LIPIcs Leibniz International Proceedings in Informatics

# Deterministic Automata and Extensions of Weak MSO

**Mikołaj Bojańczyk, Szymon Toruńczyk***
University of Warsaw
{bojan,szymtor}@mimuw.edu.pl

ABSTRACT. We introduce a new class of automata on infinite words, called min-automata. We prove that min-automata have the same expressive power as weak monadic second-order logic (weak MSO) extended with a new quantifier, the recurrence quantifier. These results are dual to a framework presented in [2], where max-automata were proved equivalent to weak MSO extended with an unbounding quantifier. We also present a general framework, which tries to explain which types of automata on infinite words correspond to extensions of weak MSO. As another example for the usefulness framework, apart from min- and max-automata, we define an extension of weak MSO with a quantifier that talks about ultimately periodic sets.

## Introduction

In [2], a new class of languages of infinite words was defined. This class had two equivalent descriptions: in terms of a deterministic counter automaton (called a max-automaton), and in terms of an extension of weak monadic second-order logic (weak MSO). The argument raised in [2] was that there are robust extensions of $\omega$-regular languages, extensions that have descriptions in terms of both automata and logic. This paper further investigates that argument. These are the contributions:

1. We define a type of automaton dual to max-automata, called a min-automaton, and prove that it is equivalent to a certain extension of weak MSO.
2. We show that min- and max-automata fit in a general picture, where deterministic automata with prefix-closed acceptance conditions define extensions of weak MSO.
3. As another example of the general picture, we present an extension of weak MSO, together with a corresponding automaton, that talks about ultimately periodic sets.

Below we describe these contributions in more detail.

**Min-automata.** A max-automaton, as defined in [2], works as follows. It is a deterministic automaton, but it also has a finite set $C$ of counters, which store natural numbers. Each transition is decorated by a sequence of counter operations, which are from the set

$$Op = \{c := c + 1\,, \quad c := \max(d, e) \qquad : \qquad c, d, e \in C\}.$$

(The toolkit of operations in [2] was slightly different, but the simpler one above is equivalent.) There are two key properties of the model. First, the automaton is deterministic, which is important for the connection with weak MSO. Second, the choice of the next state is not influenced by the counter values, but only the current state and input letter; one can

*Work partially funded by the Polish government grant no. N206 008 32/0810


Foundations of Software Technology and Theoretical Computer Science (Kanpur) 2009.
Editors: Ravi Kannan and K. Narayan Kumar; pp 73–84
Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany.
Digital Object Identifier: 10.4230/LIPIcs.FSTTCS.2009.2308

somehow think of the counter operations being applied after the run is chosen. The only place where the counters are read is the acceptance condition, which is a boolean combination of conditions

$$\limsup_{i\to\infty} val(c, a_1 a_2 \dots a_i) = \infty,$$

where $val(c, u)$ is the value of counter $c$ after reading a finite prefix $u$ of the input word.

The main contribution of [2] is that max-automata are equivalent to weak MSO extended with a quantifier, called the unbounding quantifier. The unbounding quantifier binds a set variable $X$ in a formula $\varphi(X)$ and is true if there are sets $X$ of arbitrarily large finite size that satisfy $\varphi(X)$.

If an automaton with the max operation has a matching logic, then what about min? What if we use lim inf instead of lim sup in the acceptance condition? In this paper we analyze such an automaton model, called a min-automaton, where min is used instead of max, and the acceptance condition uses lim inf instead of lim sup. We show that min-automata also have a corresponding logic. Note that there are other combinations, which we do not study here, such as automata that use max and lim inf.

What is the logic that corresponds to min-automata? As was the case for max-automata, this is an extension of weak MSO, where a new quantifier is added. The quantifier for min-automata, which we introduce in this paper and call the recurrence quantifier, says: "there is some $n \in \mathbb{N}$ such that infinitely many sets of size $n$ satisfy $\varphi(X)$". One of our main results, Theorem 8, is that min-automata have the same expressive power as weak MSO extended with the recurrence quantifier.

**General Framework.** Although we think that min-automata are interesting in their own right, we also think that they are part of a bigger picture for deterministic automata on infinite words. The bigger picture is that any "reasonable" acceptance condition seems to give a robust class of languages extending weak MSO. We present some preliminary results that attempt to formalise these ideas.

