\rangle})^+$. Then for a data language $\mathcal{L} \subseteq \mathbb{D}(\Gamma^{\langle \mathsf{Reg}\rangle})^+$, the renaming by

$$h(\mathcal{L}) = \{h(u) \mid u \in \mathcal{L}\} \subseteq \mathbb{D}(\Delta^{\langle \mathsf{Reg}\rangle})^+.$$

Similarly, for a data language $\mathcal{L}' \subseteq \mathbb{D}(\Delta^{\langle \mathsf{Reg}\rangle})^+$, we define the inverse renaming by

$$h^{-1}(\mathcal{L}') = \{u \in \mathbb{D}(\Gamma^{\langle \mathsf{Reg}\rangle})^+ \mid h(u) \in \mathcal{L}'\} \subseteq \mathbb{D}(\Gamma^{\langle \mathsf{Reg}\rangle})^+.$$

Lemma 9.4. *Let Γ, Δ be alphabets and $h : \Gamma \to \Delta$ a renaming.*

(a) *If $\mathcal{L} \subseteq \mathbb{D}(\Gamma^{\langle \mathsf{Reg}\rangle})^+$ is recognizable by a visibly register automaton over Γ, $\mathbb{D}$ and* Reg*, then $h(\mathcal{L})$ is recognizable by a visibly register automaton over Δ, $\mathbb{D}$ and* Reg*.*
(b) *If $\mathcal{L} \subseteq \mathbb{D}(\Delta^{\langle \mathsf{Reg}\rangle})^+$ is recognizable by a visibly register automaton over Δ, $\mathbb{D}$ and* Reg*, then $h^{-1}(\mathcal{L})$ is recognizable by a visibly register automaton over Γ, $\mathbb{D}$ and* Reg*.*

Proof. Let Γ and Δ be alphabets, and $h : \Gamma \to \Delta$ a renaming.

(a) Let $\mathcal{A} = (Q, \mathsf{Reg}, I, T, F)$ be a visibly register automaton over Γ, $\mathbb{D}$ and Reg recognizing the data language $\mathcal{L} \subseteq \mathbb{D}(\Gamma^{\langle \mathsf{Reg}\rangle})^+$. We construct a visibly register automaton $\mathcal{A}' = (Q, \mathsf{Reg}, I, T', F)$ with the same state space Q over Δ, $\mathbb{D}$ and Reg recognizing $h(\mathcal{L})$ as follows: the set of transitions T' consists of all transitions $t' = \big(p \xrightarrow[\phi,\Lambda]{(h(a),\theta)} q\big)$ where $t = \big(p \xrightarrow[\phi,\Lambda]{(a,\theta)} q\big)$ is a transition in T. Now one can see that $\mathcal{L}(\mathcal{A}') = h(\mathcal{L})$.

(b) Let $\mathcal{B} = (Q, \mathsf{Reg}, I, T, F)$ be a visibly register automaton over Δ, $\mathbb{D}$ and Reg recognizing the data language $\mathcal{L} \subseteq \mathbb{D}(\Delta^{\langle \mathsf{Reg}\rangle})^+$. We obtain a visibly register automaton $\mathcal{B}' = (Q, \mathsf{Reg}, I, T', F)$ over Γ, $\mathbb{D}$ and Reg recognizing $h^{-1}(\mathcal{L}) \subseteq \mathbb{D}(\Gamma^{\langle \mathsf{Reg}\rangle})^+$ by defining T' as follows: the set of transitions T' consists of all transitions $t' = \big(p \xrightarrow[\phi,\Lambda]{(a,\theta)} q\big)$ where $t = \big(p \xrightarrow[\phi,\Lambda]{(h(a),\theta)} q\big)$ is a transition in T. It can be seen that $\mathcal{L}(\mathcal{B}') = h^{-1}(\mathcal{L})$. $\square$

Now let $\beta \in \mathsf{Bool}(\Sigma, \mathbb{D})$ be a formula and Reg the set of all second-order variables X occurring in a subformula of β of the form $R(X, x)$. Using the standard encoding of free variables, we encode the set of all pairs $(w, \sigma|_{\mathsf{Free}(\beta)})$ such that $(w,\sigma) \in \mathbb{D}\Sigma^+_V$ and $(w,\sigma) \models \beta$ as the data language $\mathcal{L}(\varphi) \subseteq \mathbb{D}(\Gamma^{\langle \mathsf{Reg}\rangle})^+$ where $\Gamma = \Sigma \times \{0,1\}^{\mathsf{Free}(\beta)\setminus \mathsf{Reg}}$. Using Lemmas 9.3 and 9.4 and Theorem 9.2, one can show by induction the following theorem.

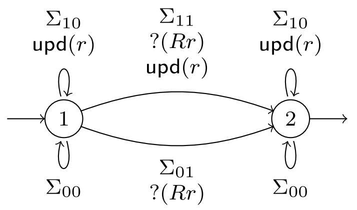

Fig. 4. The visibly register automaton $\mathcal{A}$ for the formula $R(X,x)$.

Theorem 9.5. *Let $\beta \in \mathsf{Bool}(\Sigma,\mathbb{D})$ be a formula and Reg the set of all second-order variables X occurring in a subformula of β of the form $R(X,x)$. Then, there exists a deterministic visibly register automaton $\mathcal{A}$ over Γ, $\mathbb{D}$ and Reg such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\beta)$.*

Proof. Let $\beta \in \mathsf{Bool}(\Sigma,\mathbb{D})$ and Reg the set of all second-order variables X occurring in a subformula of β of the form $R(X,x)$. We show that for each formula β the language $\mathcal{L}(\beta)$ can be recognized by a deterministic visibly register automaton $\mathcal{A}$ over the extended alphabet $\Gamma^{\langle \mathsf{Reg}\rangle}$ where $\Gamma = \Sigma \times \{0,1\}^{\mathsf{Free}(\beta)\setminus\mathsf{Reg}}$, $\mathbb{D}$ and Reg.

