Foundations of Software Technology and Theoretical Computer Science (2009) Submission

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 ω-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

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

$$\limsup_{i\to\infty} val(c, a_1a_2\dots a_i) = \infty,$$

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 bounding 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 ω-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
δ	The state transition function, $\delta\colon Q \times A \to Q$
γ	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 γ 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 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).

[†]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.

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 proved in the appendix (Appendix B), 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 Σ_3 of the Borel hierarchy, while deterministic max-automata can only recognize languages that are boolean combinations of Σ_2 languages.

For min-automata, we show in the appendix (Appendix C) 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 Π_2 of the Borel hierarchy, while deterministic min-automata can recognize even Σ_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. The solution we present in the appendix 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 max- and min-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 ∞ 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 (see Appendix A.2 for details).

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

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

In the appendix we will show Proposition 6, which says that every automaton can be modified into a matrix form as in the above example. 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

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

We call this a *min-automaton in matrix form.*

PROPOSITION 6. *Every min-automaton can be transformed into an equivalent min-automaton in matrix form. If the input automaton has n states and m counters, the resulting 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. The details are in Appendix A.2. ∎

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 from Appendix C, which shows that 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. 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 $\exists_{\mathsf{fin}} X$ 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 McNaughton theorem, define all ω-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 bounded size that satisfy $\varphi(X)$. More precisely, $\mathsf{R}X.\phi(X)$ 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 $\phi(X)$ is satisfied in w.

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

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

A word $a^{n_1}ba^{n_2}b\ldots$ satisfies $\mathsf{R}X.\phi(X)$ if and only if $\liminf n_i < \infty$. Therefore, the set of words with infinitely many b's that satisfy $\mathsf{R}X.\phi(X)$ 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.*

The easier part of this theorem, the translation from automata to logic, is presented in Appendix D. The more difficult part, from logic to automata, is a consequence of Corollary 11 and Theorem 12, which are stated in the next section.

3 General framework

In the previous section, we defined min-automata and proved 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. The proof in this paper and in [2] share some similarities. In this section, we would like to bring out these similarities, by introducing a more abstract framework.

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.*

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 ∞.

We are mainly interested in *prefix-independent* acceptance conditions, namely languages $F \subseteq B^\omega$ that satisfy if $F = B^*F$. 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 following theorem shows that automata with prefix-independent acceptance conditions compose well with weak existential quantification. Since the theorem talks about automata and languages, the notion of weak existential quantification is given 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 $X \subseteq \mathbb{N}$, 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 $F \subseteq B^\omega$ be any prefix independent acceptance condition, and let L be a language accepted by an F-automaton over the alphabet $A \times \{0,1\}$. Then the language*

$$\exists_{\mathsf{fin}} L = \{w \in A^\omega : \exists_{\mathsf{fin}} X.\ w \otimes X \in L\}$$

is a boolean combination of languages accepted by F-automata and Büchi automata.

COROLLARY 11. *Let $\mathscr{F}$ be a class of prefix-independent languages that contains any Büchi acceptance condition and is closed under boolean combinations. Then the class of languages recognized by automata with acceptance conditions in $\mathscr{F}$ is closed under boolean combinations and weak existential and universal quantification.*

THEOREM 12. *Let $F \subseteq B^\omega$ be any prefix-independent acceptance condition, and let L be a language accepted by an F-automaton over the alphabet $A \times \{0,1\}$. Then the language*

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

is a boolean combination of languages accepted by F-automata and min-automata .

An analogous theorem, where R is replaced by U and *min*-automata are replaced by *max*-automata holds.

The above theorem, together with Theorem 10 immediately imply the difficult part of Theorem 8 when applied to acceptance conditions corresponding to min-automata, namely that min-automata are captured by weak MSO extended with the R quantifier. The analogous result linking max-automata with the quantifier U also follows. One can even deduce the following fact.

COROLLARY 13. *Boolean combinations of max-automata and min-automata capture exactly weak MSO extended by both the recurrence quantifier* R *and the unbounding quantifier* U.

The above corollary also gives a normal form for weak MSO with the quantifiers R and U. Take a formula φ 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 $\mathsf{R}X.\phi(X)$ or $\mathsf{U}X.\phi(X)$, where $\phi(X)$ is a formula of weak MSO without R or U. In other words, nesting the quantifiers R and 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 P, is used to say that a set of positions is ultimately periodic. Specifically, if φ is a formula, and x is a first-order variable free in φ, then $\mathsf{P}x.\varphi(x)$ is true in a word if the set of positions x that satisfy φ 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 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 PS_L is called an *ultimately periodic acceptance condition*.

THEOREM 14. *Let $F \subseteq B^\omega$ be any prefix-independent acceptance condition, and let L be a language accepted by an F-automaton over the alphabet $A \times \{0,1\}$. Then the language*

$$\mathsf{P}L = \{w \in A^\omega : \mathsf{P}x.\, w \otimes \{x\} \in L\}$$

is a boolean combination of languages recognized by automata whose acceptance condition is either F, Büchi, or ultimately periodic.

COROLLARY 15. *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 14. ∎

THEOREM 16. *Satisfiability is decidable for weak MSO extended with the ultimately periodic quantifier.*

PROOF. By Corollary 15, 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 ω-regular language, we may assume one conjunct is ω-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 ω-regular and each K_i is either a language PS_{L_i} or its complement, for some ω-regular language L_i. It is not difficult to see that the following language over alphabet $\{0,1\}^n$ is ω-regular:

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

The emptiness problem boils down to testing if the set M above contains a word, whose projection onto coordinates i corresponding to languages PS_{L_i} is an ultimately periodic word, and whose projection onto coordinates i corresponding to complements of languages PS_{L_i} is not ultimately periodic. This way we have reduced our satisfiability problem to the following combinatorial result, which is solved in the appendix.

THEOREM 17. *The following problem is decidable*

- *Input: An ω-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 17 where the language L is not ω-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 ω-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 a theory of quantifiers for weak MSO over words. We could define a quantifier Q to be simply a set of sets of finite subsets of $\mathbb{N}$. For instance, the U quantifier is the set of sets which contain arbitrarily large subsets, while $\exists$ is the set of nonempty sets (of subsets). We could then find a common framework for Theorems 10, 12 and 14 with the following definition: a quantifier Q is captured by a family of acceptance conditions $\mathscr{F}$ if for every acceptance condition F (not necessarily in $\mathscr{F}$), and every language $L \subseteq (A \times \{0,1\})^\omega$ recognized by an F-automaton, the language

$$\mathsf{Q}L = \{w \in A^\omega : \{X : w \otimes X \text{ for some } X \subseteq \mathbb{N}\} \in \mathsf{Q}\}$$

is recognized by a boolean combination of F-automata, $\mathscr{F}$-automata, and Büchi automata. We would like to study this theory further. One of the questions would be: what conditions should be satisfied by Q so that any formula using Q is equivalent to a boolean combination of formulas where Q is used once, as the outermost quantifier.

