Theory Comput Syst (2011) 48: 554–576
DOI 10.1007/s00224-010-9279-2

Weak MSO with the Unbounding Quantifier

Mikołaj Bojańczyk

Published online: 28 July 2010

Abstract A new class of languages of infinite words is introduced, called the *max-regular languages*, extending the class of ω-regular languages. The class has two equivalent descriptions: in terms of automata (a type of deterministic counter automaton), and in terms of logic (weak monadic second-order logic with a bounding quantifier). Effective translations between the logic and automata are given.

Keywords Automata · Monadic second-order logic

1 Introduction

This paper introduces a new class of languages of infinite words, the class of *max-regular languages*, which contains all ω-regular languages. Max-regular languages can be described in terms of automata, and also in terms of a logic. A typical max-regular language is the set of words in $(a+b)^\omega$ where the distance between consecutive b's is unbounded, i.e. the language

$$L = \{a^{n_1}ba^{n_2}ba^{n_3}\ldots : \limsup n_i = \infty\}. \tag{1}$$

A practical motivation for this class can be found in verification. For instance a max-regular language could specify that a system responds to requests with bounded delay. We will begin, however, with a more fundamental motivation, which is the question: what is a regular language of infinite words?

There is little doubt as to what is a regular language of finite words. For instance, the requirement that the Myhill-Nerode equivalence relation has finitely many equivalence classes uniquely determines which languages of finite words should be regular.

Author supported by ERC Starting Grant "Sosna".

M. Bojańczyk (✉)
University of Warsaw, Warsaw, Poland
e-mail: bojan@mimuw.edu.pl

Other notions, such as finite semigroups, or monadic-second order logic also point to the same class.

For infinite words, however, there is more doubt. Of course, the class of ω-regular languages has much to justify calling it regular, but some doubts remain as to its uniqueness. Consider, for instance, the language L mentioned above, or the set K of ultimately periodic words, i.e. words of the form wv^ω, say over alphabet a, b. None of these languages are ω-regular. However, under a popular variant of Myhill-Nerode equivalence for infinite words, given by Arnold in [2], both languages have exactly one equivalence class.

Should these languages be called regular? If yes, what is the appropriate notion of regularity? In this paper we propose a notion of regular languages, which are called *max-regular languages*, that captures the language L, but not the language K. This new notion has many properties that one would wish from regular languages. The class is (effectively) closed under boolean operations, including negation. There is a finite index Myhill-Nerode relation, and equivalence classes are regular languages of finite words. There is an automaton model, there is a logical description, and translations between the two are effective. Emptiness is decidable. Membership is decidable (although since we deal with infinite words, the membership test is for certain finitely presented inputs, such as ultimately periodic words).

So, what is this new class? One definition is in terms of logic. The max-regular languages are the ones that can be defined by formulas of weak monadic second-order logic extended with the unbounding quantifier. The term "weak" means that only quantification over finite sets is allowed, denoted by quantifiers $\exists_{\text{fin}}$ and $\forall_{\text{fin}}$. The unbounding quantifier $\mathrm{U}X.\varphi(X)$ says that the size of sets X satisfying $\varphi(X)$ is unbounded, i.e.

$$\mathrm{U}X.\varphi(X) = \bigwedge_{n\in\mathbb{N}} \exists_{\text{fin}} X\big(\varphi(X) \wedge |X| \geq n\big). \tag{2}$$

This is not the first paper about logic with the unbounding quantifier. The first paper, which introduced[1] the unbounding quantifier, is [3]. The setting was much more general than here: the logic was not weak (i.e. quantification over infinite sets was allowed), and the models considered were infinite trees and not infinite words. Nevertheless, [3] contains very little as far as decidability or automata models are concerned. The only decidability result concerns a fragment with very limited quantification. The study of the unbounding quantifier was continued in [4], where the logic is also allowed to quantify over infinite sets, but the models are restricted from infinite trees to infinite words. The results in [4] are substantially more advanced than in [3]. In particular there is an automaton model, called an ωBS-utomaton, which is also the basis of the work in this paper. Unfortunately, ωBS-utomata do not capture all of the logic, but only a fragment with limited quantification patterns. Even for this limited fragment, proving equivalence of the logic with ωBS-utomata is a very difficult task, due to nondeterminism of the automata.

The basic idea in this paper is to restrict the set quantification to finite sets (i.e. weak quantification), while keeping the unbounding quantifier. It turns out that

[1]The quantifier introduced in [3] was actually the negation of U, saying that the size is bounded.

with this restriction, lots of the problems encountered in [4] are avoided, and the resulting class is surprisingly robust. Note that for infinite words and without unbounding quantification, weak monadic second-order logic has the same expressive power as full monadic second-order logic; this is no longer true when the unbounding quantifier is allowed (we prove this using topological techniques).

The main contribution of this paper is Theorem 5, which shows that over infinite words, weak monadic second-order logic with the unbounding quantifier has the same expressive power as deterministic max-automata. Max-automata are a new automaton model, introduced in this paper. A max-automaton is a finite automaton equipped with counters, which store natural numbers. The important thing is that the counters are not read during the run (and therefore do not influence the control of the automaton), which avoids the usual undecidability problems of counter machines. The counters are only used in the acceptance condition, which requires some counter values to be bounded, and some to be unbounded.

To the best of the author's knowledge, quantifiers similar to the unbounding quantifier have only been considered in [3, 4]. On the other hand, the idea to use automata with quantitative acceptance conditions has a long history, going back to weighted automata of Schützenberger [14] (see [9] for a recent paper on weighted automata and related logics). Automata very similar to the ωBS-automata of [4] have also appeared in the literature under the name of *distance desert automata* in [12], or *R-automata* in [1]. One important application, see [12], of these automata is that they can be used to solve the famous star-height problem,[2] providing simpler techniques and better complexities than in the original proof of Hashiguchi [10]. (The reduction from the star-height problem is not to emptiness of the automata, but to something called *limitedness*.) Other problems that can be tackled using this type of automata include the star-height of tree languages [7] or the Mostowski index of ω-regular languages [8].

2 The Automaton

We begin our presentation with the automaton model. Fix a set of counters C, which will store natural numbers. We allow the following operations:

$$\begin{array}{ll} c := c + 1, & \text{increment counter } c; \\ c := 0, & \text{reset counter } c; \\ c := \max(d, e), & \text{store in counter } c \text{ the maximal value of counters } d, e. \end{array}$$

For a counter valuation $v \in \mathbb{N}^C$ and a finite sequence of counter operations π, we define $v\pi \in \mathbb{N}^C$ to be the counter valuation obtained from v by applying all the operations in π.

A *max automaton* is a finite automaton where each transition is labeled by a finite sequence of counter operations. A *run* of the automaton is a sequence of transitions that is consistent with the input word. Fix a run ρ, and let π_i be the sequence of

[2] This is the question of calculating the least number of nested stars in a regular expression (without negation) that defines a regular language $L \subseteq \Sigma^*$.

counter operations that labels the first i transitions in ρ. For an initial counter valuation $v \in \mathbb{N}^C$ and a counter $c \in C$ we define a sequence

$$\rho_c \in \mathbb{N}^\omega = v(c), v\pi_1(c), v\pi_2(c), \ldots.$$

The *acceptance condition* of a max automaton is a boolean combination of conditions: "the sequence ρ_c is bounded". Stated differently, the acceptance condition is a boolean combination[3] of conditions

$$\limsup \rho_c < \infty.$$

Note that changing the initial counter valuation v does not change the limsup, so the initial counter valuation is irrelevant to the accepted language, and need not be supplied with the automaton.

Unless otherwise stated, all max automata are deterministic.