- If β is one of the formulas $P_a(x)$, $x \le y$ or $x \in X$, we can construct a deterministic visibly register automaton $\mathcal{A}_\beta$ using the same approach as for the formulas in MSO.
- The automaton $\mathcal{A}_{R(X,x)}$ for the formula $\beta = R(X,x)$ over the alphabet $\Gamma^{\langle X\rangle}$ where $\Gamma = \Sigma \times \{0,1\}^{\{x\}}$ is depicted in Fig. 4. Here, Σ_{ij} with $i,j \in \{0,1\}$ means any letter in $\Gamma^{\langle X\rangle}$ whose X-component is i and x-component is j and $r = X$ is the register of $\mathcal{A}_{R(X,x)}$. This automaton can guess at which step the last transition labeled with a letter in $\Gamma^{\langle X\rangle}$ with a 1 in the X-row is taken and updates the register r at this transition. Then it checks that whenever a transition is labeled with a letter including a 1 in the x-row, the data value loaded into the register at the last position labeled with a letter including a 1 in the X-row satisfies the guard Rr.
- If β is one of the formulas $\beta \vee \beta$, $\neg\beta$ or $\exists x.\beta$, we use the standard approach as for the formulas in MSO, applying Lemmas 9.3 and 9.4. □

10. Definability equals recognizability

In this section, we give the proof of Theorem 8.3. First, we show that recognizability implies definability.

Theorem 10.1. *Let $\mathcal{A}$ be a WRA over Σ, $\mathbb{D}$ and $\mathbb{S}$. Then there exists a sentence $\varphi \in \mathsf{wEMSO}^{\mathsf{res}}(\Sigma,\mathbb{D},\mathbb{S})$ such that $[\![\varphi]\!] = [\![\mathcal{A}]\!]$.*

Proof. Let $\mathcal{A} = (Q, \mathsf{Reg}, I, T, F, \langle \mathsf{wt}_{\mathsf{trans}}, \mathsf{wt}_{\mathsf{data}}\rangle)$. First, using $\mathsf{Bool}(\Sigma,\mathbb{D})$-formulas, we describe runs of $\mathcal{A}$. For this, we fix an enumeration $(t_i)_{1\le i\le n}$ of T and an enumeration $(r_j)_{1\le j\le m}$ of Reg. Assume that $t_i = \left(q_i \xrightarrow[\phi_i,\Lambda_i]{a_i} q_i'\right)$, for $1 \le i \le n$. Then, we associate with every transition t_i a second-order variable X_i which keeps track of positions where t_i is taken and associate with every register r_j a second-order variable Y_j which keeps track of positions where r_j is updated. Now, a run of $\mathcal{A}$ can be described using the formula $\beta \in \mathsf{Bool}(\Sigma,\mathbb{D})$ with the set of free variables $\mathsf{Free}(\beta) = \{X_1,\dots,X_n,Y_1,\dots,Y_m\}$ defined by

$$\beta = \mathsf{Partition} \wedge \mathsf{Labels} \wedge \mathsf{Initial} \wedge \mathsf{Consistent} \wedge \mathsf{Final} \wedge \mathsf{Registers}$$

where

- $\mathsf{Partition} = \forall x. \bigvee_{1\le i\le n}((x \in X_i) \wedge \bigwedge_{j\ne i}(x \notin X_j))$ demands that values of variables $X_1,\dots,X_n$ form a partition of the domain of an input data word.
- $\mathsf{Labels} = \forall x. \bigwedge_{a\in\Sigma}(P_a(x) \to \bigvee(x \in X_i \mid 1 \le i \le n,\ a_i = a))$ demands that the transition labels of a run are compatible with an input data word.
- $\mathsf{Initial} = \exists x.(\forall y.(x \le y) \wedge \bigvee(x \in X_i \mid 1 \le i \le n,\ q_i \in I))$ demands that a run starts in an initial state from I.
- $\mathsf{Consistent} = \forall x.\forall y.((y = x+1) \to \bigvee(x \in X_i \wedge y \in X_j \mid 1 \le i,j \le n, q_i' = q_j))$ demands that every two successive transitions are matching, i.e., they are connected via the same state.
- $\mathsf{Final} = \exists x.(\forall y.(y \le x) \wedge \bigvee(x \in X_i \mid 1 \le i \le n,\ q_i \in F))$ demands that a run ends in a final state from F.
- For all $x \in V_1$ and $\phi \in \mathsf{Guard}(\mathsf{Reg},\mathbb{D})$, let $f_x(\phi) \in \mathsf{Bool}(\Sigma)$ be obtained from ϕ by replacing every subformula Rr_j with $R \in \mathcal{R}$ and $j \in \{1,\dots,m\}$ by $R(Y_j,x)$. Then $\mathsf{Registers} = \forall x. \bigvee_{1\le i\le n}(x \in X_i \wedge \bigwedge_{j\in\Lambda_i} x \in Y_j \wedge \bigwedge_{j\in\mathsf{Reg}\setminus\Lambda_i} x \notin Y_j \wedge f_x(\phi_i))$ checks that whenever a transition t_i is taken (i.e. $x \in X_i$), each register $j \in \Lambda_i$ is updated (i.e., $x \in Y_j$), and each register $j \notin \Lambda_i$ is not updated (i.e., $x \notin Y_j$), and a register valuation and a new data value must satisfy the register guard ϕ_i (i.e., $f_x(\phi_i)$ holds true).