One consequence of our results is a normal form theorem: any formula of weak MSO extended with both the unbounding quantifier (the quantifier related to max-automata) and the recurrence quantifier (the quantifier related to min-automata) is effectively equivalent to a boolean combination of formulas, each of which has at most one occurrence of the new quantifiers (bounding or recurrence). In other words, mutual nesting of the new quantifiers does not contribute to the expressive power. This normal form can be used to decide satisfiability for weak MSO extended with both quantifiers, since the algorithm only needs to test emptiness for boolean combinations of (actually, conjunctions of) max- and min-automata.

**Ultimately Periodic Quantifier.** As an example of the bigger picture, we consider an extension of weak MSO with the ultimately periodic quantifier. This quantifier binds a first-order variable in a formula $\varphi(x)$ and says that the set of word positions that satisfy $\varphi(x)$ is ultimately periodic. We present an equivalent automaton model, where the acceptance condition says that certain states appear in an ultimately periodic way, and certain other states

do not. Using this model, and some combinatorics, we prove that satisfiability is decidable for weak MSO with the ultimately periodic quantifier.

**Background and related work.** The idea of considering extensions of $\omega$-regular languages is not new, dating back to the sixties. One line of work has been to add new predicates, such as a predicate $square(x)$, which holds for positions that are square numbers. This line was started by [7], and continued in [5, 11, 10].

More closely related to this paper is the work on the unbounding quantifier. This quantifier was introduced in [3]. The satisfiability problem for full MSO (as opposed to weak MSO, the subject of this paper) extended with the unbounding quantifier was tackled [4]. By introducing an automaton model, called a BS-automaton, [4] provided some fragments of full MSO with the unbounding quantifier that have decidable satisfiability over infinite words. A BS-automaton is a counter automaton with acceptance conditions as in max- and min-automata, but, crucially, it is nondeterministic. Nondeterminism is important for full MSO, where existential quantification over infinite sets is allowed. Nondeterminism also increases the flexibility of the model (for instance, the max and min operations become redundant). There is no free lunch, however: nondeterministic BS-automata are not closed under complement, and it is not clear what is the correct automaton model for full MSO with the unbounding quantifier. It is still an open problem if full MSO extended with the unbounding quantifier has decidable satisfiability over infinite words.

BS-automata have also been considered in [1], under the name of R-automata. BS-automata are also closely related to distance desert automata, which were used by Kirsten to decide the star height problem [8]. A tree variant of distance desert automata was introduced in [6], to decide star height for tree languages.

**Acknowledgments.** We would like to thank Eryk Kopczyński, Sławomir Lasota, Aymeric Vincent and Thomas Wilke for many stimulating discussions.

## 1 Min-automata

In this section we introduce min-automata. The idea is that a min-automaton has a finite set of counters that store natural numbers, and each transition is labeled by a finite sequence of counter operations, taken from the set

$$Op_C = \{c := c + 1\,, \quad c := \min(d, e) \qquad : \qquad c, d, e \in C\}.$$

Formally, a deterministic min-automaton consists of:

| | |
|---|---|
| $A$ | The alphabet of the automaton |
| $Q$ | A finite set of states of the automaton |
| $C$ | A finite set of counters of the automaton |
| $\delta$ | The state transition function, $\delta\colon Q \times A \to Q$ |
| $\gamma$ | The counter update function, $\gamma\colon Q \times A \to (Op_C)^*$ |
| $q_0$ | The initial state, $q_0 \in Q$ |
| $v_0$ | The vector of initial counter values, $v_0 \in \mathbb{N}^C$ |
| $F$ | The acceptance condition, described below. |

Given a finite word $w \in A^*$, the automaton produces a unique run $\rho \in Q^*$. By applying the counter update function $\gamma$ to this run, we get a sequence $\pi \in (Op_C)^*$ of counter operations. By applying this sequence of operations to the initial counter valuation $v_0$, we get a counter valuation written $val(c, w)$.

The acceptance condition $F$ is the only place where the counters are read. It talks about the asymptotic[†] values of the counters when reading an input word $a_1 a_2 \cdots \in A^\omega$. It is a boolean combination of conditions

$$\liminf_{i \to \infty} val(c, a_1 \cdots a_i) = \infty. \tag{1}$$

In the automaton, the above condition is represented in the formula $F$ by an atom $c$ for short.

