Monitor Logics for Quantitative Monitor Automata*

Erik Paul

Institute of Computer Science, Leipzig University, Leipzig, Germany
epaul@informatik.uni-leipzig.de

Abstract

We introduce a new logic called Monitor Logic and show that it is expressively equivalent to Quantitative Monitor Automata.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases Quantitative Monitor Automata, Nested Weighted Automata, Monitor Logics, Weighted Logics

Digital Object Identifier 10.4230/LIPIcs.MFCS.2017.14

1 Introduction

In the last years, there has been increasing interest in quantitative features of the specification and analysis of systems. Such quantitative aspects include the consumption of a certain resource or the output of a benefit. Both weighted automata and weighted logics [7] are means to achieve this quantitative description of systems. They can be employed for both finite and infinite input.

Very recently, Chatterjee et al. introduced a new automaton model operating on infinite words [4]. *Quantitative Monitor Automata* are equipped with a finite number of monitor counters. At each transition, a counter can be started, terminated, or the value of the counter can be increased or decreased. The term "monitor" stems from the fact that the values of the counters do not influence the behavior of the automaton. The values of the counters when they are terminated provide an infinite sequence of weights, which is evaluated into a single weight using a *valuation function*.

Quantitative Monitor Automata possess several interesting features. They are expressively equivalent to a subclass of *Nested Weighted Automata* [3], an automaton model which for many valuation functions has decidable emptiness and universality problems. Quantitative Monitor Automata are also very expressive. As an example, imagine a storehouse with a resource which is restocked at regular intervals. Between restocks, demands can remove one unit of this resource at a time. Such a succession of restocks and demands can be modeled as an infinite sequence over the alphabet $\{\mathsf{restock}, \mathsf{demand}\}$. Interesting quantitative properties of such a sequence include the long-term average demand, the minimum demand and the maximum demand between restocks. These properties can be described using Quantitative Monitor Automata. At every restock a counter is started, counting the number of demands until the next restock. An appropriate valuation function then computes the desired property. For the average demand, this can be achieved with the *Cesàro mean* which was introduced to automata theory by Chatterjee et. al in [2]. Note that behaviors like these cannot be

* This work was supported by Deutsche Forschungsgemeinschaft (DFG), Graduiertenkolleg 1763 (QuantLA).

modeled using *weighted Büchi-automata* [11, 12] or their extension with valuation functions [8]. In the latter model, the Cesàro mean of any sequence is bounded by the largest transition weight in the automaton. This is not the case for Quantitative Monitor Automata.

In this paper, we develop a logic which is expressively equivalent to Quantitative Monitor Automata. Our main results are the following.

- We introduce a new logic which we call *Monitor Logic*.
- We show that this Monitor Logic is expressively equivalent to Quantitative Monitor Automata.
- We show various closure properties of Quantitative Monitor Automata and prove that Muller and Büchi acceptance conditions provide the same expressive power.

The relationship between automata and logics plays a large role in specification and verification. Statements are often easier to formulate in the form of a logic formula rather than directly as an automaton. Consequently, the fundamental Büchi-Elgot-Trakhtenbrot Theorem [1, 10, 17], which established the coincidence of regular languages with languages definable in monadic second order logic, has found use in many areas of application. An extension to semiring weighted automata was later given by Droste and Gastin [6].

Our logic is equipped with three quantifiers. A sum quantifier to handle the computations on the counters, a valuation quantifier to handle the valuation, and a third quantifier to combine the weights of all runs on a word. Our biggest challenge was to find appropriate restrictions on the use of the quantifiers. Without any restrictions the logic would be too powerful, which we also formally prove using counter examples. The most important result of our considerations is that the computations of the sum quantifier should depend on an MSO-definable condition.

We note that our constructions are effective. Given a formula from our logic, we can effectively construct a Quantitative Monitor Automaton describing this formula. Conversely, for every automaton we can effectively construct a formula with the automaton's behavior.

2 Preliminaries

Let $\mathbb{N} = \{0, 1, 2, \ldots\}$ denote the natural numbers, $\mathbb{Z}$ the integers and $\mathbb{R}$ the reals. An alphabet Σ is a finite set. An infinite word over Σ is a sequence $w = a_0a_1a_2\ldots$ from Σ. The set of infinite words over Σ is denoted by Σ^ω. The set of finite words Σ^* over Σ is defined as the set of finite sequences $a_0a_1\ldots a_n$ from Σ. The empty word is denoted by ε. A mapping $\Sigma^\omega \to \mathbb{R} \cup \{\infty\}$ is called a *series*.

A *(nondeterministic) Muller automaton over* Σ (NMA) is a tuple $\mathcal{A} = (\Sigma, Q, q_0, \mathcal{F}, \delta)$ where (1) Σ is an alphabet, (2) Q is a finite set of states, (3) $q_0 \in Q$ is the initial state, (4) $\mathcal{F} \subseteq \mathcal{P}(Q)$ is the set of final sets and (5) $\delta \subseteq Q \times \Sigma \times Q$ is the set of transitions.

Let $a_0a_1\ldots \in \Sigma^\omega$ be an infinite word. A *run of* $\mathcal{A}$ *over* w is an infinite sequence of transitions $r_w = (t_i)_{i\geq 0}$ so that $t_i = (q_i, a_i, q_{i+1}) \in \delta$ for all $i \geq 0$. We denote by $\mathrm{In}^Q(r_w)$ the set of states which appear infinitely many times in r_w, i.e.

$$\mathrm{In}^Q(r_w) = \{q \in Q \mid \forall i \exists j \geq i : t_j = (q, a_j, q_{j+1})\}.$$

A run r_w of $\mathcal{A}$ over $w \in \Sigma^\omega$ is called *accepting* if $\mathrm{In}^Q(r_w) \in \mathcal{F}$, that is, if the states which appear infinitely many times in r_w form a set in $\mathcal{F}$. In this case we say that w is *recognized (accepted) by* $\mathcal{A}$. The set of accepting runs over a word $w \in \Sigma^\omega$ is denoted by $\mathrm{Acc}_{\mathcal{A}}(w)$. The *infinitary language* of $\mathcal{A}$, denoted by $\mathcal{L}_\omega(\mathcal{A})$, is the set of all infinite words that are accepted by $\mathcal{A}$. A language $L \subseteq \Sigma^\omega$ is called ω-recognizable if there is a Muller automaton $\mathcal{A}$ so that $L = \mathcal{L}_\omega(\mathcal{A})$.

A Muller automaton $\mathcal{A} = (\Sigma, Q, q_0, \mathcal{F}, \delta)$ is called *deterministic* if the set of transitions δ can be interpreted as a function $Q \times A \to Q$, i.e. if for every $(p, a) \in Q \times A$ there exists exactly one $q \in Q$ with $(p, a, q) \in \delta$. It is well known and will be important to us that for each Muller automaton $\mathcal{A}$, we can effectively construct a deterministic Muller automaton with the same language [15, 16].