Below we compare max automata to ωBS-automata, an automaton model defined in [4]. A ωBS-automaton is defined like a max automaton, with three differences. First, it is nondeterministic. Second, a ωBS-automaton does not have the max counter operation, only the increment and reset. Third, the acceptance condition in a ωBS-automaton is a boolean combination of two types of conditions: "the sequence ρ_c is bounded", called B-conditions, and "the sequence ρ_c tends to infinity", called S-conditions.[4] In other words, the limit properties tested by an ωBS-automaton are

$$\limsup \rho_c = \infty, \qquad \liminf \rho_c = \infty$$

and their negations, while a max automaton is only allowed the lim sup. It turns out that the max operation can be simulated using nondeterminism, so nondeterministic max automata are essentially those ωBS-automata that only talk about lim sup and not lim inf in their acceptance condition.

Theorem 1 *Every max automaton is effectively equivalent to a nondeterministic ωBS-automaton.*

Proof Fix a max automaton $\mathcal{A}$ with counters C. An ωS-automaton is a ωBS-automaton where the acceptance condition is a *positive* boolean combination of S-conditions. For every counter c, we will define a nondeterministic ωS-automaton $\mathcal{O}_c$, whose input alphabet consists of counter operations, and which accepts an infinite sequence $o_1 o_2 \cdots$ of counter operations if and only if

$$\limsup_{n\to\infty} \bar{0} o_1 \cdots o_n(c) = \infty.$$

[3]Formally speaking, the acceptance condition is a propositional boolean formula, whose variables are counters. A run is accepting if the formula is made true by the valuation which assigns "true" to the variables that correspond to bounded counters, and "false" to the other variables.

[4]As defined in [4], the acceptance condition in a ωBS-automaton is a *positive* boolean combination of B-conditions and S-conditions. However, the negation of a B-condition can be expressed, by using nondeterminism, as an S-condition. This is because a sequence is unbounded if and only if it has a subsequence that tends to infinity. Likewise, the negation of an S-condition can be expressed using B-conditions.

In the above, $\bar{0}$ is the zero vector in $\mathbb{N}^C$. By [4], any boolean combination of ωS-automata is equivalent to a ωBS-automaton. In particular, there is a ωBS-automaton $\mathcal{O}$ that accepts exactly those infinite sequence of counter operations that satisfy the acceptance condition in the max automaton $\mathcal{A}$. To get an automaton equivalent with $\mathcal{A}$, we simulate the actions of $\mathcal{O}$ online, by doing the transitions and producing the appropriate counter operations.

We are left with defining the automaton $\mathcal{O}_c$.

Below we define what it means for a sequence of counter operations π to *transfer a counter c to a counter d*. The idea is that after performing the operations in π, the value of counter d is at least as big as the original value of counter c, or possibly bigger, if there were some increments or max operations involving bigger counters along the way. The definition is by induction on the length of π. The empty sequence of operations transfers every counter to itself. The operation $c := c + 1$ transfers every counter to itself. The operation $c := 0$ transfers every counter to itself, except c. The operation $c = \max(d, e)$ transfers every counter other than c to itself, and also both d and e to c. If π_1 transfers c to e and π_2 transfers e to d, then the concatenation $\pi_1\pi_2$ transfers c to d. This definition is designed so that the following property holds: if π transfers c to d, then $v\pi(d) \geq v(c)$ holds for any counter valuation $v \in \mathbb{N}^C$. In a similar way to the above, we define a notion of *transferred with at least one increment*, which guarantees $v\pi(d) \geq v(c) + 1$. The transfer relation is regular in the following sense: for any counters c and d, the set of words π that transfer counter c to d is a regular language of finite words, likewise for transfers with at least one increment. Sequences of counter operations $\pi_1, \ldots, \pi_n$ are called a *c-trace* (of length n) if there is a sequence of counters $c_0, \ldots, c_n$ with $c_n = c$ such that each π_i transfers counter c_{i-1} to c_i with an increment. From the very definition we get the following fact.

Fact 2 *Counter c is unbounded if and only if there are arbitrarily long c-traces.*

The automaton $\mathcal{O}_c$ works as follows. It accepts if and only if the infinite sequence of counter operations π on input can be split into finite sequences as

$$\pi = \pi_{1,0}\pi_{1,1}\cdots\pi_{1,n_1}\pi_{2,0}\pi_{2,1}\cdots\pi_{2,n_2}\cdots$$

where for each i, the sequences $\pi_{i,1}, \ldots, \pi_{i,n_i}$ form a c-trace, and the sequence n_i tends to infinity. The split is guessed using nondeterminism, and the limit property of n_i is tested using one counter, which is incremented after each $\pi_{i,j}$ and reset before each $\pi_{i,1}$. □

The construction above works also for nondeterministic max automata, but we stick to deterministic max automata, since this is the only kind used in this paper. Since emptiness is decidable for ωBS-automata, an immediate corollary of the above theorem is that emptiness is decidable for max automata, even for the nondeterministic variant.

Theorem 3 *Emptiness is decidable for max automata.*

An important ingredient in the proof of Theorem 1 is the conversion of a boolean combination of ωS-automata to a single ωBS-automaton. This conversion, as presented in [4], is nontrivial in two ways: the proof is difficult, and the construction incurs a non-elementary state explosion. In preliminary work with Szymon Toruńczyk, we present a direct emptiness proof, which achieves the optimal PSPACE complexity.

2.1 Myhill-Nerode Equivalence

In this section we show that languages recognized by deterministic max automata have finitely many equivalence classes for a strong Myhill-Nerode equivalence relation. By strong we mean that the equivalence relation is finer than, say, the Arnold congruence.

We actually prove a stronger result, since instead of deterministic max automata, we consider nondeterministic ωBS-automata, which capture them by Theorem 1.

For a language $L \subseteq \Sigma^\omega$, call two finite words L-equivalent if they can be swapped a finite or infinite number of times without L noticing. Formally, words $w, v \in \Sigma^*$ are called L-equivalent if both conditions below hold.

$$u_1 w u_2 \in L \iff u_1 v u_2 \in L \qquad \text{for } u_1 \in \Sigma^*, u_2 \in \Sigma^\omega$$

$$u_1 w u_2 w u_3 w \cdots \in L \iff u_1 v u_2 v u_3 v \cdots \in L \qquad \text{for } u_1, u_2, \ldots \in \Sigma^*.$$

Theorem 4 *Every language L recognized by a max-automaton has finitely many classes of L-equivalence.*

Proof Consider an ωBS-automaton, with input alphabet Σ, states Q, transitions δ and counters C. Consider a finite run $\rho \in \delta^*$ of this automaton. For every counter $c \in C$, there are three things that ρ can do with counter c, which we denote using symbols ϵ, I, R. These things are: (ϵ) neither increment nor reset c; (I) increment c at least once but never reset it; and (R) reset c at least once, possibly incrementing it several times. The profile of a run is defined as the vector $\{\epsilon, I, R\}^C$ which says what the run does to each counter. The profile of a word in Σ^* is defined as the set of all triples in

$$(q, v, p) \in Q \times \{\epsilon, I, R\}^C \times Q$$

such that some run over the word begins in state q, has profile v, and ends in state p. It is not difficult to see that two words with the same profile are L-equivalent. The theorem follows, since there are finitely many profiles for words. □

3 The Logic

We consider an extension of weak monadic second-order logic, called *weak MSO with the unbounding quantifier*. Recall that weak monadic second-order logic is an extension of first-order logic that allows quantification over finite sets (the restriction to finite sets is the reason for the name "weak"). In weak MSO with the unbounding quantifier, we further add the *unbounding quantifier* $U X$, as defined in (2).

Example Consider the set L from (1). This language is not regular, but defined by the following formula of weak MSO with the unbounding quantifier logic:

$$\forall x \exists y (y \geq x \wedge b(y)) \wedge \mathrm{U} X \forall x \leq y \leq z \quad x, z \in X \Rightarrow a(y) \wedge y \in X.$$

The main result of this paper is that the logic and automata coincide.