Another direction is investigating the exact relationship with the existing theory of distance automata and formal power series. We hope that such an investigation would result in an upper bound for the emptiness of min- and max- automata, hopefully PSPACE.

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.

Appendices

A Min-automata

A.1 Eliminating $\top$ from min-automata

LEMMA 18. *Min-automata with counter values in $\{0,1,\ldots\}$ recognize the same languages as min-automata with counter values in $\{0,1,\ldots,\top\}$. In other words, an automaton $\mathcal{A}$ with $\top$-values can be converted into an equivalent automaton $\mathcal{A}'$ without $\top$-values.*

PROOF. By storing the information about $\top$-valued counters in the state.

Let Q denote the states of $\mathcal{A}$ and C denote its counters. A state of $\mathcal{A}'$ is a pair $(q, C_\top)$, where $q \in Q$ corresponds to the state of $\mathcal{A}$ and $C_\top \subseteq C$ shows which counters have value $\top$. The first component of the state $(q, C_\top)$ is modified accordingly to the state transformation function of $\mathcal{A}$. The second component, can be updated basing on the counter update function of $\mathcal{A}$ (as it does not need to know the actual counter values, but only the $C_\top$ from the previous step). $\mathcal{A}'$ is equipped with the same counters as $\mathcal{A}$. The counter update function of $\mathcal{A}'$ differs, however, since we must get rid of the $\top$-assignments.

$\mathcal{A}'$ must ignore all the increments done to a counter $c \in C_\top$. This is to ensure that the counter does not converge to ∞ while it represents $\top$ (but actually stores some finite value).

This way, by equipping $\mathcal{A}'$ with the same acceptance condition as $\mathcal{A}$, we obtain an automaton which recognizes the same language. ∎

Remarks: The above translation results in a growth of the state space by a factor exponential in the number of counters.

In fact, it can be shown that this blowup is inevitable in some cases, so $\top$-valued counters allow a more succinct description of a min-automaton. On the other hand, for a min-automaton without $\top$-valued counters, the vector of initial counter values is of no significance, since a finite change of the initial values does not affect the acceptance of a word. Thus, we may omit this information in such an automaton, while in an automaton with $\top$-valued counters, the initial information about which counters are finite and which are defined might affect the acceptance of a word.

A.2 Equivalence of min-automata with min-automata in matrix form

In this section we will show that each min-automaton can be converted into a min-automaton in matrix form which recognizes the same language, and vice-versa. Let $\mathcal{A}$ be a min-automaton.

First, let us transform $\mathcal{A}$ into an automaton of an intermediate form, which assigns a counter transition matrix with each state transition. In other words, we convert the update function, $\gamma\colon Q \times A \to (Op_C)^*$ into a function $\gamma\colon Q \times A \to \mathbb{M}_C\mathcal{T}$ which results in the same counter operations. We will call such an automaton a *multi-state min-automaton in matrix form*.

The counter operation

$$c := c + 1$$

is equivalent to applying the matrix $M_{c:=c+1}$, defined by:

$$\begin{array}{lcll} M_{c:=c+1}[c,c] & = & 1 & \\ M_{c:=c+1}[d,d] & = & 0 & \text{for } d \neq c \\ M_{c:=c+1}[d,e] & = & \top & \text{for } d,e \neq c \end{array}$$

The counter operation

$$c := min(d,e)$$

is equivalent to applying the matrix $M_{c:=min(d,e)}$, such that

$$\begin{array}{lcll} M_{c:=min(d,e)}[d,c] & = & 0 & \\ M_{c:=min(d,e)}[e,c] & = & 0 & \\ M_{c:=min(d,e)}[f,f] & = & 0 & \text{for } f \neq c \\ M_{c:=min(d,e)}[f,g] & = & \top & \text{for remaining positions } [f,g] \end{array}$$

A sequence of operations $o_1; o_2; \ldots; o_n$ corresponds to the composition of the matrices $M_{o_1} \cdot M_{o_2} \cdots M_{o_n}$. To prove the result below, which was stated in the main part of the paper, it only remains to remove states.

Proposition 6. *Every min-automaton can be transformed into an equivalent min-automaton in matrix form. If the input automaton has n states and m counters, the resulting automaton has $(m+1) \times n$ counters.*

PROOF. Let $\mathcal{A}$ be a multi-state min-automaton in matrix form, with states Q and counters C. Let $\gamma\colon Q \times A \to \mathbb{M}_C\mathcal{T}$ be its counter update function and $\delta\colon Q \times A \to Q$ be its state transition function.

Let us consider an automaton $\mathcal{A}'$ whose counters C' are the disjoint union of $(Q \times C)$ and C, and whose counter update function $\gamma'\colon A \to \mathbb{M}_{C'}\mathcal{T}$ is defined as follows. For (q_1, c_1) and (q_2, c_2) in $Q \times C \subseteq C'$,

$$\gamma'(a)[(q_1,c_1),(q_2,c_2)] = \begin{cases} \top & \text{if } \delta(q_1,a) \neq q_2 \\ \gamma(q_1,a)[c_1,c_2] & \text{if } \delta(q_1,a) = q_2 \end{cases}$$

Let us ignore the counters $C \subseteq C'$ for the moment.

The interpretation of the above construction is such that after reading a finite word w, a counter (q,c) of $\mathcal{A}'$ is defined and has value k iff $\mathcal{A}'$ is in state q and its counter c has value k. In other words, each counter of $\mathcal{A}$ has one copy (q,c) corresponding to each state q of $\mathcal{A}$, and only one of those copies (the one corresponding to the active state of $\mathcal{A}$) is defined at any moment. To make this description accurate, we need to define the initial counter values of $\mathcal{A}'$, so that (q,c) is set to $\top$ if q is not an initial state of $\mathcal{A}$, otherwise its initial value is the same as of counter c in $\mathcal{A}$.

In this way, we ensure that the counters of $\mathcal{A}'$ simulate the states and the counter values of $\mathcal{A}$. However, we still need to define a corresponding acceptance condition for $\mathcal{A}'$. Note that, for a counter c of $\mathcal{A}$, its value is swapped among the corresponding counters (q,c) of $\mathcal{A}'$, so we cannot determine the asymptotic behavior of c by looking at any *particular* counter (q,c) of $\mathcal{A}'$. In order to do that, we need an extra counter of $\mathcal{A}'$, which will keep the actual value of the counter c, which is also stored in one of the counters (q, c) (but we don't know which one). That's what we need the counters $C \subseteq C'$ for. Each counter $c \in C \subseteq C'$ will be an output-only counter (so that it does not influence other counters). The update function function is defined as follows.

$$\gamma'(a)[(q_1, c_1), c_2] = \gamma(q_1, a)[c_1, c_2]$$

This way, each of the counters c of $\mathcal{A}'$ will store exactly the same value as the same counter c o $\mathcal{A}$. Therefore, we may simply equip $\mathcal{A}'$ with the acceptance condition of $\mathcal{A}$. ∎

The following proposition describes the conversion in the other way around.

