BOUNDEDNESS IN LANGUAGES OF INFINITE WORDS

MIKOŁAJ BOJAŃCZYK AND THOMAS COLCOMBET

Warsaw University
e-mail address: bojan@mimuw.edu.pl

Cnrs/Liafa/Université Paris Diderot, Paris 7
e-mail address: thomas.colcombet@liafa.jussieu.fr

ABSTRACT. We define a new class of languages of ω-words, strictly extending ω-regular languages.

One way to present this new class is by a type of regular expressions. The new expressions are an extension of ω-regular expressions where two new variants of the Kleene star L^* are added: L^B and L^S. These new exponents are used to say that parts of the input word have bounded size, and that parts of the input can have arbitrarily large sizes, respectively. For instance, the expression $(a^B b)^\omega$ represents the language of infinite words over the letters a, b where there is a common bound on the number of consecutive letters a. The expression $(a^S b)^\omega$ represents a similar language, but this time the distance between consecutive b's is required to tend toward the infinite.

We develop a theory for these languages, with a focus on decidability and closure. We define an equivalent automaton model, extending Büchi automata. The main technical result is a complementation lemma that works for languages where only one type of exponent—either L^B or L^S—is used.

We use the closure and decidability results to obtain partial decidability results for the logic MSOLB, a logic obtained by extending monadic second-order logic with new quantifiers that speak about the size of sets.

1. INTRODUCTION

In this paper we introduce a new class of languages of infinite words. The new languages of this kind—called ωBS-regular languages—are defined using an extended form of ω-regular expressions. The extended expressions can define properties such as "words of the form $(a^*b)^\omega$ for which there is an upper bound on the number of consecutive letters a". Note that this bound depends upon the word, and for this reason the language is not ω-regular. This witnesses that ωBS-regular languages are a proper extension of ω-regular languages.

The expressions for ωBS-regular languages are obtained by extending the usual ω-regular expressions with two new variants of the Kleene star L^*. These are called the bounded exponent L^B and the strongly unbounded exponent L^S. The idea behind B is that the language L in the expression L^B must be iterated a bounded number of times,

the bound being fixed for the whole word. For instance, the language given in the first paragraph is described by the expression $(a^B b)^\omega$. The idea behind S is that the number of iterated concatenations of the language L must tend toward infinity (i.e., have no bounded subsequence).

This paper is devoted to developing a theory for the new languages. There are different motivations for such a study. The first one is the interest of the model itself: it extends naturally the standard framework of ω-regular languages, while retaining some closure and decidability properties. We also show that, just as ω-regular expressions define the same languages of infinite words as the ones definable in monadic second-order logic, the class of ωBS-regular languages also admits a logical counterpart. The relevance of the model is also quite natural: the use of bounding arguments in proofs is very common, and the family of ωBS-regular languages provides a unified framework for developing such arguments. A notable example is the famous star-height problem, to which we will return later in the introduction. Another application of our results, which is presented in this paper, is an algorithm deciding if an ω-automatic graph has bounded out-degree. We believe that more problems are related to this notion of regularity with bounds. In this paper, we concentrate on a basic theoretical study of the model.

The first step in this study is the introduction of a new family of automata over infinite words, extending Büchi automata, which we call bounding automata. Bounding automata have the same expressive power as ωBS-regular expressions. (However, the translations between bounding automata and ωBS-regular expressions are more involved than in the case of ω-regular languages.) A bounding automaton is a finite automaton equipped with a finite number of counters. During a run, these counters can be incremented and reset, but not read. The counter values are used in the acceptance condition, which depends on their asymptotic values (whether counter values are bounded or tend toward infinity). Thanks to the equivalence between automata and expressions, and using simple automata constructions, we obtain the closure of ωBS-regular languages under union, intersection and projection.

Unfortunately, ωBS-regular automata cannot be determinized. Even more problematic, ωBS-regular languages are not closed under complement. However, we are able to identify two fragments of ωBS-regular languages that complement each other. We show that the complement of a language that only talks about bounded sequences is a language that only talks about sequences tending toward infinity; and vice versa. The difficult proof of this complementation result is the technical core of the paper.

Finally, we present a logic that captures exactly the ωBS-regular languages, i.e. is equivalent to both the ωBS-regular expressions and automata. As it is well known, languages defined by ω-regular expressions are exactly the ones definable in monadic second-order logic. What extension of this logic corresponds to ωBS-regular expressions? Our approach is to add a new quantifier, called the existential unbounding quantifier $\mathbb{U}$. A formula $\mathbb{U}X.\phi(X)$ is true if it is possible to find sets satisfying $\phi(X)$ of arbitrarily large size. Every ωBS-regular language can be defined in monadic second-order logic extended with $\mathbb{U}$. However, due to the failure of the closure under complementation, the converse does not hold. By restricting the quantification patterns, we identify fragments of the logic that correspond to the various types of ωBS-regular expressions introduced in this paper.

Related work. This work tries to continue the long lasting tradition of logic/automata correspondences initiated by Büchi [3, 4] and continued by Rabin [12], only to mention the

most famous names (see [14] for a survey). We believe that bounding properties extend the standard notion of regularity in a meaningful way, and that languages defined by our extended expressions have every right to be called regular, even though they are not captured by Büchi automata. For instance, every ωBS-regular language L has a finite number of quotients $w^{-1}L$ and Lw^{-1}. (Moreover, the right quotients Lw^{-1} are regular languages of finite words.) Unfortunately, we do not have a full complementation result.

The quantifier $\mathbb{U}$ in the logic that describes ωBS-regular languages was already introduced in [1]. More precisely, the quantifier studied in [1] is $\mathbb{B}$: a formula $\mathbb{B}X.\phi$ expresses that there is a bound on the size of sets X satisfying property ϕ. This formula is semantically equivalent to $\neg(\mathbb{U}X.\phi)$. Although [1] went beyond words and considered infinite trees, the proposed satisfiability algorithm worked for a very restricted fragment of the logic with no (not even partial) complementation. Furthermore, no notion of automata or regular expression was proposed.

Boundedness properties have been considered in model-checking. For instance, [2] considered systems described by pushdown automata whose stack size is unbounded.

Our work on bounds can also be related to cardinality restrictions. In [11], Klaedtke and Ruess considered an extension of monadic second-order logic, which allowed cardinality constraints of the form

$$|X_1| + \cdots + |X_n| \leq |Y_1| + \cdots + |Y_m| \ .$$

In general, such cardinality constraints (even $|X| = |Y|$) lead to undecidability of the satisfiability problem. Even though cardinality constraints can express all ωBS-regular languages, the decidable fragments considered in [11] are insufficient for our purposes: those fragments capture only a small fragment of ωBS-regular languages.

Finally, the objects we manipulate are related to the (restricted) star-height problem. This problem is, given a natural number k and a regular language of finite words L, to decide if L can be defined by a regular expression where the nesting depth of Kleene-stars is at most k. This problem, first raised by Eggan in 1963 [7], has a central role in the theory of regular languages. It has been first shown decidable by Hashiguchi [8] via a difficult proof. The correctness of this proof is still unclear. A new proof, due to Kirsten [10], reduces the star-height problem to the limitedness problem for nested distance desert automata. The latter are finite state automata, which assign a natural number to each word. The limitedness problem is the question whether there is a bound on the numbers produced by a given automaton. Nested distance desert automata—we were not aware of their existence when developing the theory of ωBS-regular languages—happen to be syntactically equivalent to the hierarchical B-automata that we use as an intermediate object in the present work. The semantics of the two models are also tightly connected, and it is possible to derive from the result of our paper the decidability of the limitedness problem for nested distance desert automata (though using our more general results, we obtain a non-elementary complexity, which is far beyond the optimal PSPACE algorithm of Kirsten).

Structure of the paper. In Section 2, we formally define the ωBS-regular expressions that are the subject of this paper. We introduce two restricted types of expressions (where the B and S exponents are prohibited, respectively) and give an overview of the closure properties of the respective expressions. In Section 3, we introduce our automata models and show that they are equivalent to the ωBS-regular expressions. In Section 4, we show how our results can be applied to obtain a decision procedure for satisfiability in an extension

of monadic second-order logic. The main technical result, which concerns closure under complementation, is given at the end of the paper, in Section 5.

We would like to thank the anonymous referees, who contributed enormously to this paper, through their many thoughtful and helpful comments.

2. Regular Expressions with Bounds

In this section we define ωBS-regular expressions. The expressions are the first of three means of defining languages with bounds. The other two—automata and logic—are presented in the next sections.

2.1. **Definition.** In a nutshell, to the standard operations used in ω-regular expressions, we add two variants of the Kleene star $*$: the B and S exponents. These are used to constrain the number of iterations, or more precisely the asymptotic behavior of the number of iterations (this makes sense since the new exponents are used in the context of an ω exponent. When the B exponent is used, the number of iterations has to be bounded by a bound which depends on the word. When the S exponent is used, it has to tend toward infinity. For instance, the expression $(a^B b)^\omega$ represents the words in $(a^* b)^\omega$ where the size of sequences of consecutive a's is bounded. Similarly, the expression $(a^S b)^\omega$ requires the size of (maximal) sequences of consecutive a's to tend toward infinity. These new expressions are called ωBS-regular expressions. A more detailed definition is presented below.

Just as an ω-regular expression uses regular expressions of finite words as building blocks, for ωBS-regular expressions one also needs first to define a finitary variant, called BS-regular expressions. Subsection 2.1.1 presents this finitary variant, while Subsection 2.1.2 introduces ωBS-regular expressions.

2.1.1. *BS-regular expressions.* In the following we will write that an infinite sequence of natural numbers $g \in \mathbb{N}^\omega$ is *bounded* if there exists a global bound on the $g(i)$'s, i.e., if $\limsup_i g(i) < +\infty$. This behavior is denoted by the letter B. The infinite sequence is *strongly unbounded* if it tends toward infinity, i.e., $\liminf_i g(i) = +\infty$. This behavior is denoted by the letter S. Let us remark that an infinite sequence is bounded iff it has no strongly unbounded infinite subsequence, and that an infinite sequence is strongly unbounded iff it has no bounded infinite subsequence.

A *natural number sequence* g is a finite or infinite sequence of natural numbers that we write in a functional way: $g(0), g(1), \ldots$ Its length is $|g|$, which may possibly be ∞. We denote by $\mathbb{N}^\infty$ the set of sequences of natural numbers, both finite and infinite. A sequence of natural numbers g is *non-decreasing* if $g(i) \leq g(j)$ holds for all $i \leq j < |g|$. We write $g \leq n$ to say that $g(i) \leq n$ holds for all $i < |g|$. For g a non-decreasing sequence of natural numbers, we define g' to be $g'(i) = g(i+1) - g(i)$ for all i such that $i + 1 < |g|$. The sequence g is of *bounded difference* if g' is either finite or bounded; it is of *strongly unbounded difference* if the sequence g' is either finite or strongly unbounded.

A *word sequence* $\vec{u}$ over an alphabet Σ is a finite or infinite sequence of finite words over Σ, i.e., an element of $(\Sigma^*)^\infty$. The components of the word sequence $\vec{u}$ are the finite words $u_0, u_1, \ldots$; we also write $\vec{u} = \langle u_0, u_1, \ldots \rangle$. We denote by ε the finite word of length 0, which is different from the word sequence of length 0 denoted $\langle \rangle$. We denote by $|\vec{u}|$ the length of the word sequence $\vec{u}$.

A *language of word sequences* is a set of word sequences. The finitary variant of ωBS-regular expressions will describe languages of word sequences. Note that the finiteness concerns only the individual words in the sequences, while the sequences themselves will sometimes be infinitely long.

We define the following operations, which take as parameters languages of word sequences K, L.

The *concatenation* of K and L is defined by

$$K \cdot L = \{\langle u_0 v_0, u_1 v_1 \ldots \rangle \ : \ \vec{u} \in K, \ \vec{v} \in L, \ |\vec{u}| = |\vec{v}|\} \ .$$

The *mix* of K and L (which is *not* the union) is defined by

$$K + L = \{\vec{w} \ : \ \vec{u} \in K, \ \vec{v} \in L, \ \forall i < \max(|\vec{w}|, |\vec{u}|).w_i \in \{u_i\} \cup \{v_i\}\} \ ,$$

with the convention that $\{u_i\}$ (respectively, $\{v_i\}$) is $\emptyset$ if $i \geq |\vec{u}|$ (respectively, if $i \geq |\vec{v}|$).

The $*$ *exponent* of L is defined by grouping words into blocks:

$$L^* = \{\langle u_0 \ldots u_{g(0)-1}, u_{g(0)} \ldots u_{g(1)-1}, \ldots \rangle \ : \ \vec{u} \in L, \ g \in \mathbb{N}^\infty, \ g \leq |u|, \ g \text{ non-decreasing}\} \ .$$

The *bounded exponent* L^B is defined similarly:

$$L^B = \{\langle u_0 \ldots u_{g(0)-1}, u_{g(0)} \ldots u_{g(1)-1}, \ldots \rangle \ :$$
$$\vec{u} \in L, \ g \in \mathbb{N}^\infty, \ g \leq |u|, \ g \text{ non-decreasing of bounded difference}\} \ .$$

And the *strongly unbounded exponent* L^S is defined by:

$$L^S = \{\langle u_0 \ldots u_{g(0)-1}, u_{g(0)} \ldots u_{g(1)-1}, \ldots \rangle \ :$$
$$\vec{u} \in L, \ g \in \mathbb{N}^\infty, \ g \leq |u|, \ g \text{ non-decreasing of strongly unbounded difference}\} \ .$$

We now define BS-regular expressions using the above operators.

Definition 2.1. A *BS-regular expression* has the following syntax (a being some letter of the given finite alphabet Σ):

$$e = \ \emptyset \mid \varepsilon \mid \bar{\varepsilon} \mid a \mid e \cdot e \mid e + e \mid e^* \mid e^B \mid e^S \ .$$

As usual, we often omit the $\cdot$ symbols in expressions.

The semantic $[\![e]\!]$ of a BS-regular expression e is the language of word sequences defined inductively by the following rules:

- $[\![\emptyset]\!] = \{\langle\rangle\}$.
- $[\![\bar{\varepsilon}]\!] = \{\langle\rangle, \langle\varepsilon\rangle, \langle\varepsilon, \varepsilon\rangle, \ldots\}$, i.e., all finite word sequences built with ε.
- $[\![\varepsilon]\!] = [\![\bar{\varepsilon}]\!] \cup \{\langle\varepsilon, \varepsilon, \ldots\rangle\}$, i.e., all word sequences built with ε.
- $[\![a]\!] = \{\langle\rangle, \langle a\rangle, \langle a, a\rangle, \ldots\} \cup \{\langle a, a, \ldots\rangle\}$, i.e., all word sequences built with a.
- $[\![e \cdot f]\!] = [\![e]\!] \cdot [\![f]\!]$, $[\![e + f]\!] = [\![e]\!] + [\![f]\!]$, $[\![e^*]\!] = [\![e]\!]^*$, $[\![e^B]\!] = [\![e]\!]^B$, and $[\![e^S]\!] = [\![e]\!]^S$.

A language of word sequences is called *BS-regular* if it is obtained by evaluating a BS-regular expression. The *B-regular* (respectively, *S-regular*) languages correspond to the particular case where the expression does not use the exponent S (respectively, B).

Example 2.2. The B-regular expression a^B represents the finite or infinite word sequences that consist of words from a^* where the number of a's is bounded:

$$[\![a^B]\!] = \{\langle a^{f(0)}, a^{f(1)}, \ldots\rangle \ : \ f \in \mathbb{N}^\infty \text{ is bounded}\} \ .$$

The language of word sequences $[\![a^B \cdot (b \cdot a^B)^S]\!]$ consists of word sequences where the number of consecutive a's is bounded, while the number of b's in each word of the word sequence is strongly unbounded.

Except for the two extra exponents B and S and the constant $\bar{\varepsilon}$, these expressions coincide syntactically with the standard regular expressions. Therefore, one may ask, how do our expressions correspond to standard regular expressions on their common syntactic fragment, which includes the Kleene star $*$, concatenation $\cdot$ and union $+$? The new expressions are a conservative extension of standard regular expressions in the following sense. If one takes a standard regular expression e defining a language of finite words L and evaluates it as a BS-regular expression, the resulting language of word sequences is the set of word sequences where every component belongs to L, i.e. $[\![e]\!] = \{\vec{u} \ : \ \forall i < |u|. \, u_i \in L\}$.

In the fact below we present two important closure properties of languages defined by BS-regular expressions. We will often use this fact, sometimes without explicitly invoking it.

Fact 2.3. For every BS-regular language L, the following properties hold:
(1) $L + L = L$,
(2) $L = \{\langle u_{f(0)}, u_{f(1)}, \dots \rangle \ : \ \vec{u} \in L, \ f \in \mathbb{N}^\infty, \ f \le |u|, f \text{ strongly unbounded}\}$.

Proof. A straightforward structural induction. □

Item 2 implies that BS-regular languages are closed under taking subsequences. That is, every BS-regular language of word sequences is closed under removing, possibly infinitely many, component words from the sequence. In particular every BS-regular language is closed under taking the prefixes of word sequences.

2.1.2. *ωBS-regular expressions.* We are now ready to introduce the ωBS-regular expressions. These describe languages of ω-words. From a word sequence we can construct an ω-word by concatenating all the words in the word sequence:

$$\langle u_0, u_1, \dots \rangle^\omega = u_0 u_1 \dots$$

This operation—called the *ω-power*—is only defined if the word sequence has nonempty words on infinitely many coordinates. The ω-power is naturally extended to languages of word sequences by taking the ω-power of every word sequence in the language (where it is defined).

Definition 2.4. An *ωBS-regular expression* is an expression of the form:

$$e = \sum_{i=1}^{n} e_i \cdot f_i^\omega$$

in which each e_i is a regular expression, and each f_i is a BS-regular expression. The expression is called ωB-*regular* (respectively, ωS-regular) if all the expressions f_i are B-regular (respectively, S-regular).

The semantic interpretation $[\![e]\!]$ is the language of ω-words

$$\bigcup_{i=1\dots n} [\![e_i]\!]_F \cdot [\![f_i]\!]^\omega,$$

in which $[\![\cdot]\!]_F$ denotes the standard semantic of regular expressions, and $\cdot$ denotes the concatenation of a language of finite words with a language of ω-words. Following a similar

tradition for regular expressions, we will often identify an expression with its semantic interpretation, writing, for instance, $w \in (a^B b)^\omega$ instead of $w \in [\![(a^B b)^\omega]\!]$.

A language of ω-words is called ωBS-*regular* (respectively, ωB-regular, ωS-regular) if it is equal to $[\![e]\!]$ for some ωBS-regular expression e (respectively, ωB-regular expression e, ωS-regular expression e).

In ωBS-regular expressions, the language of finite words can be $\{\varepsilon\}$; to avoid clutter we omit the language of finite words in such a situation, e.g., we write a^ω for $\{\varepsilon\} \cdot a^\omega$.

This definition differs from the definition of ω-regular expressions only in that the ω exponent is applied to BS-regular languages of word sequences instead of regular word languages. As one may expect, the standard class of ω-regular languages corresponds to the case of ωBS-regular languages where neither B nor S is used (the presence of $\bar{\varepsilon}$ does not make any difference here).