Theorem 5 *Weak MSO with the unbounding quantifier defines exactly the same languages as deterministic max-automata. Translations both ways are effective.*

The more difficult direction in Theorem 5, from logic to automata, is presented in Sect. 4. The easier direction is given below. The idea is to say that there are arbitrarily large c-traces, as in Fact 2 from the proof of Theorem 1.

Lemma 6 *Every max-automaton can be effectively translated to an equivalent formula of weak MSO with the unbounding quantifier.*

Proof Fix a max-automaton with input alphabet Σ. For two positions $x \leq y$ in an input word $w \in \Sigma^\omega$, we write $\pi[w, x, y]$ for the sequence of counter operations that labels the transitions of the automaton which read the part of w between x and y (beginning with the transition that reads x and ending with the transition that reads y). Note that $\pi[w, x, y]$ also depends on the positions before x, since these determine the state of the automaton before reading x. Let Π be the finite set of words over counter operations that label the individual transitions of the automaton. For each $\pi \in \Pi$ we write a formula $\varphi_\pi(x)$ that is true in a position x of an input word w if and only if $\pi[w, x, x]$ is π. This formula $\varphi_\pi(x)$ only needs to quantify over finite sets, to talk about the run of the automaton over positions prior to x. Using the formulas φ_π, we write for any two counters c and d a formula $\varphi_{c,d}(x, y)$ of weak MSO which holds whenever $\pi[w, x, y]$ transfers counter c to d with an increment. Finally, using the formulas $\varphi_{c,d}$, we write for any counter c a formula $\varphi_c(X)$, with a free set variable, which holds in a word w for a set $X = \{x_1 < \cdots < x_n\}$ if and only if the words

$$\pi[w, x_1, x_2 - 1], \quad \pi[w, x_2, x_3 - 1], \quad \ldots, \quad \pi[w, x_{n-1}, x_n - 1]$$

form a c-trace. Note that the formula φ_c does not use the unbounding quantifier. The key property is that $\mathrm{U} X \varphi_c(X)$ holds in a word w if and only if counter c is unbounded in the unique run over w. A boolean combination these formulas can express the acceptance condition of the automaton. □

The formulas that are sufficient to simulate a deterministic max-automaton are of a special type, which gives a normal form for weak MSO with the unbounding quantifier:

Proposition 7 *Every formula of weak unbounding logic is equivalent to a boolean combination of formulas* $\mathrm{U} X \varphi(X)$, *where* $\varphi(X)$ *does not use the unbounding quantifier.*

Proof By translating a formula into an automaton and then back into a formula. □

4 From Logic to Automata

We now turn to the more difficult part of Theorem 5, which says that every formula of weak MSO with the unbounding quantifier can be effectively translated to an equivalent deterministic max automaton.

To simplify the translation, we use the usual technique of removing first-order quantification, as in [16]. That is, first-order quantification is replaced by three new predicates "set X has one element", "set X is included in set Y" and "all elements of set X are before all elements of set Y". Together with weak second-order quantification, these new three predicates can be used to simulate first-order quantification, so the logic is the same.

The translation is defined by induction on the size of the formula. For purposes of the induction, we do the translation also for formulas with free variables. What is the word language corresponding to a formula $\varphi(X_1, \ldots, X_n)$? This language contains words annotated with valuations for the free set variables. We use the usual encoding, where the label of a word position $x \in \mathbb{N}$ is extended with a bit vector in $\{0,1\}^n$ that says which of the sets $X_1, \ldots, X_n$ contain position x. More formally, for sets of word positions $X_1, \ldots, X_n \subseteq \mathbb{N}$ and an infinite word $w \in \Sigma^\omega$, we define the word

$$w \otimes X_1 \otimes \cdots \otimes X_n \in (\Sigma \times \{0,1\}^n)^\omega$$

as follows. On position x, the new word has a tuple $(a, b_1, \ldots, b_n)$, with a the label of the x-th position of the original word w, and the value of bit b_i being 1 if and only if position x belongs to the set X_i, for $i = 1, \ldots, n$. With this notation, we can define the set of words satisfying a formula $\varphi(X_1, \ldots, X_n)$ to be

$$L_\varphi = \{w \otimes X_1 \otimes \cdots \otimes X_n : w, X_1, \ldots, X_n \models \varphi\}.$$

We will show that for every formula $\varphi(X_1, \ldots, X_n)$ of weak MSO with the unbounding quantifier, there is a max automaton recognizing the language L_φ. The proof is by induction on the size of the formula φ. The induction base, which corresponds to the predicates "set X has one element", "set X is included in set Y" and "all elements of set X are before all elements of set Y" is easy, since all of these are ω-regular languages, and we have:

Lemma 8 *Deterministic max automata capture all ω-regular languages.*

Proof Recall that a *deterministic Muller automaton* is a deterministic automaton where the acceptance condition is a boolean combination of conditions "state q appears infinitely often". Deterministic Muller automata capture all regular languages, see [16], so it suffices to simulate a deterministic Muller automaton by a max automaton. We add a new counter c_q for each state q of the automaton. Each time state q appears, counter c_q is incremented. The counters are never reset. In a run of this automaton, a state appears infinitely often if and only if its counter is unbounded. Therefore, the Muller acceptance condition is encoded in the unbounding condition of a max automaton. □

The induction step for boolean operations—including negation—is no more difficult, since the automata are deterministic and the accepting condition is closed under boolean operations. We are left with weak second-order quantification and the unbounding quantifier. We first deal with weak quantification, in Sect. 4.1, while the unbounded quantifier is treated in Sect. 4.2.

4.1 Weak Existential Quantification

This section is devoted to showing:

Proposition 9 *Languages recognized by deterministic max automata are closed under weak quantification. In other words, if L is a language over $\Sigma \times \{0,1\}$ recognized by a deterministic max automaton, then there is a deterministic max automaton recognizing*

$$\{w \in \Sigma^\omega : w \otimes X \in L \text{ for some finite set } X\}.$$

A convenient way to prove this result would be to use nondeterministic automata. Unfortunately, as we will later show, nondeterministic max automata are strictly more powerful than weak MSO with the unbounding quantifier, so we cannot use this strategy. We will have to do the existential quantification directly in the deterministic automata.

The proof technique is actually very generic. It would work for any model of deterministic automata that recognize all ω-regular languages and satisfies some relaxed assumptions, mainly that the acceptance condition is prefix-independent. A more general framework can be found in [5].

Fix a deterministic max automaton $\mathcal{A}$ that recognizes L, with state space Q and transition set δ. The input alphabet of $\mathcal{A}$ is $\Sigma \times \{0,1\}$.

Consider a word $w \in \Sigma^\omega$. In the proof we will be interested in runs over the word $w \otimes \emptyset$. Intuitively speaking, the word $w \otimes \emptyset$ is important because for every finite set X, the words $w \otimes X$ and $w \otimes \emptyset$ agree on almost all positions. More precisely speaking, we will study partial runs on the word $w \otimes \emptyset$, which are defined as follows. A *partial run* in $w \in \Sigma^\omega$ is a run that begins in any position of the word $w \otimes \emptyset$ (not necessarily the first position) and in any state (not necessarily the initial one). In other words, this is a word in $\bot^* \delta^\omega \cup \bot^\omega$ that is consistent with the word $w \otimes \emptyset$ on those positions where it is defined (i.e. where it is not $\bot$). 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 $\bot^\omega$, which has no seed configuration.) Here, a configuration is a pair (q, x), where x is a position in the word and q is the state of the automaton before reading position x. 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. We will be interested in finite sets of spanning runs.