PROPOSITION 19. *A min-automaton in matrix form can be converted into an equivalent min-automaton.*

PROOF. Let C be the set of counters of the min-automaton $\mathcal{A}$ in matrix form (recall that such an automaton has no state). A matrix of counter transformations M transforms a counter $c \in C$ in the following way.

$$c := \min_d \{d + M[c, d]\}$$

One problem is that all those counter operations should be performed simultaneously, i.e. the counter d on the right-hand side represents the "old" counter value, while the counter c on the left-hand side represents the "new" counter value. To ensure simultaneity, we need to introduce some auxiliary counters (which temporarily store the copies of the old counter values). Let us introduce a counter $t_{c,d}$ per each pair of counters $c, d \in C$. To simulate the matrix operation M, we first set $t_{c,d}$ to be equal to d, afterwards we increase it $M[c, d]$ times. Finally, we set c to be equal to $\min_d t_{c,d}$. This operation can be sequentially simulated by the operations of the form $c := min(d, e)$.

The obtained automaton is a min-automaton with possibly $\top$-valued counters, and one state. To get rid of the $\top$-valued counters, apply Lemma 18. ∎

COROLLARY 20. *A min-automaton with $\top$-valued counters can be converted into an equivalent min-automaton with $\top$-valued counters, which uses only one state.*

PROOF. In course of the translation described above, we obtained a min-automaton with $\top$-valued counters, which uses only one state. ∎

A.3 Diagrams of min-automata in matrix form

Just as it is convenient to display a nondeterministic finite automaton as a labeled graph, we can depict min-automata in matrix forms using similar diagrams.

Let $\mathcal{A}$ be a min-automaton in matrix form with counters C and counter transformation function $\gamma \colon A \to \mathbb{M}_C\mathcal{T}$.

We represent the automaton $\mathcal{A}$ as a graph with vertices C and an edge with label $a \in A$ from vertex c to vertex d if

$$\gamma(a)[c, d] < \top.$$

We say that this edge *is incrementing by* $\gamma(a)[c,d]$. Usually, it only matters whether it is incrementing by 0 or a finite nonzero number. Edges incrementing by a nonzero number will be simply called *incrementing edges*. For a path in the diagram, we will say that it *has* N *increments* if it contains N incrementing edges.

We will describe the counters with finite initial values as *initial counters*.

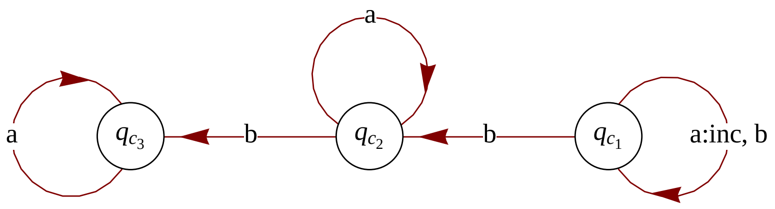
Let u be an input word. The condition that during the run of $\mathcal{A}$ on the word $u = a_1 a_2 \ldots$,

$$\liminf_{i\to\infty} val(c, a_1 \ldots a_i) < \infty$$

is equivalent to writing:

> There exists a bound N such that for arbitrarily long prefixes w of u, there exists a path in the diagram of $\mathcal{A}$ labeled by w, which starts in an initial counter and ends in counter c and has N increments.

Example 21. The automaton from Example 5 is depicted in the diagram below. The increasing edge is marked with the symbol *inc*. We want the input word u to have prefixes



labeling arbitrarily long paths in the diagram, having a bounded number of increments and ending in d. Such paths must stay in z for a long time and when approaching a short block ba^*b, move to c and finally to d.

B Complexity

In this appendix, we prove

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

The theorem is proved by reducing the universality problem for nondeterministic finite automata to the emptiness problem.

Proof. Let $\mathcal{A}$ be a nondeterministic automaton over the alphabet A. We will construct a deterministic *min*-automaton $\mathcal{A}'$ over an extended alphabet $A \cup \{\$\}$ such that universality of $\mathcal{A}$ (over finite words) is equivalent to universality of $\mathcal{A}'$ (over infinite words).

We will construct $\mathcal{A}'$ by constructing its diagram as of a min-automaton in matrix form. This automaton, however, will not use $\top$-valued counters, so conversion of $\mathcal{A}'$ to a min-automaton can be done in linear time.

Let us consider the diagram of the automaton $\mathcal{A}$. The diagram of $\mathcal{A}'$ will have the same vertices as the diagram of $\mathcal{A}$, i.e. $\mathcal{A}'$ has counters corresponding to the states of $\mathcal{A}$. We will draw an edge from a c to d with label a and with no increment if there is an edge with label a between the corresponding states in $\mathcal{A}$. Otherwise, we draw an edge with an increment. Thus, the diagram contains all possible edges, but some incrementing and some not. Now we will draw the edges corresponding to the special letter \$. We draw such an edge with no increment between any counter corresponding to a final state and any counter corresponding to an initial state.

We also add another counter, not mentioned before, which counts the number of occurrences of the letter \$. All counters are initiated to 0.

The acceptance condition requires either of the counters not to converge to ∞.

LEMMA 22. *The nondeterministic automaton $\mathcal{A}$ is universal if and only if the deterministic min-automaton in matrix form $\mathcal{A}'$ is universal.*

PROOF. Let us assume that $\mathcal{A}$ is universal. We want to show that $\mathcal{A}'$ is universal. Let $u \in (A \cup \{\$\})^\omega$ be an infinite input word. If u has finitely many \$'s, then u is accepted by $\mathcal{A}'$, since the counter which counts \$'s does not converge to ∞. So let us assume that u has infinitely many \$'s, so that it is of the form

$$u = w_1\$w_2\$w_3\$\ldots.$$

For each n, since $\mathcal{A}$ accepts the word w_n, there is a path π_n labeled by w_n in the diagram of $\mathcal{A}'$ which contains no increments, starts in a counter corresponding to an initial state, and ends in a counter corresponding to a final state. We may connect the paths $\pi_1, \pi_2, \ldots$ into one infinite path labeled by u and with no increments, thanks to the edges which join the "final" counters with the "initial" counters. This infinite path visits some counter infinitely often, proving that this counter does not converge to ∞. Thus, u is accepted by $\mathcal{A}'$.