Then, the behavior of $\mathcal{A}$ can be described using the $\mathsf{wEMSO}^{\mathsf{res}}(\Sigma, \mathbb{D}, \mathbb{S})$-sentence

$$\varphi = \bigoplus X_1, \ldots, X_n, Y_1, \ldots, Y_m.$$
$$(\beta \otimes \bigotimes x.[\bigoplus_{i=1}^{n}(x \in X_i \otimes \mathsf{wt}_{\mathsf{trans}}(t_i) \otimes \bigotimes_{j=1}^{m} \mathsf{wt}_{\mathsf{data}}(t_i, r_j)(Y_j, x))]),$$

i.e., $[\![\varphi]\!] = [\![\mathcal{A}]\!]$. □

Remark 10.2. Now we estimate the computational complexity of the construction provided by the proof of Theorem 10.1. For the simplicity, assume that the size of the alphabet and all transition guards are $O(1)$. Let Φ be the maximal size of a transition guard in $\mathcal{A}$. Note that the formulas Partition and Consistent have the length $O(|T|^2)$ and the formula Registers has the length $O(|T| \cdot |\mathsf{Reg}|)$. Then, the constructed formula φ has the size $O(|T|^2 + |T| \cdot |\mathsf{Reg}|)$.

Now we show that definability by $\mathsf{wEMSO}^{\mathsf{res}}$-sentences implies recognizability by WRA. Our proof will follow a similar strategy as the proof of the corresponding theorem in [18], i.e., we proceed by induction on the structure of the formula, encode the values of variables as letters of an extended alphabet and apply closure properties stated in Lemma 6.1 of our paper. A crucial problem occurs with the $\bigotimes x$-quantifiers and this case requires a new proof technique, since unweighted register automata are not determinizable and our almost boolean formulas contain functions of the form $f(X, x)$. We solve this problem by translating a $\mathsf{wEMSO}^{\mathsf{res}}$-sentence into a sentence where $\bigotimes x$-quantifiers are applied to formulas of the simplified form. Then, using our Theorem 9.5, we can construct a WRA for $\bigotimes x$-formulas.

For simplicity, we denote the triple $(\Sigma, \mathbb{D}, \mathbb{S})$ by Υ. Let $x \in V_1$ be a first-order variable. We say that a formula $\kappa \in \mathsf{wFO}(\Upsilon)$ is a *semi-granular weight formula* over Υ and x if it is of the form $s \otimes f_1(X_1, x) \otimes \ldots \otimes f_r(X_r, x)$ where $s \in S$, $r \geq 0$, $f_1, \ldots, f_r \in \mathcal{F}$ and $X_1, \ldots, X_r$ are second-order variables. If $X_1, \ldots, X_r$ are pairwise distinct, then κ is called a *granular weight formula*. Let $\mathsf{Gran}[x](\Upsilon)$ denote the set of all granular weight formulas over Υ and x. We say that a formula $\gamma \in \mathsf{aBool}[x](\Upsilon)$ is a *simple almost boolean formula* over Υ and x if it is of the form $\bigoplus_{i=1}^{n}(\beta_i \otimes \kappa_i)$ where $n \geq 1$, $\kappa_1, \ldots, \kappa_n \in \mathsf{Gran}[x](\Upsilon)$ and $\beta_1, \ldots, \beta_n \in \mathsf{Bool}(\Sigma, \mathbb{D})$ are boolean formulas. We say that a formula $\psi \in \mathsf{wEMSO}^{\mathsf{res}}(\Upsilon)$ is *canonical* over Υ if whenever it contains a subformula of the form $\bigotimes x.\gamma$, γ is a simple almost boolean formula over Υ and x. Now we show that each sentence $\psi \in \mathsf{wEMSO}^{\mathsf{res}}(\Upsilon)$ can be translated into a canonical sentence over Υ:

Lemma 10.3. *Let $\psi \in \mathsf{wEMSO}^{\mathsf{res}}(\Upsilon)$ be a sentence. Then, there exists a canonical sentence ζ over Υ such that $[\![\zeta]\!] = [\![\psi]\!]$.*