3 Quantitative Monitor Automata

An *ω-valuation function* is a mapping $\mathrm{Val}\colon \mathbb{Z}^{\mathbb{N}} \to \mathbb{R} \cup \{\infty\}$ that assigns real values or ∞ to infinite sequences of integers. Typical examples of such functions are the Cesàro mean

$$\textsc{Ces}((z_i)_{i\geq 0}) = \begin{cases} \lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} z_i & \text{if this limit exists} \\ \infty & \text{otherwise,} \end{cases}$$

the supremum $\textsc{Sup}((z_i)_{i\geq 0}) = \sup_{i\geq 0} z_i$, the infimum $\textsc{Inf}((z_i)_{i\geq 0}) = \inf_{i\geq 0} z_i$, the limit superior $\textsc{LimSup}((z_i)_{i\geq 0}) = \lim_{n\to\infty} \sup_{i\geq n} z_i$ and the limit inferior $\textsc{LimInf}((z_i)_{i\geq 0}) = \lim_{n\to\infty} \inf_{i\geq n} z_i$.

For a new symbol $\mathbb{1}$ and an ω-valuation function Val, we extend the domain of Val to sequences $\mathbf{z} = (z_i)_{i\geq 0}$ from $\mathbb{Z} \cup \{\mathbb{1}\}$ as follows. If at some point $\mathbf{z}$ becomes constantly $\mathbb{1}$, we let $\mathrm{Val}(\mathbf{z}) = \infty$. Otherwise we let $\mathbf{z}'$ be the subsequence of $\mathbf{z}$ which contains all elements that are not $\mathbb{1}$ and define $\mathrm{Val}(\mathbf{z}) = \mathrm{Val}(\mathbf{z}')$.

We now define Quantitative Monitor Automata. We use a different name, however, in order to distinguish between Büchi and Muller acceptance conditions.

A *Büchi automaton with monitor counters* (BMCA) $\mathcal{A}$ is a tuple $(\Sigma, Q, I, F, \delta, n, \mathrm{Val})$ where (1) Σ is the alphabet, (2) Q is a finite set of states, (3) $I \subseteq Q$ is the set of initial states, (4) F is the set of accepting states, (5) δ is a finite subset of $Q \times \Sigma \times Q \times (\mathbb{Z} \cup \{s, t\})^n$, called the transition relation, such that for every $(q, a, q', \mathbf{u}) \in \delta$ at most one component of $\mathbf{u}$ contains s, (6) n is the number of counters and (7) Val is an ω-valuation function.

Intuitively, the meaning of a transition $(q, a, q', \mathbf{u})$ is that if the automaton is in state q and reads an a, it can move to state q' and start counter j if $u^j = s$, add u^j to the current value of counter j if this counter is activated and $u^j \in \mathbb{Z}$, or stop counter j if $u^j = t$. Initially, all counters are inactive. We will also call $\mathcal{A}$ an n-BMCA or a Val-BMCA, thereby stressing the number of counters or the ω-valuation function used.

Let $a_0 a_1 \ldots \in \Sigma^\omega$ be an infinite word. A *run of $\mathcal{A}$ over w* is an infinite sequence of transitions $r_w = (t_i)_{i\geq 0}$ so that $t_i = (q_i, a_i, q_{i+1}, \mathbf{u}_i) \in \delta$ for all $i \geq 0$.

A run r_w of $\mathcal{A}$ over $w \in \Sigma^\omega$ is called *accepting* if (1) $q_0 \in I$, (2) $\mathrm{In}^Q(r_w) \cap F \neq \emptyset$, (3) if $u_i^j = s$ for some $i \geq 0$, then there exists $l > i$ such that $u_l^j = t$ and for all $k \in \{i+1, \ldots, l-1\}$ we have $u_k^j \in \mathbb{Z}$, (4) if $u_i^j = t$ for some $i \geq 0$, then there exists $l < i$ such that $u_l^j = s$ and for all $k \in \{l+1, \ldots, i-1\}$ we have $u_k^j \in \mathbb{Z}$ and (5) infinitely often some counter is activated, i.e.

$$\{i \geq 0 \mid u_i^j = s \text{ for some } j\}$$

is an infinite set. The set of accepting runs over a word $w \in \Sigma^\omega$ is denoted by $\mathrm{Acc}_{\mathcal{A}}(w)$.

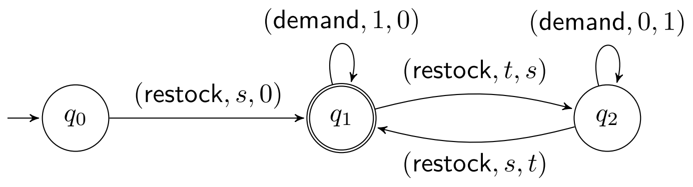
An accepting run r_w defines a sequence $\mathbf{z} = (z_i)_{i\geq 0}$ from $\mathbb{Z} \cup \{\mathbb{1}\}$ as follows. If $u_i^j = s$ for some $j \in \{1, \ldots, n\}$ and $l > i$ is such that $u_l^j = t$ and for all $k \in \{i+1, \ldots, l-1\}$ we have $u_k^j \in \mathbb{Z}$, then $z_i = \sum_{k=i+1}^{l-1} u_k^j$. If $u_i^j \neq s$ for all $j \in \{1, \ldots, n\}$, then $z_i = \mathbb{1}$. We also call $\mathbf{z}$ the *weight-sequence associated to r_w*. The weight of the run r_w is defined as $\mathrm{Val}(r_w) = \mathrm{Val}(\mathbf{z})$. The behavior of the automaton $\mathcal{A}$ is the series $[\![\mathcal{A}]\!]\colon \Sigma^\omega \to \mathbb{R} \cup \{\infty\}$

defined by $[\![\mathcal{A}]\!](w) = \inf_{r_w \in \mathrm{Acc}(w)} \mathrm{Val}(r_w)$, where the infimum over the empty set is defined as ∞. A series $\Sigma^\omega \to \mathbb{R} \cup \{\infty\}$ is called *MC-recognizable* if there exists a BMCA $\mathcal{A}$ such that $[\![\mathcal{A}]\!] = S$. The notions of *n-MC-recognizable* and Val-*MC-recognizable* are defined likewise.

A *Muller automaton with monitor counters* (MMCA) is defined like a BMCA, but instead of a set of accepting states we have a set of accepting sets $\mathcal{F} \subseteq \mathcal{P}(Q)$. The condition (2) for a run r_w on a word $w \in \Sigma^\omega$ to be accepting is then replaced by $\mathrm{In}^Q(r_w) \in \mathcal{F}$, i.e. a Muller acceptance condition.

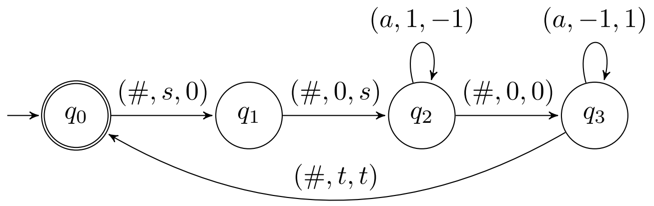
Büchi automata with monitor counters use a Büchi acceptance condition, i.e. at least one accepting state has to appear infinitely often. Lemma 3 shows that using a Muller acceptance condition does not influence the expressive power.

▶ **Example 1.** Consider the the alphabet $\Sigma = \{\mathsf{demand}, \mathsf{restock}\}$ with the ω-valuation function Val = Ces. We model a storehouse with some sort of supply that is restocked whenever restock is encountered, and one unit of the supply is removed at every demand. Given an infinite sequence of restocks and demands, we are interested in the long-time average demand between restocks. Under the assumption that every such sequence starts with a restock, this behavior is modeled by the following automaton with two monitor counters.