Now let us assume that the automaton $\mathcal{A}'$ is universal. Let $w \in A^*$ be a finite input word. Consider the infinite word $u = w\$w\$w\$\ldots$. This word u is accepted by $\mathcal{A}'$. Since u contains infinitely many \$'s, this means that some counter c corresponding to a state of $\mathcal{A}$ does not converge to ∞. Therefore, there exists a bound N and arbitrarily long paths π which end in counter c, are labeled by a prefix of u and have less than N increments. Let us consider such a path of length at least $N \times |w + 2|$. This path must contain some sub-path labeled by $\$w\$$, which contains no increments. Therefore, that path delineates an accepting run of $\mathcal{A}$ over the word w. ∎

By negating the acceptance condition of the automaton constructed in the lemma, we obtain a reduction of the universality problem for nondeterministic automata to the emptiness problem for min-automata in matrix form. As noted before, since the constructed automaton does not use $\top$-valued counters, the described reduction is done in linear-time. ∎

C Nondeterministic min-automata

In the definition of a deterministic min-automaton, by relaxing the requirement on the transition δ to be a function and allowing it to be a relation, we obtain the notion of a *nondeterministic min-automaton*. The acceptance condition then may require the *existence* of a run which satisfies the acceptance condition.

Let us consider the language K mentioned in Section 1, which was defined as follows.

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

The language K is recognized by a nondeterministic min-automaton. The nondeterministic min-automaton does not even use the *min* operation. The automaton has two states, p and q, and three counters, c, d and z.

The transitions of the automaton are described as follows.

In each state, the automaton may nondeterministically choose to move either to state p or to state q. In any state, the letter a increases the counter c. In state p, the letter b resets the counter c by copying z to c and in state q, the letter b first copies the counter c to d and then resets it. The acceptance condition requires that counter d converges to infinity. It is easy to see that this automaton recognizes the language K.

The language K is not recognized by any deterministic min-automaton. It is more difficult, as usual, to show the inexpressivity result. The rest of Section C is devoted to this. Let us assume, by contradiction, that some deterministic min-automaton in matrix form with counters C recognizes K.

Let α_1 be the surjection of the tropical semiring obtained by glueing together all numbers $1, 2, 3, \ldots$ (this relation is a semiring congruence) and let $\mathcal{T}_1$ denote the quotient semiring, and $0, 1, \top$ denote its elements (note that $1 + 1 = 1$ in $\mathcal{T}_1$). This surjection gives rise to a surjection α_1 from the semiring $\mathbb{M}_C\mathcal{T}$ of $n \times n$ matrices over the tropical semiring onto the finite semiring $\mathbb{M}_C\mathcal{T}_1$ of $n \times n$ matrices over $\mathcal{T}_1$. Let ω be the idempotent power for the multiplicative semigroup of $\mathbb{M}_C\mathcal{T}_1$.

Let $\gamma\colon A \to \mathbb{M}_C\mathcal{T}$ denote the counter transition function, and let us consider its extension $\gamma\colon A^* \to \mathbb{M}_C\mathcal{T}$ to a semigroup homomorphism. This function is defined by its values on the generators a and b.

By defining

$$\gamma'(a) = \gamma(a^\omega)$$
$$\gamma'(b) = \gamma\big((a^\omega b^\omega a^\omega)^\omega)\big)$$

we obtain another automaton $\mathcal{A}'$ which recognizes the same language K. This is because the substitution ϕ defined by $a \mapsto a^\omega$, $b \mapsto (a^\omega b^\omega a^\omega)^\omega$ preserves the language K, i.e. $w \in L$ iff $\phi(w) \in K$.

Moreover, the two projections of the generator matrices, $\hat{a} = \alpha_1(\gamma'(a))$ and $\hat{b} = \alpha_1(\gamma'(b))$ satisfy the equations:

$$\hat{a} = \hat{a}^2$$
$$\hat{b} = \hat{b}^2 = \hat{a}\hat{b} = \hat{b}\hat{a}. \qquad (*)$$

The above equations can be viewed in terms of paths in the diagram of the automaton $\mathcal{A}'$. For example, equation $(*)$ implies that if there exists a path from counter c to counter d with at least one edge labeled b, then there exists a single edge with label b from counter c to counter whose d. Moreover, if the original path made no increments, then the edge makes no increment either.

We will show that in this case, the language recognized by $\mathcal{A}'$ cannot distinguish between any two words with infinitely many b's. This is clearly a contradiction, since the word b^ω does not belong to K, while the word $a\ b\ a^2 b\ a^3 b \ldots$ does.

LEMMA 23. *Let $\mathcal{A}$ be an min-automaton over the alphabet a, b, with transition function γ defined by matrices whose projections $\hat{a}$ and $\hat{b}$ under α_1 satisfy the equation $(*)$. Then $\mathcal{A}$ either both accepts, or rejects any two words u, v with infinitely many b's.*

PROOF. Let u and v be two words having infinitely many b's. We will prove that if the counter c does not converge to ∞ while reading the word u, then it does not converge to ∞ while reading the word v either. From symmetry, we will deduce that the set of counters which converge to ∞ is the same for either word. Therefore, either both words are accepted, or both are rejected, ending the proof of the lemma.

Assuming that counter c does not converge to ∞, there exists a bound N and a sequence of paths $\pi^1, \pi^2 \ldots$ in the diagram of $\mathcal{A}$, labeled with arbitrarily long prefixes of u, ending in counter c and having a at most N increments. Without loss of generality, we may assume that this sequence of paths converges to an infinite path π^∞ labeled by u (a sequence of finite words converges to an infinite word, if any prefix of the infinite word is shared with almost all of the finite words). This infinite path has at most N increments. Although this path not necessarily visits the counter c, arbitrarily long prefixes of π^∞ may be terminated in order to obtain a path visiting the counter c and having in total no more than N increments.

Since the path π^∞ is infinite, it necessarily visits some counter d infinitely many times. Therefore, there exists some loop in the diagram of $\mathcal{A}$ from d to d during which no increment is performed. Moreover, since there are infinitely many b's in the input word, the loop may be chosen so that at least one of its edges has label b. From equation $(*)$ we deduce that in this case, for any finite word containing a letter b, there exists a loop from d to d labeled by that word an having no increments.

Moreover, the equation $(*)$ tells us that if there is a path from d to c labeled by any word over the alphabet $\{a, b\}$, then there is an edge from d to c with label b.

Putting those two observations together, we conclude that since the word v contains infinitely many b's, for arbitrarily large i we can construct a path which loops in d without incrementing the counter before reading the i-th letter b and finishes in c immediately afterwards.