Example 2.5. The expression $(a^B b)^\omega$ defines the language of ω-words containing an infinite number of b's where the possible number of consecutive a's is bounded. The language $(a^S b)^\omega$ corresponds to the case where the lengths of maximal consecutive sequences of a's tend toward infinity. The language $(a + b)^* a^\omega + ((a^* b)^* a^S b)^\omega$ is more involved. It corresponds to the language of words where either there are finitely many b's (left argument of the sum), or the number of consecutive a's is unbounded but not necessarily strongly unbounded (right argument of the sum). This is the complement of the language $(a^B b)^\omega$.

Fact 2.6. Emptiness is decidable for ωBS-regular languages.

Proof. An ωBS-regular language is nonempty if and only if one of the languages $M \cdot L^\omega$ in the union defining the language is such that the regular language M is nonempty and the BS-regular language L contains a word sequence with infinitely many nonempty words. Therefore the problem boils down to checking if a BS-regular language contains a word sequence with infinitely many nonempty words. Let $\mathcal{N}$ be the set of BS-regular languages with this property, and let $\mathcal{I}$ be the set of BS-regular languages that contain at least one infinite word sequence (possibly with finitely many nonempty words). The following rules determine which languages belong to $\mathcal{I}$ and $\mathcal{N}$.

- $K + L \in \mathcal{I}$ iff $K \in \mathcal{I}$ or $L \in \mathcal{I}$.
- $K \cdot L \in \mathcal{I}$ iff $K \in \mathcal{I}$ and $L \in \mathcal{I}$.
- L^* and L^B always belong to $\mathcal{I}$.
- $L^S \in \mathcal{I}$ iff $L \in \mathcal{I}$.
- $K + L \in \mathcal{N}$ iff $K \in \mathcal{N}$ or $L \in \mathcal{N}$.
- $K \cdot L \in \mathcal{N}$ iff either $K \in \mathcal{I}$ and $L \in \mathcal{N}$, or $K \in \mathcal{N}$ and $L \in \mathcal{I}$.
- L^*, L^B and L^S belong to $\mathcal{N}$ iff $L \in \mathcal{N}$. $\qquad\square$

The constant $\bar{\varepsilon}$ will turn out to be a convenient technical device. In practice, $\bar{\varepsilon} \cdot L$ restricts a language of word sequences L to its finite sequences. We denote by $\bar{L}$ the language of word sequences $\bar{\varepsilon} \cdot L$. This construction will be used for instance when dealing with intersections: as an example, the equality $L^B \cap L^S = \bar{L}^*$ holds when L does not contain ε. It turns out that $\bar{\varepsilon}$ is just syntactic sugar, as stated by:

Proposition 2.7. *Every ωBS-regular expression (respectively, ωB-regular and ωS-regular ones) is equivalent to one without the $\bar{\varepsilon}$ constant.*

Note that this proposition does not mean that $\bar{\varepsilon}$ can be eliminated from BS-regular expressions. It can only be eliminated under the scope of the ω-power in ωBS-regular

expressions. For instance, $\bar{\varepsilon}$ is necessary to capture the BS-regular language $\bar{a}$. However, once under the ω exponent, $\bar{\varepsilon}$ becomes redundant; for instance $(\bar{a})^\omega$ is equivalent to $\emptyset$, while $(\bar{a} + b)^\omega$ is equivalent to $(a + b)^* b^\omega$.

Proposition 2.7 will follow immediately once the following lemma is established:

Lemma 2.8. *Let $T \in \{BS, B, S\}$. For every T-regular expression, there is an equivalent one of the form $\bar{M} + L$ where M is a regular expression and L is T-regular and does not use $\bar{\varepsilon}$.*

Proof. By structural induction, we prove that for every BS-regular expression L can be equivalently expressed as

$$L = \bar{M} + K \ ,$$

where M is obtained from L by replacing exponents B and S by Kleene stars and replacing $\bar{\varepsilon}$ by ε, and K is obtained from L by replacing $\bar{\varepsilon}$ by $\emptyset$.

Note that Fact 2.3 is used implicitly above. $\qquad\square$

2.2. **Summary: The Diamond.** In this section we present Figure 1, which summarizes the technical contributions of this paper. We call this figure *the diamond*. Though not all the material necessary to understand this figure has been provided yet, we give it here as a reference and guide to what follows.
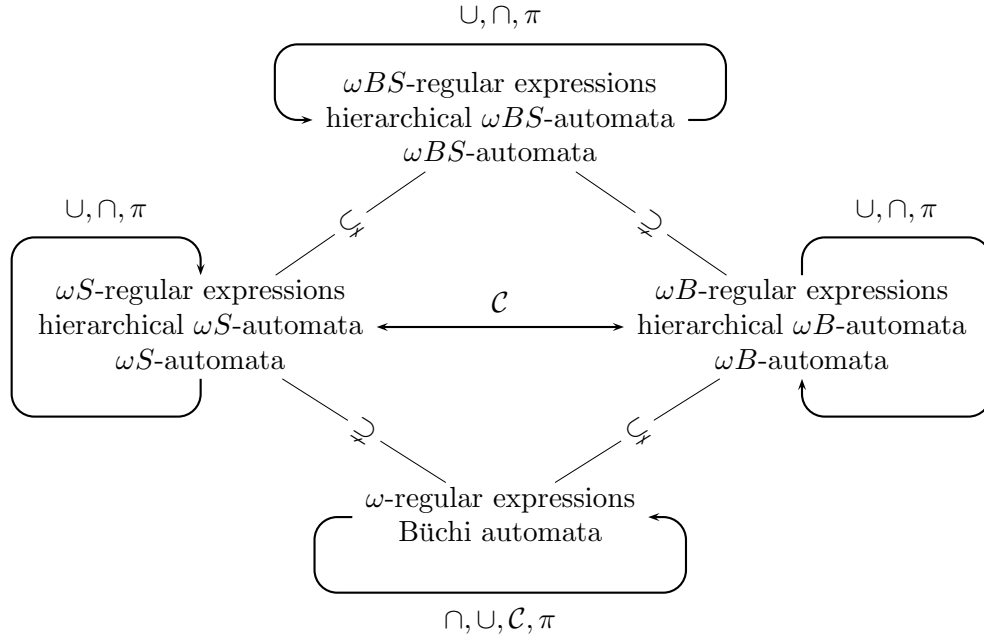

Figure 1: The diamond

The diamond illustrates the four variants of languages of ω-words that we consider: ω-regular, ωB-regular, ωS-regular and ωBS-regular languages. The inclusions between the

four classes give a diamond shape. We show in Section 2.3 below that the inclusions in the diamond are indeed strict.

To each class of languages corresponds a family of automata. The automata come in two variants: "normal automata", and the equally expressive "hierarchical automata". The exact definition of these automata as well as the corresponding equivalences are the subject of Section 3.

All the classes are closed under union, since ωBS-regular expressions have finite union built into the syntax. It is also easy to show that the classes are closed under projection, i.e., images under a letter to letter morphism (the operation denoted by π in the figure), and more generally images under a homomorphism. From the automata characterizations we obtain closure under intersection for the four classes; see Corollary 3.4. For closure under complement, things are not so easy. Indeed, in Section 2.3 we show that ωBS-regular languages are not closed under complement. However, some complementation results are still possible. Namely Theorem 5.1 establishes that complementing an ωB-regular language gives an ωS-regular language, and vice versa. This is the most involved result of the paper, and Section 5 is dedicated to its proof.

2.3. **Limits of the diamond.** In this section we show that all the inclusions depicted in the diamond are strict. Moreover, we show that there exists an ωBS-regular language whose complement is not ωBS-regular.

Lemma 2.9. *Every ωB-regular language over the alphabet $\{a, b\}$, which contains a word with an infinite number of b's, also contains a word in $(a^B b)^\omega$.*

Proof. Using a straightforward structural induction, one can show that a B-regular language of word sequences L satisfies the following properties.
- If L contains a sequence in a^*, then it contains a sequence in a^B.
- If L contains a sequence in $(a^* b)^+ a^*$, then it contains a sequence in $(a^B b)^+ a^B$.

The statement of the lemma follows. ☐

Corollary 2.10. *The language $(a^S b)^\omega$ is not ωB-regular. The language $(a^B b)^\omega$ is not ωS-regular.*

Proof. Being ωB-regular for $(a^S b)^\omega$ would contradict Lemma 2.9, since it contains a word with an infinite number of b's but does not intersect $(a^B b)^\omega$.

For the second part, assume that the language $(a^B b)^\omega$ is ωS-regular. Consequently, so is the language $(a^B b)^\omega + (a + b)^* a^\omega$. Using Theorem 5.1, its complement $((a^* b)^* a^S b)^\omega$ would be ωB-regular. But this is not possible, by the same argument as above. A proof that does not use complementation—along the same lines as in the first part—can also be given. ☐

We now proceed to show that ωBS-regular languages are not closed under complement. We start with a lemma similar to Lemma 2.9.

Lemma 2.11. *Every ωBS-regular language over the alphabet $\{a, b\}$ that contains a word with an infinite number of b's also contains a word in $(a^B b + a^S b)^\omega$.*

Proof. As in Lemma 2.9, one can show the following properties of a BS-regular language of word sequences L.
- If L contains a word sequence in a^*, then it contains one in $a^B + a^S$.
- If L contains a word sequence in $(a^* b)^+ a^*$, then it contains one in $(a^B b + a^S b)^+ (a^B + a^S)$.

The result directly follows. ☐

Corollary 2.12. *The complement of $L = (a^B b + a^S b)^\omega$ is not ωBS-regular.*

Proof. The complement of L contains the word

$$a^1 \ ba^1ba^2 \ ba^1ba^2ba^3 \ ba^1ba^2ba^3ba^4b \ \ldots \ ,$$

and consequently, assuming it is ωBS-regular, one can apply Lemma 2.11 to it. It follows that the complement of L should intersect L, a contradiction. ☐

3. Automata

In this section we introduce a new automaton model for infinite words, which we call ωBS-automata. We show that these automata have the same expressive power as ωBS-regular expressions.

   We will actually define two models: ωBS-automata, and hierarchical ωBS-automata. We present them, successively, in Sections 3.1 and 3.2. Theorem 3.3, which shows that both models have the same expressive power as ωBS-regular expressions, is given in Section 3.3. Section 3.3 also contains Corollary 3.4, which states that ωBS-regular languages are closed under intersection. The proof of Theorem 3.3 is presented in Sections 3.5 and 3.6.

3.1. **General form of ωBS-automata.** An ωBS-*automaton* is a tuple $(Q, \Sigma, q_I, \Gamma_B, \Gamma_S, \delta)$, in which Q is a finite set of *states*, Σ is the input alphabet, $q_I \in Q$ is the *initial state*, and Γ_B and Γ_S are two disjoint finite sets of *counters*. We set $\Gamma = \Gamma_B \cup \Gamma_S$. The mapping δ associates with each letter $a \in \Sigma$ its *transition relation*

$$\delta_a \subseteq Q \times \{\mathrm{i}, \mathrm{r}, \epsilon\}^\Gamma \times Q \ .$$

The counters in Γ_B are called *bounding counters*, or *B-counters* or *counters of type B*. The counters in Γ_S are called *unbounding counters*, or *S-counters* or *counters of type S*. Given a counter α and a transition (p, v, q), the transition is called a *reset* of α if $v(\alpha) = \mathrm{r}$; it is an *increment of α* if $v(\alpha) = \mathrm{i}$.

   When the automaton only has counters of type B, i.e., if $\Gamma_S = \emptyset$ (respectively, of type S, i.e., if $\Gamma_B = \emptyset$), then the automaton is called an ωB-*automaton* (respectively, an ωS-*automaton*).

   A *run* ρ of an ωBS-automaton over a finite or infinite word $a_1 a_2 \cdots$ is a sequence of transitions $\rho = t_1 t_2 \cdots$ of same length such that for all i, t_i belongs to δ_{a_i}, and the target state of the transition t_i is the same as the source state of the transition t_{i+1}.

   Given a counter α, every run ρ can be uniquely decomposed as

$$\rho = \rho_1 t_{n_1} \rho_2 t_{n_2} \ldots$$

in which, for every i, t_{n_i} is a transition that does a reset on α, and ρ_i is a subrun (sequence of transitions) that does no reset on α. We denote by $\alpha(\rho)$ the sequence of natural numbers, which on its i-th position has the number of occurrences of increments of α in ρ_i. This sequence can be finite if the counter is reset only a finite number of times, otherwise it is infinite. A *run* ρ over an ω-word is *accepting* if the source of its first transition is the initial state q_I, and for every counter α, the sequence $\alpha(\rho)$ is infinite and furthermore, if α is of type B then $\alpha(\rho)$ is bounded and if α is of type S then $\alpha(\rho)$ is strongly unbounded.

Example 3.1. Consider the following automaton with a single counter (the counter action is in the parentheses):

$$a(i), b(r) \qquad a(\epsilon), b(\epsilon)$$

$$b(\epsilon)$$

$$q \qquad p$$

$$b(r)$$
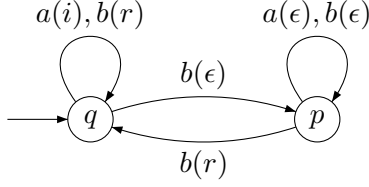
Assume now that the unique counter in this automaton is of type B. We claim that this automaton recognizes the language $L = (a^B b(a^*b)^*)^\omega$, i.e., the set of ω-words of the form $a^{n_0} b a^{n_1} b \ldots$ such that $\liminf_i n_i < +\infty$. We only show that the automaton accepts all words in L, the converse inclusion is shown by using similar arguments. Let $w = a^{n_0} b a^{n_1} b \ldots$ be a word in L. There exists an infinite set $K \subseteq \mathbb{N}$ and a natural number N such that $n_k < N$ holds for all $k \in K$. Without loss of generality we assume that $0 \in K$ (by replacing N by $\max(n_0, N)$). We now construct an accepting run of the automaton on the word w. This run uses state q in each position $i \in \mathbb{N}$ such that the distance between the last occurrence of b before position i and the first occurrence of b after position i is at most N. For other positions the state p is used and the value of the counter is left unchanged. In this way, the value of the counter will never exceed N. Since furthermore K is infinite, the counter will be reset infinitely many times. This proves that the run is accepting. If the counter is of type S, then the automaton recognizes the language $(a^S b(a^*b)^*)^\omega$.

Although this is not directly included in the definition, an ωBS-automaton can simulate a Büchi automaton, and this in two ways: by using either unbounding, or bounding counters (and therefore Büchi automata are captured by both ωS-automata and ωB-automata). Consider then a Büchi automaton, with final states F. One way to simulate this automaton is to have an ωB-automaton with the same state space, and one bounding counter, which is reset every time a state from F is visited, and never incremented. In this case, the accepting condition for ωBS-automata collapses to visiting F infinitely often, since a bounded value is assured by the assumption on not doing any increments. Another way to simulate the Büchi automaton is to use an unbounding counter. Whenever the simulating automaton visits a state in F, it nondeterministically decides to either increment the unbounding counter, or to reset it. It is easy to see that the accepting state is seen infinitely often if and only if there is a policy of incrementing and resetting that satisfies the accepting condition for unbounding counters.

3.2. **Hierarchical automata.** Hierarchical ωBS-automata are a special case of ωBS-automata where a stack-like discipline is imposed on the counter operations.

An ωBS-automaton is called *hierarchical* if its set of counters is $\Gamma = \{1, \ldots, n\}$ and whenever a counter $i > 1$ is incremented or reset, the counters $1, \ldots, i - 1$ are reset. Therefore, in a hierarchical automaton, a transition (q, v, r) can be of three forms:

- $v(1) = \cdots = v(n) = \epsilon$, i.e., no counter is affected by the transition. In this case we write ϵ for v.
- $v(1) = \cdots = v(k) = r$ and $v(k+1) = \cdots = v(n) = \epsilon$, i.e., the maximal affected counter is k, and it is reset by the transition. In this case we write R_k for v.
- $v(1) = \cdots = v(k-1) = r$, $v(k) = i$ and $v(k+1) = \cdots = v(n) = \epsilon$., i.e., the maximal affected counter is k, and it is incremented by the transition. In this case we write I_k for v.

It is convenient to define for a hierarchical automaton its *counter type*, defined as a word in $\{B + S\}^*$. The length of this word is the number of counters; its i-th letter is the type of counter i.

Example 3.2. Consider the following hierarchical ωBS-automaton:

$$a(I_1), b(R_1), c(I_2), d(R_2)$$
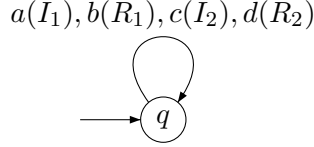
In this picture, we use once more the convention that the resets and increments are in parentheses. If this automaton has counter type $T_1 T_2$ with $T_1, T_2 \in \{B, S\}$, then it recognizes the language

$$\left(\left((a^{T_1}b)^* a^{T_1} c \right)^{T_2} (a^{T_1}b)^* a^{T_1} d \right)^\omega \ .$$

3.3. **Equivalence.** The key result concerning the automata is that the hierarchical ones are equivalent to the non-hierarchical ones, and that both are equivalent to the ωBS-regular expressions. Furthermore, this equivalence also holds for the fragments where only B-counters or S-counters are allowed.

Theorem 3.3. *Let $T \in \{BS, B, S\}$. The following are equivalent for a language $L \subseteq \Sigma^\omega$.*
(1) *L is ωT-regular.*
(2) *L is recognized by a hierarchical ωT-automaton.*
(3) *L is recognized by an ωT-automaton.*

Before establishing this result, we mention an important application.

Corollary 3.4. *The classes of ωBS-regular, ωB-regular and ωS-regular languages are closed under intersection.*

Proof. The corresponding automata (in their non-hierarchical form) are closed under intersection using the standard product construction. □

The implication from (1) to (2) is straightforward since hierarchical automata are a special case of general automata. In the following sections, we show the two difficult implications in Theorem 3.3: that expressions are captured by hierarchical automata (Section 3.5) and that automata are captured by expressions (Section 3.6). First, we introduce in Section 3.4 the notion of word sequence automata.

3.4. **Word sequence automata.** In proving the equivalence of hierarchical ωBS-automata and ωBS-regular languages, we will use a form of automaton that runs over word sequences. A *word sequence BS-automaton* $\mathcal{A}$ is defined as an ωBS-automaton, except that we add a set of accepting states, i.e., it is a tuple $(Q, \Sigma, q_I, F, \Gamma_B, \Gamma_S, \delta)$ in which $Q, \Sigma, q_I, \Gamma_B, \Gamma_S, \delta$ are as for ωBS-automata, and $F \subseteq Q$ is a set of *accepting states*.

A word sequence BS-automaton $\mathcal{A}$ *accepts* an infinite sequence of finite words $\vec{u}$ if there is a sequence $\vec{\rho}$ of finite runs of $\mathcal{A}$ such that:

• for all $i \in \mathbb{N}$, the run ρ_i is a run over the word u_i that begins in the initial state and ends in an accepting state;

- for every counter α of type B, the sequence of natural numbers

$$\max(\alpha(\rho_0)), \max(\alpha(\rho_1)), \ldots$$

  (in which max is applied to a finite sequence of natural numbers with the obvious meaning) is bounded;
- for every counter α of type S, the sequence of natural numbers

$$\min(\alpha(\rho_0)), \min(\alpha(\rho_1)), \ldots$$

  (in which min is applied to a sequence of natural numbers with the obvious meaning) is strongly unbounded.