When for the valuation function we take Inf or Sup, the automaton above describes the lowest or highest demand ever encountered, for the latter assuming that the demands are bounded.

▶ **Example 2** ([4]). Consider the alphabet $\Sigma = \{a, \#\}$ and the language L consisting of words $(\#^2 a^* \# a^* \#)^\omega$. On these words, we consider the quantitative property "the maximal block-length difference between even and odd positions", i.e. the value of the word $\#\# a^{m_1} \# a^{m_2} \#\#\# \dots$ shall be $\sup_{i \geq 1} |m_{2i-1} - m_{2i}|$. With the choice Val = Sup, a BMCA realizing this behavior is the following.



Each $(\#^2 a^{m_1} \# a^{m_2} \#)$-block is processed by starting both counters on the first two #'s, accumulating m_1 into the first counter and accumulating $-m_1$ into the second, reading #, then accumulating $m_1 - m_2$ into the first counter and $-m_1 + m_2$ into the second, and finally terminating both counters on the last #. Thus, the associated weight-sequence for only this block is $(m_1 - m_2, -m_1 + m_2, \mathbb{1}, \dots, \mathbb{1})$. Clearly, the final value of counter one is always the negative of the final value in counter two. Since our ω-valuation function is Sup, only the positive counter value actually plays a role in the value assigned to the whole word, and this positive value is $|m_1 - m_2|$.

In the rest of this section, we prove various closure properties for automata with monitor counters and that BMCA and MMCA have the same expressive power.

▶ **Lemma 3.** *Büchi automata with monitor counters are expressively equivalent to Muller automata with monitor counters.*

Proof. The proof is similar to the standard construction to show that Büchi automata are expressively equivalent to Muller automata, see for example [9]. ◀

The next lemma shows that MC-recognizable series are closed under projections and their preimage. Given two alphabets Σ and Γ and a mapping $h\colon \Sigma \to \Gamma$ and thus a homomorphism $h\colon \Sigma^\omega \to \Gamma^\omega$, we define for every $S\colon \Sigma^\omega \to \mathbb{R} \cup \{\infty\}$ the *projection* $h(S)\colon \Gamma^\omega \to \mathbb{R} \cup \{\infty\}$ by

$$h(S)(w) = \inf\{S(v) \mid h(v) = w\}$$

for every $w \in \Gamma^\omega$. Moreover, if $S'\colon \Gamma^\omega \to \mathbb{R} \cup \{\infty\}$, then we put $h^{-1}(S') = S' \circ h$, i.e. $h^{-1}(S')\colon \Sigma^\omega \to \mathbb{R} \cup \{\infty\}$, $w \mapsto S'(h(w))$.

▶ **Lemma 4.** *Let Σ and Γ be two alphabets, $h\colon \Sigma \to \Gamma$ be a mapping and* Val *be an ω-valuation function.*

(i) *If $S\colon \Sigma^\omega \to \mathbb{R} \cup \{\infty\}$ is* Val*-MC-recognizable, then the projection $h(S)\colon \Gamma^\omega \to \mathbb{R} \cup \{\infty\}$ is also* Val*-MC-recognizable.*

(ii) *If $S'\colon \Gamma^\omega \to \mathbb{R} \cup \{\infty\}$ is* Val*-MC-recognizable, then $h^{-1}(S')\colon \Sigma^\omega \to \mathbb{R} \cup \{\infty\}$ is also* Val*-MC-recognizable.*

For two series $S_1, S_2\colon \Sigma^\omega \to \mathbb{R} \cup \{\infty\}$, the minimum $\min(S_1, S_2)$ of S_1 and S_2 is defined pointwise, i.e.

$$\min(S_1, S_2)(w) = \min\{S_1(w), S_2(w)\}.$$

Applying the usual union construction for automata, we can show that the minimum of two MC-recognizable series is MC-recognizable as well.

▶ **Lemma 5.** *For any given ω-valuation function* Val*, the* Val*-MC-recognizable series are closed under minimum.*

Let $L \subseteq \Sigma^\omega$ and $S\colon \Sigma^\omega \to \mathbb{R} \cup \{\infty\}$. The *intersection of L and S* is the series $L \cap S\colon \Sigma^\omega \to \mathbb{R} \cup \{\infty\}$ defined for $w \in \Sigma^\omega$ by

$$L \cap S(w) = \begin{cases} S(w) & \text{if } w \in L \\ \infty & \text{otherwise.} \end{cases}$$

The intersection of a recognizable language with an MC-recognizable series is MC-recognizable as well.

▶ **Lemma 6.** *Let* Val *be an ω-valuation function, let $L \subseteq \Sigma^\omega$ be ω-recognizable and $S\colon \Sigma^\omega \to \mathbb{R} \cup \{\infty\}$ be* Val*-MC-recognizable. Then $L \cap S$ is also* Val*-MC-recognizable.*

Proof. The proof is similar to the standard product construction to show that recognizable languages are closed under intersection. ◀

4 Monitor MSO logic

We first want to give a motivation for the quantifiers and restrictions we use in our logic. We are looking for a logic which is expressively equivalent to automata with monitor counters. It is clear that we need a valuation quantifier in order to model the valuation done by the automata. The question is which types of formulas should be allowed in the scope of the valuation quantifier. From [8] it follows that allowing only *almost Boolean* formulas (see below) is too weak. We would only describe Muller automata over valuation monoids, and these are strictly weaker than automata with monitor counters [4].

We therefore have to allow at least some other quantifier in the scope of the valuation quantifier. Taking into account the automaton model we want to describe, this should be a sum quantifier. Most weighted logics [6, 9, 8, 13, 14, 5] use quantifiers that act unconditionally on the whole input, i.e. on the whole word, tree or picture. However, in Lemma 11 we will see that in our case, an unrestricted sum quantifier quickly gets out of hand.

The intention of the sum quantifier as we define it here is to have a sum quantifier which acts on infinite words, but still computes only finite sums on a given word. The computation of the sum quantifier depends on a first order variable x and a second order variable X provided to it. The variable X serves as a "list" of start and stop positions, and the variable x indicates where the summation on the infinite word should take place. Simply put, the sum is evaluated to $\mathbb{1}$ if x does not point to a position in X or there is no successor of x in X. Otherwise, if y is x's successor in X, the sum is taken from $x+1$ to $y-1$.

Intuitively, each sum quantifier corresponds to a counter. In a run of an automaton with monitor counters, not more than one counter can be started at each letter of the given word. Therefore, we use Boolean formulas to choose which counter to use. We combine these choices between counters into so-called x-*summing* formulas, where x is the first order variable provided to each sum quantifier in the formula.