In particular, the class of languages accepted by min-automata is closed under complementation, since replacing the acceptance condition $F$ by $\neg F$ gives an automaton recognizing the complement language, thanks to determinism. Closure under alternative and conjunction follows from the usual cartesian product construction.

If the counters would influence the states, such as by having a zero-test counter operation, we would lose all the robust decidability of the model. It is crucial that as far as choosing the states is concerned, a min-automaton behaves just like a finite deterministic automaton.

**EXAMPLE 1.** With each infinite sequence of natural numbers $n_1, n_2, n_3 \ldots$, we may associate an infinite word

$$a^{n_1} b\, a^{n_2} b\, a^{n_3} b \ldots$$

Let $L$ be the set of words associated with sequences where $\liminf n_i < \infty$. Then $L$ is recognized by a deterministic min-automaton with one state, three counters $c, d, z$ and the following instructions.

- when reading $a$, do $c := c + 1$,
- when reading $b$, do $d := min(c, c)$; $c := z$.

The initial valuation is $(0, 0, 0)$. Counter $c$ stores the size of the current $a$ block, while counter $d$ stores the size of the last complete $a$ block. Counter $z$ always stores 0, and is used to reset

[†]Since the acceptance condition is insensitive to finite perturbations, the initial counter valuation does not influence the accepted language. The initial counter valuation will play a role for automata in matrix form.

counter $c$ when a block of $a$'s is finished. The acceptance condition is $F = \neg c \wedge \neg d$: both counters $c$ and $d$ should have $\liminf < \infty$ (counter $z$ is not mentioned in the acceptance condition).

The above example shows how counter operations $c := 0$ and $d := c$ can be implemented in the model.

The following lower bound on the complexity of emptiness is via a reduction from the universality problem for nondeterministic automata. This is also a partial answer to a question posed in [2], which asked about the complexity of emptiness for max-automata (the same proof works for max-automata).

**THEOREM 2.** *Emptiness is* PSPACE*-hard for min-automata.*

**Determinism.** Does determinism restrict the expressive power of min-automata? It does for max-automata: in [2], it was shown that nondeterministic max-automata can, while deterministic max-automata cannot, recognize the language

$$L = \{a^{n_1} b\, a^{n_2} b\, a^{n_3}\, b \ldots : \liminf n_i < \infty\}.$$

The reason why a nondeterministic max-automaton can recognize $L$ is that a sequence has $\liminf < \infty$ if and only if it has a subsequence of $\limsup < \infty$, and the subsequence can be nondeterministically guessed. The reason why deterministic max-automata cannot recognize this language is that $L$ is on level $\Sigma_3$ of the Borel hierarchy, while deterministic max-automata can only recognize languages that are boolean combinations of $\Sigma_2$ languages.

For min-automata, one can prove that nondeterministic min-automata can, while deterministic min-automata cannot, recognize the language

$$K = \{a^{n_1} b\, a^{n_2} b\, a^{n_3}\, b \ldots : \limsup n_i = \infty\}.$$

The reason why a nondeterministic min-automaton can recognize $K$ is the same as in the counterexample for max-automata. However, how does one prove that a deterministic min-automaton cannot recognize $K$? The topological argument no longer works, since $K$ is on level $\Pi_2$ of the Borel hierarchy, while deterministic min-automata can recognize even $\Sigma_3$ languages, such as the language $L$. One idea would be to change the topology, to one where min-automata would be simpler than max-automata, but we could not find such a topology. Our solution uses pumping arguments.

**Relationship with BS-automata.** In this section we talk about translating min- and max-automata into BS-automata, as defined in [4]. BS-automata are like min- or max-automata, with three differences: (i) they are nondeterministic; (ii) they do not have the min and max counter operations, only increment and reset; and (iii) the acceptance condition can speak of both $\liminf$ and $\limsup$. In [2] it was shown how to convert a max-automaton to a nondeterministic BS-automaton. The same technique works for min-automata, so we get:

**THEOREM 3.** *Every max-automaton is effectively equivalent to a nondeterministic BS-automaton. The same holds for min-automata.*

**COROLLARY 4.** *Emptiness is decidable for boolean combinations of min- and max-automata.*

PROOF. Since max- and min-automata are closed under boolean operations, the problem is equivalent to testing emptiness for positive boolean combinations. Since BS-automata are closed under positive boolean combinations, every boolean combination of max- and min-automata is effectively equivalent to a BS-automaton. Emptiness of BS-automata is decidable by [4]. ∎

The complexity of the above procedure is quite high, especially due to the high cost of translating a max-automaton into a BS-automaton (the current algorithm is nonelementary). It would be nice to get an upper bound that is closer to the PSPACE lower bound from Theorem 2.