This sequence of paths proves that counter c does not converge to ∞ when $\mathcal{A}'$ reads the word v. As noted before, this ends the proof of the lemma. ∎

Therefore, we have shown by contradiction, that no min-automaton can recognize the language K.

COROLLARY 24. *The class of languages recognized by deterministic min-automata is not closed under the second order monadic quantifier $\exists$.*

D Weak MSO with the recurrence quantifier

In this appendix, we prove Theorem 8, which says that min-automata recognize the same languages as can be defined in weak MSO with the recurrence quantifier.

The translation from logic to automata is the more difficult translation, and follows from more general Theorems 10 and 12.

The translation from automata to logic is described below. Let $\mathcal{A}$ be a min-automaton in matrix form. The acceptance condition of $\mathcal{A}$ is a boolean combination of formulas of the form

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

For a given infinite input word $u = a_1 a_2 \ldots$, the run of $\mathcal{A}$ on the word u satisfies the negation of the above formula if and only if (see Appendix A.3) the following formula α_c is holds in u:

> There exists a bound N such that for arbitrarily long prefixes w of u, there exists a path in the diagram of $\mathcal{A}$ labeled by w, which starts in an initial counter and ends in counter c and has N increments.

Since the logic is closed under boolean combinations, we only need to take care of the case when the acceptance condition of $\mathcal{A}$ is the single atomic formula α_c.

Let the counters of $\mathcal{A}$ be $C = \{c_1, c_2, \ldots, c_n\}$. We will need an auxiliary formula $\phi(X, X_1, \ldots, X_n)$ with $n+1$ free variables ranging over finite sets. The formula $\phi(X, X_1, \ldots, X_n)$ verifies that the sets $X_1, \ldots, X_n$ form a partition of some finite prefix of the input word, which we interpret as a path in the diagram of $\mathcal{A}$ – if position i is in set X_k the path visits counter c_k in step i – and then checks that this run is a valid path in the diagram of $\mathcal{A}$, having increments precisely at the positions belonging to X, and that the path begins in an initial counter and ends in state c. The formula ϕ can be defined using first order quantification.

Then, the formula α_c is equivalent to

$$\mathsf{R}X.\ \exists_{\mathsf{fin}} X_1 \ldots \exists_{\mathsf{fin}} X_n.\ \phi(X, X_1, \ldots, X_n).$$

Therefore, $\mathcal{A}$ accepts the word u if and only if it satisfies the above formula of weak MSO with the recurrence quantifier. Note that the resulting formula is of a very specific form.

E General framework

In this section, we prove the two main theorems from 3.

The $\exists_{\mathsf{fin}}$ quantifier

Theorem 10. *Let $F \subseteq B^\omega$ be any prefix independent acceptance condition, and let L be a language accepted by an F-automaton over the alphabet $A \times \{0,1\}$. Then the language*

$$\exists_{\mathsf{fin}} L = \{w \in A^\omega : \exists_{\mathsf{fin}} X.\ w \otimes X \in L\}$$

is a boolean combination of languages accepted by F-automata and Büchi automata.

The proof is almost identical to the proof of a special case from [2].

Fix a deterministic max-automaton $\mathcal{B}$ that recognizes L, with state space Q. A *partial run* in an infinite word w is a run that begins in any position of the word (not necessarily the first position) and in any state (not necessarily the initial one). In other words, this is a word in $\perp^*\delta^\omega \cup \perp^\omega$, where δ is the set of transitions of $\mathcal{B}$, that is consistent with the word w on those positions where it is defined (i.e. where it is not $\perp$). Since the automaton is deterministic, a partial run is uniquely specified by giving the first configuration where it is defined, this is called the *seed configuration*. (There is also the undefined partial run $\perp^\omega$, which has no seed configuration.) Here, a configuration is a pair (q, x), where q is a state and x is a word position. Note that we do not include the counter values in the seed configuration, since the acceptance condition is not sensitive to finite perturbations.

We say that two partial runs *converge* if they agree from some position on. Equivalently, they converge if they share some configuration, or both are undefined. We say a set of partial runs *spans* a word w if every partial run over w converges with some run from the set. Usually, we will be interested in finite sets of spanning runs.

We now describe how the spanning partial runs will be encoded in the output of the transducer. When speaking of spanning partial runs, we mean spanning partial runs of the automaton $\mathcal{B}$. A single partial run can be encoded as an infinite word over the alphabet $Q \times \{0,1\}$. The idea is that $\{0,1\}$ is used as a marker, with 0 meaning "ignore the prefix until this position", and 1 meaning "do not ignore". Formally, an infinite word

$$(q_1, a_1)(q_2, a_2), \ldots \quad \in \quad (Q \times \{0,1\})^\omega$$

is interpreted as the partial run which on position i has $\perp$ if $a_j = 0$ for some $j \geq i$, otherwise it has q_i. Note that if the word above has infinitely many positions j with $a_j = 0$, then the partial run is nowhere defined, i.e. it is $\perp^\infty$. If we want to encode n partial runs, we use n parallel word sequences, encoded as a single sequence over the product alphabet

$$(Q \times \{0,1\})^n \,.$$

The following lemma was shown in [2]:

LEMMA 25. *Let $n = |Q|$. There is a transducer*

$$f: \quad \Sigma^\omega \quad \to \quad ((Q \times \{0,1\})^n)^\omega$$

such that for any word w, the output $f(w)$ encodes n spanning partial runs.

We are now ready to prove Theorem 10. By properties of spanning sets of runs, a word $w \in \Sigma^\omega$ belongs to the language of the theorem if and only if there is some $i = 1, \ldots, n$ such that the following three properties hold:

(A) The i-th run encoded by $f(w)$ is defined, i.e. the encoding does not contain infinitely many canceling 0s.

(B) The i-th run encoded by $f(w)$ satisfies the accepting condition in the automaton $\mathcal{B}$.

(C) There is some finite set $X \subseteq \mathbb{N}$ such that the run of $\mathcal{B}$ over $w[X]$ converges with the i-th run encoded by $f(w)$.

Properties (A) and (C) are ω-regular, so they are recognized by boolean combinations of Büchi automata. Property (B) is recognized by an F-automaton.

The R quantifier

We shall prove the theorem mentioned in Section 3:

Theorem 12. *Let $F \subseteq B^\omega$ be any prefix-independent acceptance condition, and let L be a language accepted by an F-automaton over the alphabet $A \times \{0,1\}$. Then the language*

$$\mathsf{R}L = \{w \in A^\omega : \mathsf{R}X.\, w \otimes X \in L\}$$

is a boolean combination of languages accepted by F-automata and min-automata .