We provide a countable set $\mathcal{V}$ of first and second order variables, where lower case letters like x and y denote first order variables and capital letters like X and Y denote second order variables. We define a three step logic over an alphabet Σ according to the following grammars.

$$\begin{aligned}
\beta &::= P_a(x) \mid x \leq y \mid x \in X \mid \neg\beta \mid \beta \vee \beta \mid \exists x.\beta \mid \exists X.\beta\\
\psi &::= k \mid \beta \mathbin{?} \psi : \psi\\
\zeta_x &::= \mathbb{1} \mid \beta \mathbin{?} \zeta_x : \zeta_x \mid \bigoplus\nolimits^{x,X} y.\psi\\
\varphi &::= \beta \mathbin{?} \varphi : \varphi \mid \min(\varphi,\varphi) \mid \inf x.\varphi \mid \inf X.\varphi \mid \mathrm{Val}\, x.\zeta_x
\end{aligned}$$

where $x, y, X \in \mathcal{V}$, $a \in \Sigma$ and $k \in \mathbb{Z}$. The formulas β are called *Boolean* or *MSO* formulas, the formulas ψ *almost Boolean* formulas, the formulas ζ_x x-*summing* formulas and the formulas φ *monitor MSO* (mMSO) formulas. We remark that within an x-summing formula, the first order variable provided to each sum quantifier is always x. This restriction is not imposed on the second order quantifiers, i.e. $\beta \mathbin{?} \bigoplus^{x,X} y.\psi_1 : \bigoplus^{x,Z} y.\psi_2$ is an x-summing formula, but $\beta \mathbin{?} \bigoplus^{x,X} y.\psi_1 : \bigoplus^{z,Z} y.\psi_2$ is neither an x-summing nor a z-summing formula. Also note that the x-summing formulas are only auxiliary formulas, see Remark 7 later on.

The set of free variables $\mathrm{Free}(\varphi)$ is defined as usual, i.e. $\exists$, inf and Val bind variables, and in $\bigoplus^{x,X} y.\psi$ the variable y is bound. A formula without free variables is called a *sentence*.

Let $w \in \Sigma^\omega$. We put $\mathrm{dom}(w) = \{0, 1, 2, \dots\}$ and denote the i-th letter of w by w_i, i.e. $w = w_0w_1w_2\dots$. Let $\mathcal{V}$ be a finite set of first and second order variables with $\mathrm{Free}(\varphi) \subseteq \mathcal{V}$. A $(\mathcal{V}, w)$-assignment is a mapping $\rho\colon \mathcal{V} \to \mathrm{dom}(w) \cup \mathcal{P}(\mathrm{dom}(w))$ where every first order variable is mapped to an element of $\text{dom}(w)$ and every second order variable is mapped to a subset of $\text{dom}(w)$. The update $\rho[x \to i]$ for $i \in \text{dom}(w)$ is defined as $\rho[x \to i](x) = i$ and $\rho[x \to i](\mathcal{X}) = \rho(\mathcal{X})$ for all $\mathcal{X} \in \mathcal{V} \setminus \{x\}$. The update $\rho[X \to I]$ for $I \subseteq \text{dom}(w)$ is defined similarly. We encode $(\mathcal{V}, w)$-assignments as usual with an extended alphabet $\Sigma_{\mathcal{V}} = \Sigma \times \{0,1\}^{\mathcal{V}}$. Here, we refer to a word over the alphabet $\Sigma_{\mathcal{V}}$ by (w, ρ), where w is the projection to Σ and ρ is the projection to $\{0,1\}^{\mathcal{V}}$. A word over $\Sigma_{\mathcal{V}}$ represents an assignment if and only if for every first order variable the respective row in the extended word contains exactly one 1, in which case (w, ρ) is called *valid*.

It is not difficult to see that the set

$$N_{\mathcal{V}} = \{(w, \rho) \in \Sigma_{\mathcal{V}}^{\omega} \mid (w, \rho) \text{ is valid}\}$$

is ω-recognizable. Let $(w, \rho) \in \Sigma_{\mathcal{V}}^{\omega}$. For a Boolean formula β we define the satisfaction relation $(w, \rho) \models \beta$ as usual: if (w, ρ) is not valid, then $(w, \rho) \models \beta$ does not hold; otherwise we define it as follows.

$$\begin{array}{lcl}
(w,\rho) \models P_a(x) & \iff & w_{\rho(x)} = a \\
(w,\rho) \models x \leq y & \iff & \rho(x) \leq \rho(y) \\
(w,\rho) \models x \in X & \iff & \rho(x) \in \rho(X) \\
(w,\rho) \models \neg\beta & \iff & (w,\rho) \models \beta \text{ does not hold} \\
(w,\rho) \models \beta_1 \vee \beta_2 & \iff & (w,\rho) \models \beta_1 \text{ or } (w,\rho) \models \beta_2 \\
(w,\rho) \models \exists x.\beta & \iff & (w,\rho[x \to i]) \models \beta \text{ for some } i \in \text{dom}(w) \\
(w,\rho) \models \exists X.\beta & \iff & (w,\rho[X \to I]) \models \beta \text{ for some } I \subseteq \text{dom}(w).
\end{array}$$

Let β be an MSO formula. We will write Σ_β for $\Sigma_{\text{Free}(\beta)}$ and N_β for $N_{\text{Free}(\beta)}$. We recall the fundamental Büchi Theorem [1], namely that for $\text{Free}(\beta) \subseteq \mathcal{V}$ the language

$$\mathcal{L}_{\mathcal{V}}(\beta) = \{(w, \rho) \in N_{\mathcal{V}} \mid (w, \rho) \models \beta\}$$

defined by β over $\Sigma_{\mathcal{V}}$ is ω-recognizable. We abbreviate $\mathcal{L}(\beta) = \mathcal{L}_{\text{Free}(\beta)}(\beta)$.

Conversely, every ω-recognizable language $L \subseteq \Sigma^\omega$ is definable by an MSO sentence β, i.e. $L = \mathcal{L}(\beta)$.

We now come to the semantics of the remaining formulas. Let Val be an ω-valuation function. For an almost Boolean, x-summing or monitor MSO formula η we define the *semantics* $[\![\eta]\!]_{\mathcal{V}}(w, \rho)$ *of* η *under the* $(\mathcal{V}, w)$*-assignment* ρ as follows: if (w, ρ) is not valid, then $[\![\eta]\!]_{\mathcal{V}}(w, \rho) = \infty$; otherwise the semantics are defined as follows.

$$\begin{array}{rcl}
[\![k]\!]_{\mathcal{V}}(w,\rho) & = & k \\
[\![\beta \mathbin{?} \psi_1 : \psi_2]\!]_{\mathcal{V}}(w,\rho) & = & \begin{cases} [\![\psi_1]\!]_{\mathcal{V}}(w,\rho) & \text{if } (w,\rho) \models \beta \\ [\![\psi_2]\!]_{\mathcal{V}}(w,\rho) & \text{otherwise} \end{cases} \\
[\![\bigoplus^{x,X} y.\psi]\!]_{\mathcal{V}}(w,\rho) & = & \begin{cases} \displaystyle\sum_{i=\rho(x)+1}^{\min\{j \in \rho(X) \mid j > \rho(x)\}-1} [\![\psi]\!]_{\mathcal{V}}(w,\rho[y \to i]) & \begin{array}{l}\text{if } \rho(x) \in \rho(X) \text{ and} \\ \{j \in \rho(X) \mid j > \rho(x)\} \neq \emptyset\end{array} \\ \mathbb{1} & \text{otherwise.} \end{cases} \\
[\![\min(\varphi_1,\varphi_2)]\!]_{\mathcal{V}}(w,\rho) & = & \min\{[\![\varphi_1]\!]_{\mathcal{V}}(w,\rho), [\![\varphi_2]\!]_{\mathcal{V}}(w,\rho)\} \\
[\![\inf x.\varphi]\!]_{\mathcal{V}}(w,\rho) & = & \inf_{i \in \text{dom}(w)} [\![\varphi]\!]_{\mathcal{V}}(w,\rho[x \to i]) \\
[\![\inf X.\varphi]\!]_{\mathcal{V}}(w,\rho) & = & \inf_{I \subseteq \text{dom}(w)} [\![\varphi]\!]_{\mathcal{V}}(w,\rho[X \to I]) \\
[\![\text{Val}\, x.\zeta_x]\!]_{\mathcal{V}}(w,\rho) & = & \text{Val}(([\![\zeta_x]\!]_{\mathcal{V}}(w,\rho[x \to i]))_{i \in \text{dom}(w)}).
\end{array}$$

We write $[\![\eta]\!]$ for $[\![\eta]\!]_{\text{Free}(\eta)}$. To indicate the ω-valuation function Val or the alphabet Σ used, we may denote the set of monitor MSO formulas by $\text{mMSO}(\Sigma, \text{Val})$.