BS-automata do not have the min operation, and yet they are still able to capture min-automata. The translation from min-automata to BS-automata introduces nondeterminism. One might ask: is the min counter operation necessary in a deterministic min-automaton? (After removing the min-operation, we add a substitution operation $c := d$ and a reset operation $c := 0$, and we still keep the acceptance condition that talks about lim inf.) Notice how the automaton in Example 1 does not really use the min operation, only the substitution. In preliminary work, we have proved that min-automata without min are less expressive.

Below we describe the separating example. The alphabet is $a, b, c, d$. Let

$$w = a^{n_1} b a^{n_2} b \cdots a^{n_k} b$$

be a word in $(a^*b)^+$. For $\sigma \in \{c, d\}$ we define $\overline{w\sigma}$ to be $\min(n_1, \ldots, n_k)$ if $\sigma = c$ and $\infty$ otherwise. The separating language is

$$\{w_1\sigma_1 w_2\sigma_2 \ldots \in ((a^*b)^+(c+d))^\omega : \liminf \overline{w_i\sigma_i} = \infty\}$$

It is easy to define a min-automaton that recognizes the above language. The proof that an automaton without min cannot recognize this language requires a pumping argument, and will be given in a full version of this paper.

**A matrix representation.** In this section we represent the automata by matrices.

We extend slightly the definition of min-automata and allow an additional value $\top$, called the *undefined* value. As far as the min operation is concerned, the values are ordered $0 < 1 < \ldots < \top$. We extend addition to the new counter values by setting:

$$\top + x = x + \top = \top \qquad \text{for all } x.$$

We write $\mathcal{T}$ for the extended set $\{0, 1, 2, \ldots, \top\}$ of counter values. Together with the two operations above $\mathcal{T}$ forms a semiring, where the addition operation is min and the multiplication operation is $+$. This semiring is called *the tropical semiring*, or $(min, +)$ semiring, see e.g. [9].

The new counter values can be eliminated, by storing in the states the information about which counters are $\top$. The undefined counter value $\top$ will become important in the matrix representation, where it will be used to eliminate states from the automaton.

Let $\mathbb{M}_C\mathcal{T}$ denote the semiring of $C \times C$ matrices with entries from $\mathcal{T}$. Suppose that $M \in \mathbb{M}_C\mathcal{T}$. We can treat $M$ as a counter operation, which changes a counter valuation, treated as a vector $v \in \mathcal{T}^C$, to $v \cdot M \in \mathcal{T}^C$. This type of operation can be implemented by a min-automaton, possibly after introducing auxiliary counters.

**EXAMPLE 5.** Let us return to the automaton from Example 1. When reading a letter $a$, the automaton would perform the operations $c := c + 1$. In matrix form, this is written as

$$\begin{pmatrix} c & d & z \end{pmatrix} \quad := \quad \begin{pmatrix} c & d & z \end{pmatrix} \cdot \begin{pmatrix} 1 & \top & \top \\ \top & 0 & \top \\ \top & \top & 0 \end{pmatrix}.$$

When reading $b$, the automaton would do $d := min(c, c);\ c := z$. In matrix form, this is

$$\begin{pmatrix} c & d & z \end{pmatrix} \quad := \quad \begin{pmatrix} c & d & z \end{pmatrix} \cdot \begin{pmatrix} \top & 0 & \top \\ \top & \top & \top \\ 0 & \top & 0 \end{pmatrix}.$$

Every sequence of counter operations can be represented in a matrix form as in the above example. In a *min-automaton in matrix* form, the counter operations are implemented by matrices, and the choice of the matrix only depends on the last letter seen (so there is no state). Such an automaton is given by an initial vector and a matrix for each letter of the input alphabet, so it is a tuple

$$\langle A, C, \gamma : A \to \mathbb{M}_C\mathcal{T}, v_0 \in \mathcal{T}^C, F \rangle.$$