Let $\mathcal{B}$ be an F-automaton over the alphabet $A \times \{0,1\}$ recognizing the language L. The language $\mathsf{R}L$ contains precisely those words w such that there exist infinitely many sets X of bounded size for which $w \otimes X$ is accepted by $\mathcal{B}$. We must prove that the language $\mathsf{R}L$ is recognized by a boolean combination of F-automata and min-automata. The proof is split into three steps.

In what follows, by *F-regular language*, we will mean a language recognized by an F-automaton.

Step 1: Reduction to *F-guarded* min-automata. Let w be an infinite word over the alphabet A. The *acceptance cost* of a prefix v of w is the size of the smallest set X which marks a subset of positions of the prefix v such that:

1. X contains the last position of v
2. $w \otimes X$ is accepted by $\mathcal{B}$.

If no set X has the above properties, we define the acceptance cost to be ∞. The usefulness of the above definition comes from the following observation.

LEMMA 26. *A word w belongs to the language $\mathsf{R}L$ if and only if the sequence of acceptance costs of prefixes of w has a bounded subsequence.*

Note that, by prefix independence of the acceptance condition F, the requirement 2 in the definition of acceptance cost is satisfied if and only if there exists a state q of $\mathcal{B}$ such that:

2′. $\mathcal{B}$ reaches the state q after reading $v \otimes X$

2″. If v' is the word w with the prefix v removed, then $\mathcal{B}$ accepts the word $v' \otimes \emptyset$ starting from the state q.

For a state q of $\mathcal{B}$ and a prefix v of the input word, let the *cost of reaching* a state q be the size of the smallest set X which satisfies conditions 1 and 2′, or ∞ if no such set exists.

LEMMA 27. *There exists a min-automaton $\mathcal{A}$ which, after reading a finite word v, stores in its counters the costs of reaching each state of $\mathcal{B}$.*

PROOF. For a state q of $\mathcal{B}$ and a prefix v of the input word, let us define the *charge for passing* through a state q as the size of the smallest set X which satisfies condition 2′ but not necessarily 1, i.e. the set X doesn't need to contain the last position of the word v.

The automaton $\mathcal{A}$ will store in its counters both the cost of reaching q and the charge for passing through q, for each state q of $\mathcal{B}$. Those values can be updated in each step of the automaton, basing on the transition of the automaton $\mathcal{B}$ and the counter values obtained in the previous step, using the min instruction. (In this place, the min instruction is used.)

A problem arises with storing the value ∞. One solution is to allow min-automata to have ∞-valued counters (where any sequence containing ∞ is considered unbounded) and assignments of the form $c := \infty$. One can prove that this does not increase the expressive power of min-automata, similarly to Lemma 18. Indeed, information about which counters have value ∞ may be stored in the state of the automaton (resulting in an exponential blowup) and the automaton should steadily increase all ∞-valued counters in each step, so that the acceptance condition is obeyed. ∎

Let $\mathcal{A}$ be the automaton described in the lemma. By knowing the cost of reaching a state q, the automaton $\mathcal{A}$ keeps track of the smallest size of a marking set X which satisfies the conditions 1 and $2'$ listed above. However, there is no way a min-automaton can verify the condition $2''$ basing just on the prefix it has read so far.

This motivates us to introducing the following definition.

DEFINITION 28. *An F-guarded* min-automaton $\mathcal{A}$ *is a min-automaton whose instruction set is extended by instructions of the form:*

if L then output(c),

where L is a F-regular language and c is some counter of $\mathcal{A}$.

The semantics is such that $\mathcal{A}$ outputs the value of counter c provided that the suffix of the input word (from the position following the current position onwards) belongs to L. By *outputs the value of counter c* the following is meant. We equip the automaton $\mathcal{A}$ with an additional *output counter* c_o, such that the only allowed operation involving c_o is $c_o := c$, denoted $output(c)$. The counter c_o can be taken into account in the acceptance condition of $\mathcal{A}$ in the same way as other counters. We will say for instance that the values output by counter c have a bounded subsequence, referring to the values of c_o. We may assume that, for a given counter c, the $output(c)$ instruction is always guarded by the same F-regular language L (to guard by some other language, use a different counter).

PROPOSITION 29. *If L is an F-regular language over the alphabet $A \times \{0,1\}$ then* RL *is recognized by an F-guarded min-automaton.*

PROOF. Let $\mathcal{A}$ be the automaton from the previous lemma and let c_q be the counter holding the cost of reaching the state q, for each state q of $\mathcal{A}$. We modify the automaton $\mathcal{A}$ so that in each step and for all states of $\mathcal{A}$, it performs the instruction

if L_q then output(c_q),

where L_q is the language of words w such that $\mathcal{B}$ accepts $w \otimes \emptyset$ starting from the state q. The languages L_q are clearly F-regular.

By Lemma 26, a word w belongs to RL if and only if the values output by one of the counters c_q has a bounded subsequence. Thanks to this, choosing a suitable acceptance condition for the automaton $\mathcal{A}$ makes it recognize precisely the language RL. ∎

In the following two steps we shall prove the complementary part of the theorem.

PROPOSITION 30. *Languages accepted by F-guarded min-automata are equivalent to boolean combinations of F-automata and min-automata.*

Step 2: Reduction to *regular-guarded* automata. Let L' be a regular language of *finite* words over the alphabet A. Then, an instruction

$$\textit{if } L' \cdot A^\omega \textit{ then } output(c),$$

will be called a *regular-guarded* output instruction. Thus, a regular-guarded instruction only inquires a near (finite) future of the input word. Min-automata equipped with regular-guarded outputs will be called regular-guarded min-automata.

Let $\mathcal{A}$ be an min-automaton with an F-regular guarded instruction

$$o: \quad \textit{if } L \textit{ then } output(c),$$

where the language L is recognized by an F-automaton $\mathcal{B}$. Let w be an input word. Let us imagine a run of $\mathcal{A}$ on w. Each time the instruction o is performed, a query to $\mathcal{B}$ is made. This query initiates a new partial run σ of $\mathcal{B}$ on the suffix of the word. If σ joins a fixed partial run ρ, we will call the execution of o which initiated σ a *ρ-query*. Note that this execution has an effect only if σ is an accepting run, which is equivalent to ρ being an accepting run.

We will use the following fact.

LEMMA 31. *The values output by counter c have a bounded subsequence if and only if for some partial run ρ the following conditions hold:*
- *The values output by the ρ-queries have a bounded subsequence*
- *The partial run ρ is accepted by $\mathcal{B}$.*

That these conditions are sufficient is clear, and their necessity follows from the fact that there are only finitely many nonequivalent partial runs.

Let us assume for a moment that $\mathcal{A}$ uses only one F-guarded instruction.