The variants of *word sequence B-automata* and *word sequence S-automata* are defined as expected. The same goes for the hierarchical automata.

An equivalent way for describing the acceptance of a word sequence by a BS-automaton $\mathcal{A}$ is as follows. Consider the ωBS-automaton $\mathcal{A}'$ obtained from $\mathcal{A}$ by a) removing the set of accepting states F, b) adding a new symbol $\square$ to the alphabet, and c) setting $\delta_\square$ to contain the transitions (q, R, q_I) for $q \in F$ with $R(\alpha) = \mathrm{r}$ for every counter α. Then $\mathcal{A}$ accepts the word sequence $\langle v_0, v_1, \ldots \rangle$ iff $\mathcal{A}'$ accepts the ω-word $v_0 \square v_1 \square \ldots$

3.5. **From expressions to hierarchical automata.** This section is devoted to showing one of the implications in Theorem 3.3:

Lemma 3.5. *Every ωBS-regular (respectively, ωB-regular, ωS-regular) language can be recognized by a hierarchical ωBS-automaton (respectively, ωB-regular, ωS-regular).*

There are two main difficulties:
- Our word sequence automata do not have ε-transitions, which are used in the proof for finite words. Instead of introducing a notion of ε-transition and establishing that such transitions can be eliminated from automata, we directly work on word sequence automata without ε-transitions.
- When taking the mix or the concatenation of two languages L, K defined by hierarchical word sequence automata, there are technical difficulties with combining the counter types of the automata for L and K.

We overcome these difficulties by first rewriting an expression into normal form before compiling it into a hierarchical word sequence automaton. The basic idea is that we move the mix $+$ to the top level of the expression, and also remove empty words and empty iterations. To enforce that no empty words occur, we use the exponents L^+, L^{S+} and L^{B+} which correspond to $L \cdot L^*$, $L \cdot L^S$ and $L \cdot L^B$, respectively.

We say that a BS-regular expression is *pure* if it is constructed solely with the single letter constants a and $\bar{a}$, concatenation $\cdot$, and the exponents $+$, $B+$ and $S+$. We say a BS-regular expression is in *normal form* if it is a mix $e_1 + \cdots + e_n$ of pure BS-regular expressions $e_1, \ldots, e_n$. An ωBS-regular expression is in *normal form* if all the BS-regular expressions in it are in normal form.

In Section 3.5.1, we show that every BS-regular language (with no occurrence of ε) can be described by a BS-regular expression in normal form. Then, in Section 3.5.2, we show that every ωBS-regular expression in normal form can be compiled into a hierarchical word sequence automaton.

3.5.1. *Converting an expression into normal form.* Given a BS-regular language L, let us define $clean(L)$ to be the set of word sequences in L that have nonempty words on all coordinates. Remark that, thanks to Fact 2.3, L^ω is the same as $(clean(L))^\omega$. Therefore, we only need to work on sequence languages of the form $clean(L)$. Using Fact 2.3 one obtains without difficulty:

Fact 3.6. For every BS-regular language L, either $L = clean(L)$, $L = \bar\varepsilon + clean(L)$, or $L = \varepsilon + clean(L)$.

The following lemma concludes the conversion into normal form:

Lemma 3.7. *For every BS-regular language L, $clean(L)$ can be rewritten as $L_1 + \cdots + L_n$, where each L_i is pure.*

The proof of this lemma has a similar flavor as the proof of the analogous result for finite words, which says that union can be shifted to the topmost level in a regular expression.

Proof. The proof is by induction on the structure.
- For $L = \emptyset, \varepsilon, \bar\varepsilon, a$, the claim is obvious.
- Case $L = K \cdot M$. There are nine subcases (according to Fact 3.6):
  - If K is $clean(K)$ and M is $clean(M)$ then
  $$clean(K \cdot M) = clean(K) \cdot clean(M) .$$

  We then use the induction assumption on $clean(K)$ and $clean(M)$, and concatenate the two unions of pure expressions. This concatenation is also a union of pure expressions, since
  $$\sum_i K_i \cdot \sum_j M_j = \sum_{i,j} K_i \cdot M_j .$$
  - If K is $clean(K)$ and M is $\bar\varepsilon + clean(M)$ then
  $$clean(K \cdot M) = \overline{clean(K)} + clean(K) \cdot clean(M) .$$

  In the above, the language $\overline{clean(K)}$ consists of finite prefixes of sequences in $clean(K)$. A pure expression for this language is obtained from $clean(K)$ by replacing every exponent with $+$ and every letter a with $\bar a$.
  - If K is $clean(K)$ and M is $\varepsilon + clean(M)$ then
  $$clean(K \cdot M) = clean(K) + clean(K) \cdot clean(M) .$$

  - The six other cases are similar.
- Case $L = K + M$. We have $clean(K + M) = clean(K) + clean(M)$.
- Case $L = K^*$. We have $clean(K^*) = (clean(K))^+$. By induction hypothesis, this becomes $(L_1 + \cdots + L_n)^+$, for pure expressions L_i. We need to show how the mix $+$ can be moved to the top level. For $n = 2$, we use:
$$
\begin{aligned}
(L_1 + L_2)^+ =& L_1^+ + (L_2^+ \cdot L_1^+)^+ + (L_2^+ \cdot L_1^+)^+ \cdot L_2^+ + \\
& + (L_1^+ \cdot L_2^+)^+ + (L_1^+ \cdot L_2^+)^+ \cdot L_1^+ + L_2^+ .
\end{aligned}
$$
The general case is obtained by an inductive use of this equivalence.

- Case $L = K^S$. This time, we use $clean(K^S) = (clean(K))^{S+}$, and get by induction an expression of the form $(L_1 + \cdots + L_n)^{S+}$, for pure expressions L_i. We only do the case of $n = 2$, the general case is obtained by induction on n:

$$(L_1 + L_2)^{S+} = (L_1 + L_2)^* \cdot L_1^{S+} \cdot (L_1 + L_2)^*$$
$$+ (L_1 + L_2)^* \cdot L_2^{S+} \cdot (L_1 + L_2)^*$$
$$+ L_2^* \cdot (L_1^+ \cdot L_2^+)^{S+} \cdot L_1^* \ .$$

The right side of the equation is not yet in the correct form, i.e., it is not a mix of pure expressions, but it can be made so using the mix, the concatenation and the $*$ exponent cases described above (resulting in a mix of 102 pure expressions).
- Case $L = K^B$. Same as for case K^*, in which the exponent $*$ is replaced by B and the exponent $+$ is replaced by $B+$. $\qquad\square$

3.5.2. *From normal form to automata.* Here we show that every expression in normal form can be compiled into a hierarchical automaton. Furthermore, if the expression is ωB-regular (respectively, ωS-regular), then the automaton has the appropriate counter type.

We begin by showing how to expand counter types:

Lemma 3.8. *A hierarchical BS-automaton of counter type $tt' \in \{B, S\}^*$ can be transformed into an equivalent one of counter type tBt'. A hierarchical BS-automaton of counter type $tSt' \in \{B, S\}^*$ can be transformed into an equivalent one of counter type $tSSt'$.*

Proof. Let $\mathcal{A}$ be the automaton. For the first construction, we insert a new counter of type B at the correct position, i.e., between counter $|t|$ and $|t| + 1$, and reset it as often as possible, i.e., we construct a new automaton $\mathcal{A}'$ of counter type tBt' which is similar to $\mathcal{A}$ in all respects but every transition (p, v, q) of $\mathcal{A}$ becomes a transition (p, v', q) in $\mathcal{A}'$ with:

$$v' = \begin{cases} R_1 & \text{if } v = \epsilon \text{ and } |t| = 0 \\ \epsilon & \text{if } v = \epsilon \text{ and } |t| > 0 \\ I_k & \text{if } v = I_k \text{ and } k \leq |t| \\ R_k & \text{if } v = R_k \text{ and } k < |t| \\ R_{|t|+1} & \text{if } v = R_{|t|} \\ I_{k+1} & \text{if } v = I_k \text{ and } k > |t| \\ R_{k+1} & \text{if } v = R_k \text{ and } k > |t| \end{cases}$$

For every ω-word u, this translation gives a natural bijection between the runs of $\mathcal{A}$ over u and the runs of $\mathcal{A}'$ over u. This translation of runs preserves the accepting condition. Hence the language accepted by $\mathcal{A}$ and $\mathcal{A}'$ are the same.

For the second construction, we split the S counter into two nested copies. The automaton chooses nondeterministically which one to increment. Formally, we transform every transition (p, v, q) of $\mathcal{A}$ into possibly multiple transitions in $\mathcal{A}'$, namely the transitions

(p, v', q) with $v' \in V$ in which:

$$V = \begin{cases} \{\epsilon\} & \text{if } v = \epsilon \\ \{I_k\} & \text{if } v = I_k \text{ and } k \leq |t| \\ \{R_k\} & \text{if } v = R_k \text{ and } k \leq |t| \\ \{I_{|t|+1}, I_{|t|+2}\} & \text{if } v = I_{|t|+1} \\ \{R_{|t|+2}\} & \text{if } v = R_{|t|+1} \\ \{I_{k+1}\} & \text{if } v = I_k \text{ and } k > |t| + 1 \\ \{R_{k+1}\} & \text{if } v = R_k \text{ and } k > |t| + 1 \end{cases}$$

For every input ω-word u, this transformation induces a natural surjective mapping from runs of $\mathcal{A}'$ onto runs of $\mathcal{A}$. Accepting runs are mapped by this translation to accepting runs of $\mathcal{A}$. Hence the language accepted by $\mathcal{A}'$ is a subset of the one accepted by $\mathcal{A}$. For the converse inclusion, one needs to transform an accepting run ρ of $\mathcal{A}$ over u into an accepting run of $\mathcal{A}'$ over u. For this one needs to decide each time a transition $(p, I_{|t|+1}, q)$ is used by ρ, whether to use the transition $(p, I_{|t|+1}, q)$ or the transition $(p, I_{|t|+2}, q)$ of $\mathcal{A}'$. For this, for all maximal subruns of ρ of the form

$$\rho_0'(p_1, I_{|t|+1}, q_1)\rho_1' \cdots (p_n, I_{|t|+1}, q_n)\rho_n'$$

in which the counter $|t| + 1$ is never reset, and such that the counter $|t| + 1$ is never incremented in the runs ρ_i', one replaces it by the run of $\mathcal{A}'$:

$$\rho_0''(p_1, I_{x_1}, q_1)\rho_1'' \cdots (p_n, I_{x_n}, q_n)\rho_n''$$

where

$$x_i = \begin{cases} |t| + 2 & \text{if } i \text{ is a multiple of } \lceil \sqrt{n} \rceil \\ |t| + 1 & \text{otherwise,} \end{cases}$$

and ρ_i'' is ρ_i' in which each counter $k > |t| + 1$ is replaced by counter $k + 1$. This operation transforms an accepting run of $\mathcal{A}$ into an accepting run of $\mathcal{A}'$. □

We will use the following corollary of Lemma 3.8, which says that any number of hierarchical BS-automata can be transformed into equivalent ones that have comparable counter types.

Corollary 3.9. *Given hierarchical BS-automata $\mathcal{A}_1, \ldots, \mathcal{A}_n$, there exist (respectively) equivalent hierarchical BS-automata $\mathcal{A}_1', \ldots, \mathcal{A}_n'$, such that for all $i, j = 1 \ldots n$ the counter type of $\mathcal{A}_i'$ is a prefix of the counter type of $\mathcal{A}_j'$ or vice versa. Furthermore, if $\mathcal{A}_1, \ldots, \mathcal{A}_n$ are B-automata (respectively, S-automata), then so are $\mathcal{A}_1', \ldots, \mathcal{A}_n'$.*

Recall that we want to compile a normal form expression

$$M \cdot (L_1 + \cdots + L_n)^\omega$$

into a hierarchical ωBS-automaton (or ωB, or ωS automaton, as the case may be). This is done in the next two lemmas. First, Lemma 3.10 translates each L_i into a hierarchical word sequence automaton, and then Lemma 3.11 combines these word sequence automata into an automaton for the infinite words $(L_1 + \cdots + L_n)^\omega$. Since prefixing the language M is a trivial operation for the automata, we thus obtain the desired Lemma 3.5, which says that expressions can be compiled into hierarchical automata.

Lemma 3.10. *The language of word sequences described by a pure BS-regular (respectively, B-regular, respectively, S-regular) expression can be recognized by a hierarchical word sequence BS-automaton (respectively, B-automaton, respectively, S-automaton).*

Proof. By induction on the operations that appear in a pure expression.

- Languages accepted by hierarchical word sequence automata are closed under concatenation. Let us compute an automaton recognizing $L \cdot L'$ where L, L' are languages recognized by hierarchical word sequence automata $\mathcal{A}, \mathcal{A}'$ respectively. Using Corollary 3.9, we can assume without loss of generality that the type of $\mathcal{A}$ is a prefix of the type of $\mathcal{A}'$ (or the other way round, which is a symmetric case). Remark that since L and L' are pure, no state in $\mathcal{A}$ or $\mathcal{A}'$ is both initial and final.

  We do the standard concatenation construction for finite automata (by passing from a final state of $\mathcal{A}$ to the initial state of $\mathcal{A}'$), except that when passing from $\mathcal{A}$ to $\mathcal{A}'$ we reset the highest counter available to $\mathcal{A}$.

- Languages accepted by hierarchical word sequence automata are closed under the + exponent. We use the standard construction: linking all final states to the initial state while resetting all counters. In order to have non-empty words on all coordinates, the initial state cannot be accepting (if it is accepting, we add a new initial state).

- Languages accepted by hierarchical word sequence automata are closed under the $S+$ exponent. We add a new counter of type S of rank higher than all others. We then proceed to the construction for the + exponent as above, except that we increment the new counter whenever looping from a final state to the initial one.

- For the $B+$ exponent, we proceed as above, except that the new counter is of type B instead of being of type S. $\qquad\square$

The compilation of normal form expressions into automata is concluded by the following lemma:

Lemma 3.11. *Let $L_1, \ldots, L_n$ be sequence languages recognized by the hierarchical BS-automata $\mathcal{A}_1, \ldots, \mathcal{A}_n$ respectively. The language $(L_1 + \cdots + L_n)^\omega$ is recognized by an ωBS-automaton. Likewise for ωB-automata and ωS-automata.*

Proof. Thanks to Corollary 3.9, we can assume that the counter type of $\mathcal{A}_i$ is a prefix of the counter type of $\mathcal{A}_j$, for $i \leq j$. We use this prefix assumption to share the counters between the automata: we assume that the counters in $\mathcal{A}_i$ are a subset of the counters in $\mathcal{A}_j$, for $i \leq j$. Under this assumption, the hierarchical automaton for infinite words can nondeterministically guess a factorization of the infinite word in finite words, and nondeterministically choose one of the automata $\mathcal{A}_1, \ldots, \mathcal{A}_n$ for each factor. $\qquad\square$

3.6. **From automata to expressions.** This section is devoted to showing the remaining implication in Theorem 3.3:

Lemma 3.12. *Every language recognized by an ωBS-automaton (respectively, an ωB-automaton, an ωS-automaton) can be defined using an ωBS-regular (respectively, ωB-regular, ωS-regular) expression.*

Before we continue, we would like to remark that although long, the proof does not require any substantially original ideas. Basically, it consists of observations of the type: "if a word contains many a's and a b, then it either has many a's before the b or many a's

after the b". Using such observations and Kleene's theorem for finite words, we obtain the desired result.

We begin by introducing a technical tool called external constraints. These constraints are then used in Section 3.6.2 to prove Lemma 3.6.

3.6.1. *External constraints.* External constraints provide a convenient way of constructing ωBS-regular languages. Let e be one of $0, +, S, B$. Given a symbol $a \notin \Sigma$ and a word sequence language L over $\Sigma \cup \{a\}$, we denote by $L[a : e]$ the word sequence language

$$L \cap ((\Sigma^* \cdot a)^e \cdot \Sigma^*)$$

with the standard convention that $L^0 = [\![\varepsilon]\!]$. This corresponds to restricting the word sequences in L to ones where the number of occurrences of a satisfies the constraint e.

Here we show that external constraints can be eliminated:

Lemma 3.13. *BS-regular languages of word sequences are closed under external constraints. In other words, if L is a BS-regular language over Σ and a is a letter in Σ, then $L[a : e]$ is a BS-regular language over Σ. B-regular languages are closed under external constraints of type $0, +, B$. S-regular languages are closed under external constraints of type $0, +, S$.*

Proof. Structural induction. The necessary steps are shown in Figure 2. For some of the equivalences, we use closure properties of Fact 2.3. □

3.6.2. *Controlling counters in BS-regular expressions.* In this section we show how BS-regular languages can be intersected with languages of specific forms. We use this to write an ωBS-regular expression that describes successful runs of an ωBS-regular automaton, thus completing the proof of Lemma 3.12.

In the following lemma, the languages L should be thought of as describing runs of a BS-automaton. The idea is that the language K in the lemma constrains runs that are good from the point of view of one of the counters. The set of labels I can be thought of as representing the transitions which increment the counter, R as representing the transitions which reset the counter, and A as representing the transitions which do not modify it. The intersection in the lemma forces the counter operations to be consistent with the acceptance condition.

Given a word sequence language L and a subset A of the alphabet, denote by $L \bowtie A^*$ the set of word sequences that give a word sequence in L if all letters from A are erased (i.e., a very restricted form of the shuffle operator). For instance $B^* \bowtie A^*$ is the same as $(A + B)^*$.