Proof. First, using the commutativity and distributivity of the data semiring $\mathbb{S}$, we can replace every almost boolean formula $\gamma \in \mathsf{aBool}[x](\Upsilon)$ occurring in ψ by a formula $\gamma' = \bigoplus_{i=1}^{n}(\beta_i \otimes \kappa_i)$ where $\beta_1, \ldots, \beta_n \in \mathsf{Bool}(\Sigma, \mathbb{D})$ and each κ_i is a semi-granular weight formula over Υ and x which is of the form $s_i \otimes \bigotimes_{k=1}^{r} f_{ik}(Y_k, x)$. We do this by induction on the structure of γ. If γ is a boolean formula $\beta \in \mathsf{Bool}(\Sigma, \mathbb{D})$, it can be replaced by the formula $\beta \otimes \mathbf{1}$ which is a formula of the desired form. In case γ is the constant $s \in S$ we replace it by the formula $\mathsf{True} \otimes s$. In case $\gamma = f(Y, x)$, where $f \in \mathcal{F}$, Y is a second-order variable and x is a first-order variable, we replace it by the formula $\mathbf{1} \otimes \mathsf{True} \otimes f(Y, x)$ which is also of the desired form. Now we let $\gamma_1 = \bigoplus_{i=1}^{m}(\beta_i \otimes \kappa_i)$ and $\gamma_2 = \bigoplus_{i=m+1}^{n}(\beta_i \otimes \kappa_i)$ where $\beta_i, \ldots, \beta_n \in \mathsf{Bool}(\Sigma, \mathbb{D})$ and each κ_i is a semi-granular weight formula over Υ and x which is of the form $s_i \otimes \bigotimes_{k=1}^{r} f_{ik}(Y_k, x)$. Assume that $\gamma = \gamma_1 \oplus \gamma_2$. Then clearly we have $\gamma_1 \oplus \gamma_2 = \bigoplus_{i=1}^{n}(\beta_i \otimes \kappa_i)$ which is a semi-granular weight formula over Υ and x. Now let γ_1 be as above, and let $\gamma_2 = \bigoplus_{j=1}^{n}(\beta'_j \otimes \kappa'_j)$ where $\beta'_1, \ldots, \beta'_n \in \mathsf{Bool}(\Sigma, \mathbb{D})$ and each κ'_i is a semi-granular weight formula over Υ and x which is of the form $s'_j \otimes \bigotimes_{k=1}^{r} f'_{jk}(Y_k, x)$. Assume that $\gamma = \gamma_1 \otimes \gamma_2$. Then we have $\gamma_1 \otimes \gamma_2 = \bigoplus_{i=1}^{m} \bigoplus_{j=1}^{n}(\beta_i \wedge \beta'_j) \otimes (\kappa_i \otimes \kappa'_j)$ where

$$\begin{aligned}\kappa_i \otimes \kappa'_j &= \left(s_i \otimes \bigotimes_{k=1}^{r} f_{ik}(Y_k, x)\right) \otimes \left(s'_j \otimes \bigotimes_{k=1}^{r} f'_{jk}(Y_k, x)\right) \\ &= (s_i \otimes s'_j) \otimes \bigotimes_{k=1}^{r}\left(f_{ik}(Y_k, x) \otimes f'_{jk}(Y_k, x)\right).\end{aligned}$$

Therefore, $\gamma_1 \otimes \gamma_2$ is also a semi-granular weight formula over Υ and x. Note that the second equality is obtained by the commutativity of the data semiring $\mathbb{S}$.

Now let $\eta \in \mathsf{wEMSO}^{\mathsf{res}}(\Upsilon)$ be the sentence obtained after these replacements. Second, we replace semi-granular weight formulas in η by granular weight formulas. The idea is the following. Assume that $\eta = \bigoplus X_1, \ldots, X_k.\varphi$ with $\varphi \in \mathsf{wFO}^{\mathsf{res}}(\Upsilon)$. In the case when φ contains a semi-granular formula $\kappa = s \otimes f_1(Y_1, x) \otimes \ldots \otimes f_l(Y_l, x) \otimes \ldots \otimes f_j(Y_j, x) \otimes \ldots \otimes f_r(Y_r, x)$ with $l \neq j$ and $Y_l = Y_j$, then we take a fresh second-order variable Z and replace η by the sentence $\bigoplus X_1, \ldots, X_k, Z.([\forall z.(z \in Z \leftrightarrow z \in Y_l)] \otimes \tilde{\varphi})$ where $\tilde{\varphi}$ is obtained from φ by replacing the variable Y_j in κ by the fresh variable Z. Following this process, in finitely many steps we can get rid of all repeating second-order variables in semi-granular weight formulas. We call this formula ζ. Now for a data word $(w, \sigma) \in \mathbb{D}\Sigma^+$ we have:

$$\begin{aligned}[\![\zeta]\!] &= [\![\bigoplus X_1, \ldots, X_k, Z.([\forall z.(z \in Z \leftrightarrow z \in Y_l)] \otimes \tilde{\varphi})]\!](w, \sigma) \\ &= \sum_{I \subseteq \mathsf{dom}(w)} [\![\bigoplus X_1, \ldots, X_k([\forall z.(z \in Z \leftrightarrow z \in Y_l)] \otimes \tilde{\varphi})]\!](w, \sigma[Z/I])\end{aligned}$$

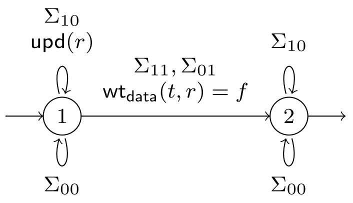

Fig. 5. The WRA $\mathcal{A}$ for the formula $f(X,x)$.

$$= [\![\bigoplus X_1, \ldots, X_k([\forall z.(z \in Z \leftrightarrow z \in Y_l)] \otimes \tilde{\varphi})]\!](w, \sigma[Z/\sigma(Y_l)])$$
$$= [\![\bigoplus X_1, \ldots, X_k.\tilde{\varphi})]\!](w, \sigma[Z/\sigma[Y_l]])$$
$$= [\![\bigoplus X_1, \ldots, X_k.\varphi]\!](w, \sigma) = [\![\psi]\!].$$

This completes the proof. □

Theorem 10.4. *Let $\psi \in \mathsf{wEMSO}^{\mathsf{res}}(\Sigma, \mathbb{S}, \mathbb{D})$ be a sentence. Then there exists a WRA $\mathcal{A}$ over Σ, $\mathbb{S}$ and $\mathbb{D}$ such that $[\![\mathcal{A}]\!] = [\![\psi]\!]$.*