Lemma 10 *For every word w, there is a set of at most $|Q|$ spanning runs.*

Proof We begin with some arbitrary configuration, and take the partial run ρ_1 that begins in that configuration. If $\{\rho_1\}$ is spanning, then we are done. Otherwise, we take some partial run ρ_2 that does not converge with ρ_1, and see if the set $\{\rho_1, \rho_2\}$ is spanning. If it is not, we add a third partial run ρ_3, and so on. This process terminates after at most $|Q|$ steps, because if two partial runs do not converge, then they must use different states on each position where they are both defined. So $|Q|$ partial runs that do not converge will use up all the states. □

To prove Proposition 9, we use a result stronger than Lemma 10. We will show that not only the spanning set of runs exists, but it can also be computed by a (deterministic, letter-to-letter) transducer. By *transducer* we mean a finite deterministic automaton where each transition is equipped with an output letter, from an output alphabet Γ. Therefore, the transducer defines a function $f : \Sigma^\omega \to \Gamma^\omega$. The transducer does not have any accepting conditions (using bounds or even parity or Muller), it just scans the word and produces its output. It is easy to see that deterministic max automata are closed under preimages of transducers, as stated in the following lemma.

Lemma 11 *If $f : \Sigma^\omega \to \Gamma^\omega$ is a transducer and $\mathcal{A}$ is a deterministic max automaton with input alphabet Γ, then there is a deterministic max automaton with input alphabet Σ recognizing the set of words $w \in \Sigma^\omega$ such that $f(w)$ is accepted by $\mathcal{A}$.*

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 $\mathcal{A}$ in Proposition 9.

A single partial run will be encoded by a sequence of transitions $\rho \in \delta^\omega$ and a set of *invalidating* positions X. The partial run encoded by ρ and X is obtained from ρ by putting $\perp$ on all positions that are in X or are followed by an invalidating position. In particular, if X contains infinitely many positions, then the encoded partial run is $\perp^\omega$. To encode n partial runs as a single word, we use n words of the form $\rho \otimes X$ written in parallel

$$\rho_1 \otimes X_1 \otimes \rho_2 \otimes X_2 \otimes \cdots \otimes \rho_n \otimes X_n \in \big((\delta \times \{0,1\})^n\big)^\omega.$$

With the encoding of spanning runs defined, we are now ready to present the stronger version of Lemma 10.

Lemma 12 *Let $n = |Q|$. There is a transducer*

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

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

Proof The idea is to implement the proof of Lemma 10 in a transducer. The states of the transducer will be permutations of the state space, i.e. tuples from Q^n where each state appears exactly once. The initial state is any arbitrarily chosen permutation.

When reading an input letter $a \in \Sigma$ in state π, the transducer f works as follows. It is supposed to output a tuple

$$(t_1, x_1, \ldots, t_n, x_n) \in (\delta \times \{0,1\})^n.$$

In the above, t_i is the transition of the underlying deterministic automaton $\mathcal{A}$ that is chosen when reading letter $(a,0)$ in the state on the i-th coordinate of π. The invalidating bits x_i will be defined below. Let πa be the tuple of target states in $(t_1, \ldots, t_n)$. This tuple is not necessarily a permutation and cannot be used as a state of the transducer, since on some coordinates $i \in \{1, \ldots, n\}$, the transition t_i might have the same target state as one of the transitions $t_1, \ldots, t_{i-1}$. Let $I = \{i_1, \ldots, i_k\}$ be these coordinates, and let $\{p_1, \ldots, p_m\}$ be the states that do not appear in πa. These two sets have the same size, i.e. $k = m$. We can now correct πa to be a permutation σ, by replacing its coordinate i_1 with the state p_1, the coordinate i_2 with state p_2, and so on. Note that on the coordinates from I, the new permutation σ has a value unrelated to the one from πa (i.e. σ begins a new run), while on coordinates from outside I, the new permutation σ simply continues the runs from π. This is signified in the output of the transducer, which sets the invalidating bit x_i to 1 if $i \in I$ and to 0 otherwise. □

We are now ready to prove Proposition 9. Let $w \in \Sigma^\omega$ be an input word of the max automaton, and let

$$f(w) = \rho_1 \otimes X_1 \otimes \rho_2 \otimes X_2 \otimes \cdots \otimes \rho_n \otimes X_n$$

be the n spanning runs that are output by the transducer f. The word w is accepted by the max automaton if and only if there is some $i \in \{1, \ldots, n\}$ such that the following two properties hold:

(A) The i-th spanning run is defined (i.e. the set X_i contains finitely many invalidating positions) and satisfies the accepting condition in the automaton $\mathcal{A}$.
(B) There is some finite set $X \subseteq \mathbb{N}$ such that the run of $\mathcal{A}$ over $w \otimes X$ converges with the i-th spanning run.

Since languages recognized by max automata are closed under union and intersection, it suffices to show that for each fixed i, both properties (A) and (B) are recognized by deterministic max automata. For property (A), we use Lemma 11 on preimages. Property (B), on the other hand, is an ω-regular property, which can be recognized by a deterministic max automaton thanks to Lemma 8.

4.2 Unbounding Quantification

We now turn to the more difficult step in translating a formula to a max automaton, where the outermost operation in the formula is the U quantifier.

Proposition 13 *Languages recognized by deterministic max automata are closed under unbounding quantification. In other words, if L is a language over $\Sigma \times \{0,1\}$ recognized by a deterministic max automaton, then so is*

$$\mathrm{U}L = \{w \in \Sigma^\omega : w \otimes X \in L \text{ for arbitrarily large finite sets } X\}.$$

Fix a deterministic max automaton $\mathcal{A}$ recognizing the language L in the proposition. Let Q be its state space. For a state $q \in Q$ and a finite word $w \in \Sigma^*$, we define $\max(q, w)$ be the maximal size of a set $X \subseteq \{1, \ldots, |w|\}$ such that the automaton $\mathcal{A}$ reaches state q after reading $w \otimes X$. As stated in the lemma below, these numbers can be computed in the counters of a deterministic max automaton (not surprisingly, using the max operation).

Lemma 14 *There is a deterministic max automaton with counters $\{c_q\}_{q \in Q}$ such that the value of c_q after reading $w \in \Sigma^*$ is exactly* $\max(q, w)$.

Proof We use the following inductive property, for $w \in \Sigma^*$ and $a \in \Sigma$

$$\max(q, wa) = \max_{(p,i)} \max(p, w) + i.$$

In the expression above, the maximum ranges over the pairs $(p, i) \in Q \times \{0, 1\}$ such that automaton $\mathcal{A}$ changes state from p to q when reading the input label (a, i). We assume that the maximum is zero when there is no such pair. To evaluate the subexpressions, several bookkeeping counters are used. □

We will use the values from the above lemma to capture the unbounding quantifier. However, some more effort is needed: it is not the case that an input word $w = a_1 a_2 \cdots$ belongs to $\mathrm{U}L$ if and only if the values $\max(q, a_1 \cdots a_n)$ are unbounded. In general, only the left to right implication holds. The right to left implication may fail since a value $\max(q, a_1 \cdots a_n)$ is relevant only if the run of $\mathcal{A}$ over w that begins in configuration (q, n) can be extended to an accepting one over the rest of the word. The correct characterization is given below:

Lemma 15 *A word $a_1 a_2 \cdots \in \Sigma^\omega$ belongs to* $\mathrm{U}L$ *if and only if for some state q, the values*

$$\{\max(q, a_1 \cdots a_n) : a_{n+1} a_{n+2} \cdots \in K_q\}$$

are unbounded, where

$$K_q = \{w \in \Sigma^\omega : w \otimes \emptyset \text{ is accepted by } \mathcal{A} \text{ when starting in } q\}.$$

As suggested by the above lemma, to recognize the language $\mathrm{U}L$ it would be convenient to have an extension of max automata, where the limsup in the acceptance condition would not look at all values assumed by a counter, but only at values assumed in positions that have a certain future. Below, we introduce such an extension of max automata and show that it can be simulated by a standard max automaton, thus completing the proof of Proposition 13.