After reading a word $a_1 \cdots a_n$, the counter valuation is

$$v_0 \cdot \gamma(a_1) \cdot \gamma(a_2) \cdots \gamma(a_n).$$

**PROPOSITION 6.** *For every min-automaton one can construct an equivalent min-automaton in matrix form. If the input automaton has $n$ states and $m$ counters, the output automaton has $(m + 1) \times n$ counters.*

PROOF. [sketch] By storing the state information in the counters which use the value $\top$. Each counter has one copy corresponding to each of the automaton states, and all but one of the copies are undefined at any moment. ∎

What is the point of the matrix representation? One advantage is that it underlies the close connection with existing work on distance automata and formal power series, where matrices over the tropical semiring play an important role. We would like to further investigate this connection, especially how the PSPACE upper bound on the limitedness problem for distance automata can be used for testing emptiness of min automata.

Another advantage is that we can eliminate states from the automaton. This is more an advantage of the $\top$ counter value. Having a stateless automaton enormously simplifies combinatorics, for instance in the proof that deterministic min-automata cannot recognize the language $K$ defined earlier, and hence nondeterministic min-automata cannot be determinized.

## 2 Weak MSO with the recurrence quantifier

In [2], max-automata were proved to have the same expressive power as weak MSO extended with a new quantifier, called the unbounding quantifier (denoted U). For min-automata, the situation is the same, only a different quantifier is needed. Before introducing the new quantifier, we recall the definition of weak MSO. In weak MSO over infinite words we may:

- quantify over finite sets of positions (the $\exists_{\text{fin}} X$ quantifier) and single positions (the $\exists x$ quantifier),
- verify that a position belongs to a set of positions ($x \in X$),
- verify that one position comes before another ($x \leq y$),
- check the label standing on a position ($a(x)$ for each label $a \in A$),
- use boolean operations ($\wedge, \vee, \neg$).

Weak MSO corresponds to deterministic Muller automata over infinite words, which, thanks to the theorem of McNaughton, define all $\omega$-regular languages. The goal of this section is to show this correspondence for min-automata, by adding a new quantifier, called the recurrence quantifier.

**The recurrence quantifier** The recurrence quantifier, written R, binds a set variable $X$ in a formula $\varphi(X)$ and is true if there are infinitely many sets $X$ of equal size that satisfy $\varphi(X)$. More precisely, $\mathsf{R}X.\varphi(X)$ is satisfied in a word $w$ if there exists a number $N \in \mathbb{N}$ and infinitely many sets $X$ of size $N$ such that $\varphi(X)$ is satisfied in $w$.

**EXAMPLE 7.** Let $\varphi$ be a formula with a free set-variable $X$ which says that $X$ is connected and has at least two $b$'s. Formally,

$$\varphi(X) = \wedge \begin{cases} \forall x \forall y \forall z & x \in X \quad \wedge \quad z \in X \quad \wedge \quad x \leq y \leq z \quad \Rightarrow \quad y \in X \\ \exists x \exists y & x < y \quad \wedge \quad b(x) \quad \wedge \quad b(y) \quad \wedge \quad x \in X \quad \wedge \quad y \in X \end{cases}$$

A word $a^{n_1} b\, a^{n_2} b \ldots$ satisfies $\mathsf{R}X.\varphi(X)$ if and only if $\liminf n_i < \infty$. Therefore, the set of words with infinitely many $b$'s that satisfy $\mathsf{R}X.\varphi(X)$ is the language $L$ from Example 1.

**THEOREM 8.** *Weak MSO logic with the recurrence quantifier recognizes the same class of languages as min-automata.*

This theorem is a special case of Theorem 11, stated in the next section.

## 3 General framework

In the previous section, we defined min-automata and stated that they are equivalent to weak MSO with the recurrence quantifier. This is analogous to the situation for max-automata, where the appropriate quantifier is the unbounding quantifier. Converting an automaton into a formula is straightforward, while converting a formula into an automaton can be done thanks to some general properties shared by min- and max-automata. We would like to bring out these similarities, by introducing a more abstract framework.