We will prove this theorem by induction on the structure of a subformula ζ of ψ. As usual, whenever ζ contains free variables, ζ will be translated into a WRA over the extended alphabet $\Sigma \times \{0,1\}^{\mathsf{Free}(\zeta)}$. We fix Υ to stand for the triple $(\Sigma, \mathbb{S}, \mathbb{D})$ and we show the following lemmas:

Lemma 10.5. *Let Σ be an alphabet, $\mathbb{D}$ a data structure and $\mathbb{S} = \langle (S, +, \cdot, \mathbf{0}, \mathbf{1}), \mathcal{F} \rangle$ a data semiring over $\mathbb{D}$. The semantics of all boolean formulas $\mathsf{Bool}(\Sigma, \mathbb{D})$ and all formulas $s \in S$ and $f(X,x) \in \mathcal{F}$ are recognizable by a WRA.*

Proof. Let $\beta \in \mathsf{Bool}(\Sigma, \mathbb{D})$. By Theorem 9.5, we can construct a deterministic visibly register automaton $\mathcal{A}$ for β which will be considered as a register automaton. It can be easily transformed into a desired WRA for β by using $\mathbf{1}$ for $\mathsf{wt}_{\mathsf{trans}}$ and $\mathbf{1}^{D \times D}$ for $\mathsf{wt}_{\mathsf{data}}$. The automaton construction for the constant s is straightforward. For the formula $f(X,x)$ over the alphabet $\Gamma^{\langle X \rangle}$ where $\Gamma = \Sigma \times \{0,1\}^{\{x\}}$, the automaton is depicted in Fig. 5. The idea behind the construction is the same as the one for the formula $R(X,x)$. Note that in Fig. 5, for the transition from the state 1 to the state 2, $\mathsf{wt}_{\mathsf{trans}}$ is $\mathbf{1}$ and $\mathsf{wt}_{\mathsf{data}}$ is the weight function f. For all the other transitions we assign $\mathbf{1}$ to $\mathsf{wt}_{\mathsf{trans}}$ and $\mathbf{1}^{D \times D}$ to $\mathsf{wt}_{\mathsf{data}}$. □

Lemma 10.6. *Let $\zeta_1, \zeta_2 \in \mathsf{wFO}(\Upsilon)$ such that $[\![\zeta_1]\!]$ and $[\![\zeta_2]\!]$ are recognizable. Then $[\![\zeta_1 \oplus \zeta_2]\!]$ and $[\![\zeta_1 \otimes \zeta_2]\!]$ are also recognizable.*

Proof. Let $V = \mathsf{Free}(\zeta_1 \oplus \zeta_2) = \mathsf{Free}(\zeta_1 \otimes \zeta_2) = \mathsf{Free}(\zeta_1) \cup \mathsf{Free}(\zeta_2)$. By applying Lemma 7.1, we can see that $[\![\zeta_1]\!]_V$ and $[\![\zeta_2]\!]_V$ are both recognizable. Now by Lemma 6.1 (a), $[\![\zeta_1 \oplus \zeta_2]\!] = [\![\zeta_1]\!]_V \oplus [\![\zeta_2]\!]_V$ and by Lemma 6.1 (b), $[\![\zeta_1 \otimes \zeta_2]\!] = [\![\zeta_1]\!]_V \otimes [\![\zeta_2]\!]_V$ are recognizable. □

Lemma 10.7. *Let $\zeta' \in \mathsf{wFO}(\Upsilon)$ such that $[\![\zeta']\!]$ is recognizable. Then $[\![\bigoplus x.\zeta']\!]$ and $[\![\bigoplus X.\zeta']\!]$ are recognizable.*

Proof. Consider the formula $\zeta = \bigoplus x.\zeta'$ for $\zeta' \in \mathsf{wEMSO}^{\mathsf{res}}(\Sigma, \mathbb{S}, \mathbb{D})$ such that $[\![\zeta']\!]$ is recognizable. Let $V = \mathsf{Free}(\bigoplus x.\zeta')$. Note that $x \notin V$. Consider the projection $h : \Sigma^+_{\mathcal{V} \cup \{x\}} \to \Sigma^+_V$ which erases the x-row in $\Sigma_{V \cup \{x\}}$. Then for a word $(w, \sigma) \in \mathbb{D}\Sigma^+_{V \cup \{x\}}$ where σ is a valid w-assignment, we have

$$\begin{aligned}[\![\bigoplus x.\zeta']\!](w,\sigma) &= \sum([\![\zeta']\!](w, \sigma[x/i]) \mid i \in \mathsf{dom}(w)) \\ &= \sum([\![\zeta']\!](w, \sigma') \mid h(w,\sigma') = (w,\sigma)) \\ &= h([\![\zeta']\!])(w,\sigma).\end{aligned}$$

Due to Lemma 6.2 (a), $[\![\bigoplus x.\zeta']\!]$ is recognizable. The proof for the case $\bigoplus X.\zeta'$ is similar. □

Lemma 10.8. *Let $\gamma \in \mathsf{aBool}[x](\Upsilon)$ be a simple almost boolean formula over Υ and x. Then $\bigotimes x.\gamma$ is recognizable.*