An *guarded output max automaton* is like a max automaton, except that each counter c is associated with a *guard* K_c, which is a language recognized by a max automaton. In the acceptance condition, instead of saying that the values of counter c are unbounded, we say that the values of counter c are unbounded in positions n

such that the suffix $a_n a_{n+1} \cdots$ belongs to K_c. By Lemma 15, the language UL is recognized by a guarded output max automaton.

We will show that guarded outputs are redundant, and can be simulated by non-guarded outputs. This completes the proof Proposition 13. The difficulty in the proof below is that we are dealing with deterministic automata, while a guard looks to the future.

Proposition 16 *Every guarded output max automaton is equivalent to a max automaton.*

Proof Let $\mathcal{A}$ be a guarded output max automaton. A *trivial guard* is one that accepts all words. We will show how to take a counter with a nontrivial guard and replace it by a finite number of counters with trivial guards. Once all guards are trivial, we are left with a max automaton.

Let c be a counter with a nontrivial guard K_c, recognized by a max automaton $\mathcal{B}$.

In the construction, we will use a concept of *thread*. A thread consists of a state of the automaton $\mathcal{B}$, as well as a number, which corresponds to the value of counter c at some point where the guard was satisfied. Note that a thread does not contain information about values of the counters of automaton $\mathcal{B}$. The idea is that threads will be alive for only finitely many steps, so the counters of $\mathcal{B}$ are not relevant. We will denote threads by τ. If $a \in \Sigma$ is an input letter, then we write τa for the thread obtained from τ by updating the state according to a (and leaving the number unchanged).

The (non-guarded) max automaton $\mathcal{C}$ that simulates $\mathcal{A}$ works as follows. It keeps track of the state of $\mathcal{A}$ and of its counter values. (So $\mathcal{C}$ has all the counters of $\mathcal{A}$, but also some other ones, defined below.) At each point, the simulating automaton contains a finite set $\{\tau_1, \ldots, \tau_i\}$ of *active* threads. There will be at most one thread per state of $\mathcal{B}$, so the set of threads can be stored using finitely many counters and the finite memory of the automaton. This set of active threads is initially empty. After reading a new letter of the input a, the automaton does the following operations.

- Whenever $\mathcal{A}$ does a guarded output with the guard K_c, a new thread τ is created, with the initial state of $\mathcal{B}$ and the current value of c.
- The threads $\tau_1, \ldots \tau_i, \tau$ are updated to $\tau_1 a, \ldots, \tau_i a, \tau a$.
- If two active threads have the same state, then they are merged, and only the greater number is kept (using the max operation).

Similarly to the proof of Proposition 9, the automaton $\mathcal{C}$ will also read the output of a transducer f that computes spanning partial runs of the automaton $\mathcal{B}$ used for the guards. Recall that the transducer f outputs n spanning partial runs of the automaton $\mathcal{B}$, where n is the number of states in $\mathcal{B}$.

The automaton $\mathcal{C}$ accepts a word w if and only if there is some $i = 1, \ldots, n$ such that:

(A) The i-th run encoded by $f(w)$ is defined (i.e. the encoding contains finitely many invalidating positions) and satisfies the accepting condition in the automaton $\mathcal{B}$.
(B) For every m, some thread storing a number greater than m converges with i-th run encoded by $f(w)$.

Since deterministic max automata are closed under finite union, we only need to show the construction for some fixed i. As in the previous section, property (A) is recognized by a deterministic max automaton. For property (B), we create a new counter c, which contains the maximal number stored in a thread τ that had the same state as ρ_i. (Once a thread has the same state as ρ_i, it continues to have the same state until ρ_i is invalidated.) □

5 Eliminating Max

In this section, we show that the max counter operation can be eliminated, at the cost of equipping the automaton with ω-regular lookahead. Using the automaton without max but with lookahead, we give another decision procedure for emptiness of the automaton, this time without referring to the constructions of [4].

A *lookahead transducer* is a function $f : \Sigma^\omega \to \Gamma^\omega$ such that the label of position x in the output $f(w)$ is determined by a regular property of position x in the input w. One way of presenting such a lookahead transducer is by giving a weak MSO formula $\varphi_a(x)$ over the input alphabet Σ for each letter a of the output alphabet Γ, such that in any word $w \in \Sigma^\omega$, each position x satisfies exactly one of the formulas $\varphi_a(x)$, thus determining the label of x in $f(w)$. Another way would be using a type of automaton, where the transition function reads not only the current input letter, but can ask ω-regular queries about the suffix of the input word that it has not yet read.

A *limsup automaton* is a max automaton that does not use the max counter operation. A *lookahead limsup automaton* is given by a lookahead transducer $f : \Sigma^\omega \to \Gamma^\omega$ and a limsup automaton over alphabet Γ. It accepts a word $w \in \Sigma^\omega$ if the limsup automaton accepts the image $f(w)$.

Theorem 17 *Max automata and lookahead limsup automata capture the same languages. Translations both ways are effective.*

The translation of a lookahead limsup automaton to a max automaton can be done via the logic. It is easy to see that languages defined by weak MSO with the unbounding quantifier are closed under inverse images of lookahead transducers (simply use the formulas of the lookahead transducer instead of letters). In particular, any language recognized by a lookahead limsup automaton is also recognized by a max automaton.

The converse translation also uses the logic. We convert the max automaton into a formula of the logic, and then put the formula into a boolean combination of formulas $\mathrm{U}X\varphi(X)$, using Proposition 7. Since lookahead limsup automata are closed under boolean combinations, it suffices to prove the following proposition. (A less wasteful reduction to the proposition below could also be presented, which would not pass through the logic.)

Proposition 18 *For any ω-regular language L over alphabet $\Sigma \times \{0, 1\}$, the language $\mathrm{U}L$ is recognized by a lookahead limsup automaton.*

In the proof we use factorization forests. We present these in the following section. Next, in Sect. 5.2, we prove Proposition 18.

5.1 Factorization Forests

In the discussion below, it will be more convenient to talk about edges than positions in a word. Consider a word w, either finite or infinite. An *edge* in this word is the space between two positions. We identify edges with numbers, so that edge x is the space between its *source position* x and its *target position* $x+1$. There are one or two dummy edges: an edge 0 with an undefined source, and in case of a finite word with n positions, a dummy edge n undefined target. For edges $x \leq y$, we write $w[x..y]$ for the infix of w that begins in the target of x and ends in the source of y. Given a set of edges E, an E-factor is any infix $w[x..y]$ for $x < y \in E$.

Consider a monoid morphism $\alpha : \Sigma^* \to S$ and a word $w \in \Sigma^\omega$. We say $(s, f) \in S^2$ is a *tail pair* in w if there is an infinite set of edges E such that all E-factors have value f under α, and all prefixes $w[0..x]$ have value s under α, for $x \in E$. We say a language $L \subseteq \Sigma^\omega$ is *recognized* by a α if there is a set of *accepting pairs* $F \subseteq S^2$ such that if L is the set of words that have an accepting tail pair. A language is ω-regular if and only if it is recognized by a morphism into a finite monoid, see [13].

Fix a finite set of colors C, with a linear order $\leq$. An *edge coloring* in a word—finite or infinite—is a function σ that maps each edge to a color. We say edges $x_1 < \cdots < x_n$ are *neighboring* with respect to σ if they are all have the same color c and all edges between x_1 and x_n have colors at most c. Let $\alpha : \Sigma^* \to S$ be a monoid morphism. The edge coloring σ is called an *α-factorization forest* if for any set of neighboring edges, all the factors have the same value under α.

An important result of Imre Simon says that for any morphism α, a finite set of colors is sufficient to produce factorization forests for all finite words. This result can be strengthened in two ways: by allowing ω-words and by requiring the factorization forest to be produced by a lookahead transducer. This strengthening is stated below.