LEMMA 32. *Let M be the language consisting of all words $w \otimes X \in (A \times \{0,1\})^\omega$ such that:*
1. *X marks a single position x*
2. *The unique partial run ρ of $\mathcal{B}$ on the suffix of w which starts at x in the initial state of $\mathcal{B}$ is accepting*
3. *The values output by the ρ-queries in $\mathcal{A}$ have a bounded subsequence.*

Then M is recognized by a boolean combination of F-automata and regular-guarded min-automata over the alphabet $A \times \{0,1\}$.

PROOF. Condition 1 on the input words is ω-regular, so can be checked by a min-automaton. Condition 2 can be tested by an F-automaton, since the acceptance condition F is prefix-independent.

To test the third condition, we will construct a regular-guarded min-automaton $\mathcal{C}$, which consists of two components, one simulating the state behavior of $\mathcal{B}$ and the other simulating $\mathcal{A}$ (formally, this is defined using a cartesian product). We will call those components the $\mathcal{B}$ and the $\mathcal{A}$ component. The precise behavior of those components is described below.

The $\mathcal{B}$ component is initiated after encountering the marked position x and computes the partial run ρ.

For any state q of $\mathcal{B}$, let L_q be the language of finite words w for which $\mathcal{B}$ transforms its initial state and the state q into the same state. The language L_q will be used to determine whether the partial run initiating at the current position converges with the partial run ρ.

The $\mathcal{A}$ component, given an the input word $w \otimes X$, simulates the behavior of $\mathcal{A}$ on w, with the exception that when the automaton $\mathcal{A}$ invokes the operation o, instead, a regular-guarded operation

$$\mathit{if}\ L_q \cdot A^\omega\ \mathit{then}\ \mathit{output}(c),$$

is performed, where q is the current state of the $\mathcal{B}$ component (if the $\mathcal{B}$ component has not initiated yet, the instruction is ignored).

The acceptance condition of $\mathcal{C}$ requires values output to counter c to have a bounded subsequence. The described automaton $\mathcal{C}$ is a regular-guarded min-automaton which verifies condition 3. ∎

What does the language $\exists_{\mathsf{fin}} M$ consist of? By Lemma 31, $\exists_{\mathsf{fin}} M$ consists of those input words w, for which values output by counter c in $\mathcal{A}$ have a bounded subsequence. Hence, if $\mathcal{A}$ used only one output instruction, that instruction can be eliminated by a using boolean combinations with the language $\exists_{\mathsf{fin}} M$.

In the following step we will see that regular-guarded min-automata are equivalent to min-automata. Therefore, M is in fact recognized by a boolean combination of min-automata and F-automata. From Theorem 10 we can deduce that $\exists_{\mathsf{fin}} M$ is also recognized by a boolean combination of min-automata and F-automata. This implies that $\mathcal{A}$ is equivalent to a boolean combination of min-automata and F-automata.

In the case that $\mathcal{A}$ uses more than one F-guarded instruction, one may inductively apply the above reasoning and replace each such instruction by a boolean combination of min-regular and F-regular languages.

Therefore, to finish the prove of Theorem 12, we only need to prove that regular-guarded min-automata recognize min-regular languages, which we do in the next step. Note that we have already dismissed the F-regular languages from the problem.

Step 3: Eliminating guards from regular-guarded automata. The following lemma is the last step in the proof of Theorem 12.

LEMMA 33. *Languages accepted by regular-guarded min-automata are min-regular.*

PROOF. Let $\mathcal{A}$ be a regular-guarded min-automaton with a regular-guarded operation

$$o: \quad \mathit{if}\ L \cdot A^\omega\ \mathit{then}\ \mathit{output}(c).$$

For simplicity, we will assume that this is the only regular-guarded instruction used by $\mathcal{A}$ and that no other instruction of $\mathcal{A}$ outputs values of the counter c. As previously, we will show that we can eliminate this single operation by constructing a min-regular automaton $\mathcal{C}$ which recognizes whether the values output by the counter c have a bounded subsequence. The general case where there are more output instructions may be obtained by inductively eliminating output operations.

Let $\mathcal{B}$ be the automaton recognizing the language L. This time, $\mathcal{B}$ is just a finite automaton. We will consider the following min-automaton $\mathcal{B}'$ associated to $\mathcal{B}$ as described below.

The automaton $\mathcal{B}'$ has one state and counters corresponding to the states of $\mathcal{B}$, with possibly $\top$ values. The counter transitions of $\mathcal{B}'$ correspond to the state transitions of $\mathcal{B}$ – when a state q is transformed to a state r by $\mathcal{B}$, then $\mathcal{B}'$ transfers a value of the counter corresponding to q to the counter corresponding to r. If two states q, q' are transformed to the same state r, then $\mathcal{B}'$ transfers the smaller of the two values. In other words, $\mathcal{B}'$ is a min-automaton with the same diagram as the finite automaton $\mathcal{B}$.

The min-automaton $\mathcal{B}'$ has an ability of tracing runs of $\mathcal{B}$ and additionally associating values with those runs. A state of a traced run corresponds to a finite value of a counter; when two traced runs merge, the smaller value is chosen. This the second place in the proof of Theorem 12 where the min instruction is used.

The automaton $\mathcal{B}'$ is initialized with an empty set of traced runs, or, equivalently, with $\top$ counter values. Normally this would imply that the counters of $\mathcal{B}'$ will stay infinite for ever. However, in what follows, we will use the automaton $\mathcal{B}'$ as an internal automaton for the min-automaton $\mathcal{C}$, which has the ability to initiate a new run traced by $\mathcal{B}'$. It is done by setting the counter associated to the initial state of $\mathcal{B}$ to the value of the counter c of $\mathcal{C}$. Let us call this the *trigger*(c) operation.

Finally we will define the automaton $\mathcal{C}$. It is simply the automaton $\mathcal{A}$ equipped in the auxiliary automaton $\mathcal{B}'$ which runs simultaneously. Formally, this is constructed by using a cartesian product. Whenever $\mathcal{A}$ wishes to perform the operation o, the automaton $\mathcal{C}$ invokes the operation *trigger*(c) in $\mathcal{B}'$. The acceptance condition of $\mathcal{C}$ requires that either of the counters of $\mathcal{B}'$ associated to the accepting states of $\mathcal{B}$ has a bounded subsequence.

The construction guarantees that $\mathcal{C}$ accepts if and only if the values output by $\mathcal{A}$ have a bounded subsequence. Thus, we can eliminated the guarded output instruction by replacing it by a boolean combination with the min-automaton $\mathcal{C}$. ∎

Altogether, the steps 1, 2 and 3 show that min-regular automata are closed under the R quantifier, proving Theorem 12. ∎

F Ultimately Periodic Quantifier

In this part of the appendix, we prove

Theorem 17. *The following problem is decidable*

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

We assume that the language L is given by a nondeterministic Büchi automaton $\mathcal{A}$. A *component* in the automaton is a maximal set of mutually reachable states.