**The automaton side** The control structure of deterministic min-automata, max-automata, Büchi automata, etc. is always the same, it is only the mode of acceptance that changes. We give an abstract definition below, by modeling an acceptance condition as a language $F \subseteq B^\omega$. The definition uses the notion of a letter to letter transducer, by which we understand a finite deterministic automaton with input alphabet $A$, whose transitions are labelled by letters of an output alphabet $B$. This transducer maps every word in $A^*$ to a word in $B^*$ of same length. We will use a transducer on infinite words, where it will give a function $A^\omega \to B^\omega$. Note that the transducers have no acceptance condition.

**DEFINITION 9.** *An automaton with acceptance condition $F \subseteq B^\omega$ (or simply $F$-automaton) $\mathcal{A}$ is a deterministic letter-to-letter transducer with input alphabet $A$ and output alphabet $B$. We say that $\mathcal{A}$ accepts an input word $w \in A^\omega$ if the output word belongs to $F$. Languages accepted by $F$-automata are called $F$-regular.*

One example of this definition is a Büchi automaton. In this case, the acceptance condition is any language of the form $(B^*C)^\omega \subseteq B^\omega$, for $C \subseteq B$. In a similar way we can encode Muller or parity automata.

For min- or max-automata, the same can be done. In this case, the alphabet of the acceptance condition consists of words over the set of counter operations, and the acceptance condition contains those infinite sequences of counter operations where the appropriate limits are $\infty$.

We are mainly interested in *prefix-independent* acceptance conditions, namely languages $F \subseteq B^\omega$ that satisfy $F = B^*F$. All the examples mentioned above are prefix-independent. (In the case of min- or max-automata, to get prefix-independence we should not use the matrix form of automata, but the original definition, where the counters have values in $\mathbb{N}$.)

**The logic side** Let us call a *locus* any family $\mathcal{X}$ of finite sets of positions. Let a given input word be fixed. A formula $\varphi(X)$ with a free set-type variable $X$ defines its locus $\mathcal{X}_\varphi$ as the family of finite sets of positions $X$ which satisfy $\varphi$. A *locus property* $\mathsf{Q}$ is any set of loci. If $\mathsf{Q}$ is a locus property, then we write $\mathsf{Q}X.\varphi(X)$ if $\mathcal{X}_\varphi \in \mathsf{Q}$. The quantifiers $\exists_{\mathsf{fin}}$, $\mathsf{U}$, $\mathsf{R}$, $\mathsf{P}$ (defined in the next section) all arise in this fashion. For instance, for a locus $\mathcal{X}$, $\mathcal{X} \in \exists_{\mathsf{fin}}$ if it is nonempty, while $\mathcal{X} \in \mathsf{U}$ if it contains arbitrarily large sets.

For two loci $\mathcal{X}$ and $\mathcal{Y}$, we write $\mathcal{X} \simeq \mathcal{Y}$ if $\mathcal{X}$ and $\mathcal{Y}$ differ by a finite number of sets. We call $\mathsf{Q}$ *finitely invariant* if $\mathsf{Q}$ is invariant under $\simeq$, i.e. if $\mathcal{X} \in \mathsf{Q}$ and $\mathcal{X} \simeq \mathcal{Y}$, then $\mathcal{Y} \in \mathsf{Q}$. Examples of finitely invariant locus properties are $\mathsf{U}, \mathsf{R}, \mathsf{P}$. On the other hand, $\exists_{\mathsf{fin}}$ is not finitely invariant.

A $\mathsf{Q}$-*formula* is a formula $\mathsf{Q}X.\varphi(X)$ where $\varphi(X)$ is a formula of WMSO with only one free variable, namely $X$. An open $\mathsf{Q}$-*formula* is a $\mathsf{Q}$-formula where $\varphi$ is open in the following sense: if a word $w$ together with a set $X$ satisfies $\varphi(X)$, then there is some finite prefix of $w$ such that changing the word $w$ on positions outside the prefix does not affect the truth value of $\varphi(X)$.

**Quantifier elimination** Here we present our main result, which shows how quantifiers can be denested in the scope of a formula of WMSO. Since the theorem talks about automata and languages, a quantifier is viewed as an operation on languages, which takes a language

over an alphabet $A \times \{0,1\}$ and returns a language over an alphabet $A$. In the following, for a word $w$ over alphabet $A$ and a set of positions $X$, we write $w \otimes X$ for the word over alphabet $A \times \{0,1\}$ that has the labels of $w$ on the first coordinate and the characteristic function of $X$ on the second coordinate.