Theorem 19 *For any morphism $\alpha : \Sigma^* \to S$ there is a finite set of colors $C, \leq$ and a lookahead transducer $f : \Sigma^\omega \to C^\omega$ which maps each word to an α-factorization forest.*

Proof We give two proofs. The second proof was suggested by one of the anonymous referees.

The first proof uses a theorem of Colcombet [6]. Note that a factorization forest requires colors on edges, while a lookahead transducer produces colors for positions. We solve this discrepancy by having the color for an edge output on its target position. A theorem of Colcombet [6], which says that there is a deterministic transducer (so the lookahead transducer does not use the lookahead) which produces for every word w an edge coloring τ that satisfies

$$\alpha(w[x..y]) = \alpha(w[x..z]) \quad \text{for neighboring edges } x < y < z. \tag{3}$$

The edge coloring τ can be upgraded into a factorization forest in the following way. Consider an edge x, and let x' be the first neighboring edge after x'. The edge x' may be undefined, if x has no neighboring edges to the right. We define the *type of x* to be the pair $(\tau(x), \alpha(w[x..x']))$. If x' is undefined, the second coordinate is $\perp$. We order the types lexicographically, using some arbitrary order on $S \cup \{\perp\}$. Let σ be the edge

coloring that maps each edge x to its type. Note that σ needs to look to the future, to check the type. We claim that σ is a α-factorization forest. Indeed, consider a set of neighboring edges E. Let (i, s) be the type of these edges. If E has one edge, then it has no factors, so there is nothing to prove. Otherwise, E has at least two edges. In particular, s is defined, since at least the first edge in E has a right neighbor, and therefore a closest right neighbor, under τ. Below, we show that every E-factor is mapped to s by α. Let $w[x..y]$ be such a factor, and let x' be the first neighboring edge after x, under τ.

$$\alpha(w[x..y]) = \alpha(w[x..x'] \cdot w[x'..y]) \overset{(3)}{=} \alpha(w[x..x']) = s.$$

This completes the first proof. We now do the second proof.

First note that factorization forests exist for ω-words. Indeed, consider an ω-word $w \in \Sigma^\omega$. By the Ramsey theorem, w can be decomposed as an infinite concatenation $w = w_0 w_1 \cdots$ so that all the words $w_1, w_2, \ldots$ are mapped to the same idempotent in S. Apply the Simon theorem to the morphism $\alpha : \Sigma^* \to S$, yielding a finite set of colors which is sufficient to get α-factorization forests for all the finite words $w_0, w_1, \ldots$. Finally, add a new color (bigger than all the previous colors), and use it to recolor all the edges that connect two consecutive words w_i and w_{i+1}, for $i \in \mathbb{N}$. It is easy to see that the resulting edge coloring is a α-factorization forest.

We have shown that a finite set of colors, call it C, is sufficient to produce some α-factorization forests for every word $w \in \Sigma^\omega$. We will use the uniformization theorem for MSO on ω-words to show that an α-factorization forest can be output by a lookahead transducer. We say that a partition $\{X_c\}_{c \in C}$ of $\mathbb{N}$ encodes an α-factorization forest in a word $w \in \Sigma^\omega$ if we get an α-factorization by coloring each edge with the color corresponding to the unique set containing the edge target. It is not difficult to see that there is an MSO formula with free variables $\{X_c\}_{c \in C}$ and input alphabet Σ, which says that the free variables encode an α-factorization forest in its input ω-word. By the uniformization theorem [15] for MSO on ω-words, there exists an MSO formula φ, with free variables $\{X_c\}_{c \in C}$ and input alphabet Σ, such that for every input word $w \in \Sigma^\omega$, the formula φ holds for exactly one choice of sets $\{X_c\}_{c \in C}$, which encodes an α-factorization forest. From this formula φ we can get a lookahead transducer. □

5.2 Proof of Proposition 18

For the rest of the proof, we fix a morphism

$$\alpha : (\Sigma \times \{0, 1\}) \to S$$

that recognizes the language L. Consider a finite word $w \in \Sigma^*$. We say $s \in S$ is a *possible value* in w if $s = \alpha(w \otimes X)$ holds for some set X. If X is nonempty, we say s is a *possible nonempty* value. If X is empty, we say s is the *empty value*. Let

$$\beta : \Sigma \to P(S) \times S$$

be the function that maps each word w to: the set of possible nonempty values of w, and the empty value of w. One can equip the target of β with a monoid structure so

that β is a monoid morphism, as described below

$$(S_1, s_1) \cdot (S_2, s_2) = ((s_1 \cup S_1) \cdot (s_2 \cup S_2), s_1 \cdot s_2) \quad \text{for } s_1, s_2 \in S,\ S_1, S_2 \subseteq S.$$

We will be using factorization forests for the morphism β.

Lemma 20 *Fix a word $w \in \Sigma^\omega$ together with a β-factorization. The word w belongs to $\mathrm{U}L$ if and only if it satisfies the following condition.*

(*) *For any sufficiently large $n \in \mathbb{N}$, there is a sequence of edges*

$$0 = x_0 < x_1 < \cdots < x_n < x_{n+1} < \cdots \tag{4}$$

such that $x_1, \ldots, x_n$ are neighboring and (seu, f) is an accepting pair for elements $s, e, u, f \in S$ such that

- *s is a possible value in $w[x_0..x_2]$*
- *e is idempotent and a possible nonempty value in $w[x_2..x_3], \ldots, w[x_{n-2}..x_{n-1}]$*
- *u is a possible value in $w[x_{n-1}..x_{n+1}]$*
- *f is the empty value in $w[x_{n+1}..x_{n+2}], w[x_{n+2}..x_{n+3}] \ldots$.*

Proof The bottom up implication is straightforward, so we only show that $w \in \mathrm{U}L$ implies condition (*). Let $n \in \mathbb{N}$. If we choose a sufficiently large set X with $w \otimes X \in L$, we can find neighboring edges

$$x_1 < \cdots < x_n$$

such that X contains at least one element between each two consecutive edges. Since $w \otimes X$ belongs to L it must have an accepting pair (t, f), which is given by edges

$$x_{n+1} < x_{n+2} < \cdots .$$

By removing a finite number of edges, we can assume without loss of generality that x_{n+1} is after x_n and that the finite set X contains no positions after x_{n+1}. We write x_0 for the edge whose target is the first position. For $i \in \mathbb{N}$, we define w_i to be the part of w between edges x_i and x_{i+1}, and we define s_i to be the value assigned by α to the part of $w \otimes X$ between positions x_i and x_{i+1}. By assumption on the edges $x_{n+1}, x_{n+2}, \ldots$, all the elements $s_{n+1}, s_{n+2}, \ldots$ are equal to f. Since X has no positions after x_{n+1}, the element f satisfies the condition required by (*).

A simple monoid lemma, which can be proved using a Ramsey argument, says that in any sufficiently long sequence $s_1, \ldots, s_n \in S$ there must be indexes $1 < i < j < n$ such that $s_i \cdots s_j$ is an idempotent, call it e. Since the edges $x_1, \ldots, x_n$ are neighboring, the possible nonempty values for a word $w_i \cdots w_j$ are always the same, regardless of the choice $1 \leq i \leq j < n$. In particular e, which is a possible nonempty value in $w_i \cdots w_j$, is also a possible nonempty value in all the words $w_2, \ldots, w_{n-1}$, as required by (*). Since $s_1 \cdots s_{i-1}$ is a possible nonempty value in $w_1 \cdots w_{i-1}$, it is also a possible nonempty value in w_1, and therefore s defined as $s_0 \cdots s_{i-1}$ is a possible (even nonempty) value in $w_0 w_1$, as required by (*). The same argument works for u defined as $s_{j+1} \cdots s_n$. □

Lemma 21 *Property* (*) *can be recognized by a lookahead limsup automaton.*

Proof Fix a word $w \in \Sigma^\omega$ together with a β-factorization forest. A *neighborhood* is a maximal (with respect to inclusion) set of neighboring positions. There is one infinite neighborhood. The general idea is that the automaton will calculate the factorization forest and accept if neighborhoods that satisfy a certain property are of unbounded size. The lookahead is used to calculate the factorization forest and determine which neighborhoods satisfy the certain property.