▶ Remark 7. From the semantics defined here it is clear that any x-summing sentence ζ_x is semantically equivalent to $\mathbb{1}$. In this sense, the x-summing formulas constitute no meaningful fragment of our logic, and are only auxiliary formulas for the construction of monitor MSO formulas.

Note also that for (w,ρ) valid, we have $[\![\mathrm{Val}\, x.\mathbb{1}]\!](w,\rho)=\infty$. By abuse of notation, we can thus define the abbreviation $\infty = \mathrm{Val}\, x.\mathbb{1}$.

▶ Remark 8. The condition used in the definition of the sum quantifier is definable by the MSO formula

$$\mathrm{notLast}(x,X) = x\in X \wedge \exists y.(y\in X\wedge x<y),$$

where $x<y$ is an abbreviation for $x\le y\wedge\neg(y\le x)$. We can therefore also write

$$[\![\textstyle\bigoplus^{x,X} y.\psi]\!](w,\rho)=\begin{cases}\sum_{i=\rho(x)+1}^{\min\{j\in\rho(X)\mid j>\rho(x)\}-1}[\![\psi]\!](w,\rho[y\to i]) & \text{if } (w,\rho)\models\mathrm{notLast}(x,X)\\ \mathbb{1} & \text{otherwise.}\end{cases}$$

In Lemma 12 we will see that the first order variable x is necessarily also the variable which is quantified by Val. If we define an unrestricted sum quantifier $\bigoplus y.\psi$ by

$$[\![\textstyle\bigoplus y.\psi]\!](w,\rho)=\sum_{i\in\mathrm{dom}(w)}[\![\psi]\!](w,\rho[y\to i]),$$

we can write our restricted sum quantifier as

$$[\![\textstyle\bigoplus^{x,X} y.\psi]\!](w,\rho)=$$
$$[\![\mathrm{notLast}(x,X)\ ?\ \textstyle\bigoplus y.(x<y\wedge\forall z.((x<z\wedge z\le y)\to\neg z\in X)\ ?\ \psi:0):\mathbb{1}]\!](w,\rho).$$

▶ **Example 9.** Consider Example 1 again, i.e. the alphabet $\Sigma=\{\mathsf{demand},\mathsf{restock}\}$ with the ω-valuation function Val = Ces. Then the formula

$$\varphi=\inf X.\left(\forall z.(z\in X\leftrightarrow P_{\mathsf{restock}}(z))\ ?\ \mathrm{Val}\,x.\textstyle\bigoplus^{x,X} y.1:\infty\right)$$

describes the average total demand between two restocks. We recall that ∞ is simply an abbreviation for the formula $\mathrm{Val}\,x.\mathbb{1}$. As in Example 1, if we take Inf or Sup for the valuation function, the formula above describes the lowest or highest demand ever encountered.

▶ **Lemma 10** (Consistency Lemma). *Let $\varphi\in\mathrm{mMSO}(\Sigma,\mathrm{Val})$ and $\mathcal{V}$ be a finite set of variables with $\mathcal{V}\supseteq\mathrm{Free}(\varphi)$.*

(i) *For any valid $(w,\rho)\in\Sigma_{\mathcal{V}}^{\omega}$ we have $[\![\varphi]\!]_{\mathcal{V}}(w,\rho)=[\![\varphi]\!](w,\rho\restriction_{\mathrm{Free}(\varphi)})$.*

(ii) *$[\![\varphi]\!]$ is MC-recognizable if and only if $[\![\varphi]\!]_{\mathcal{V}}$ is MC-recognizable.*

The following lemma shows that the use of an unrestricted sum quantifier leads to not MC-recognizable series.

▶ **Lemma 11.** *Consider the unrestricted sum quantifier from Remark 8*

$$[\![\textstyle\bigoplus y.\psi]\!](w,\rho)=\sum_{i\in\mathrm{dom}(w)}[\![\psi]\!](w,\rho[y\to i]),$$

the ω-valuation function Val *defined by*

$$\mathrm{Val}((z_i)_{i\ge0})=\begin{cases}\sum_{i=0}^{\infty} z_i & \textit{if this sum converges}\\ \infty & \textit{otherwise}\end{cases}$$

and the alphabet $\Sigma=\{a,b\}$. Then for the almost Boolean formula

$$\psi=y\le x\wedge\forall z.(z\le x\to P_a(z))\ ?\ -1:0,$$

the formula $\varphi=\mathrm{Val}\,x.\bigoplus y.\psi$ is not Val*-MC-recognizable.*

Proof (sketch). One easily checks that

$$[\![\varphi]\!](w) = \begin{cases} -\frac{m(m+1)}{2} & \text{if } w = a^m b w' \text{ for some } w' \in \Sigma^\omega \\ \infty & \text{if } w = a^\omega. \end{cases}$$

The idea is now that with only finitely many transitions, and therefore only finitely many different weights, this quadratic growth cannot be realized if only transitions up to the first b in each word influence the weight of the runs. But once the automaton has read this first b, it cannot distinguish between the words anymore. Under appropriate assumptions, we can therefore combine runs from different words to obtain a contradiction. ◀

The next lemma shows that the first order variable x provided to the sum quantifier is necessarily the variable that Val quantifies.

▶ **Lemma 12.** *Consider the ω-valuation function* Val *defined by*

$$\mathrm{Val}((z_i)_{i\geq 0}) = \begin{cases} \frac{1}{z_0} & \text{if } 0 < z_0 = z_1 = z_2 = \dots \\ -1 & \text{otherwise.} \end{cases}$$

and the alphabet $\Sigma = \{a\}$. We define the abbreviation

$$(y = x + 1) = x \leq y \wedge \neg(y \leq x) \wedge \forall z.(z \leq x \vee y \leq z).$$

Then for the Boolean formula

$$\beta(X) = \forall x_1.\forall x_2.((x_1 \in X \wedge x_2 = x_1 + 1) \to \neg(x_2 \in X)),$$

the formula $\varphi = \inf X. \inf z. \left(\beta(X) \mathrel{?} \mathrm{Val}\, x. \bigoplus^{z,X} y.1 : \infty\right)$ *is not* Val*-MC-recognizable.*

Proof (sketch). One can check that $[\![\varphi]\!](a^\omega) = 0$. For a BMCA $\mathcal{A}$ realizing this series, the weight-sequence associated to each run has to be constant, and there must be a sequence of runs such that this constant grows arbitrarily large. The latter fact can be exploited to show that there must be a run whose associated weight-sequence is not constant, which leads to the contradiction $[\![\mathcal{A}]\!](a^\omega) = -1$. ◀

5 The main result

In this section, we want to show that the MC-recognizable series coincide with the series definable by monitor MSO formulas from our logic. In Lemma 14, we show how a given MMCA can be described by a monitor MSO formula. To show that every series definable by a monitor MSO formula is also MC-recognizable, we show by induction on the structure of the formula how to construct an MMCA with the same behavior as the formula. We first formulate our main theorem.