Lemma 3.14. *Let Σ be an alphabet partitioned into sets A, I, R. Let L be a BS-regular word sequence language over Σ. Then $K \cap L$ is also BS-regular, for K being one of:*

$$(I^B R)^+ I^B \ \bowtie \ A^*, \quad (I^S R)^+ I^* \ \bowtie \ A^* \ , \quad I^*(RI^S)^+ \ \bowtie \ A^*, \quad I^S(RI^S)^+ \ \bowtie \ A^* \ .$$

Similarly, B-regular languages are closed under intersection with the first language, and S-regular languages are closed under intersection with the three other languages.

$$\varepsilon[a : +] = \emptyset$$
$$\varepsilon[a : e] = \varepsilon \qquad \text{for } e \in \{0, B\}$$
$$\varepsilon[a : S] = \bar{\varepsilon}$$
$$\bar{\varepsilon}[a : e] = \bar{\varepsilon} \qquad \text{for } e \in \{0, B, S\}$$
$$\bar{\varepsilon}[a : +] = \emptyset$$

$$b[a : +] = \emptyset \qquad \text{for } b \neq a$$
$$b[a : 0] = b \qquad \text{for } b \neq a$$
$$b[a : S] = \bar{b} \qquad \text{for } b \neq a$$
$$b[a : B] = b \qquad \text{for } b \neq a$$

$$a[a : 0] = \emptyset$$
$$a[a : S] = \bar{a}$$
$$a[a : e] = a \qquad \text{for } e \in \{B, +\}$$

$$(L + L')[a : e] = L[a : e] + L'[a : e] \qquad \text{for } e \in \{0, +, B, S\}$$

$$(L \cdot L')[a : e] = L[a : e] \cdot L'[a : e] \qquad \text{for } e \in \{B, 0\}$$
$$(L \cdot L')[a : e] = L[a : e] \cdot L' + L \cdot L'[a : e] \qquad \text{for } e \in \{+, S\}$$

$$L^*[a : 0] = (L[a : 0])^*$$
$$L^*[a : +] = L^* \cdot L[a : +] \cdot L^*$$
$$L^*[a : B] = (L[a : B] \cdot (L[a : 0])^*)^B$$
$$L^*[a : S] = L^* \cdot (L[a : +] \cdot L^*)^S + L^* \cdot L[a : S] \cdot L^*$$

$$L^B[a : 0] = (L[a : 0])^B$$
$$L^B[a : +] = L^B \cdot L[a : +] \cdot L^B$$
$$L^B[a : B] = (L[a : B])^B$$
$$L^B[a : S] = \bar{\varepsilon} + L^B \cdot L[a : S] \cdot L^B$$

$$L^S[a : 0] = (L[a : 0])^S$$
$$L^S[a : +] = L^S \cdot L[a : +] \cdot L^* + L^* \cdot L[a : +] \cdot L^S$$
$$L^S[a : B] = (L[a : B] \cdot (L[a : 0])^*)^B \cdot (L[a : 0])^S \cdot (L[a : B] \cdot (L[a : 0])^*)^B$$
$$L^S[a : S] = L^S \cdot L[a : S] \cdot L^* + L^* \cdot L[a : S] \cdot L^S + (L^* \cdot L[a : +])^S \cdot L^*$$

Figure 2: Elimination of external constraints.

Proof. In the proof we rewrite $K \cap L$ into an expression with external constraints, which will use and constrain letters from some new alphabet Σ' disjoint with Σ. In the proof, we will consider equality up to the removal of letters from Σ'. We then eliminate the external constraints using Lemma 3.13, and then erase the letters from Σ' (erasing letters is allowed, since languages described by our expressions are closed under homomorphic images).

We begin by showing that $(I^e \bowtie A^*) \cap L$ is BS-regular, for $e = 0, +, B, S$. Let $a \in \Sigma'$ be a new symbol. The transformation is simple: replace everywhere in the expression the letters i in I by $i \cdot a$, and constrain the resulting language by $[a : e]$.

For $K = (I^B R)^+ I^B \bowtie A^*$, the construction is by induction on the size of the expression defining L. We use the following equalities (in which $K' = I^B \bowtie A^*$):

$$K \cap b = b \quad \text{if } b \in R \text{ and } \emptyset \text{ otherwise}$$

$$K \cap (L + L') = (K \cap L) + (K \cap L')$$

$$K \cap (L \cdot L') = (K \cap L) \cdot ((K \cap L') + (K' \cap L')) + (K' \cap L) \cdot (K \cap L')$$

$$K \cap L^* = ((K' \cap L^*) \cdot (K \cap L))^+ \cdot (K' \cap L^*) \, .$$

The remaining cases, namely $K \cap L^B$ and $K \cap L^S$, can be reduced to the L^* case as follows. First one rewrites L^B (respectively, L^S) as $(La)^*[a : B]$ (respectively, $(La)^*[a : S]$), where $a \in \Sigma'$ is a new letter (recall that we consider here equality of languages up to removal of letters from Σ'). Second, we use the associativity of intersection, i.e.,

$$K \cap ((La)^*[a : e]) = (K \cap (La)^*)[a : e] \,, \qquad \text{for } \ e = B, S \,.$$

For the case where K is either $((I^S R)^+ I^*) \bowtie A^*$ or $(I^* (R I^S)^+) \bowtie A^*$, a slightly more tedious transformation is involved. This is also done by induction. To make the induction pass, we generalize the result to languages K of the form

$$K_{e,f} = I^e R (I^S R)^* I^f \bowtie A^* \,, \qquad \text{where } e, f \in \{*, S\} \,.$$

The transformations for $L = b$ and $L + L'$ are as follows:

$$K_{e,f} \cap b = \begin{cases} b & \text{if } b \in R \text{ and } e = f = *, \\ \emptyset & \text{otherwise,} \end{cases}$$

$$K_{e,f} \cap (L + L') = (K_{e,f} \cap L) + (K_{e,f} \cap L') \,.$$

For sequential composition $L \cdot L'$, we use the convenient operation $\sqcup$ over the exponents $\{*, S\}$, defined by

$$* \sqcup * = * , \qquad \text{and } e \sqcup f = S \text{ otherwise.}$$

The transformation for sequential composition $L \cdot L'$ is then the following:

$$
\begin{aligned}
K_{e,f} \cap (L \cdot L') = & \sum_{e' \sqcup f' = S} (K_{e,e'} \cap L) \cdot (K_{f',f} \cap L') \\
& + \sum_{e' \sqcup f' = e} (I^{e'} \cap L) \cdot (K_{f',f} \cap L') \\
& + \sum_{e' \sqcup f' = f} (K_{e,e'} \cap L) \cdot (I^{f'} \cap L')
\end{aligned}
$$

The rule for L^* is the most complex one. For conceptual simplicity we use an infinite sum. This can be transformed into a less readable but correct expression using standard methods for regular languages.

For $n \geq 1$, let L_n be the mix of all languages of the form

$$(I^{e_0} \cap L^*) \cdot (K_{f_1, g_1} \cap L) \cdot (I^{e_1} \cap L^*) \cdots (K_{f_n, g_n} \cap L) \cdot (I^{e_n} \cap L^*) \,,$$

where the exponents $e_i, f_i, g_i \in \{*, S\}$ satisfy

$$g_i \sqcup e_i \sqcup f_{i+1} = S \,, \qquad e_0 \sqcup f_1 = e \,, \qquad \text{and } g_n \sqcup e_n = f \,.$$

The language L_n corresponds to those word sequences, where the reset is done in n separate iterations of L. The language $K_{e,f} \cap L^*$ is then equal to the infinite mix $L_1 + L_2 + \cdots$ $\square$

We now use the above lemma to complete the proof of Lemma 3.12. Consider an ωBS-automaton $\mathcal{A}$. We will present an expression not for the recognized words, but the accepting runs. The result then follows by projecting each transition onto the letter it reads. Without loss of generality we assume that a transition uniquely determines this letter.

Given a counter α, let I_α represent the transitions that increment this counter, let R_α represent the transitions that reset it and let A_α be the remaining transitions. Let Γ_B be the set of bounded counters of $\mathcal{A}$ and let Γ_S be its unbounded counters. Given a state q, we define $Pref_q$ to be the language of finite partial runs starting in the initial state and ending in state q, and $Loop_q$ to be the language of nonempty finite partial runs starting and ending in state q. Those languages are regular languages of finite words, and hence can be described via regular expressions. We use those expressions as word sequence expressions.

The following lemma concludes the proof of Lemma 3.12, by showing how the operations from Lemma 3.14 can be used to check that a run is accepting.

Lemma 3.15. *A run ρ visiting infinitely often a state q is accepting iff there is a partition of Γ_S into two sets $\Gamma_{S,*}$ and $\Gamma_{*,S}$ (either of which may be empty) such that*

$$\rho \in Pref_q \cdot (Loop_q \cap L_B \cap L_{S,*} \cap L_{*,S})^\omega \ ,$$

where the languages L_B, $L_{S,*}$ and $L_{*,S}$ are defined as follows:

$$L_B = \bigcap_{\alpha \in \Gamma_B} ((I_\alpha^B R_\alpha)^+ I_\alpha^B) \bowtie A_\alpha^* \ ,$$

$$L_{e,f} = \bigcap_{\alpha \in \Gamma_{e,f}} (I_\alpha^e R_\alpha (I_\alpha^S R_\alpha)^* I_\alpha^f) \bowtie A_\alpha^* \ , \qquad for \ (e,f) = (*,S), (S,*) \ .$$

Proof. It is not difficult to show that membership in $Pref_q(Loop_q \cap L_B \cap L_{S,*} \cap L_{*,S})^\omega$ is sufficient for ρ to be accepting.

For the other direction, consider an accepting run ρ. Let us consider an increasing infinite sequence of positions $u_1, u_2, \ldots$ in the run ρ such that for each n, all counters are reset between u_n and u_{n+1} and the run assumes state q at position u_n. Such a sequence can be found since each counter is reset infinitely often. Consider now an unbounded counter α. For each n we define b_n^α to be the number of increments of α in ρ happening between u_n and the last reset before u_n, likewise, we define a_n^α to be the number of increments of α in ρ happening between u_n and the next reset of α after u_n. By extracting a subsequence, we may assume that either b_n^α is always greater than a_n^α, or a_n^α is always greater than b_n^α. In the first case b_n^α is strongly unbounded and we put α into $\Gamma_{*,S}$; in the second case a_n^α is strongly unbounded and we put α into $\Gamma_{*,S}$. We iterate this process for all unbounded counters. $\square$

4. MONADIC SECOND-ORDER LOGIC WITH BOUNDS

In this section, we introduce the logic MSOLB. This is a strict extension of monadic second-order logic (MSOL), where a new quantifier $\mathbb{U}$ is added. This quantifier expresses that a property is satisfied by arbitrarily large sets. We are interested in the satisfiability problem: given a formula of MSOLB, decide if it is satisfied in some ω-word. We are not able to solve this problem in its full generality. However, the diamond properties from the previous sections, together with the complementation result from Section 5, give an interesting partial solution to the satisfiability problem.

In Section 4.1 we introduce the logic MSOLB. In Section 4.2 we present some decidable fragments of the logic MSOLB, and restate the diamond picture in this new framework. In Section 4.3 we show how the unbounding quantifier can be captured by our automaton model. In Section 4.4 we present an application of our logical results; namely we provide an algorithm that decides if an ω-automatic graph has bounded degree.

4.1. **The logic.** Recall that monadic second-order logic (MSOL for short) is an extension of first-order logic where quantification over sets is allowed. Hence a formula of this logic is made of atomic predicates, boolean connectives ($\wedge, \vee, \neg$), first-order quantification ($\exists x.\varphi$ and $\forall x.\varphi$) and set quantification (also called monadic second-order quantification) ($\exists X.\varphi$ and $\forall X.\varphi$) together with the membership predicate $x \in X$. A formula of MSOL can be evaluated in an ω-word. In this case, the universe of the structure is the set $\mathbb{N}$ of word positions. The formula can also use the following atomic predicates: a binary predicate $x \leq y$ for order on positions, and for each letter a of the alphabet, a unary predicate $a(x)$ that tests if a position x has the label a. This way, a formula that uses the above predicates defines a language of ω-words: this is the set of those ω-words for which it is satisfied.

In the logic MSOLB we add a new quantifier, the *existential unbounding quantifier* $\mathbb{U}$, which can be defined as the following infinite conjunction:

$$\mathbb{U}X.\varphi := \bigwedge_{N \in \mathbb{N}} \exists X. \left(\varphi \wedge |X| \geq N \right) .$$

The quantified variable X is a set variable and $|X|$ denotes its cardinality. Informally speaking, $\mathbb{U}X.\varphi(X)$ says that the formula $\varphi(X)$ is true for sets X of arbitrarily large cardinality. If $\varphi(X)$ is true for some infinite set X, then $\mathbb{U}X.\varphi(X)$ is immediately true. Note that φ may contain other free variables than just X.

From this quantifier, we can construct other meaningful quantifiers:
- The *universal above quantifier* $\mathbb{A}$ is the dual of $\mathbb{U}$, i.e., $\mathbb{A}X.\varphi$ is a shortcut for $\neg\mathbb{U}X.\neg\varphi$. It is satisfied if all the sets X above some threshold of cardinality satisfy property φ.
- Finally, the *bounding quantifier* $\mathbb{B}$ is syntactically equivalent to the negation of the $\mathbb{U}$ quantifier. Historically, this was the first quantifier to be studied, in [1]. It says that a formula $\mathbb{B}X.\varphi$ holds if there is a bound on the cardinality of sets satisfying property φ.

Over finite structures, MSOLB and MSOL are equivalent: a subformula $\mathbb{U}X.\varphi$ can never be satisfied in a finite structure, and consequently can be removed from a formula. Over infinite words, MSOLB defines strictly more languages than MSOL. For instance the formula

$$\mathbb{B}X. [\forall x \in X. \ a(x)] \wedge [\forall x \leq y \leq z. \ (x, z \in X) \to (y \in X)]$$

expresses that there is a bound on the size of contiguous segments made of a's. Over the alphabet $\{a, b\}$, this corresponds to the language $(a^B b)^\omega$. As mentioned previously (recall Corollary 2.10), this language is not regular. Hence, this formula is not equivalent to any MSOL formula. This motivates the following decision problem:

> Is a given formula of MSOLB satisfied over some infinite word?

We do not know the answer to this question in its full generality (this problem may yet be proved to be undecidable). However, using the diamond (Figure 1), we can solve this question for a certain class of formulas. This is the subject of Sections 4.2 and 4.3. In Section 4.4, we use the logic MSOLB to decide if a graph has bounded outdegree, for graphs interpreted in the natural numbers via monadic formulas.

4.2. **A decidable fragment of MSOLB.** A classical approach for solving satisfiability of monadic second-order logic is to translate formulas into automata (this is the original approach of Büchi [3, 4] for finite and infinite words, which has been later extended by Rabin to infinite trees [12], see [14] for a survey). To every operation in the logic corresponds a language operation. As languages recognized by automata are effectively closed under those operations, and emptiness is decidable for automata, the satisfaction problem is decidable for MSOL. We use the same approach for MSOLB. Unfortunately, our automata are not closed under complement, hence we cannot use them to prove satisfiability for the whole logic, which is closed under complement.

$$\vee, \wedge, \exists, \mathbb{U}$$

ωBS-regular languages
BS-formulas

$$\vee, \wedge, \exists, \forall, \mathbb{U} \qquad\qquad \vee, \wedge, \exists, \forall, \mathbb{A}$$

ωS-regular languages
S-formulas
$\qquad \neg \qquad$
ωB-regular languages
B-formulas

ω-regular languages
MSOL-formulas

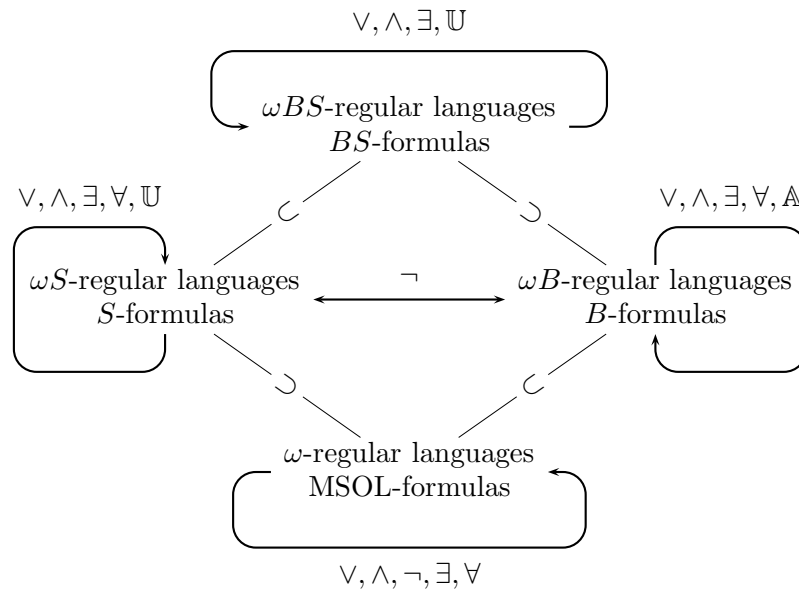$$\vee, \wedge, \neg, \exists, \forall$$

Figure 3: Logical view of the diamond

For this reason, we consider the following fragments of the logic MSOLB, which are not, in general, closed under complementation.

Definition 4.1. We distinguish the following syntactic subsets of MSOLB formulas:

- The *B-formulas* include all of MSOL and are closed under $\vee, \wedge, \forall, \exists$ and $\mathbb{A}$.
- The *S-formulas* include all of MSOL and are closed under $\vee, \wedge, \forall, \exists$ and $\mathbb{U}$.
- The *BS-formulas* include all B-formulas and S-formulas, and are closed under $\vee, \wedge, \exists$ and $\mathbb{U}$.

Note that in this definition, B-formulas and S-formulas are dual in the sense that the negation of an S-formula is logically equivalent to a B-formula, and vice versa. The above fragments are tightly connected to ωBS-regular languages according to the following fact:

Fact 4.2. BS-formulas define exactly the ωBS-regular languages. Likewise for B-formulas, and S-formulas, with the corresponding languages being ωB-regular and ωS-regular.

Proof. (Sketch.) Thanks to the closure properties of ωBS-regular languages, each BS-formula can be translated into an ωBS-regular language. We use here the standard coding of the valuation of free variables in the alphabet of the word, see, e.g. [14]. Closure under

$\vee$ and $\wedge$ is a direct consequence of closure under $\cup$ and $\cap$. Closure under $\exists$ corresponds to closure under projection, which is straightforward for non-deterministic automata. Closure under universal quantification follows as dual of the existential quantifications (using the complementation result, Theorem 5.1). Closure under $\mathbb{U}$ of ωS-regular and ωBS-regular languages is the subject of Section 4.3, and more precisely Proposition 4.4. Closure under $\mathbb{A}$ of ωB-regular languages is obtained by duality (once more using Theorem 5.1).

For the converse implication, a formula of the logic can use existential set quantifiers in order to guess a run of the automaton, and then check that this run is accepting using the new quantifiers. $\qquad\square$

These fragments are summarized in the logical view of the diamond presented in Figure 3. Also, from this fact we get that BS-formulas are not expressively complete for MSOLB, since MSOLB is closed under negation, while ωBS-regular languages are not closed under complementation.

Since emptiness for BS-regular languages is decidable by Fact 2.6, we obtain the following decidability result:

Theorem 4.3. *The problem of satisfiability over $(\omega, <)$ is decidable for BS-formulas.*

In the proof of Fact 4.2, we left out the proof of closure under $\mathbb{U}$. We present it in the next section.

4.3. Closure under existential unbounding quantification ($\mathbb{U}$).

Here we show that the classes of ωS- and ωBS-regular languages are closed under application of the quantifier $\mathbb{U}$. This closure is settled by Proposition 4.4.

In order to show this closure, we need to describe the quantifier $\mathbb{U}$ as a language operation, in the same way existential quantification corresponds to projection. Let Σ be an alphabet, and consider a language $L \subseteq (\Sigma \times \{0,1\})^\omega$. Given a word $w \in \Sigma^\omega$ and a set $X \subseteq \mathbb{N}$, let $w[X] \in (\Sigma \times \{0,1\})^\omega$ be the word obtained from w by setting the second coordinate to 1 at the positions from X and to 0 at the other positions. We define $\mathbb{U}(L)$ to be the set of those words $w \in \Sigma^\omega$ such that for every $N \in \mathbb{N}$ there is a set $X \subseteq \mathbb{N}$ of at least N elements such that $w[X]$ belongs to L.