For a neighborhood we define the following three pieces of information, which are called the neighborhood's *profile*.

- *Starter type*—the set of possible values in the neighborhood's *starter*, which is defined as the part of w before the first edge in the neighborhood.
- *Factor type*—the value under β of any of its factors.
- *Remainder type*—the set of tail pairs for the *remainder*, which is defined as the suffix of w after the last edge in the neighborhood.

The remainder and its type are defined only when the neighborhood is finite. For infinite neighborhoods, the profile contains only the starter type and the factor type. The two types of profile are called *finite* and *infinite* profiles, respectively.

Given a profile, we say $s \in S$ is a *possible nonempty factor value* if s appears in the second coordinate of the factor type (recall that a value of β stores nonempty possible values on the second coordinate, and the empty value on the first coordinate). An element $s \in S$ is called a *possible factor value* if it is a possible nonempty factor value, or it belongs to the second coordinate of the factor type. In other words, a possible factor value is corresponds to a value under α of some decoration of the factor with a set that is either empty or nonempty.

We say a neighborhood is *n-large* if there is a sequence of edges as in (4) that satisfies the properties in (*) and such that $x_1, \ldots, x_n$ are all in the neighborhood. Property (*) is equivalent to saying that there are n-large neighborhoods for arbitrarily large n.

Lemma 22 *There is a set of profiles* Π *such that for every neighborhood, the following conditions are equivalent*: (a) *being n-large*; *and* (b) *having profile in* Π *and at least n-edges.*

We first show how Lemma 22 concludes the proof of Lemma 21. The lookahead limsup automaton works as follows. The automaton has a counter c for each possible color of the factorization forest. After reading the prefix of a word $w \in \Sigma^\omega$ that ends in an edge x, the counter values satisfy the following invariant.

> Let c be any counter, and consider the (at most one) neighborhood that contains c-colored edges both before and after x. If this neighborhood exists and has a profile in the set Π, counter c stores the number of edges in the neighborhood that are before x. Otherwise, counter c stores 0.

It is not difficult to design an automaton that maintains this invariant. It uses the lookahead to calculate the factorization forest and the profiles of neighborhoods. The

automaton accepts if any of the counters c is unbounded. (This will also be the case if the profile of the unique infinite neighborhood belongs to Π.)

We now prove Lemma 22. We only show it when a finite profile is in Π, the criterion for infinite profiles is defined in a similar way. A finite profile is included in Π if there are elements

$$s_1, s_2, e, u_1, u_2, f \in S \quad \text{with } e, f \text{ idempotent} \tag{5}$$

such that s_1 is in the starter type, s_2, u_1 are possible factor values, e is a possible nonempty factor value, (u_2, f) is a pair in the remainder type, and $(s_1 s_2 e u_1 u_2, f)$ is an accepting pair.

It is not difficult to show that the above condition is sufficient for a finite neighborhood with n edges to be n-large. The accepting pair (seu, f) required in property (*) is defined by $s = s_1 s_2$ and $u = u_1 u_2$, while e and f are as taken from the profile.

For the converse implication, we will show that the profile of an n-large finite neighborhood belongs to Π. Let y and z be the first and last edges in the neighborhood. Let the positions x_i be defined as in (4), and likewise s, e, u, f. Since e is a possible nonempty value in some factor of the neighborhood, it is a possible nonempty factor in the profile. Recall that s is a value in $w[0..x_2]$. Since y is between 0 and x_2, there must some $s_1, s_2 \in S$ such that $s = s_1 s_2$, s_1 is a possible value in $w[0..y]$ and s_2 is a possible value in $w[y..x_2]$. Since $w[0..y]$ is the starter, the element s_1 belongs to the starter type of π. Since $w[y..x_2]$ is a factor of the neighborhood, the element s_2 is a possible factor value. In a similar way, we show that u can be decomposed as $u = u_1 u_2$ for some elements $u_1, u_2 \in S$ such that u_1 is a possible value in $w[x_{n-1}, z]$ and u_2 is a possible value in $w[z..x_{n+1}]$.

This completes the proof of Lemma 22, and therefore also of Lemma 21. □

5.3 Emptiness for Lookahead limsup Automata

One advantage of eliminating max in a limsup automaton is that emptiness can be now decided in an elementary and self-contained fashion, without referring to the complicated machinery in [4]. Below we show how to decide emptiness for lookahead limsup automata. This procedure also extends to max automata and weak MSO with the unbounding quantifier, by using the translations from Theorems 17 and 5.

Theorem 23 *Emptiness is decidable for lookahead limsup automata.*

Proof In the proof we will use *nondeterministic* limsup automata. Since they capture nondeterministic Büchi automata, nondeterministic limsup automata are closed under inverse images of lookahead transducers. Therefore, every lookahead limsup automaton can be converted into an equivalent nondeterministic limsup automaton without lookahead.

We will now show that emptiness for nondeterministic limsup automata is in NP-complete. NP-hardness is immediate, since emptiness is at least as hard as solving satisfiability for boolean formula in the acceptance condition. We show membership in NP.

Fix a nondeterministic limsup automaton $\mathcal{A}$. Let δ be its transitions and Q its states. Our strategy is to convert each accepting run of $\mathcal{A}$ into a certain normal form, and show that accepting runs in normal form can be found in NP.

Fix an accepting run $\rho \in \delta^\omega$. Let U be the counters which are unbounded (under the counter operations labeling this run), likewise let B be the bounded counters. For a position x in ρ, the *state in* x is the source state of the transition labeling x. Let q be a state that appears infinitely often in ρ. Let x be a position with state q such that after x every counter in B is either reset infinitely often, or never incremented. Consider now a counter $c \in U$. Since counter c assumes arbitrarily large values, we can find arbitrarily long sequences of positions $x_1 < \cdots < x_n$ to the right of x which are labeled by an increment on counter c, and such that counter c is not reset between positions x_1 and x_i. Let m be the maximal value assumed by the bounded counters. If n is chosen greater than $m \cdot |B| \cdot |Q|$, we can find two positions $x_i < x_j$ with the same state q_c such that each bounded counter is either not incremented between them, or it is reset at least once. Let π_c be the part of the accepting run between x and x_i. Let ρ_c be the part of the accepting run between x_i and x_j. Let σ_c be the part of the accepting run between x_j and some position $y > x_j$ with state q.

Above, we have shown how that if there is a run, then we can find

$$B, U \subseteq C \quad q \in Q \quad \{\pi_c \in \delta^*, \rho_c \in \delta^*, \sigma_c \in \delta^*\}_{c \in U} \quad \{q_c \in Q\}_{c \in U} \tag{6}$$

such that B, U satisfy the acceptance condition[5] in the limsup automaton and the following conditions hold for each $c \in U$

1. Run π_c goes from q to q_c, run ρ_c goes from q_c to q_c, and run σ_c goes from q_c to q.
2. Run ρ_c contains at least one increment and no reset on c.
3. For every counter $d \in B$, run ρ_c contains either no increment, or some reset on d. Likewise for $\pi_c \rho_c \sigma_c$.

Conversely, if we can find elements as in (6), there exists an accepting run, namely

$$\pi_0 \quad \pi_{c_1} \rho_{c_1} \sigma_{c_1} \cdots \pi_{c_n} \rho_{c_n} \sigma_{c_n} \quad \pi_{c_1} (\rho_{c_1})^2 \sigma_{c_1} \cdots \pi_{c_n} (\rho_{c_n})^2 \sigma_{c_n} \quad \cdots$$

where π_0 is a run that leads the automaton to state q—we assume all states in $\mathcal{A}$ are reachable—and $c_1, \ldots, c_n$ is some enumeration of the counters in U.