Proof. Let $\zeta = \bigotimes x.\gamma$. Let Reg be the set of all second-order variables Y such that γ has a subformula of the form $R(Y,x)$ with $R \in \mathcal{R}$ or $f(Y,x)$ with $f \in \mathcal{F}$. Let $(Y_k)_{1 \le k \le r}$ be an enumeration of Reg. Now by Lemma 10.3, we can transform γ into the form $\gamma' = \bigoplus_{i=1}^{n}(\beta_i \otimes s_i \otimes \bigotimes_{k=1}^{r} f_{ik}(Y_k, x))$ where $\beta_1, \ldots, \beta_n \in \mathsf{Bool}(\Sigma, \mathbb{D})$, $s_i \in S$ and $f_{ik} \in \mathcal{F}$. We also may assume for simplicity that, for all $i \in \{1, \ldots, n\}$ and $k \in \{1, \ldots, r\}$, β_i has a subformula of the form $R(Y_k, x)$. This means that $\mathsf{Reg} \subseteq \mathsf{Free}(\beta_i)$ for all $i \in \{1, \ldots, n\}$.

Let $\tilde{S} \subseteq S$ be the set of all s_i appearing in γ' and $\tilde{\mathcal{F}} \subseteq \mathcal{F}$ the set of all f_{ik} appearing in γ'. Consider the extended alphabet $\Delta = \Sigma \times \{1, ..., n\} \times \tilde{S} \times \tilde{\mathcal{F}}^r$. A data word in $\mathbb{D}\Delta^+_{\mathsf{Free}(\zeta)}$ will be written by $(w, u, v, \vec{g}, \sigma)$ where $(w, \sigma) \in \mathbb{D}\Sigma^+_{\mathsf{Free}(\zeta)}$, $u = u_1...u_{|w|} \in \{1, ..., n\}^+$, $v = v_1...v_{|w|} \in \tilde{S}^+$ and $\vec{g} = (g^{(1)}, ..., g^{(r)}) \in (\tilde{\mathcal{F}}^+)^r$ and $g^{(i)} = g^{(i)}_1...g^{(i)}_{|w|} \in \tilde{\mathcal{F}}^+$ for all $i \in \{1, ..., r\}$.

We let $\xi \in \mathsf{Bool}(\Delta, \mathbb{D})$ be of the form

$$\xi = \forall x. \bigvee_{i=1}^n \left(\left(\bigvee_{a,v,g_1,...,g_r} P_{(a,i,v,g_1,...,g_r)}(x) \right) \to \left(\tilde{\beta}_i \wedge \bigvee_a P_{(a,i,s_i,f_{i1},...,f_{ir})}(x) \right) \right)$$

where each formula $\tilde{\beta}_i \in \mathsf{Bool}(\Delta, \mathbb{D})$ is obtained from β_i by replacing every predicate $P_a(x)$ with $a \in \Sigma$ and $x \in V_1$ by the formula

$$\bigvee \left(P_{(a,u,v,g_1,...,g_r)}(x) \mid u \in \{1, ..., n\}, v \in \tilde{S}, g_1, ..., g_r \in \tilde{\mathcal{F}} \right).$$

Note that $\mathsf{Free}(\xi) = \mathsf{Free}(\zeta)$ and, in particular, $\mathsf{Reg} \subseteq \mathsf{Free}(\xi)$.

By Theorem 9.5, there exists a deterministic visibly register automaton $\mathcal{A} = (Q, \mathsf{Reg}, I, T, F)$ over $\Delta \times \{0,1\}^{\mathsf{Free}(\xi)\setminus\mathsf{Reg}}$, Reg and $\mathbb{D}$ such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\xi)$. Note that $\mathcal{A}$ can be considered as a register automaton over the alphabet $\Delta \times \{0,1\}^{\mathsf{Free}(\xi)}$ and $\mathbb{D}$. We construct a WRA $\mathcal{A}' = (Q, \mathsf{Reg}, I, T, F, \mathsf{wt}_{\mathsf{trans}}, \mathsf{wt}_{\mathsf{data}})$ over $\Delta \times \{0,1\}^{\mathsf{Free}(\xi)}$, $\mathbb{D}$ and $\mathbb{S}$ where $\mathsf{wt}_{\mathsf{trans}}$ and $\mathsf{wt}_{\mathsf{data}}$ are defined as follows. For any transition $t \in T$ with $\mathsf{label}(t) = (a, \iota, s, f_1, ..., f_r)$, we let $\mathsf{wt}_{\mathsf{trans}}(t) = s$ and, for any register $Y_i \in \mathsf{Reg}$ $(i \in \{1, ..., r\})$, we let $\mathsf{wt}_{\mathsf{data}}(t, Y_i) = f_i$.

By Lemma 6.2 (a), we can construct a WRA $\mathcal{B}$ over $\Sigma \times \{0,1\}^{\mathsf{Free}(\xi)}$, $\mathbb{D}$ and $\mathbb{S}$ such that $[\![\mathcal{B}]\!] = h([\![\mathcal{A}']\!])$. Our goal now is to show that $[\![\mathcal{B}]\!] = [\![\zeta]\!]$.