**THEOREM 10.** *Let $F$ be a prefix-independent acceptance condition and let $\mathsf{Q}$ be a locus property. If $L$ is an $F$-regular language over the alphabet $A \times \{0,1\}$, then the language*

$$\mathsf{Q}L = \{w \in A^\omega : \mathsf{Q}X.[w \otimes X \in L]\}$$

*is a boolean combination of $F$-regular languages, $\omega$-regular languages, and $\mathsf{Q}$-formulas. Moreover, if $\mathsf{Q}$ is finitely invariant, then the $\mathsf{Q}$-formulas are open.*

Here is an important corollary of the above result.

**THEOREM 11.** *Weak MSO extended by both the recurrence quantifier $\mathsf{R}$ and the unbounding quantifier $\mathsf{U}$ defines the same languages as boolean combinations of max-automata and min-automata. If the formula does not use $\mathsf{R}$, then min-automata are not used in the combination, likewise for $\mathsf{U}$ and max-automata.*

The above theorem also gives a normal form for weak MSO with the quantifiers $\mathsf{R}$ and $\mathsf{U}$. Take a formula $\varphi$ of the logic, compile it into a boolean combination of automata as in the above corollary, and then compile each of those automata back into a formula. What we end up with is a boolean combination of formulas of the form $\mathsf{R}X.\varphi(X)$ or $\mathsf{U}X.\varphi(X)$, where $\varphi(X)$ is a formula of weak MSO without $\mathsf{R}$ or $\mathsf{U}$. In other words, nesting the quantifiers $\mathsf{R}$ and $\mathsf{U}$ does not contribute anything to the expressive power of weak MSO.

## 4 Ultimately Periodic Quantifier

In this section we present another extension of weak MSO, and use the general framework to show that its emptiness problem is decidable.

The *ultimately periodic quantifier*, written $\mathsf{P}$, is used to say that a set of positions is ultimately periodic. Specifically, if $\varphi$ is a formula, and $x$ is a first-order variable free in $\varphi$, then $\mathsf{P}x.\varphi(x)$ is true in a word if the set of positions $x$ that satisfy $\varphi$ is ultimately periodic (the variable $x$ gets bound by the quantifier).

We now use the framework from the previous section to present an automaton model that captures weak MSO extended with the ultimately periodic quantifier. For $L \subseteq A^\omega$ and a word $a_1a_2 \ldots \in A^\omega$, we write

$$\text{suffix}_L(a_1a_2\ldots) = \{i \in \mathbb{N} : a_ia_{i+1}\ldots \in L\}$$

We define $\mathsf{PS}_L$ to be the set of words $w \in A^\omega$ where $\text{suffix}_L(w)$ is ultimately periodic. Any language of the form $\mathsf{PS}_L$ is called an *ultimately periodic acceptance condition*.

**COROLLARY 12.** *Weak MSO extended with the ultimately periodic quantifier has the same expressive power as boolean combinations of deterministic automata with Büchi and ultimately periodic acceptance conditions.*

PROOF. The nontrivial translation, from logic to automata, follows from Theorem 10. ∎

**THEOREM 13.** *Satisfiability is decidable for weak MSO extended with the ultimately periodic quantifier.*

PROOF. By Corollary 12, it suffices to decide emptiness for a boolean combination of deterministic automata with Büchi and ultimately periodic acceptance conditions. (The translations between formulas and automata are effective.) Since the acceptance conditions concerned are closed under homomorphic images, we may assume that the same transducer $f : A^\omega \to B^\omega$ is used by all automata. We may also assume that the boolean combination is in DNF form, and as far as emptiness is concerned, has only one disjunct (which is a conjunction of, possibly negated, acceptance conditions). Finally, by collapsing the Büchi languages into a single $\omega$-regular language, we may assume one conjunct is $\omega$-regular, and all others involve ultimately periodic acceptance conditions.