Therefore, proving the theorem boils down to finding runs as described above, by a nondeterministic polynomial time machine. The machine uses nondeterminism to guess the sets B, U, as well as the states q and $\{q_c\}_{c \in U}$. Also, for each counter $c \in U$ the machine nondeterministically guesses sets

$$\delta_{\pi,c}, \delta_{\rho,c}, \delta_{\sigma,c} \subseteq \delta$$

which represent the transitions that will be used by the runs π_c, ρ_c, σ_c. These sets have to be consistent with condition 3 above. Namely, for every counter $d \in B$, if the

[5]Formally speaking, the boolean propositional formula in the acceptance condition is made true by the valuation which assigns "true" to the counters in B and "false" to the other counters.

set $\delta_{\rho,c}$ contains a transition that increments d, then it must also contain a transition that resets d; likewise for the union $\delta_{\pi,c} \cup \delta_{\rho,c} \cup \delta_{\sigma,c}$. After doing all this guessing, the machine has to answer for each $c \in U$ the following question: is there a run π_c from q to q_c that uses exactly the transitions from $\delta_{\pi,c}$? (And similar questions for ρ and σ.) These questions can be answered in polynomial time (even deterministic polynomial time). □

6 Problems with Nondeterminism

In this section we show that nondeterministic max automata are more expressive than deterministic ones.

Theorem 24 *Nondeterministic max automata recognize strictly more languages than deterministic ones.*

Contrast this result with the situation for Muller or parity automata, which are equally expressive in the deterministic and nondeterministic variants. Since full monadic second-order can capture nondeterministic automata by existentially quantifying over infinite sets, the above theorem immediately implies:

Corollary 25 *Full monadic second-order logic with the unbounding quantifier is stronger than weak monadic second-order with the unbounding quantifier.*

The separating language in Theorem 24 is

$$L = \{a^{n_1} b a^{n_2} b a^{n_3} b \ldots : \liminf n_i < \infty\}. \tag{7}$$

This language is captured by a nondeterministic max automaton. The automaton uses nondeterminism to output a subsequence of $n_1, n_2, \ldots$ and accepts if this subsequence is bounded.

It remains to show that the language L cannot be recognized by a deterministic max automaton. For this, we will use topological complexity. In Lemmas 26 and 27, we will show that every language recognized by a deterministic max automaton is a boolean combination of sets on level Σ_2 in the Borel hierarchy, while the language L is not.

Below we briefly describe the Borel hierarchy, a way of measuring the complexity of a subset of a topological space. See [13] for more on the topological approach to languages of infinite words. The topology that we use on words is that of the Cantor space, as described below. A set of infinite words (over a given alphabet Σ) is called *open* if it is a union

$$\bigcup_{i \in I} w_i \Sigma^\omega, \quad w_i \in \Sigma^*,$$

with the index set I being possibly infinite. In other words, membership of a word w in an open set is assured already by a finite prefix of w. For the Borel hierarchy,

as far as max automata are concerned, we will only be interested in the first two levels $\Sigma_1, \Pi_1, \Sigma_2, \Pi_2$. The open subsets are called Σ_1, the complements of these (the closed subsets) are called Π_1. Countable intersections of open subsets are called Π_2, the complements of these (countable unions of closed subsets) are called Σ_2.

Lemma 26 *Any language accepted by a deterministic max automaton is a boolean combination of Σ_2 sets.*

Proof Fix a max automaton $\mathcal{A}$, and a counter c of this automaton. We will examine the topological complexity of the set of runs of this automaton. For any fixed n, the following set of runs is clearly open:

Counter c has value at least n at some point in the run.

In particular, its complement

Counter c has value at most n at all points in the run.

is a closed set of runs. By taking a countable union of the above over $n \in \mathbb{N}$, we deduce that the property

The values of counter c are bounded.

is a Σ_2 property. In particular, the set of accepting runs of any max automaton is a boolean combination of Σ_2 sets. Since the automata are deterministic, the function that maps an input word to its run is continuous, i.e. preimages of open sets are also open. Since preimages of continuous functions preserve the levels of the hierarchy, we conclude that any language accepted by a deterministic max automaton is a boolean combination of Σ_2 sets. □

Lemma 27 *The language L is not a boolean combination of Σ_2 sets.*

Proof Consider the mapping from $\mathbb{N}^\omega$ to $\{a, b\}^\omega$ defined by

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

This is a continuous mapping. The preimage, under this mapping, of the language L is the set of sequences with $\liminf n_i < \infty$. This set is known not to be a boolean combination of Σ_2 sets, see Exercise 23.2 in [11]. □

7 Conclusion

This paper is intended as a proof of concept. The concept is that ω-regular languages can be extended in various ways, while still preserving good closure properties and decidability. The class presented in this paper, languages recognized by (deterministic) max regular automata, is closed under boolean operations, inverse morphisms, and quotients. It is not closed under morphic images (which corresponds to nondeterminism on the automaton side).

Acknowledgement I would like to thank the anonymous referees for their helpful comments.

References

1. Abdulla, P.A., Krcál, P., Yi, W.: R-automata. In: CONCUR, pp. 67–81 (2008)
2. Arnold, A.: A syntactic congruence for rational omega-language. Theor. Comput. Sci. **39**, 333–335 (1985)
3. Bojańczyk, M.: A bounding quantifier. In: Computer Science Logic. Lecture Notes in Computer Science, vol. 3210, pp. 41–55. Springer, Berlin (2004)
4. Bojańczyk, M., Colcombet, T.: Omega-regular expressions with bounds. In: Logic in Computer Science, pp. 285–296 (2006)
5. Bojańczyk, M., Toruńczyk, S.: Deterministic automata and extensions of weak MSO. Submitted
6. Colcombet, T.: Factorization forests for infinite words and applications to countable scattered linear orderings. Theor. Comput. Sci. **411**(4–5), 751–764 (2010). doi:10.1016/j.tcs.2009.10.013, DBLP, http://dblp.uni-trier.de
7. Colcombet, T., Löding, C.: The nesting-depth of disjunctive mu-calculus for tree languages and the limitedness problem. In: Computer Science Logic. Lecture Notes in Computer Science, vol. 5213. Springer, Berlin (2008)
8. Colcombet, T., Löding, C.: The non-deterministic Mostowski hierarchy and distance-parity automata. In: International Colloquium on Automata, Languages and Programming. Lecture Notes in Computer Science, vol. 5126, pp. 398–409. Springer, Berlin (2008)
9. Droste, M., Gastin, P.: Weighted automata and weighted logics. Theor. Comput. Sci. **380**(1–2), 69–86 (2007)
10. Hashiguchi, K.: Algorithms for determining relative star height and star height. Inf. Comput. **78**(2), 124–169 (1988)
11. Kechris, A.S.: Classical Descriptive Set Theory. Graduate Texts in Mathematics, vol. 156. Springer, Berlin (1995)
12. Kirsten, D.: Distance desert automata and the star height problem. Theor. Inform. Appl. **39**(3), 455–511 (2005)
13. Perrin, D., Pin, J.-É.: Infinite Words. Elsevier, Amsterdam (2004)
14. Schützenberger, M.P.: On the definition of a family of automata. Inf. Control **4**, 245–270 (1961)
15. Siefkes, D.: The recursive sets in certain monadic second order fragments of arithmetic. Arch. Math. Log. Grundlagenforschung **17**, 71–80 (1975)
16. Thomas, W.: Languages, automata, and logic. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Language Theory, vol. III, pp. 389–455. Springer, Berlin (1997)