Indeed, let $(w, \sigma) \in \mathbb{D}\Sigma^+_{\mathsf{Free}(\zeta)}$ with $w = (a_1, d_1)...(a_{|w|}, d_{|w|})$. Then:

$$\begin{aligned}[\![\mathcal{B}]\!](w,\sigma) &= \sum \left([\![\mathcal{A}']\!](w, u, v, \vec{g}, \sigma) \mid u \in \{1, ..., n\}^{|w|}, v \in (\tilde{S})^{|w|}, \vec{g} \in ((\tilde{\mathcal{F}})^{|w|})^r \right) \\ &= \sum \left(\prod_{j=1}^{|w|} v_j \cdot g^{(1)}_j(\delta^{(1)}_j, d_j) \cdot ... \cdot g^{(r)}_j(\delta^{(r)}_j, d_j) \,\middle|\, (w, u, v, \vec{g}, \sigma) \in \mathcal{L}(\mathcal{A}) \right)\end{aligned}$$

where $\delta^{(1)}_j, ..., \delta^{(r)}_j$ are data stored in registers before taking the j-th transition (note that these data values are uniquely determined by $(w, u, v, \tilde{g}, \sigma)$, since $\mathcal{A}$ is deterministic). Moreover, since $\mathcal{A}$ is a visibly register automaton, we have $g^{(k)}_j(\delta^{(k)}_j, d_j) = [\![g^{(k)}_j(Y_k, x)]\!](w, \sigma[x/j])$.

Since $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\xi)$, we have

$$\begin{aligned}&[\![\mathcal{B}]\!](w,\sigma) \\ &= \sum_{u \in \{1,...,n\}^{|w|}} \left(\prod_{j=1}^{|w|} v_j \cdot g^{(1)}_j(\delta^{(1)}_j, d_j) \cdot ... \cdot g^{(r)}_j(\delta^{(r)}_j, d_j) \,\middle|\, (w, u, v, \vec{f}, \sigma) \models \xi \right) \\ &= \sum_{u \in \{1,...,n\}^{|w|}} \prod_{j=1}^{|w|} \left(s_{u(j)} \cdot \prod_{k=1}^{r} [\![f_{u(j),k}(Y_k, x)]\!](w, \sigma[x/j]) \,\middle|\, (w, \sigma[x/j]) \models \beta_{u(j)} \right)\end{aligned}$$

One the other hand:

$$\begin{aligned}&[\![\zeta]\!](w,\sigma) \\ &= \prod_{j=1}^{|w|} [\![\gamma]\!](w, \sigma[x/j]) \\ &= \prod_{j=1}^{|w|} \sum_{i=1}^{n} \left(s_i \cdot \prod_{k=1}^{r} [\![f_{i,k}(Y_k, x)]\!](w, \sigma[x/j] \,\middle|\, (w, \sigma[x/j]) \models \beta_i \right) \\ &= \sum_{u \in \{1,...,n\}^{|w|}} \prod_{j=1}^{|w|} \left(s_{u(j)} \cdot \prod_{k=1}^{r} [\![f_{u(j),k}(Y_k, x)]\!](w, \sigma[x/j]) \,\middle|\, (w, \sigma[x/j]) \models \beta_{u(j)} \right).\end{aligned}$$

Then $[\![\mathcal{B}]\!](w,\sigma) = [\![\zeta]\!](w,\sigma)$ and hence the claim follows. □

The proof of Theorem 10.4 is by induction on the structure of a subformula ζ of ψ, applying Lemmas 10.5–10.8.

Remark 10.9. Since the classical existential MSO over a finite alphabet can be embedded into our new logic wEMSO$^{\mathrm{res}}$, the complexity of the translation of a formula into a WRA is non-elementary in the size of a formula.

11. Conclusion

We introduced a model of weighted register automata and gave an expressively equivalent weighted logic. On the one hand, our results show the robustness of the automata-theoretic approach, help to understand better the behaviors of weighted register automata and can also help to find applications for the setting of timed words. On the other hand, our expressive equivalence result could be used as a basis for the quantitative verification of systems with data, e.g., for the study of quantitative extensions of temporal logics on data words [22]. An important open question concerns algorithmic properties of weighted register automata. We believe that the optimal reachability problem for weighted register automata is decidable for various examples considered in this paper. It could be also interesting to extend our results to the setting of infinite data words and data trees and to investigate in the setting of data words the cases where the weight measure cannot be modeled using semirings (e.g., average or discounted costs [16,24], energy problems [13,23] and weighted register automata with multiple cost parameters [12]). Note that these nonclassical weight measures have been extensively studied in the setting of weighted timed automata. The new research could study the connection between the quantitative and qualitative settings for weighted register automata. Note that for weighted timed automata such a connection was established in [20] via a Nivat theorem. It could be also interesting to compare the expressive power of our register automata model with the data automata model of [11]. We believe that they are incomparable. An extension of class register automata and the logic captured by them [9], where data words have been considered as behavioral models of concurrent systems, to the weighted setting can attract attention, as well.

Our comparison of timed and register automata in terms of data structures could also be of independent interest. On the one hand, the results shown for register automata over arbitrary data structures could be applied to the special case of the timed data structure. On the other hand, we believe that for data structures enjoying similar properties to the timed data structure our comparison results could be used to transfer results for timed automata to some new data structures. The data structure considered in Example 2.2(d) could be a possible candidate for these considerations.

Last but not least, the future research should focus on optimization of the translation procedures given in this article and investigation of fragments of $\text{wEMSO}^{\text{res}}$ with reasonable complexity of the translation into weighted register automata.