Summing up: we want to decide if the transducer $f$ can output a word in an intersection $K \cap K_1 \cap \cdots \cap K_n$, where $K$ is $\omega$-regular and each $K_i$ is either a language $\mathsf{PS}_{L_i}$ or its complement, for some $\omega$-regular language $L_i$. It is not difficult to see that the following language over alphabet $\{0,1\}^n$ is $\omega$-regular:

$$M = \{X_1 \otimes \cdots \otimes X_n : \text{ exists } w \in f(A^\omega) \cap K \text{ such that } X_i = \mathrm{suffix}_{L_i}(w) \text{ for all } i = 1, \ldots, n\}$$

(here $\otimes$ combines characteristic functions of sets into a word over the product alphabet).

The emptiness problem boils down to testing if the set $M$ above contains a word, whose projection onto coordinates $i$ corresponding to languages $\mathsf{PS}_{L_i}$ is an ultimately periodic word, and whose projection onto coordinates $i$ corresponding to complements of languages $\mathsf{PS}_{L_i}$ is not ultimately periodic. This way we have reduced our satisfiability problem to the following combinatorial result, which can be solved using standard automata techniques. ∎

**THEOREM 14.** *The following problem is decidable*

- *Input: An $\omega$-regular language $L \subseteq B^\omega$, letter-to-letter homomorphisms $\pi_i : B^\omega \to B_i^\omega$ for $i = 1, \ldots, n$, and a set $F \subseteq \{1, \ldots, n\}$.*
- *Question: Is there some $w \in L$ such that $F = \{i : \pi_i(w)$ is ultimately periodic$\}$.*

We could go even further, and consider an extension of weak MSO where all the new quantifiers mentioned in this work are allowed: the bounding quantifier, the recurrence quantifier, and the ultimately periodic quantifier. As previously, the automaton model would simply be boolean combinations of the three automata models: min-automata, max-automata, and automata with ultimately periodic acceptance condition. The emptiness problem would require solving a variant of Theorem 14 where the language $L$ is not $\omega$-regular, but recognized by a nondeterministic BS-automaton (since these are strong enough to capture both max- and min-automata).

## 5 Conclusions

In this paper we presented several new classes of languages of infinite words. These classes are robust: they have good closure properties, they admit logical and automaton characterizations, they have decidable emptiness. We hope that the examples from this paper,

together with the max-automata from [2], offer convincing proof that there are interesting generalizations of the concept of $\omega$-regular language. The general theme is to look at deterministic automata with conditions that talk about asymptotic behavior, conditions more subtle than the usual "state $q$ appears infinitely often".

One direction of future research is investigating the exact relationship between min-automata and the existing theory of distance automata and formal power series. Preliminary results show that such connections result in a PSPACE-upper bound for deciding emptiness of boolean combinations of min- and max-automata.

Finally, we intend to investigate a similar theory for tree languages.

## References

[1] P. A. Abdulla, P. Krcál, and W. Yi. R-automata. In *CONCUR*, pages 67–81, 2008.

[2] M. Bojańczyk. Weak MSO with the unbounding quantifier. submitted.

[3] M. Bojańczyk. A bounding quantifier. In *Computer Science Logic*, volume 3210 of *Lecture Notes in Computer Science*, pages 41–55, 2004.

[4] M. Bojańczyk and T. Colcombet. Omega-regular expressions with bounds. In *Logic in Computer Science*, pages 285–296, 2006.

[5] O. Carton and W. Thomas. The monadic theory of morphic infinite words and generalizations. In *Mathematical Foundations of Computer Science*, volume 1893 of *Lecture Notes in Computer Science*, pages 275–284, 2000.

[6] T. Colcombet and C. Löding. The nesting-depth of disjunctive mu-calculus for tree languages and the limitedness problem. In *Computer Science Logic*, volume 5213 of *Lecture Notes in Computer Science*, 2008.

[7] C. C. Elgot and M. O. Rabin. Decidability and undecidability of extensions of second (first) order theory of (generalized) successor. *Journal of Symbolic Logic*, 31:169–181, 1966.

[8] D. Kirsten. Distance desert automata and the star height problem. *Theoretical Informatics and Applications*, 39(3):455–511, 2005.

[9] J.-É. Pin. Tropical semirings. In *Idempotency*, pages 50–69. Cambridge University Press, 1998.

[10] A. Rabinovich. On decidability of monadic logic of order over the naturals extended by monadic predicates. *Inf. Comput.*, 205(6):870–889, 2007.

[11] A. Rabinovich and W. Thomas. Decidable theories of the ordering of natural numbers with unary predicates. In *CSL*, pages 562–574, 2006.