Restated in terms of this operation, closure under unbounding quantification becomes:

Proposition 4.4. *Both ωS and ωBS-regular languages are closed under $\mathbb{U}$.*

We begin with a simple auxiliary result. A *partial sequence* over an alphabet Σ is a word in $\perp^* \Sigma^\omega$, in which $\perp \notin \Sigma$ is a fresh symbol. A partial sequence is *defined* at the positions where it does not have value $\perp$, it is *undefined* at positions of value $\perp$. We say that two partial sequences *meet* if there is some position where they are both defined and have the same value.

Lemma 4.5. *Let I be an infinite set of partial sequences over a finite alphabet Σ. There is a partial sequence in I that meets infinitely many partial sequences from I.*

Proof. A *constrainer* for I is an infinite word c over $P(\Sigma)$ such that for each $i \in \mathbb{N}$, the i-th position of every sequence in I is either undefined or belongs to c_i. The size of a constrainer is the maximal size of a set it uses infinitely often.

We prove the statement of the lemma by induction over the size of a constrainer for I. Since every I admits a constrainer of size $|\Sigma|$, this concludes the proof.

The base case is when I admits a constrainer of size 1; in this case, every two partial sequences in I meet, so any partial sequence in I satisfies the statement of the lemma. Consider now a set I with a constrainer c of size n. Take some sequence s in I. If s meets infinitely many sequences from I, then we are done. Otherwise let $J \subseteq I$ be the (infinite) set of sequences that do not meet s. One can verify that d is a constrainer for J, where d is defined by $d_i = c_i \setminus \{s_i\}$. Moreover, d is of size $n-1$ (since $s_i = \bot$ can hold only for finitely many i). We then apply the induction hypothesis. □

Let L be a language of infinite words over $\Sigma \times \{0,1\}$ recognized by an ωBS-automaton. We want to show that the language $\mathbb{U}(L)$ is also recognized by a bounding automaton. Consider the following language:

$$K = \{w[X] : \text{ for some } Y \supseteq X, w[Y] \in L\} .$$

This language is downward closed in the sense that if $w[X]$ belongs to K, then $w[Y]$ belongs to K for every $Y \subseteq X$. Furthermore, clearly $\mathbb{U}(L) = \mathbb{U}(K)$. Moreover, if L is recognized by an ωBS-automaton (resp. ωS-automaton), then so is K. Let then $\mathcal{A}$ be an ωBS-automaton recognizing K. We will construct an ωBS-automaton recognizing $\mathbb{U}(K)$.

Given a word $w \in \Sigma^\omega$, we say that a sequence of sets $X_1, X_2, \ldots \subseteq \mathbb{N}$ is an *an unbounding witness* for K if for every i, the word $w[X_i]$ belongs to K and the sizes of the sets X_i are unbounded. An unbounding witness is *sequential* if there is a sequence of numbers $a_1 < a_2 < \cdots$ such that for each i, all elements of the set X_i are between a_i and $a_{i+1} - 1$.

The following lemma is a consequence of K being downward closed.

Lemma 4.6. *A word that has an unbounding witness for K also has a sequential one.*

Let $X_1, X_2, \ldots$ be a sequential unbounding witness and let $a_1 < a_2 \cdots$ be the appropriate sequence of numbers. Let $\rho_1, \rho_2, \ldots$ be accepting runs of the automaton $\mathcal{A}$ over the words $w[X_1], w[X_2], \ldots$ Such runs exist by definition of the unbounding witness. A sequential witness $X_1, X_2, \ldots$ is called a *good witness* if every two runs ρ_i and ρ_j agree on almost all positions.

Lemma 4.7. *A word belongs to $\mathbb{U}(K)$ if and only if it admits a good witness.*

Proof. By Lemma 4.6, a word belongs to $\mathbb{U}(K)$, if and only if it admits a sequential witness. Let X_i, a_i and ρ_i be as above. For i, let s_i be the partial sequence that has $\bot$ at positions before a_{i+1} and agrees with ρ_i after a_{i+1}. By applying Lemma 4.5 to the set $\{s_1, s_2 \ldots\}$, we can find a run ρ_i and a set $J \subseteq \mathbb{N}$ such that for every $j \in J$, the runs ρ_i and ρ_j agree on some position x_j after a_{j+1}. For $j \in J$, let ρ'_j be a run that is defined as ρ_j at positions before x_j and is defined as ρ_i at positions after x_j. Since modifying the counter values over a finite set of positions does not violate the acceptance condition, the run ρ'_j is also an accepting run over the word $w[X_j]$. For every $j, k \in J$, the runs ρ'_j and ρ'_k agree on almost all positions (i.e., positions after both x_j and x_k). Therefore the sequential witness obtained by using only the sets X_j with $j \in J$ is a good witness. □

Lemma 4.8. *Words admitting a good witness can be recognized by a bounding automaton.*

Proof. Given a word w, the automaton is going to guess a sequential witness

$$a_1 < a_2 < \cdots \qquad X_1 \subseteq [a_1, a_2 - 1], X_2 \subseteq [a_2, a_3 - 1] \ldots$$

and a run ρ of $\mathcal{A}$ over w and verify the following properties:
- The run ρ is accepting;

- There is no bound on the size of the X_i's;
- For every i, some run over $w[X_i]$ agrees with ρ on almost all positions.

The first property can be obviously verified by an ωBS-automaton. For the second property, the automaton nondeterministically chooses a subsequence of $X_1, X_2, \ldots$ where the sizes are strongly unbounded. The third property is a regular property. The statement of the lemma then follows by closure of bounding automata under projection and intersection.   □

4.4. **Bounds on the out-degree of a graph interpreted on sets.** In this section, somewhat disconnected from the rest of the paper, we show how to use the logic MSOLB for solving a non-trivial question concerning ω-automatic structures. An ω-*automatic (directed) graph (of injective presentation)* is a graph described by two formulas of MSOL, which are interpreted in the natural numbers (hence the term ω-automatic). The vertexes of the ω-automatic graph are sets of natural numbers. The first formula $\delta(X)$ has one free set variable, and says which sets of natural numbers will be used as vertexes of the graph. The second formula $\varphi(X, Y)$ has two free set variables, and says which vertexes of the graph are connected by an edge (if $\varphi(X, Y)$ holds, then both $\delta(X)$ and $\delta(Y)$ must hold). The original idea of automaticity has been proposed by Hodgson [9] via an automata theoretic presentation. The more general approach that we use here of logically defining a structure in the powerset of another structure is developed in [6]. We show in this section that it is possible to decide, given the formulas $\delta(X)$ and $\varphi(X, Y)$ of MSOL defining an ω-automatic graph, whether this graph has bounded out-degree or not.

Let $\varphi(X, Y)$ be a formula of MSOLB with two free set variables. This formula can be seen as an edge relation on sets, i.e., it defines a directed graph with sets as vertexes. We show here that MSOLB can be used to say that this edge relation has unbounded out-degree. The formula presented below will work on any structure—not just $(\omega, <)$—but the decision procedure will be limited to infinite words, since it requires testing satisfiability of MSOLB formulas.

In the following, we say that a set Y is a *successor* of a set X if $\varphi(X, Y)$ holds. We begin by defining the notion of an X-witness. This is a set witnessing that the set X has many successors. (The actual successors of X form a set of sets, something MSOLB cannot talk about directly.) An X-*witness* is a set Y such that every two elements $x, y \in Y$ can be separated by a successor of X, that is:

$$\forall x, y \in Y.\ x \neq y \rightarrow\ \exists Z.\varphi(X, Z) \wedge (x \in Z \leftrightarrow y \notin Z) . \tag{4.1}$$

We claim that the graph of φ has unbounded out-degree if and only if there are X-witnesses of arbitrarily large cardinality. This claim follows from the following fact:

Fact 4.9. If X has more than 2^n successors, then it has an X-witness of size at least n. If X has n successors, then all X-witnesses have size at most 2^n.

Proof sketch. For the first statement, one first shows that X has at least n successors that are boolean independent, i.e., there exists $X_1, \ldots, X_n$ successors of X such that for all $i = 1 \ldots n$, X_i is not a boolean combination of $X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n$. From n boolean independent successors one can then construct by induction an X-witness of size n.

For the second statement, consider X with n successors as well as an X-witness. To each element w of the X-witness, associate the characteristic function of '$w \in Y$' for Y ranging over the successors of X. If the X-witness had more than 2^n elements, then at

least two would give the same characteristic function, contradicting the definition of an X-witness. $\qquad\square$

As witnessed by (4.1), being an X-witness can be defined by an MSOL formula. Therefore, the existence of arbitrarily large X-witnesses is expressible by an MSOLB formula with a single $\mathbb{U}$ operator at an outermost position. This formula belongs to one of the classes with decidable satisfiability in Theorem 4.3 below. This shows:

Proposition 4.10. *It is decidable if an ω-automatic graph has unbounded out-degree.*

5. Complementation

5.1. **The complementation result.** The main technical result of this paper is the following theorem:

Theorem 5.1. *The complement of an ωS-regular language is ωB-regular, while the complement of an ωB-regular language is ωS-regular.*

The proof of this result is long, and takes up the rest of this paper. We begin by describing some proof ideas. For the sake of this introduction, we only consider the case of recognizing the complement of an ωS-regular language by an ωB-regular automaton.

Consider first the simple case of a language described by an ωS-automaton $\mathcal{A}$ which has a single counter. Furthermore, assume that in every run, between any two resets, the increments form a single connected segment. In other words, between two resets of the counter, the counter is first left unaffected for some time, then during the n following transitions the counter is always incremented, then it is not incremented any more before reaching the second reset. Below, we use the name *increment interval* to describe an interval of word positions (a set of consecutive word positions) corresponding to a maximal sequence of increments. We do not, however, assume that the automaton is deterministic (the whole difficulty of the complementation result comes from the fact that we are dealing with nondeterministic automata). This means that the automaton resulting from the complementation construction must check that all possible runs of the complemented automaton are rejecting.

The complement ωB-automaton $\mathcal{B}$ uses a single B-counter, which ticks as a clock along the ω-word (independently of any run of $\mathcal{A}$). A tick of the clock is a reset of the counter. Between every two ticks, the counter is constantly incremented. Since the counter is a B-counter, the ticks have to be at bounded distance from each other. We say that an interval of word positions is *short* (with respect to this clock) if it contains at most one tick of the clock. If the clock ticks at most every N steps, then short intervals have length at most $2N - 1$. Reciprocally, if an interval has length at most N, then it is short with respect to every clock ticking with a tempo greater than N. According to these remarks, being short is a fair approximation of the length of an interval.

The complement automaton $\mathcal{B}$ works by guessing the ticks of a clock using non-determinism together with a B-counter, and then checks the following property: every run of $\mathcal{A}$ that contains an infinite number of resets, also contains an infinite number of short increment intervals. Once the clock is fixed, checking this is definable in monadic second-order logic, i.e., can be checked by a finite state automaton without bounding conditions. Using this remark it is simple to construct $\mathcal{B}$.

It is easy to see that if $\mathcal{B}$ accepts an ω-word, then this word is not accepted by $\mathcal{A}$. The converse implication is a consequence of the following compactness property: if no run of $\mathcal{A}$ is accepting, then there is a threshold $N \in \mathbb{N}$ such that every run of $\mathcal{A}$ either has less than N increments between two resets infinitely often, or does finitely many resets. Such a property can be established using Ramsey-like arguments.

Consider now a single counter ωS-automaton, but without the constraint of increments being performed during intervals. In our construction, we use an algebraic decomposition result, Simon's factorization forest theorem [13]. Using Simon's theorem, we reduce the complementation problem to a bounded number of instances of the above construction. In this case, the complement ωB-automaton uses one counter for each level of the factorization, the result being a structure of nested clocks. As above, once the ticks of the clocks are fixed, checking if a run makes few increments can be done by a finite state automaton without any bounding conditions.

Finally, for treating the general case of ωS-automata with more than one counter, we use automata in their hierarchical form and do an induction on the number of counters.

5.2. **Overview the complementation proof.** Theorem 5.1 talks about complementing two classes, and two proofs are necessary. The two proofs share a lot of similarities, and we will try to emphasize common points.

From now on, a *hierarchical automaton* $\mathcal{A}$ with states Q, input alphabet Σ and counters Γ is fixed. Either all the counters are of type S or all the counters are of type B. We will denote this type by $T \in \{B, S\}$, and by $\bar{T}$ we will denote the other type. We say $f \in \mathbb{N}^{\mathbb{N}}$ is a *B-function* if it is bounded, and an *S-function* if it is strongly unbounded. We set $\preceq$ to be $\leq$ if $T = S$ and $\geq$ if $T = B$. The idea is that greater means better. For instance, if $n \preceq m$, then replacing n increments by m increments leads to a run that is more likely to be accepting.

In this terminology, a run of an ωT-automaton is accepting if its counter values (at the moment before they are reset) are greater than some T-function for the order $\preceq$. When complementing an ωT-automaton our goal is to show that all runs are rejecting, i.e., for every run, either a counter is reset a finite number of times, or infinitely often the number of increments between two resets is smaller than a $\bar{T}$-function with respect to the order $\preceq$.

Our complementation proof follows the scheme introduced by Büchi in his seminal paper [3] (we also refer the reader to [15]). Büchi establishes that languages recognized by nondeterministic Büchi automata are closed under complement. In this proof Büchi did not determinize the automata as usual in the finite case (this is impossible for Büchi automata). Instead, he used a novel technique which allows to directly construct a nondeterministic automaton for the complement of a recognizable language. The key idea is that—thanks to Ramsey's theorem—each ω-word can be cut into a prefix followed by an infinite sequence of finite words, which are indistinguishable in any context by the automaton (we also say that those words have same type). Whether or not the ω-word is accepted by that automaton depends only on the type of the prefix and the type appearing in the infinite sequence. The automaton accepting the complement guesses this cut and checks that the two types correspond to a rejected word. For this reason the proof can be roughly decomposed into two parts. The first part shows that each word can be cut in the way specified. The second part shows that a cut can be guessed and verified by a Büchi automaton.

Our proof strategy is similar. That is why, in order to help the reader gather some intuition, we summarize below Büchi's complementation proof, in terms similar to our own proof for the bounding automata. Then, in Section 5.2.2, we outline our own proof.

5.2.1. *The Büchi proof.* In this section, we present a high-level overview of Büchi's complementation proof.

A *Büchi specification* describes properties of a finite run of a given Büchi automaton. It is a positive boolean combination of atomic specifications, which have three possible forms:

(1) The run begins with state p;
(2) The run ends with state q;
(3) The run contains/does not contain an accepting state.

We will use the letter τ to refer to specifications, be they Büchi, or the more general form introduced later on for bounding automata.

The following statement shows the Büchi strategy for complementation. The statement could be simplified for Büchi automata, but we choose the presentation below to stress the similarities with our own strategy, as stated in Proposition 5.3.

Proposition 5.2. *Let $\mathcal{B}$ be a Büchi automaton with input alphabet Σ. One can compute Büchi specifications $\tau_1, \ldots, \tau_n$ and regular languages $L_1, \ldots, L_n \subseteq \Sigma^*$ such that the following statements are equivalent for every infinite word $u \in \Sigma^\omega$:*

(A) The automaton $\mathcal{B}$ rejects the word u;
(B) There is some $i = 1, \ldots, n$ such that u admits a factorization $u = wv_1v_2\cdots$ where:
  (i) The prefix w belongs to L_i;
  (ii) For every j, every run ρ over the finite word v_j satisfies τ_i.

Moreover, for each $i = 1, \ldots, n$, the language $K_i \subseteq \Sigma^*$ of words where every run satisfies τ_i is regular.

In the above statement, the prefix w could also be described in terms of specifications, but we stay with just the regular languages L_i to accent the similarities with Proposition 5.3.

Proposition 5.2 immediately implies closure under complementation of Büchi automata, since the complement of the language recognized by $\mathcal{B}$ is the union

$$L_1 K_1^\omega + \cdots + L_n K_n^\omega \ ,$$

which is clearly recognizable by a Büchi automaton.

The "Moreover.." part of the proposition is straightforward, while the equivalence of conditions (A) and (B) requires an application of Ramsey's Theorem.

Our proof follows similar lines, although all parts require additional work. In particular, our specifications will be more complex, and will require more than just a regular language to be captured. The appropriate definitions are presented in the following section.

5.2.2. *Preliminaries.* Below we define the type of a finite run. Basically, the type corresponds to the information stored in a Büchi specification, along with some information on the counter operations. The type contains: 1) the source and target state of the run, like in a Büchi specification; and 2) some information on the counter operations that happen in

the run. Since we want to have a finite number of types, we can only keep limited information on the counter operations (in particular, we cannot keep track of the actual number of increments). Formally, a *type* t is an element of:

$$Q \times \{\emptyset, \{\text{-}\}, \{\text{-}\circ, \circ\text{-}\}, \{\text{-}\circ, \circ\circ, \circ\text{-}\}\}^{\Gamma} \times Q \ .$$

To a given run, we associate its type in the following way. The two states contain respectively the source (i.e. first) and target (i.e. last) state of the partial run. (A partial run is a finite run that does not necessarily begin at the beginning of the word, and does not necessarily end at the end of the word.) The middle component associates to each counter $\alpha \in \Gamma$ a *counter profile* denoted—by slight abuse—$t(\alpha)$. The counter profile is $t(\alpha) = \emptyset$ when there is no increment nor reset on counter α. The counter profile is $t(\alpha) = \{\text{-}\}$ when the counter is incremented but not reset. The counter profile is $t(\alpha) = \{\text{-}\circ, \circ\text{-}\}$ when the counter is reset just once, while $t(\alpha) = \{\text{-}\circ, \circ\circ, \circ\text{-}\}$ is used for the other cases, when the counter is reset at least twice.

Note that the counter profiles are themselves sets, and elements of these sets have a meaningful interpretation. Graphically, each symbol among $\text{-}, \text{-}\circ, \circ\text{-}, \circ\circ$ represents a possible kind of sequence of increments of a given counter. The circle $\circ$ symbolizes a reset starting or ending the sequence, while the dash - represents the sequence of increments itself. For instance, $\text{-}\circ$ identifies the segment that starts at the beginning of the run and end at the first occurrence of a reset of the counter. Given a type t, we use the name t-*event* for any pair

$$(\alpha, c) \in \Gamma \times \{\text{-}, \text{-}\circ, \circ\circ, \circ\text{-}\}, \qquad \text{with } c \in t(\alpha) \ .$$

The set of t-events is denoted by $events(t)$. Given a t-event (α, c) and a finite run ρ of type t, the value $val(\rho, \alpha, c)$ is the natural number defined below:

$val(\rho, \alpha, \text{-})$ the number of increments on counter α in the run.
$val(\rho, \alpha, \text{-}\circ)$ the number of increments on counter α before the first reset of counter α.
$val(\rho, \alpha, \circ\text{-})$ the number of increments on counter α after the last reset of counter α.
$val(\rho, \alpha, \circ\circ)$ the minimal (with respect to $\preceq$) number of increments on counter α between two consecutive resets on counter α.