▶ **Theorem 13.** *Let Σ be an alphabet and* Val *an ω-valuation function. A series $S\colon \Sigma^\omega \to \mathbb{R} \cup \{\infty\}$ is* Val*-MC-recognizable if and only if there is a monitor MSO sentence $\varphi \in \mathrm{mMSO}(\Sigma, \mathrm{Val})$ with $[\![\varphi]\!] = S$.*

In the following lemma, we show the first direction, namely how to obtain a formula for a given MMCA.

▶ **Lemma 14.** *For every* Val*-MMCA $\mathcal{A}$, there exists a sentence $\varphi \in \mathrm{mMSO}(\Sigma, \mathrm{Val})$ with $[\![\mathcal{A}]\!] = [\![\varphi]\!]$.*

Proof. For first order variables x and y and second order variables $X_1, \dots, X_k$ we define the MSO formulas

$$\begin{aligned}
\text{first}(x) &= \forall y. x \leq y \\
x < y &= x \leq y \wedge \neg(y \leq x) \\
(y = x+1) &= x < y \wedge \forall z.(z \leq x \vee y \leq z) \\
\text{partition}(X_1, \dots, X_k) &= \forall x. \bigvee_{i=1}^{k} \left(x \in X_i \wedge \bigwedge_{j \neq i} \neg(x \in X_j) \right).
\end{aligned}$$

Now let $\mathcal{A} = (\Sigma, Q, I, \mathcal{F}, \delta, n, \text{Val})$ be an n-MMCA. For every $(p, a, q, \mathbf{u}) \in \delta$ we choose a second order variable $X_{(p,a,q,\mathbf{u})}$ and with $k = |\delta|$ we fix a bijection $v \colon \{1, \dots, k\} \to \delta$. For $i \in \{1, \dots, k\}$ we write X_i for $X_{v(i)}$ and $\bar{X}$ for $(X_1, \dots, X_k)$. Furthermore, we fix second order variables $Y_1, \dots, Y_n$ and write $\bar{Y}$ for $(Y_1, \dots, Y_n)$. For $j \in \{1, \dots, n\}$ and $\star \in \{s, t\}$ we abbreviate

$$(u^j(x) = \star) = \bigvee_{\substack{(p,a,q,\mathbf{u}) \in \delta \\ u^j = \star}} x \in X_{(p,a,q,\mathbf{u})}.$$

Intuitively, we use the variables $\bar{X}$ to encode runs, i.e. by assigning the transition $v(i)$ to every position in X_i. The variables $\bar{Y}$ are used to mark the starts and stops of the counters in $\bar{X}$. In the following, we define the MSO formula muller$(\bar{X})$ which checks that $\bar{X}$ encodes a run of $\mathcal{A}$ satisfying the Muller acceptance condition, and the MSO formula valid$(\bar{X})$ which checks that $\bar{X}$ encodes an accepting run. The MSO formula valid$^*(\bar{X}, \bar{Y})$ asserts that the positions in $\bar{Y}$ conform to the starts and stops of the counters in $\bar{X}$. The precise formulas are as follows.

$$\text{matched}(\bar{X}) = \bigwedge_{(p,a,q,\mathbf{u}) \in \delta} \forall x. \big(x \in X_{(p,a,q,\mathbf{u})} \to P_a(x) \big)$$

$$\wedge \forall x. \forall y. \left(y = x+1 \to \bigvee_{q \in Q} \left(\bigvee_{(p,a,q,\mathbf{u}),(q,a',p',\mathbf{u}') \in \delta} (x \in X_{(p,a,q,\mathbf{u})} \wedge y \in X_{(q,a',p',\mathbf{u}')}) \right) \right)$$

$$\text{muller}(\bar{X}) = \text{partition}(\bar{X}) \wedge \text{matched}(\bar{X}) \wedge \exists x. \left(\text{first}(x) \wedge \bigvee_{\substack{(p,a,q,\mathbf{u}) \in \delta \\ p \in I}} x \in X_{(p,a,q,\mathbf{u})} \right)$$

$$\wedge \bigvee_{F \in \mathcal{F}} \left(\exists x. \forall y. x \leq y \to \left(\left(\bigvee_{\substack{(p,a,q,\mathbf{u}) \in \delta \\ q \in F}} y \in X_{(p,a,q,\mathbf{u})} \right) \right. \right.$$

$$\left. \left. \wedge \bigwedge_{q \in F} \exists z. \left(y \leq z \wedge \bigvee_{(p,a,q,\mathbf{u}) \in \delta} z \in X_{(p,a,q,\mathbf{u})} \right) \right) \right)$$

$$\text{valid}(\bar{X}) = \text{muller}(\bar{X}) \wedge \forall x. \exists y. (x \leq y \wedge \bigvee_{j=1}^{n} u^j(y) = s) \wedge$$

$$\bigwedge_{j=1}^{n} \forall x. \Big(\big((u^j(x) = s) \to \exists y. (x < y \wedge u^j(y) = t \wedge \forall z. ((x < z \wedge z < y) \to \neg(u^j(z) = s))) \big)$$

$$\wedge \big((u^j(x) = t) \to \exists y. (y < x \wedge u^j(y) = s \wedge \forall z. ((y < z \wedge z < x) \to \neg(u^j(z) = t))) \big) \Big)$$

$$\mathrm{valid}^*(\bar{X},\bar{Y}) = \mathrm{valid}(\bar{X}) \wedge \bigwedge_{j=1}^{n} \forall x.(x \in Y_j \leftrightarrow (u^j(x) = s \vee u^j(x) = t)).$$

For $(p,a,q,\mathbf{u}) \in \delta$ we let $\mathrm{wt}_j(p,a,q,\mathbf{u}) = u^j$ and for $i \in \{1,\dots,k-2\}$ and $j \in \{1,\dots,n\}$ define inductively

$$\begin{aligned}
\psi^j_{k-1} &= (y \in X_{k-1} \mathbin{?} \mathrm{wt}_j(v(k-1)) : \mathrm{wt}_j(v(k)))\\
\psi^j_i &= \left(y \in X_i \mathbin{?} \mathrm{wt}_j(v(i)) : \psi^j_{i+1}\right)\\
\zeta_{n+1} &= \mathbb{1}\\
\zeta_j &= \left((u^j(x) = s) \mathbin{?} \bigoplus\nolimits^{x,Y_j} y.\psi^j_1 : \zeta_{j+1}\right).
\end{aligned}$$

Then with $\varphi = \inf \bar{X}.\inf \bar{Y}.(\mathrm{valid}^*(\bar{X},\bar{Y}) \mathbin{?} \mathrm{Val}\, x.\zeta_1 : \infty)$, we have $[\![\mathcal{A}]\!] = [\![\varphi]\!]$. The formula ψ^j_1 evaluates to the weight for counter j in the transition at position y, i.e. it is $\mathrm{wt}_j(v(i))$ iff y is in X_i. The formula ζ_1 evaluates to the output of the counter started at position x in the run encoded by $\bar{X}$. More precisely, ζ_1 evaluates to $\bigoplus^{x,Y_j} y.\psi^j_1$ if counter j is started at position x, and to $\mathbb{1}$ if no counter is started at x. Finally, the formula φ takes the infimum over the weights of all "runs" $\bar{X}$, in the sense that assignments to $\bar{X}$ and $\bar{Y}$ only influence the value of φ if $\bar{X}$ encodes an accepting run and $\bar{Y}$ mirrors its counter starts and stops. ◀

The remainder of this section is dedicated to show the converse, namely that for every monitor MSO formula there is an MMCA with the same behavior as the formula.