References

[1] R. Alur, L. D'Antoni, J. Deshmukh, M. Raghothaman, Y. Yuan, Regular functions and cost register automata, in: LICS 2013, IEEE Computer Society, 2013, pp. 13–22.
[2] R. Alur, D.L. Dill, A theory of timed automata, Theoret. Comput. Sci. 126 (2) (1994) 183–235.
[3] R. Alur, L. Fix, T. Henzinger, Event-clock automata: a determinizable class of timed automata, Theoret. Comput. Sci. 211 (1–2) (1999) 253–273.
[4] R. Alur, P. Madhusudan, Visibly pushdown languages, in: STOC 2004, ACM, 2004, pp. 202–211.
[5] R. Alur, S. La Torre, G.J. Pappas, Optimal paths in weighted timed automata, in: HSCC 2001, in: Lecture Notes in Comput. Sci., vol. 2034, Springer, 2001, pp. 49–62.
[6] P. Babari, M. Droste, V. Perevoshchikov, Weighted register automata and weighted existential MSO logic for data words, in: ICTAC 2016, in: Lecture Notes in Comput. Sci., vol. 9965, Springer, 2016, pp. 370–384.
[7] G. Behrmann, A. Fehnker, T. Hune, K.G. Larsen, P. Petterson, J. Romijn, F. Vaandrager, Minimum-cost reachability for priced timed automata, in: HSCC 2001, in: Lecture Notes in Comput. Sci., vol. 2034, Springer, 2001, pp. 147–161.
[8] M. Bojańczyk, C. David, A. Muscholl, T. Schwentick, L. Segoufin, Two-variable logic on data words, ACM Trans. Comput. Log. 12 (4) (2011), 27 pp.
[9] B. Bollig, An automaton over data words that captures EMSO logic, in: CONCUR 2011, in: Lecture Notes in Comput. Sci., vol. 6901, Springer, 2011, pp. 171–186.
[10] B. Bollig, P. Gastin, Weighted versus probabilistic logics, in: DLT 2009, in: Lecture Notes in Comput. Sci., vol. 5583, Springer, 2009, pp. 18–38.
[11] P. Bouyer, A logical characterization of data languages, Inform. Process. Lett. 84 (2) (2002) 75–85.
[12] P. Bouyer, E. Brinksma, K.G. Larsen, Optimal infinite scheduling for multi-priced timed automata, Form. Methods Syst. Des. 32 (2008) 3–23.
[13] P. Bouyer, U. Fahrenberg, K.G. Larsen, N. Markey, J. Srba, Infinite runs in weighted timed automata with energy constraints, in: FORMATS 2008, in: Lecture Notes in Comput. Sci., vol. 5215, Springer, Heidelberg, 2008, pp. 33–47.
[14] P. Bouyer, A. Petit, D. Thérien, An algebraic characterization of data and timed languages, in: CONCUR 2001, in: Lecture Notes in Comput. Sci., vol. 2154, Springer, 2001, pp. 248–261.
[15] J.R. Büchi, Weak second order arithmetic and finite automata, Z. Math. Log. Grund. Inform. 6 (1960) 66–92.
[16] K. Chatterjee, L. Doyen, T.A. Henzinger, Quantitative languages, in: CSL 2008, in: Lecture Notes in Comput. Sci., vol. 5213, Springer, 2008, pp. 385–400.
[17] T. Colcombet, C. Ley, G. Puppies, On the use of guards for logics with data, in: MFCS 2011, in: Lecture Notes in Comput. Sci., vol. 6907, Springer, 2011, pp. 243–255.
[18] M. Droste, P. Gastin, Weighted automata and weighted logics, Theoret. Comput. Sci. 380 (1–2) (2007) 69–86.
[19] M. Droste, W. Kuich, H. Vogler (Eds.), Handbook of Weighted Automata, EATCS Monographs on Theoretical Computer Science, Springer, 2009.
[20] M. Droste, V. Perevoshchikov, A Nivat theorem for weighted timed automata and relative distance logic, in: ICALP 2014, in: Lecture Notes in Comput. Sci., vol. 8573, Springer, 2014, pp. 171–182.
[21] M. Droste, H. Vogler, Weighted automata and multi-valued logics over arbitrary bounded lattices, Theoret. Comput. Sci. 418 (2012) 14–36.
[22] S. Demri, R. Lazić, LTL with the freeze quantifier and register automata, ACM Trans. Comput. Log. 10 (3) (2009), 30 pp.
[23] U. Fahrenberg, L. Juhl, K.G. Larsen, J. Srba, Energy games in multiweighted automata, in: ICTAC 2011, in: Lecture Notes in Comput. Sci., vol. 6916, Springer, 2011, pp. 95–115.
[24] U. Fahrenberg, K.G. Larsen, Discount-optimal infinite runs in priced timed automata, Electron. Notes Theor. Comput. Sci. 239 (2009) 179–191.
[25] D. Figueira, Alternating register automata on finite data words and trees, Log. Methods Comput. Sci. 8 (1:22) (2012) 1–43.
[26] D. Figueira, P. Hofman, S. Lasota, Relating timed and register automata, in: EXPRESS 2010, in: EPTCS, vol. 41, 2010, pp. 61–75.
[27] M. Kaminski, N. Francez, Finite-memory automata, Theoret. Comput. Sci. 134 (1994) 329–363.
[28] F. Neven, T. Schwentick, V. Vianu, Finite state machines for strings over infinite alphabets, ACM Trans. Comput. Log. 5 (3) (2004) 403–435.
[29] K. Quaas, Kleene–Schützenberger and Büchi Theorems for Weighted Timed Automata, PhD thesis, Universität Leipzig, 2010.
[30] K. Quaas, MSO logics for weighted timed automata, Form. Methods Syst. Des. 38 (3) (2011) 193–222.
[31] T. Wilke, Specifying timed state sequences in powerful decidable logics and timed automata, in: FTRTFT 1994, in: Lecture Notes in Comput. Sci., vol. 863, Springer, 1994, pp. 694–715.