We comment on the last value. When $T = S$, $val(\rho, \alpha, \circ\circ)$ is the smallest number of increments on counter α that is done between two successive resets of α. When $T = B$, this is the largest number of increments. At any rate, this is the worst number of increments, as far as the acceptance condition is concerned.

A *specification* is a property of finite runs. It is a positive boolean combination of the following two kinds of *atomic specifications*:

(1) The run has type t.
(2) The run satisfies $val(\rho, \alpha, c) \preceq K$.

The first atomic specification is defined by giving t. The second atomic specification is defined by giving α and c, but not K, which remains undefined. The number K is treated as a special parameter, or free variable, of the specification. This parameter is shared by all atomic specifications. Given a value of $K \in \mathbb{N}$, we define the notion that a run ρ *satisfies a specification τ under K* in the natural manner.

Unlike the Büchi proof, it is important that the boolean combination in the specification is positive. The reason is that we will use $\bar{T}$-automata to complement T-automata, and therefore we only talk about one type of behavior (bounded, or strongly unbounded).

5.2.3. *The decomposition result.* In this section we present the main decomposition result, which yields Theorem 5.1.

Proposition 5.3. *For every ωT-automaton $\mathcal{A}$ one can effectively obtain regular languages $L_1, \ldots, L_n$ and specifications $\tau_1, \ldots, \tau_n$ such that the following statements are equivalent for every ω-word u:*

(A) The automaton $\mathcal{A}$ rejects u;

(B) There is some $i = 1, \ldots, n$ such that u admits a factorization $u = wv_1v_2\cdots$ where:

   (i) The prefix w belongs to L_i, and;

   (ii) There is a $\bar{T}$-function f such that for every j, every run ρ over v_j satisfies τ_i under $f(j)$.

Moreover, for each $i = 1, \ldots, n$, one can verify with a word sequence $\bar{T}$-automaton $\mathcal{B}_i$ if a sequence of words $v_1, v_2, \ldots$ satisfies condition (ii). (Equivalently, the set of word sequences satisfying (ii) is $\bar{T}$-regular.)

First we note that the above proposition implies Theorem 5.1, since property (B) can be recognized by an $\omega\bar{T}$-automaton. This is thanks to the "Moreover..." and the closure of $\omega\bar{T}$-automata under finite union and the prefixing of a regular language.

The rest of this paper is devoted to showing the proposition. In Sections 5.3 and 5.4 we show the first part, i.e., the equivalence of (A) and (B). Section 5.3 develops extensions of Ramsey's theorem. Section 5.4 uses these results to show the equivalence. In Sections 5.5 and 5.6 we prove the "Moreover..." part. Section 5.5 contains the construction for a single counter, while Section 5.6 extends this construction to multiple counters. The difficulty in the "Moreover..." part is that (ii) talks about "every run ρ", and therefore the construction has to keep track of many simultaneous runs.

5.3. **Ramsey's theorem and extensions.** Ramsey-like statements (as we consider them in our context) are statements of the form *"there is an infinite set $D \subseteq \mathbb{N}$ and some index $i \in I$ such that the property $P_i(x, y)$ holds for any $x < y$ in D".* This statement is relative to a family of properties $\{P_i\}_{i\in I}$. In general, the family $\{P_i\}_{i\in I}$ may be infinite. The classical theorem of Ramsey, as stated below, follows this scheme, but for a family of two properties: $P_1 = R$ and $P_2 = \mathbb{N}^2 \setminus R$.

Theorem 5.4 (Ramsey)**.** *Given $R \subseteq \mathbb{N}^2$ and an infinite set $E \subseteq \mathbb{N}$, there is an infinite set $D \subseteq E$ such that;*

- *for all $x < y$ in D, $(x, y) \in R$, or;*
- *for all $x < y$ in D, $(x, y) \notin R$.*

In the original statement of Ramsey's theorem, the set E is not used (i.e. $E = \mathbb{N}$). We use the more general, but obviously equivalent, formulation to *compose* Ramsey-like statements as follows. Assume that there are two Ramsey-like statements, one using properties $\{P_i\}_{i\in I}$, and the other using $\{Q_j\}_{j\in J}$. We can apply the two in cascade and obtain a new statement of the form "there is an infinite set $D \subseteq \mathbb{N}$ and indexes $i \in I, j \in J$ such that both $P_i(x, y)$ and $Q_j(x, y)$ hold for any $x < y$ in D". This is again a Ramsey-like statement. This composition technique is heavily used below. We will simply refer to it as the *compositionality of Ramsey-like statements* and shortcut the corresponding part of the proofs.

The following lemma is our first Ramsey-like statement which uses an infinite (even uncountable) number of properties.

Lemma 5.5. *For any $h : \mathbb{N}^2 \to \mathbb{N}$ there is an infinite set $D \subseteq \mathbb{N}$ such that either*

- *there is a natural number M such that $h(x, y) \leq M$ holds for all $x < y \in D$, or;*
- *there is an S-function g such that $h(x, y) > g(x)$ holds for all $x < y \in D$.*

Note that in the above statement, only values $h(x, y)$ for $x < y$ are relevant. This will be the case in the other Ramsey-like statements below.

Proof. By induction we construct a sequence of sets of natural numbers $D_0 \supseteq D_1 \supseteq \cdots$. The set D_0 is defined to be $\mathbb{N}$. For $n > 0$, the set D_n is defined to be the infinite set D obtained by applying Ramsey's theorem to $E = D_{n-1} \setminus \{\min D_{n-1}\}$ with the binary property $R = h(x, y) \leq n$.

Two cases may happen. Either for some n, the value of $h(x, y)$ is at most n for all $x < y$ taken from D_n. In this case the first disjunct in the conclusion of the lemma holds (with $D = D_n$ and $M = n$). Otherwise, for all n, the value $h(x, y)$ is greater than n for all $x < y$ taken from D_n. In this case, we set D to be $\{\min D_i : i \in \mathbb{N}\}$ and g to satisfy $g(\min D_i) = i$. The second conclusion of the lemma holds. $\qquad\square$

Definition 5.6. A *separator* is a pair (f, g) where f is a $\bar{T}$-function and g is a T-function.

The following lemma restates the previous one in terms of separators.

Lemma 5.7. *For any $h : \mathbb{N}^2 \to \mathbb{N}$ there is an infinite set $D \subseteq \mathbb{N}$ and a separator (f, g) such that either;*

- *for all $x < y \in D$, $h(x, y) \preceq f(x)$, or;*
- *for all $x < y \in D$, $h(x, y) \succ g(x)$.*

Lemma 5.8 below generalizes Lemma 5.7: instead of having a single element $h(x, y)$ for each $x < y$, we have a set $E_{x,y}$ of vectors. The conclusion of the lemma describes which components of the input vectors from $E_{x,y}$ are bounded or unbounded simultaneously.

When applied to complementing automata, each set $h(x, y)$ will gather information relative to the possible runs of the automaton over the part of a word that begins in position x and ends in position y. Since the automaton is nondeterministic, $h(x, y)$ contains not a single element, but a set of elements, one for each possible run. Since the automaton has many counters, and a counter may come with several events, elements of $h(x, y)$ are vectors, with each coordinate corresponding to a single event.

Before stating the lemma, we introduce some notation. Let C be a finite set, which will be used for coordinates in vectors. Given a vector $v \in \mathbb{N}^C$, a set of coordinates $\sigma \subseteq C$ and a natural number M, the expression $v \preceq_\sigma M$ means that $v(\alpha) \preceq M$ holds for all coordinates $\alpha \in \sigma$. We use a similar notation for $\succ$, i.e. $v \succ_\sigma M$ means that $v(\alpha) \succ M$ holds for all coordinates $\alpha \in \sigma$. Note that $\npreceq_\sigma$ is different from $\succ_\sigma$. The first says that $\succ_{\{\alpha\}}$ holds for some coordinate $\alpha \in \sigma$, while the second says that $\succ_{\{\alpha\}}$ has to hold for all coordinates $\alpha \in \sigma$.

Lemma 5.8. *Let C be a finite set, and for every natural numbers $x < y \in \mathbb{N}$, let $E_{x,y} \subseteq \mathbb{N}^C$ be a finite nonempty set of vectors. There is a family of coordinate sets $\Theta \subseteq \mathcal{P}(C)$, an infinite set $D \subseteq \mathbb{N}$ and a separator (f, g) such that for all $x < y$ in D,*

(1) for all $v \in E_{x,y}$, there is a coordinate set $\sigma \in \Theta$ such that $v \preceq_\sigma f(x)$, and;

(2) for all coordinate sets $\sigma \in \Theta$, there is $v \in E_{x,y}$ such that $v \preceq_\sigma f(x)$ and $v \succ_{C \setminus \sigma} g(x)$.

Proof. If we only take (1) into account, we can see Θ as a disjunction of conjunctions of boundedness constraints, i.e., a DNF formula. The property (1) says that each vector

satisfies one of the disjuncts. Keeping this intuition in mind, for two coordinate sets $\sigma, \sigma' \subseteq C$ we write $\sigma \Rightarrow \sigma'$ if $\sigma \supseteq \sigma'$. Given two families of coordinate sets $\Theta, \Theta' \subseteq \mathcal{P}(\mathcal{C})$, we write $\Theta \Rightarrow \Theta'$ if for every disjunct $\sigma \in \Theta$ there is a disjunct $\sigma' \in \Theta'$ such that $\sigma \Rightarrow \sigma'$. This notation corresponds to the following property: if (1) holds for Θ and $\Theta \Rightarrow \Theta'$, then (1) also holds for Θ'. There is a minimum element for this preorder which is $\emptyset$ (the empty disjunction, equivalent to false), and a maximum element $\{\emptyset\}$ (a single empty conjunction, equivalent to true). The preorder $\Rightarrow$ induces an equivalence relation $\Leftrightarrow$, which corresponds to logical equivalence of DNF formulas.

Let $\Theta \subseteq \mathcal{P}(C)$ be a nonempty family of coordinate sets. For two natural numbers $x < y$, we define $h_\Theta(x, y) \in \mathbb{N}$ to be

$$h_\Theta(x, y) = \min_{\preceq} \{M \ : \ \forall \, v \in E_{x,y}. \, \exists \, \sigma \in \Theta. \, v \preceq_\sigma M\} \ .$$

By applying Lemma 5.7 to the property $h_\Theta(x, y)$, we obtain an infinite set $D \subseteq \mathbb{N}$ and a separator (f, g) such that either;

(a) for all $x < y \in D$, $h_\Theta(x, y) \preceq f(x)$,
   i.e., for all $v \in E_{x,y}$ there exists $\sigma \in \Theta$ such that $v \preceq_\sigma f(x)$, or;
(b) for all $x < y \in D$, $h_\Theta(x, y) \succ g(x)$,
   i.e., there exists $v \in E_{x,y}$ such that for all $\sigma \in \Theta$, $v \npreceq_\sigma g(x)$.

Using compositionality of Ramsey-like statements, we can assume that the separator (f, g) works for all possible families Θ simultaneously. Note however, that the choice of item (a) or (b) may depend on the particular family Θ. Furthermore remark that if to Θ corresponds property (a), and $\Theta \Rightarrow \Theta'$ holds, then property (a) also corresponds to Θ'. By removing a finite number of elements of D, we can further assume that for any x in D we have $f(x) \preceq g(x)$.

Let $\Theta \subseteq \mathcal{P}(C)$ be a family of coordinate sets that satisfies property (a), but is minimal in the sense that for every family Θ' satisfying (a), the implication $\Theta \Rightarrow \Theta'$ holds. The family Θ exists since $\{\emptyset\}$ satisfies (a) for any separator (f, g) (it is even unique up to $\Leftrightarrow$). Without loss of generality, we assume that Θ does not contain two coordinate sets $\sigma' \subseteq \sigma$, since otherwise we can remove the larger coordinate set σ and still get an equivalent family.

We will show that the properties (1) and (2) of the lemma hold for this family Θ. Property (1) directly comes from (a). Let us prove (2). Fix a coordinate set σ in Θ, as well as $x < y$ in D. We need to show that for some $v \in E_{x,y}$, both $v \preceq_\sigma f(x)$ and $v \succ_{C \setminus \sigma} g(x)$. Let Θ' be the family of coordinate sets obtained from Θ by removing σ and adding all coordinate sets of the form $\sigma \cup \{\beta\}$, for β ranging over $C \setminus \sigma$. It is easy to see that $\Theta' \Rightarrow \Theta$, while $\Theta \Leftrightarrow \Theta'$ does not hold. In particular, by minimality of Θ, the family Θ' cannot satisfy property (a), and therefore it must satisfy property (b). Let v be the vector of $E_{x,y}$ existentially introduced by property (b) applied to Θ'. We will show that this vector satisfies both $v \preceq_\sigma f(x)$ and $v \succ_{C \setminus \sigma} g(x)$.

First, we show $v \preceq_\sigma f(x)$. Since the family Θ satisfies property (a), there must be a coordinate set $\sigma' \in \Theta$ such that $v \preceq_{\sigma'} f(x)$. We claim that $\sigma' = \sigma$. Indeed, otherwise σ' would belong to Θ', and by (b) we would have $v \npreceq_{\sigma'} g(x)$. This is in contradiction with $v \preceq_{\sigma'} f(x)$ since $f(x) \preceq g(x)$.

Second, we show $v \succ_{C \setminus \sigma} g(x)$. Let then β be a coordinate in $C \setminus \sigma$. By definition of Θ', the coordinate set $\sigma \cup \{\beta\}$ belongs to Θ'. By (b) for Θ', we get $v(\alpha) \succ g(x)$ for some $\alpha \in \sigma \cup \{\beta\}$. Since above we have shown $v \preceq_\sigma f(x)$ and $f(x) \preceq g(x)$, it follows that $\alpha = \beta$ and $v(\beta) \succ g(x)$. Since this is true for all $\beta \in C \setminus \sigma$, we have $v \succ_{C \setminus \sigma} g(x)$. $\quad\square$

5.4. **Descriptions.** Recall the mapping $val(\rho, \alpha, c)$, which describes the number of increments that run ρ does on counter c in the event α. By fixing a run ρ of type t, we can define $val(\rho)$ as a mapping from the set of t-events $events(t)$ to $\mathbb{N}$, i.e., a vector of natural numbers. This vector measures the number of increments in the run ρ for each event and each counter, keeping track of the numbers which are the *worst* for the acceptance condition. Note that the coordinates of $val(\rho)$ depend on the type t of ρ—more precisely on $events(t)$—and therefore the vectors $val(\rho)$ and $val(\rho')$ may not be directly comparable for runs ρ, ρ' of different types. The key property of val is that given a sequence of finite runs $\rho_1, \rho_2, \ldots$, it is sufficient to know their respective types $t_1, t_2, \ldots$ and the vectors $val(\rho_1), val(\rho_2), \ldots$ in order to decide whether or not the infinite run $\rho_1 \rho_2 \ldots$ satisfies the acceptance condition. The descriptions defined below gather this information in a finite object.

Definition 5.9 (description). A *description* is a set of pairs (t, γ) where t is a type and γ a set of t-events.

The intuition is that γ is the set of events where the counter values are bad for the automaton's acceptance condition, i.e.. small in the case when $T = S$ and large in the case when $T = B$.

A *cut* is any infinite set of natural numbers D, which are meant to be word positions. We also view a cut as a sequence of natural numbers, by ordering the numbers from D in increasing order. Given a cut $D = \{d_1 < d_2 < \cdots\}$ and an ω-word $w \in \Sigma^\omega$, we define $w|_D$ to be the infinite sequence of finite words obtained by cutting the word at all positions in D:

$$w|_D = w[d_1, \ldots, d_2 - 1], \ w[d_2, \ldots, d_3 - 1], \ \ldots$$

Note that the prefix $w[0, \ldots, d_1 - 1]$ of w up to position $d_1 - 1$ is not used here; as in the proof of Büchi, it is treated separately.

Definition 5.10 (strong description). Let $w \in \Sigma^\omega$ be an input ω-word, τ a description and $D \subseteq \mathbb{N}$ a cut. We say that τ *strongly describes* $w|_D$ if there is a separator (f, g) such that for every $x < y \in D$ the following conditions hold.
- for every partial run ρ of type t over w from position x to position y, there is a pair $(t, \gamma) \in \tau$ such that $val(\rho) \preceq_\gamma f(x)$, and;
- for every pair $(t, \gamma) \in \tau$, there is a run ρ over w of type t from position x to position y such that $val(\rho) \preceq_\gamma f(x)$ and $val(\rho) \succ_{events(t) \setminus \gamma} g(x)$.

Lemma 5.11. *For every ω-word $w \in \Sigma^\omega$, there is a description τ and a cut D such that τ strongly describes $w|_D$.*

Proof. Using Ramsey's theorem and its compositionality, we find a cut D_0 and a set of types A such that for all $x < y$ in D_0 the following conditions are equivalent;
- there is a partial run of type t between positions x and y, and;
- the type t belongs to A.

The rest follows by applying Lemma 5.8 for all types t in A, and using compositionality of Ramsey-like statements. $\qquad \square$

The problem with strong descriptions is that an $\omega\bar{T}$-automaton cannot directly check if a description strongly describes some cut word $w|_D$. There are two reasons for this. First, we need to check the description for each $x < y$ in D, and therefore deal with the overlap between the finite words $w[x, \ldots, y]$. Second, the second condition of strong description involves guessing the T-function g, and this cannot be done using a $\omega\bar{T}$-automaton.

Hence, we reduce the property to checking weak descriptions (see Definition 5.12) which are stable (see Definition 5.14). Lemma 5.15 shows that this approach makes sense.

Definition 5.12 (weak description). Given a type t, a set of events γ and a natural number N, a finite run ρ, is called *consistent with* (t, γ) *under* N if ρ has type t and $val(\rho) \preceq_\gamma N$. Given a description τ, a run ρ is called *consistent with* τ *under* N if it is consistent with some $(t, \gamma) \in \tau$ under N.

Let $w \in \Sigma^\omega$ be an input ω-word, τ a description and D a cut. We say that τ *weakly describes* $w|_D$ if there is a $\bar{T}$-function f such that for all $i \in \mathbb{N}$, every run ρ over the i-th word in $w|_D$ is consistent with τ under $f(i)$.

A weak description is a weakening of strong descriptions for two reasons: only runs between consecutive elements of the cut are considered, and only the first constraint of the strong description is kept.

The following lemma shows that weak descriptions can be expressed by specifications.

Lemma 5.13. *For every weak description there is an equivalent specification. In other words, for every description τ there is a specification τ' such that the following are equivalent for an ω-word w and a cut D ;*

- *the description τ weakly describes $w|_D$, and;*
- *there is a $\bar{T}$-function f such that for all $i \in \mathbb{N}$, every run ρ over the i-th word in $w|_D$ satisfies τ' under $f(i)$.*

Proof. All the conditions in the definition of weak descriptions can be expressed by a specification. $\qquad\square$

In Definition 5.14, we present the notion of a stable description. The basic idea is to mimic the notion of idempotency used in the case of Büchi.

Definition 5.14 (stability). A description τ is *stable* if there is a T-function h such that for all natural numbers N and all finite runs $\rho = \rho_1 \ldots \rho_k$, if each ρ_i is consistent with τ under N for all $i = 1 \ldots k$, then ρ is consistent with τ under $h(N)$.