▶ **Lemma 15.** *Let β be an MSO formula and $\varphi_1,\varphi_2 \in \mathrm{mMSO}(\Sigma,\mathrm{Val})$ such that $[\![\varphi_1]\!]$ and $[\![\varphi_2]\!]$ are MC-recognizable. Then with $\varphi = \beta \mathbin{?} \varphi_1 : \varphi_2$, the series $[\![\varphi]\!]$ is also MC-recognizable.*

Proof. Let $\mathcal{V} = \mathrm{Free}(\varphi)$. Then we have $\mathrm{Free}(\varphi_1) \subseteq \mathcal{V}$ and $\mathrm{Free}(\varphi_2) \subseteq \mathcal{V}$ and hence by Lemma 10 $[\![\varphi_1]\!]_{\mathcal{V}}$ and $[\![\varphi_2]\!]_{\mathcal{V}}$ are MC-recognizable. Due to $\mathrm{Free}(\beta) \subseteq \mathcal{V}$, the classical Büchi theorem tells us that both $\mathcal{L}_{\mathcal{V}}(\beta)$ and $\mathcal{L}_{\mathcal{V}}(\neg\beta)$ are ω-recognizable. Hence by Lemma 5 and Lemma 6, $[\![\varphi]\!] = \min(\mathcal{L}_{\mathcal{V}}(\beta) \cap [\![\varphi_1]\!]_{\mathcal{V}}, \mathcal{L}_{\mathcal{V}}(\neg\beta) \cap [\![\varphi_2]\!]_{\mathcal{V}})$ is also MC-recognizable. ◀

▶ **Lemma 16.** *Let $\varphi_1,\varphi_2 \in \mathrm{mMSO}(\Sigma,\mathrm{Val})$ be such that $[\![\varphi_1]\!]$ and $[\![\varphi_2]\!]$ are MC-recognizable. Then $\varphi = \min(\varphi_1,\varphi_2)$, the series $[\![\varphi]\!]$ is also MC-recognizable.*

Proof. Let $\mathcal{V} = \mathrm{Free}(\varphi_1) \cup \mathrm{Free}(\varphi_2)$, then by Lemma 10, $[\![\varphi_1]\!]_{\mathcal{V}}$ and $[\![\varphi_2]\!]_{\mathcal{V}}$ are also MC-recognizable. Hence by Lemma 5, $[\![\varphi]\!] = \min([\![\varphi_1]\!]_{\mathcal{V}}, [\![\varphi_2]\!]_{\mathcal{V}})$ is also MC-recognizable. ◀

▶ **Lemma 17.** *Let $\varphi \in \mathrm{mMSO}(\Sigma,\mathrm{Val})$ such that $[\![\varphi]\!]$ is MC-recognizable. Then $\inf x.\varphi$ and $\inf X.\varphi$ are also MC-recognizable.*

Proof. We show the lemma for $\inf x.\varphi$. The proof for $\inf X.\varphi$ is similar. Let $\mathcal{V} = \mathrm{Free}(\inf x.\varphi)$, then $x \notin \mathcal{V}$. We now consider the homomorphism $h\colon \Sigma^\omega_{\mathcal{V}\cup\{x\}} \to \Sigma^\omega_{\mathcal{V}}$, which erases the x-row. Then for any $(w,\rho) \in \Sigma^\omega_{\mathcal{V}}$, we have that

$$[\![\inf x.\varphi]\!]_{\mathcal{V}}(w,\rho) = \inf\{[\![\varphi]\!]_{\mathcal{V}\cup\{x\}}(w,\rho[x \to i]) \mid i \geq 0\} = h([\![\varphi]\!]_{\mathcal{V}\cup\{x\}})(w,\rho).$$

As $\mathrm{Free}(\varphi) \subseteq \mathcal{V} \cup \{x\}$, Lemma 10 shows that $[\![\varphi]\!]_{\mathcal{V}\cup\{x\}}$ is MC-recognizable and therefore by Lemma 4 (i) the series $[\![\inf x.\varphi]\!]_{\mathcal{V}}$ is MC-recognizable as well. ◀

▶ **Lemma 18.** *Let ψ be an almost Boolean formula and $\mathcal{V} \supseteq \mathrm{Free}(\psi)$. Then there are MSO formulas $\beta_1,\dots,\beta_n$ and weights $z_1,\dots,z_n \in \mathbb{Z}$ such that $\mathrm{Free}(\psi) = \bigcup_{i=1}^n \mathrm{Free}(\beta_i)$, $N_{\mathcal{V}} = \bigcup_{i=1}^n \mathcal{L}_{\mathcal{V}}(\beta_i)$, for $i \neq j$ we have $\mathcal{L}_{\mathcal{V}}(\beta_i) \cap \mathcal{L}_{\mathcal{V}}(\beta_j) = \emptyset$ and for $(w,\rho) \in N_{\mathcal{V}}$ we have $[\![\psi]\!]_{\mathcal{V}}(w,\rho) = z_i$ if and only if $(w,\rho) \in \mathcal{L}_{\mathcal{V}}(\beta_i)$.*

Proof. For $\psi = k$ with $k \in \mathbb{Z}$, we choose β_1 as any tautology, for example $\beta_1 = \exists x.x \leq x$, and $z_1 = k$.

For $\psi = \beta \mathbin{?} \psi_1 : \psi_2$ we assume that the lemma is true for ψ_1 with $\beta_1^{(1)}, \dots, \beta_{n_1}^{(1)}$ and $z_1^{(1)}, \dots, z_{n_1}^{(1)}$ and for ψ_2 with $\beta_1^{(2)}, \dots, \beta_{n_2}^{(2)}$ and $z_1^{(2)}, \dots, z_{n_2}^{(2)}$. Then for ψ we choose $\beta_1, \dots, \beta_{n_1+n_2}$ and $z_1, \dots, z_{n_1+n_2}$ as follows. For $i \in \{1, \dots, n_1\}$ we set $\beta_i = \beta \wedge \beta_i^{(1)}$ and $z_i = z_i^{(1)}$ and for $i \in \{1, \dots, n_2\}$ we set $\beta_{n_1+i} = \neg\beta \wedge \beta_i^{(2)}$ and $z_{n_1+i} = z_i^{(2)}$. ◀