LEMMA 34. *Without loss of generality we may assume that all states in the automaton are in a single component, and that $F = \{1\}$.*

PROOF. For a Büchi automaton $\mathcal{A}$, a component automaton is simply the automaton obtained from $\mathcal{A}$ by picking some component, and removing all states and transitions that do not use this component. The initial state is chosen inside the component, in any way. Since the condition to be decided in Theorem 17 is ultimately periodic, it holds for an automaton $\mathcal{A}$ iff it holds for one of its (finitely many) component automata.

To ensure $F = \{1\}$ we can combine all the projections $\pi_i : B^\omega \to B_i^\omega$ for $i \in F$ into a single homomorphism $\pi : B \to \Pi_{i \in F} B_i^\omega$. (This construction does not work correctly for the coordinates $i \notin F$, so we need many homomorphisms for the indexes that are not ultimately periodic.) ∎

Proposition 17 follows from two lemmas. The first, Lemma 35, gives equivalent descriptions of the property that we want to decide in Proposition 17. The second, Lemma 36, says that one of these equivalent descriptions is decidable. Both lemmas use the assumptions on automaton $\mathcal{A}$ from Lemma 34.

Before we state Lemma 17, we need two more definitions.

The *period* of a connected component P is the largest common multiple of the lengths of loops in P. A *prime* component is a component with period 1. In a prime component there is a threshold $k \in \mathbb{N}$ such that any two states can be connected by paths of lengths $k, k+1, \ldots$.

For $u \in B_1^*$ we define an automaton $\mathcal{A}_u$. The states and accepting states are the same as in $\mathcal{A}$. The difference is in the input alphabet, which is

$$\{v \in B^{|u|} : \pi_1(v) = u\}$$

and in the transitions, which are triples (p, v, q) such that in the automaton $\mathcal{B}$ there is a run from p to q that reads the word v. Note that even though $\mathcal{A}$ is connected, the automaton $\mathcal{A}_u$ might not be connected.

LEMMA 35. *The following three conditions are equivalent*

1. *There is some $w \in L$ such that $\pi_1(w)$ is ultimately periodic, but not $\pi_2(w), \ldots, \pi_n(w)$.*
2. *For some $u \in B_1^*$, in the automaton $\mathcal{A}_u$ there exists a prime connected component $P \subseteq Q$, such that for each $i = 2, \ldots, n$ there is a pair of transitions*

$$(p_{i,1}, u_{i,1}, q_{i,1}), (p_{i,2}, u_{i,2}, q_{i,2}) \qquad \text{such that } \pi_i(u_1) \neq \pi_i(u_2) \text{ and } p_{i,1}, q_{i,1}, p_{i,2}, q_{i,2} \in P$$

3. *There is a state p and words $u_1, u_2 \in B^*$ of same length that label a loop in p such that*

$$\pi_1(u_1) = \pi_1(u_2) \qquad \text{and } \pi_i(u_1) \neq \pi_i(u_2) \text{ for } i = 2, \ldots, n.$$

PROOF.

Implication from 1 to 2. Suppose that $w \in B^\omega$ is as in 1. From the assumption that $\pi_1(w)$ is periodic, let $v \in B_1^*$ be such that $\pi_1(w) = u^\omega$. Let us decompose w as $w_1 w_2 \cdots$ where $w_1, w_2, \ldots$ are all words in $B^{|u|}$. In the accepting run of $\mathcal{A}$ over w, let q_i be the state reached after reading the prefix $w_1 \cdots w_i$. It is not difficult to see that for each i, the triple (q_i, w_{i+1}, q_{i+1}) is a transition in $\mathcal{A}_u$. Without loss of generality we may assume that the length of u is divisible by all periods of components in $\mathcal{A}$, which guarantees that all components in $\mathcal{A}_v$ are prime. Let P be the connected component of $\mathcal{A}_u$ that contains all states from $q_0, q_1, \ldots$ that appear infinitely often. For $i = 2, \ldots, n$ the two transitions required by 2 are obtained from the fact that $\pi_i(w)$ is not ultimately periodic, and hence there are at least two different words that appear infinitely often in the sequence $\pi_i(w_1), \pi_i(w_2), \ldots$.

Implication from 2 to 3. Let $u \in B_1^*$ and $P \subseteq Q$ be as in condition 2. Let p be any state in P. Since P is prime, for any states $q_1, q_2 \in P$ there are two words $u_1, u_2 \in (B^{|u|})^*$ of the same length that label paths, in the automaton $\mathcal{A}_u$, from q_1 to p and q_2 to p, respectively. Likewise for paths from p to q_1, q_2. Using this observation, we can use the property in condition 2 to show that for each $2 = 1, \ldots, n$ there is a pair of words $v_{i,1}, v_{i,2} \in (B^{|u|})^*$ of same length such that: a) both $v_{i,1}, v_{i,2}$ label loops in state p in the automaton $\mathcal{A}_u$; and b) $\pi_i(v_{i,1}) \neq \pi_i(v_{i,2})$. The words u_1, u_2 required by condition 3 are obtained by concatenating these words:

$$u_1 = v_{1,1} \cdots v_{n,1} \qquad u_2 = v_{1,2} \cdots v_{n,2}$$

Implication from 3 to 1. Let u_1, u_2 be words as in 3. Without loss of generality we assume that that both the loops in p that label u_1, u_2 involve an accepting state. Otherwise, we concatenate to both u_1, u_2 any word u that labels a loop in p which passes an accepting state. Such a word u must exist by assumption on the automaton being connected. Let u_0 be a word which labels a path from the initial state to state p. Our assumption guarantees that any word $w \in u_0(u_1 + u_2)^\omega$ is accepted by the automaton. By assumption 3, for any two words $w_1 \neq w_2 \in u_0(u_1 + u_2)^\omega$, we have

$$\pi_1(w_1) = \pi_1(w_2) = \pi_1(u_0(u_1)^\omega) \qquad \text{and } \pi_i(w) \neq \pi_i(w') \text{ for } i = 2, \ldots, n.$$

By the first part, the projections under π_1 of all words in $u_0(u_1 + u_2)^\omega$ are ultimately periodic. By the second part, the projections under $\pi_2, \ldots, \pi_n$ of all words in $u_0(u_1 + u_2)^\omega$ are distinct. Since there are countably many ultimately periodic words, the uncountable set $u_0(u_1 + u_2)^\omega$ must contain a word such that all its projections $\pi_2, \ldots, \pi_n$ are not ultimately periodic.

∎

LEMMA 36. *One can decide if condition 3 of Lemma 35 holds.*

PROOF. Standard automata techniques. ∎