To illustrate the above definition, we present an example. In this particular example, the description will be stable for $T = B$, but it will not be stable for $T = S$. Let then $T = B$ and consider an automaton with one counter $\alpha = 1$, and one state q. We will show that for $t = (q, \{\text{-}\}, q)$ (we write $\{\text{-}\}$ instead of the mapping which to counter 1 associates $\{\text{-}\}$) and $\gamma = \{(1, \text{-})\}$, the description $\tau = \{(t, \gamma)\}$ is stable. In the case of τ, the function h from the definition of stability will be the identity function $h(N) = N$. To show stability of τ, consider any finite run ρ decomposed as $\rho_1 \cdots \rho_k$, with each ρ_i consistent with τ under N. To show stability, we need to show that

$$val(\rho) \preceq_\gamma h(N) = N \ .$$

Since $T = B$, the relation $\preceq$ is $\geq$, so we need to show that $val(\rho)$ is at least N on all events in γ. Since γ has only one event $(1, \text{-})$, this boils down to proving $val(\rho, 1, \text{-}) \geq N$. But this is simple, because

$$val(\rho, 1, \text{-}) = val(\rho_1, 1, \text{-}) + \cdots + val(\rho_k, 1, \text{-}) \ ,$$

and each ρ_i satisfies $val(\rho_i, 1, \text{-}) \geq N$ by assumption. Note that this reasoning would not go through with $T = S$, which is the reason why the above description is stable only in the case $T = B$.

We now show that strong descriptions are necessarily stable.

Lemma 5.15. *If τ is a strong description of some $w|_D$ then τ is stable.*

Proof. We prove the statement for $T = S$ first, and then for $T = B$.

Case $T = S$. In this case $\preceq$ is $\leq$. Let $\rho_1, \ldots, \rho_k$ be runs such that ρ_i is consistent with $(t_i, \gamma_i) \in \tau$ under N for all $i = 1, \ldots, k$. We need to show that the composition $\rho = \rho_1 \cdots \rho_k$ of these runs is consistent with some $(t, \gamma) \in \tau$ under $h(N)$, for some S-function h independent of $\rho_1, \ldots, \rho_k$. The function h will be a linear function, with the linear constant taken from the assumption that τ strongly describes $w|_D$. Let then (f_0, g_0) be the separator obtained by unraveling the definition of $w|_D$ being strongly described by τ. We can assume without loss of generality that f_0 is constant, equal to M. Since g_0 tends toward infinity, we can chose a natural number n such that $g_0(i) \geq M$ holds for all $i \geq n$.

We now mimic each run ρ_i by a similar run π_i over the $(i+n)$-th word in the sequence $w|_D$. By definition of a strong description, one can find for $i = 1, \ldots, k$ a run π_i over the $(i+n)$-th word in $w|_D$ such that $val(\pi_i) \leq_{\gamma_j} M$ and $val(\pi_i) >_{events(t_i)\backslash\gamma_i} g_0(i+n)$. From the last inequality together with $g_0(i+n) \geq M$, we obtain:

$$\text{for all } (\alpha, c) \in events(t_i), \qquad val(\pi_i, \alpha, c) \leq M \quad \text{implies} \quad (\alpha, c) \in \gamma_i. \tag{$\sharp$}$$

Let π be $\pi_1 \ldots \pi_k$. Since τ strongly describes $w|_D$, the run π is consistent with (t, γ) under M for some (t, γ) in τ. Formally,

$$val(\pi) \leq_\gamma M \qquad (\sharp 2) .$$

We will show that ρ is consistent with (t, γ) under $N(M+2)$, which establishes the stability of τ, under the linear function $h(N) = N(M+2)$. Let then (α, c) be an event in γ. We have to prove

$$val(\rho, \alpha, c) \leq N(M+2).$$

This is done by a case distinction depending on c.

$c = \text{-}$ This means that π does not contain any reset of α. Let $I \subseteq \{1, \ldots, k\}$ be the set of indexes j for which π_j contains an increment of counter α. By ($\sharp 2$), and since $(\alpha, \text{-})$ belongs to γ, the number of increments of counter α in π is at most M. In particular, the set I contains at most M indexes.

Let now $i \in I$. Still by ($\sharp 2$), the run π, and hence also π_i, contains at most M increments of α. By ($\sharp$), this means that $(\alpha, \text{-})$ belongs to γ_i. Hence, since ρ_i is consistent with (t_i, γ_i) under N, ρ_i contains at most N increments of α. Furthermore, for $i \notin I$, π_i does not increment α, and π_i has the same type as ρ_i. Hence ρ_i does not increment the counter α either. Summing up: at most M runs among $\rho_1, \ldots, \rho_k$ increment counter α, and those that do, do so at most N times.

Overall there are at most MN increments of α in ρ, i.e., $val(\rho, \alpha, c) \leq MN$.

$c = \text{⊸}$ Let j be the first index such that ρ_j contains a reset of α. Using the previous case, but for $k = j - 1$, we infer that the prefix $\rho_1 \ldots \rho_{j-1}$ contains at most MN increments of α.

By ($\sharp 2$), there are at most M increments of α before the first reset in π. Since this first reset occurs in π_j, the same holds for π_j. By ($\sharp$) we obtain that $(\alpha, \text{⊸})$ belongs to γ_j. Finally, since ρ_j is consistent with (t_j, γ_j) under N, there are at most N increments of α before the first reset in ρ_j.

Overall there are at most $N(M+1)$ increments of α before the first reset in ρ.

$c = \circ\!\!\text{-}$ As in the previous case.

$c = \infty$ As previously, but we need the bound $N(M+2)$ since both ends of the interval have to be considered.

Case $T = B$. In this case $\preceq$ is $\geq$. Let $\rho = \rho_1 \ldots \rho_k$ be a run such that ρ_i is consistent with $(t_i, \gamma_i) \in \tau$ under N for all $i = 1, \ldots, k$. We will show that ρ is consistent with τ under N, i.e. h is the identity function. Let then (f_0, g_0) be the separator obtained by unraveling the definition of $w|_D$ being strongly described by τ. We can assume without loss of generality that g_0 is constant, equal to M. As previously, since f_0 tends toward infinity, we can chose a natural number n such that $f_0(n+1) \geq M(N+2)$.

As in the case of $T = S$, we mimic each run ρ_i by a similar run π_i over the $(i+n)$-th word in $w|_D$. By definition of a strong description, one can find for $i = 1, \ldots, k$ a run π_i over the $(i+n)$-th word in $w|_D$ such that

$$val(\pi_i) \geq_{\gamma_j} f_0(i+n) \quad \text{and} \quad val(\pi_i) <_{events(t_i)\backslash\gamma_i} M .$$

From the last inequality we obtain:

$$\text{for all } (\alpha, c) \in events(t_i), \qquad val(\pi_i, \alpha, c) \geq M \quad \text{implies} \quad (\alpha, c) \in \gamma_i . \tag{$\sharp$}$$

Let π be $\pi_1 \ldots \pi_k$. Since τ strongly describes $w|_D$, the run π is consistent with (t, γ) under $f_0(n+1)$ for some (t, γ) in τ. In combination with $f_0(n+1) \geq M(N+2)$, we have:

$$val(\pi) \geq_{\gamma} M(N+2) . \tag{$\sharp 2$}$$

We will show that ρ is consistent with (t, γ) under N, which establishes the stability of τ. For this, let (α, c) be an event in γ, we have to prove

$$val(\rho, \alpha, c) \geq N.$$

This is done by a case distinction depending on c.

$c = \text{-}$ In this case the run π does not contain any reset of α. Let $I \subseteq \{1, \ldots, k\}$ be the set of indexes j for which π_j does an increment of counter α. By ($\sharp 2$) applied to $(\alpha, \text{-})$, there are at least MN increments (actually, at least $M(N+2)$ increments, but we only need MN here) of α in π. Two cases can happen; either I contains at least N indexes, or there is some $j \in I$ such that ρ_j contains at least M increments of α.

Consider first the case when I has at least N indexes. Since there is at least one increment of α in every ρ_i for $i \in I$, there are at least N increments of α in ρ.

Otherwise there is some $j \in I$ such that π_j contains at least M increments of α. By ($\sharp$), this means that $(\alpha, \text{-})$ belongs to γ_j. Finally, since ρ_j is consistent with γ_j under N, we obtain that ρ_j, and by consequence also ρ, contains at least N increments of α.

Overall there are at least N increments of α in ρ, i.e., $val(\rho, \alpha, c) \geq N$.

$c = \text{-}\!\circ$ Let j be the first index for which π_j contains a reset of α. By ($\sharp 2$), there are at least $M(N+1)$ increments (again, we do not need to use $M(N+2)$ here) of α before the first reset in π. Two cases are possible: either MN increments of α happen in $\pi_1 \ldots \pi_{j-1}$, or π_j contains M increments of α before the first reset of α.

In the first case we use the same argument as for $c = \text{-}$ (note that we were just using a bound of MN in this case) over the run $\pi_1 \ldots \pi_{j-1}$. We obtain that there are at least N increments of α in $\rho_1 \ldots \rho_{j-1}$.

Otherwise, there are M increments of α in π_j before the first reset. Using ($\sharp$) we deduce that $(\alpha, \text{-}\!\circ)$ belongs to γ_j. Since ρ_j is consistent with (t_j, γ_j) under N, we deduce that there are at least N increments of α in ρ_j before the first reset.

Overall there are at least N increments of α in ρ before the first reset, i.e., $val(\rho, \alpha, c) \geq N$.

$c = \circ\!\text{-}$ As in the previous case.

$c = \infty$ As previously (this time using the bound $M(N+2)$).   □

We will now show how to tell if a word is rejected by inspecting one of its descriptions. This notion of rejection will be parametrized by the set S of states in which the cut may be reached. Note that rejecting loops only make sense for stable descriptions.

Definition 5.16 (rejecting loop)**.** We say a state q is a *rejecting loop* in a description τ, if for all $(t, \gamma) \in \tau$ where the source and target state of the type t is q we have:

Case $T = S$. There exists a counter α such that either:

- $t(\alpha) = \emptyset$ or $t(\alpha) = \{\text{-}\}$, or;
- $(\alpha, \infty) \in \gamma$, or;
- Both $(\alpha, \text{-}\!\circ)$ and $(\alpha, \circ\!\text{-})$ belong to γ.

Case $T = B$. There exists a counter α such that either:

- $t(\alpha) = \emptyset$ or $t(\alpha) = \{\text{-}\}$, or;
- $(\alpha, c) \in \gamma$ for some $c \in \{\text{-}\!\circ, \infty, \circ\!\text{-}\}$.

The idea behind the above definition is as follows: if the description τ weakly describes a cut word $w|_D$, ρ is a run that assumes state q in every position from D, then the fact that q is a rejecting loop implies that ρ not accepting. Since every infinite run can be decomposed into loops, this is the key information when looking for a witness of rejection.

Given a description τ and a state p, we write $p\tau$ to denote the set of states q such that for some $(t, \gamma) \in \tau$, the source state of t is p and the target state of t is q. This notation is extended to a set of states $P\tau$ in the natural manner.

If D is a cut and w is an ω-word, then the *D-prefix* of w is defined to be the prefix of w that leads to the first position in D. Every ω-word w is decomposed into its D-prefix, and then the concatenation of words from $w|_D$.

Lemma 5.17. *Let $w|_D$ be a cut ω-word strongly described by τ. Let P be the states reachable after reading the D-prefix of w. If w is rejected, then every state in $P\tau$ is a rejecting loop.*

Proof. We only do the proof for $T = S$, the case of $T = B$ being similar. Let $D = \{d_1, d_2, \ldots\}$.

To obtain a contradiction, suppose that $q \in P\tau$ is not a rejecting loop. By definition, there must be a pair (t, γ) in τ—with q the source and target of t—such that for every counter α, none of the conditions from Definition 5.16 hold. That is, the value $t(\alpha)$ contains -$\circ$ and $\circ$-, the event (α, ∞) is outside γ, and one of the events $(\alpha, \text{-}\!\circ), (\alpha, \circ\!\text{-})$ is outside γ. Without loss of generality, let us assume $(\alpha, \text{-}\!\circ)$ is outside γ.

Let (f, g) be the separator appropriate to $w|_D$ obtained from the definition of strong descriptions. For each natural number i there is a run π_i of type t between positions d_i and d_{i+1} such that $val(\pi_i, \alpha, c) > g(d_i)$ holds for all events (α, c) not in γ.

Since $t(\alpha)$ contains -$\circ$ and $\circ$-, the counter α is reset at least once in π_i. Furthermore, since (α, ∞) is outside γ, every two consecutive resets of α in π_i are separated by at least $g(d_i)$ increments. Finally, since $(\alpha, \text{-}\!\circ)$ is outside γ, there are at least $g(d_i)$ increments of α before

the first reset in π_i. Since this holds for every counter, we obtain that the run $\pi_1\pi_2\ldots$ satisfies the accepting condition.

By assumption on $q \in P\tau$, there is some state $p \in P$ and a type in τ that has source state p and target state q. In particular, the state q can be reached in the second position of the cut D: by first reaching p after the D-prefix, and then going from p to q. From q in the second position of D, we can use the run $\pi_2\pi_3\ldots$ to get an accepting run over the word w. This contradicts our assumption that w was rejected by the automaton. □

The following lemma gives the converse of Lemma 5.17. The result is actually stronger than just the converse, since we use weaker assumptions (the description need only be weak and stable, which is true for every strong description, thanks to Lemma 5.15).

Lemma 5.18. *Let w be an ω-word, $D \subseteq \mathbb{N}$ a cut, and assume that the word sequence $w|_D$ is weakly described by a stable description τ. Let P be the states reachable after reading the D-prefix of w. If every state in $P\tau$ is a rejecting loop, then w is rejected.*

Proof. Let ρ be a run of the automaton over w. We will show that this run is not accepting.

Let q_i be the state used by ρ at position d_i. For $i < j$, we denote by $\rho_{i,j}$ the subrun of ρ that starts in position d_i and ends in position d_j. Let $t_{i,j}$ be the type of this run.

Consider now a run $\rho_{i,j}$. This run can be decomposed as

$$\rho_{i,j} = \rho_{i,i+1}\rho_{i+1,i+2}\cdots\rho_{j-1,j} \ .$$

Let f be the $\bar{T}$-function from the assumption that τ weakly describes $w|_D$. By recalling the definition of weak descriptions, there must be sets of events $\gamma_i, \ldots, \gamma_{j-1}$ such that

$$(t_{i,i+1}, \gamma_i) \in \tau \qquad \cdots \qquad (t_{j-1,j}, \gamma_{j-1}) \in \tau$$
$$val(\rho_{i,i+1}) \preceq_{\gamma_i} f(d_i) \qquad \cdots \qquad val(\rho_{j-1,j}) \preceq_{\gamma_{j-1}} f(d_{j-1}) \ .$$

Let g be a function, which to every $k \in \mathbb{N}$ assigns the maximal, with respect to $\preceq$, value among $f(k), f(k+1), \ldots$ We claim that not only g is well defined, but it is also a $\bar{T}$ function. Indeed, when $\bar{T}$ is B then there are finitely many values of f, so a maximal one exists, and g has also finitely many values. If, on the other hand, $\bar{T}$ is S, then $\preceq$ is $\geq$. In this case, $g(k)$ is the least—with respect to the standard ordering $\geq$ on natural numbers—number among $f(k), f(k+1), \ldots$. This number is well defined, furthermore, g is an S-function since f is an S-function.

Since $f(d_k) \preceq g(d_i)$ holds for any $k \geq i$, we also have

$$val(\rho_{i,i+1}) \preceq_{\gamma_i} g(d_i) \qquad \cdots \qquad val(\rho_{j-1,j}) \preceq_{\gamma_j} g(d_i) \ .$$

By assumption on stability of τ, there is an S-function h, such that for all $i < j$, there is

$$(t_{i,j}, \gamma_{i,j}) \in \tau \qquad \text{such that} \qquad val(\rho_{i,j}) \preceq_{\gamma_{i,j}} h(g((d_i))) \ .$$

Using the Ramsey theorem in a standard way, we can assume without loss of generality that all the $t_{i,j}$ are equal to the same type t, and all $\gamma_{i,j}$ are equal to the same γ. If $t(\alpha) = \{\text{-}\}$ holds for some counter α, then this counter is reset only finitely often, so the run ρ is rejecting and we are done. Otherwise, $t(\alpha) = \{\text{-}\circ, \infty, \circ\text{-}\}$ holds for all counters α.

For the rest of the proof, we only consider the case $T = S$, with B being treated in a similar way. The function g, by its definition, assumes some value M for all but finitely many arguments. Since τ is rejecting there exists a counter $\alpha \in \Gamma$ such that either $(\alpha, \infty) \in \gamma$ or both $(\alpha, \text{-}\circ)$ and $(\alpha, \circ\text{-})$ belong to γ. In the first case, an infinite number of times there are at most $h(M)$ increments of α between two consecutive resets of α. In the second case, the

same happens, but this time with at most $2h(M)$ increments. In both cases the run is not accepting. $\qquad\square$

We are now ready to establish the main lemma of this section.

Lemma 5.19. *Let $\mathcal{A}$ be an ωT-automaton. There exist regular languages $L_1, \ldots, L_n$ and stable descriptions $\tau_1, \ldots, \tau_n$ such that for every ω-word w the following items are equivalent:*
- *$\mathcal{A}$ rejects w, and;*
- *There is some $i = 1, \ldots, n$ and a cut D such that the D-prefix of w belongs to L_i, and τ_i weakly describes $w|_D$.*

Proof. We need to construct a finite set of pairs (L_i, τ_i). Each such pair (L_P, τ) comes from a set of states P and a description τ that is stable and rejects all loops in $P\tau$. The language L_P is the set of finite words v that give exactly states P (as far as reachability from the initial state is concerned) after being read by the automaton.

The bottom-up implication is a direct application of Lemma 5.18, and therefore only the top down implication remains. Let w be an ω-word rejected by $\mathcal{A}$. By Lemma 5.11, there exists a cut D and a strong—and therefore also weak—description τ of $w|_D$. Let P be the set of states reached after reading the D-prefix of w. Clearly the D-prefix of w belongs to L_P. By Lemma 5.15, the description τ is stable, and hence Lemma 5.17 can be applied to show that all loops in $P\tau$ are rejecting. $\qquad\square$

The first part of Proposition 5.2 follows, since weak descriptions are captured by specifications thanks to Lemma 5.13. Finally, we need to show that our construction is effective:

Lemma 5.20. *It is decidable if a description is stable.*

Proof. Stability can be verified by a formula of monadic second-order logic over $(\mathbb{N}, \leq)$. $\quad\square$

5.5. **Verifying single events.** We now begin the part of the complementation proof where we show that specifications can be recognized by automata. Our goal is as follows: given a specification τ, we want to construct a hierarchical sequence $\bar{T}$-automaton that accepts the word sequences that are consistent with τ. Recall that a specification is a positive boolean combination of two types of atomic conditions. In this section we concentrate solely on atomic specifications where the boolean combination consists of only one atomic condition of the form:

$\qquad$ (2) The run satisfies $val(\rho, \alpha, c) \preceq K$.