▶ **Lemma 19.** *Let ζ be an x-summing formula and $\mathcal{V} \supseteq \mathrm{Free}(\zeta)$. Then there are MSO formulas $\beta_1, \dots, \beta_n$ and formulas $\zeta_1, \dots, \zeta_n$ with $\zeta_i = \bigoplus^{x,Y_i} y.\psi_i$ for some almost Boolean formula ψ_i such that $\mathrm{Free}(\zeta) = \bigcup_{i=1}^n \mathrm{Free}(\beta_i) \cup \mathrm{Free}(\zeta_i)$, for $i \neq j$ we have $\mathcal{L}_\mathcal{V}(\beta_i) \cap \mathcal{L}_\mathcal{V}(\beta_j) = \emptyset$, for $(w, \rho) \in N_\mathcal{V}$ we have $[\![\zeta]\!]_\mathcal{V}(w, \rho) = [\![\zeta_i]\!]_\mathcal{V}(w, \rho)$ if and only if $(w, \rho) \in \mathcal{L}_\mathcal{V}(\beta_i)$ and if $(w, \rho) \notin \bigcup_{i=1}^n \mathcal{L}_\mathcal{V}(\beta_i)$ then $[\![\zeta]\!](w, \rho) = \mathbb{1}$. We can assume the variables Y_i to be pairwise distinct.*

▶ **Theorem 20.** *Let ζ be an x-summing formula. Then $[\![\mathrm{Val}\, x.\zeta]\!]$ is MC-recognizable.*

Proof (sketch). We adapt and expand an idea from [9]. Let $\beta_1, \dots, \beta_n$ and $\zeta_1, \dots, \zeta_n$ be the formulas we can find for ζ according to Lemma 19. We write $\zeta_i = \bigoplus^{x,Y_i} y.\psi_i$. Then for each $i \in \{1, \dots, n\}$, let $\beta_{i1}, \dots, \beta_{in_i}$ and $z_{i1}, \dots, z_{in_i}$ be the formulas and weights we can find for ψ_i according to Lemma 18.

The proof idea is as follows. For $\mathcal{V} = \mathrm{Free}(\mathrm{Val}\, x.\zeta)$, the mapping $[\![\mathrm{Val}\, x.\zeta]\!]$ assigns values to words from $\Sigma_\mathcal{V}^\omega$. Consider $(w, \rho) \in \Sigma_\mathcal{V}^\omega$. We can interpret each ζ_i as a counter which is stopped and then restarted at the k-th letter of w depending on whether $(w, \rho[x \to k])$ satisfies β_i. As our automata cannot stop and start a single counter at the same time, each counter i will correspond to two counters i and i' in the automaton we construct. The computations of counter i depend on $\beta_{i1}, \dots, \beta_{in_i}$. We extend the alphabet $\Sigma_\mathcal{V}$ by adding two entries for each counter to each letter in $\Sigma_\mathcal{V}$. The entries for counter i can contain an s to indicate the start of the counter, a t to indicate a stop, a number $j \in \{1, \dots, n_i\}$ to indicate that the counter is active and should add z_{ij} to its current value, or a $\perp$ to indicate that the counter is inactive. Let $\tilde{\Sigma}_\mathcal{V}$ be this new alphabet. We show that we can define an ω-recognizable language L over $\tilde{\Sigma}_\mathcal{V}$ which has all information about the counter operations encoded in the word. For example, if $(w, \rho[x \to k]) \models \beta_i$, then in the word $(w, \rho, v) \in \tilde{\Sigma}_\mathcal{V}^\omega$ corresponding to (w, ρ) the entry for counter i in the k-th letter should contain an s. Then if $(w, \rho[x \to k, y \to k+1]) \models \beta_{ij}$, the i-entry of the $k+1$-th letter should contain a j. The precise formulation of this is involved.

When we have shown that the language L is recognizable, we can construct a deterministic Muller automaton $\tilde{\mathcal{A}}$ which recognizes L. Turning $\tilde{\mathcal{A}}$ into an MMCA and applying a projection, we finally obtain the recognizability of $[\![\mathrm{Val}\, x.\zeta]\!]$. ◀

This concludes our induction, and thus the proof of Theorem 13.

6 Conclusion

We introduced a new logic which is expressively equivalent to Quantitative Monitor Automata. Since our proofs are constructive, we immediately obtain the possibility to reduce the satisfiability and equivalence problems of our logic to the emptiness and equivalence problems of Quantitative Monitor Automata. Future work could therefore focus on the investigation of this automaton model, and on the related model of Nested Weighted Automata [3].

References

1 J. Richard Büchi. Weak second-order arithmetic and finite automata. *Z. Math. Logik und Grundl. Math.*, 6:66–92, 1960.

2 Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Quantitative languages. In Michael Kaminski and Simone Martini, editors, *Proc. CSL*, volume 5213 of *LNCS*, pages 385–400. Springer, 2008.

3 Krishnendu Chatterjee, Thomas A. Henzinger, and Jan Otop. Nested weighted automata. In *Proc. LICS*, pages 725–737, 2015.

4 Krishnendu Chatterjee, Thomas A. Henzinger, and Jan Otop. Quantitative monitor automata. In Xavier Rival, editor, *Proc. SAS*, volume 9837 of *LNCS*. Springer, 2016.

5 Manfred Droste and Stefan Dück. Weighted automata and logics on graphs. In Giuseppe F. Italiano, Giovanni Pighizzini, and Donald T. Sannella, editors, *Proc. MFCS*, volume 9234 of *LNCS*, pages 192–204. Springer, 2015.

6 Manfred Droste and Paul Gastin. Weighted automata and weighted logics. *Theor. Comput. Sci.*, 380:69–86, 2007.

7 Manfred Droste, Werner Kuich, and Heiko Vogler, editors. *Handbook of Weighted Automata.* Monogr. Theoret. Comput. Sci. EATCS Ser. Springer, 2009.

8 Manfred Droste and Ingmar Meinecke. Weighted automata and weighted MSO logics for average and long-time behaviors. *Inform. Comput.*, 220–221:44–59, 2012.

9 Manfred Droste and George Rahonis. Weighted automata and weighted logics on infinite words. In *Proc. DLT*, volume 4036 of *LNCS*, pages 49–58. Springer, 2006.

10 Calvin C. Elgot. Decision problems of finite automata design and related arithmetics. *Trans. Am. Math. Soc.*, 98(1):21–51, 1961.

11 Zoltán Ésik and Werner Kuich. A semiring-semimodule generalization of ω-regular languages I. *Journal of Automata, Languages and Combinatorics*, 10(2/3):203–242, 2005.

12 Zoltán Ésik and Werner Kuich. A semiring-semimodule generalization of ω-regular languages II. *Journal of Automata, Languages and Combinatorics*, 10(2/3):243–264, 2005.

13 Ina Fichtner. Weighted picture automata and weighted logics. *Theor. Comput. Syst.*, 48(1):48–78, 2011.

14 Christian Mathissen. Weighted logics for nested words and algebraic formal power series. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *Proc. ICALP*, volume 5126 of *LNCS*, pages 221–232. Springer, 2008.

15 Wolfgang Thomas. Automata on infinite objects. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science (Vol. B)*, pages 133–191. Elsevier Science, 1990.

16 Wolfgang Thomas. Languages, automata, and logic. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages (Vol. 3)*, pages 389–455. Springer, 1997.

17 Boris Avraamovich Trakhtenbrot. Finite automata and logic of monadic predicates. *Doklady Akademii Nauk SSSR*, 140:326–329, 1961. In Russian.