In particular, only one event (α, c) is involved in the specification. Furthermore, we also assume that the counter α is the lowest-ranking counter 1. However, the result is stated so that it can then be generalized to any specification.

5.5.1. *Preliminaries and definitions.* In this section, we define transition graphs—which are used to represent possible runs of an automaton—and then we present a decomposition result for transition graphs, which follows from a result of Simon on factorization forests [13].

Transition graph. For the rest of Section 5.5, we fix a finite set of states M. This set M is possibly different from the set of states of the automaton we are complementing. The reason is that in Section 5.6, we will increase the state space of the complemented automaton inside an induction.

The first concept we need is an explicit representation of the configuration graph of an automaton reading a word, called here an M-transition graph. Fix a finite set $\mathcal{L}$ of *transition labels*, and a finite set of *states M*. An *M-transition graph G (labeled by $\mathcal{L}$) of length $k \in \mathbb{N}$* is a directed edge labeled graph, where the nodes—called *configurations* of the graph—are pairs $M \times \{0, \dots, k\}$ and the edge labels are of the form $((q, i), l, (r, i + 1))$ for q, r in M, $0 \le i < k$ and $l \in \mathcal{L}$. The vertexes of the graph are called *configurations*, their first component is called the *state*, while the second is called the *position*. The edges of the graph are called the *transitions*. We define the *concatenation* of M-transition graphs in a natural way. A *partial run* in a transition graph is just a path in the graph. A *run* in a transition graph is a path in the graph that begins in a configuration at the first position 0 and ends in a configuration at the last position. The *label* of a run is the sequence of labels on edges of the path.

A transition graph of length k can also be seen as a word of length k over the alphabet

$$\mathcal{P}(M \times \mathcal{L} \times M) \ .$$

In this case, the concatenation of transition graphs coincides with the standard concatenation of words. When speaking of regular sets of transition graphs, we refer to this representation.

Given a ωT-automaton $\mathcal{A}$ over the alphabet $\mathcal{L}$ of states Q, the *product of an M-transition graph G with $\mathcal{A}$*—noted $G \times \mathcal{A}$—is the $(M \times Q)$-transition graph which has an l-labeled edge from $((p, q), i)$ to $((p', q'), i + 1)$ whenever G has an l-labeled edge from (p, i) to $(p', i + 1)$ and (q, l, q') is a transition of the automaton $\mathcal{A}$. Furthermore, in the product graph only those configurations are kept that can be reached via a run that begins in a configuration where the second component of the state is the initial state of $\mathcal{A}$.

Factorization forest theorem of Simon, and decomposed transition graphs. From now, we will only consider transition graphs where the first component of the labeling ranges over the actions of a hierarchical automaton over counters, i.e., the label alphabet is of the form $\text{Act} \times \mathcal{L}$, with

$$\text{Act} \ = \{\varepsilon, I_1, R_1, I_2, \dots, R_n\} \ .$$

The type of a run is defined as in the previous section, i.e., a type gives the source and target states, as well as a mapping from the set of counters to $\{\emptyset, \{\text{-}\}, \{\circ, \circ\}, \{\circ, \circ\circ, \circ\}\}$. Given an M-transition graph, its *type* is the element of

$$S \ = \mathcal{P}(M \times \{\emptyset, \{\text{-}\}, \{\circ, \circ\}, \{\circ, \circ\circ, \circ\}\}^{\Gamma} \times M) \,,$$

which contains those types t such that there is a run over G of type t. This set S can be seen as a semigroup, when equipped with the product defined by:

$$s_1 \cdot s_2 = \{t_1 \cdot t_2 \ : \ t_1 \in s_1, t_2 \in s_2\} \ .$$

In the above, the concatenation of two types $t_1 \cdot t_2$ is defined in the natural way: the source state of t_2 must agree with the target state of t_1, and the counter operations are

concatenated. (An example of how counter operations are concatenated is:

$$\{\circ, \circ\!\!\!-\} \cdot \{\circ, \circ\!\!\!-\} = \{\circ, \circ\circ, \circ\!\!\!-\} \ .$$

In particular, the mapping that assigns the type to a counter transition graph is a semigroup morphism, with transition graphs interpreted as words. Two M-transition graphs G, H are called *equivalent* if they have the same type. This equivalence is clearly a congruence of finite index with respect to concatenation. A transition graph G is *idempotent* if the concatenation GG is equivalent to G.

We now define a complexity measure on graphs, which we call their *Simon* level. A transition graph G has Simon level 0 if it is of length 1. A transition graph G has Simon level at most $k + 1$ if it can be decomposed as a concatenation

$$G = HG_1 \cdots G_n H' \ ,$$

where all the transition graphs $H, G_1, \ldots, G_n, H'$ have Simon level at most k and $G_1, \ldots, G_n$ are equivalent and idempotent. The following theorem, which in its original statement concerns semigroups, is presented here in a form adapted to our context.

Theorem 5.21 (Simon [13], and [5] for the bound $|S|$)**.** *Given a finite set of states M, the Simon level of M-transition graphs is bounded by $|S|$.*

The Simon level is defined in terms of a nested decomposition of the graph into factors. We will sometimes need to refer explicitly to such decompositions; for this we will use symbols $(,),|$ and write G as $(H|G_1|\ldots|G_n|H')$, and so on recursively for the graphs $H, G_1, \ldots, G_n, H'$. Theorem 5.21 shows that each graph admits a decomposition where the nesting of parentheses in this notation is bounded by $|S|$. We refer to the transition graphs written in this format as *decomposed transition graphs*. It is not difficult to see that the set of decomposed M-transition graphs is a regular language: a finite automaton can check that the symbols $(,),|$ indeed describe a Simon decomposition (thanks to Simon's theorem, the automaton does not need to count too many nested parentheses).

A *hint* over a decomposed graph G is a subset of the positions labeled $|$ in the decomposition of G. We call those positions *hinted positions*. (Note that a hint is relative not just to a transition graph G, but also to some decomposition of this transition graph.) For $K \in \mathbb{N}$, a hint is said to be $\geq K$ if at the same nesting level of the decomposition, every two distinct hinted positions are separated by at least K non-hinted symbols $|$. Similarly, a hint is $\leq K$ if sequences of consecutive non-hinted positions at a given Simon level have length at most $K - 1$.

Let G be a transition graph of length k (in the word representation) (with labels $\mathrm{Act} \times \mathcal{L}$), along with a decomposition and a hint h. Let

$$G_1 \cdots G_k \in (\mathcal{P}(M \times \mathrm{Act} \times \mathcal{L} \times M))^*$$

be the interpretation of this graph as word. In the proofs below, it will be convenient to decorate the graph—by expanding the transition labels—so that the label of each run contains information about the decomposition and the hint. The decorated graph is denoted by (G, h) (we do not include a name for the decomposition in this notation, since we assume that the hint h also contains the information on the Simon decomposition). The graph (G, h) has the same configurations, states and transitions as G; only the labeling of the transitions changes. Instead of having a label in $\mathrm{Act} \times \mathcal{L}$, as in the graph G, a transition in the graph

(G, h) has a label in

$$\mathcal{L}_S = \text{Act} \times \mathcal{L} \times \{\bot, 1, \dots, |S|\} \times \{0, 1\}.$$

The first two coordinates are inherited from the graph G. The other coordinates are explained below. To understand the encoding, we need two properties. First, in the decomposition, there is exactly one symbol | between any two successive letters G_i, G_{i+1} of the M-transition graph seen as a word. Second, the decomposition is entirely described by the nesting depths of those symbols with respect to the parentheses. Recall also that these nesting depths are bounded by $|S|$. For a transition in the graph G_i, the coordinate $\{\bot, 1, \dots, |S|\}$ stores the nesting depth (with respect to the parentheses) of the symbol | preceding the transition graph G_i; by the first property mentioned above, the undefined value $\bot$ is used only for the first transition in transition graph. Finally, the coordinate $\{0, 1\}$ says if the symbol | is included in the hint.

Runs over decomposed graphs. As remarked above, the transition graph (G, h) only changes the labels of transitions in the graph G. Therefore with each run ρ in G we can associate the unique corresponding run over (G, h), which we denote by (ρ, h). By abuse of notation, we will sometimes write $(\rho, h) \in L$, where $L \subseteq (\mathcal{L}_S)^*$. The intended meaning is that the labeling of the run (ρ, h) belongs to the language L.

Recall that we are only going to be verifying properties for the counter $\alpha = 1$ in this section. We consider two runs over the same transition graph to be $\equiv_\circ$-equivalent, if they agree before the last 1-reset (i.e. use the same transitions on all positions up to and including the last 1-reset). In the notation, the index shows where the runs can be different. Similarly, two runs are $\equiv_\circ$-equivalent if they agree after the first 1-reset. Two runs are $\equiv_{\circ\circ}$-equivalent if they agree before the first 1-reset, after the last 1-reset, and over all 1-resets (but do not necessarily agree between two successive 1-resets). Finally, two runs are $\equiv_-$-equivalent if both increment counter 1 but do not reset it. The fundamental property of $\equiv_c$-equivalence, for $c \in \{\circ\text{-}, \text{-}\circ, \circ\circ, \text{-}\}$, is that two $\equiv_c$-equivalent runs are indistinguishable in terms of resets and increments of counters greater or equal to 2, or in terms of events $(1, c')$ with $c' \neq c$. This means that as long we are working inside a $\equiv_c$-equivalence class, the values $val(\rho, \alpha, c')$ for $(\alpha, c') \neq (1, c)$ are constant. (This also is the reason why we only work with counter 1 in a hierarchical automaton.)

The key lemmas in this section are Lemmas 5.23 and 5.24. These talk about complementing S-automata and B-automata respectively. They both follow the same structure, which can be uniformly expressed in the following lemma, using the $\preceq$ order (which is $\leq$ for complementing S-automata and $\geq$ for complementing B-automata):

Lemma 5.22. *Let c be one of $\text{-}\circ, \circ\circ, \circ\text{-}$ or -. There are two strongly unbounded functions f and g, and a regular language $L_c \subseteq (\mathcal{L}_S)^*$ such that for every natural number K, every run ρ over every M-transition graph labeled by $\mathcal{L}$ whose type contains the event $(1, c)$ satisfies:*

- *(correctness) if a hint h in the graph is $\preceq K$ and every run $\pi \equiv_c \rho$ satisfies $(\pi, h) \in L_c$, then $val(\rho, 1, c) \preceq f(K)$, and;*
- *(completeness) if a hint h in the graph is $\succeq g(K)$ and $val(\rho, 1, c) \preceq K$ then (ρ, h) belongs to L_c.*

We would like to underline here that the regular language is a regular language of finite words in the usual sense, i.e., no counters are involved.

The above lemma says that the language L_c "approximates" the runs ρ satisfying $val(\rho, 1, c) \preceq K$. This "approximation" is given by the two functions f and g. There is however a dissymmetry in this statement. The completeness clause states that if a run has a bad value on event $(1, c)$, i.e. it satisfies $val(\rho, 1, c) \preceq K$, then it is detected by L_c for all sufficiently good hints. On the other hand, in the correctness clause, all $\equiv_c$-equivalent runs must be detected by L_c in order to say that the value of ρ is bad on event $(1, c)$. The reason for the weaker correctness clause is that we will not be able to check the value on event $(1, c)$ for every run; we will only do it for runs of a special simplified form. And the simplification process happens to transform each run in a $\equiv_c$-equivalent one.

The proof of this lemma differs significantly depending on $T = S$ or $T = B$. The two cases correspond to Lemmas 5.23 and 5.24 which are instantiations of Lemma 5.22.

5.5.2. *Case of complementing an ωS-automaton.* We consider first the case of complementing an S-automaton. Therefore, the order $\preceq$ is $\leq$. To aid reading, below we restate Lemma 5.22 for the case when $T = S$.

Lemma 5.23. *Let c be one of* -○, ○○, ○- *or* -. *There are two strongly unbounded functions f and g, and a regular language $L_c \subseteq (\mathcal{L}_S)^*$ such that for every natural number K, every run ρ over every M-transition graph labeled by $\mathcal{L}$ whose type contains the event $(1, c)$ satisfies:*
- *(correctness) if a hint h in the graph is $\leq K$ and every run $\pi \equiv_c \rho$ satisfies $(\pi, h) \in L_c$, then $val(\rho, 1, c) \leq f(K)$, and;*
- *(completeness) if a hint h in the graph is $\geq g(K)$ and $val(\rho, 1, c) \leq K$ then (ρ, h) belongs to L_c.*

Slightly ahead of time, we remark that g will be the identity function, while f will be a polynomial, whose degree is the maximal Simon level of M-transition graphs, taken from Theorem 5.21.

Before proving the lemma, we would like to give some intuition about the language L_c. The idea is that we want to capture the runs which do few increments on counter 1. However, this "few" cannot be encoded in the state space of the automaton, since it can be arbitrarily large. That is why we use the hint. One can think of the hint as a clock: if the hint is $\leq K$, then the clock ticks quickly, and if the hint is $\geq K$, then the clock ticks slowly. The language L_c looks at a run and compares it to the clock. For the sake of the explanation, let us consider a piece of a run without resets of counter 1: we want to estimate if a lot of increments of counter 1 are done (at least K) or not (at most $f(K)$) by comparing it to a suitable clock (in the first case a slow clock, in the second case a quick one). The first case is when the counter is incremented in every position between two ticks of the clock; then the value of the counter concerned is considered 'big', since at least K increments are done if the ticks are $\geq K$. For the second case, when there are few increments, we have a more involved argument that uses the idempotents from the Simon decomposition.

Proof. The language L_c is defined by induction on the Simon level of the transition graph. We construct a language L_c^k which has the stated property for all M-transition graphs of Simon level at most k. Since there is a bound on the Simon level, the result follows.

For $k = 0$, the construction is straightforward, since the transition graphs are of length 1 and there is a finite number of possible runs to be considered.

We now show how to define the language L_c^{k+1} for runs in transition graphs of Simon level $k + 1$, based on the languages for runs in transition graphs of Simon level up to k.

Consider a decomposed M-transition graph

$$G = (H|G_1|\dots|G_n|H') .$$

of Simon level $k + 1$. Let h be a hint for this decomposition, which we decompose into sub-hints $(h_0|\cdots|h_{n+1})$ for the graphs $H, G_1, \dots, G_n, H'$. In the proof, instead of writing $(\rho, h) \in L_c^{k+1}$, we write that ρ is *c-captured*. Let ρ be a run over G. The run ρ can also be decomposed into subruns $(\rho_0|\rho_1|\dots|\rho_{n+1})$. Each of these runs $\rho_0, \dots, \rho_{n+1}$ is over a transition graph whose Simon level is at most k. By abuse of notation, we will talk about a subrun ρ_i being c-captured, the intended meaning being that (ρ_i, h_i) belongs to the appropriate language $L_c^{k'}$, with $k' \le k$ being the Simon level of G_i (or H if $i = 0$, or H' if $i = n + 1$).

We now proceed to define the language L_c^{k+1}. In other words, we need to say when ρ is c-captured. We only do the case of $c = \infty$, which is the most complex situation. The idea is that a run is ∞-captured if there are two consecutive resets between which the run does few increments. We define ρ to be ∞-captured if either:

(1) some ρ_i is ∞-captured, or;
(2) for some $i < j$, the subrun ρ_i is $\circ$-captured, the run ρ_j is $\circ$-captured, each of $\rho_{i+1}, \dots, \rho_{j-1}$ is --captured, and one of the following holds:
    (a) there are $m \le m'$ in $\{i, \dots, j\}$ such that: (i) one of $\rho_m, \rho_{m+1}, \dots, \rho_{m'}$ does not increment counter 1; and (ii) the source state of ρ_m and the target state of $\rho_{m'}$ are the same state q, and G_1 admits a run from q to q that increments counter 1, or;
    (b) between any two hinted (by hints on level $k + 1$) positions in $\{i, \dots, j\}$, at least one of the runs ρ_m does not increment counter 1.

It is clear that this definition corresponds to a regular property L_∞^{k+1} of the sequence of labels in a hinted runs (once the hints are provided, the statement above is first-order definable).

We try to give an intuitive description of the above conditions. The general idea is that a run gets ∞-captured if it does few increments, at least relatively to the size of the hint. The first reason why a run may do few increments between some two resets, is that it does it inside one of the component transition graphs of smaller Simon level. This is captured by condition 1. The second condition is more complicated. The idea behind 2(a) is that the run is—in a certain sense—suboptimal and can be converted into a $\equiv_\infty$-equivalent one that does "more" increments. Then the more optimal run can be shown—using conditions 1 and 2(b)—to do few increments, which implies that the original suboptimal run also did few increments.

We now proceed to show that the properties defined above satisfy the completeness and correctness conditions in the statement of the lemma.

Completeness.   For this, we set $g(K) = K$. Let ρ be a run such that

$$val(\rho, 1, \infty) \le K$$

and let h be hint over G that is $\ge K$. We have to show that ρ is ∞-captured.

Consider a minimal subrun $\rho_i \dots \rho_j$ that resets counter 1 twice and satisfies

$$val(\rho_i \dots \rho_j, 1, \infty) \le K .$$

In particular, the counter 1 is reset in the subruns ρ_i and ρ_j. If $i = j$, this means that $val(\rho_i, 1, \infty) \le K$, and hence ρ_i is ∞-captured on a level below $k+1$ by induction hypothesis. We conclude with item 1.

Otherwise, we have

$$val(\rho_i, 1, \circ\text{-}) \leq K\,,$$
$$val(\rho_{i+1}, 1, \text{-}) \leq K \quad \cdots \quad val(\rho_{j-1}, 1, \text{-}) \leq K\,,$$
$$\text{and} \quad val(\rho_j, 1, \text{-}\circ) \leq K\,.$$

By induction hypothesis, we obtain the header part of item 2. Since the run $\rho_i \ldots \rho_j$ contains less than K increments of 1, no more than K runs among $\rho_i, \ldots, \rho_j$ can increment counter 1. Since the hint h is $\geq K$, we get item (b).

Correctness. Let f' be the strongly unbounded function obtained by the induction hypothesis for Simon level k. We set

$$f(K) = (2K|M| + 2)f'(K)$$

Assume now that the hint h is $\leq K$ and take a run ρ such that every run $\pi \equiv_{\circ\circ} \rho$ is $\circ\circ$-captured. We need to show that

$$val(\rho, 1, \circ\circ) \leq f(K)\,. \tag{5.1}$$

As before, we decompose the run ρ into $(\rho_0 | \ldots | \rho_{n+1})$.

We are going to first transform ρ into a new $\equiv_{\circ\circ}$-equivalent run π which, intuitively, is more likely to have many increments on counter 1. We will then show that the new run π satisfies inequality (5.1); moreover we will show that this inequality can then be transferred back to ρ.

We begin by describing the transformation of ρ into π. This transformation is decomposed into two stages.

In the first stage, which is called the *local transformation*, we replace the subruns $\rho_0, \ldots, \rho_{n+1}$ with new equivalent ones of the same type. Each such replacement step works as follows. We take some $c = \text{-}\circ, \circ\circ, \circ\text{-}, \text{-}$ and $i = 0, \ldots, n+1$. If there is a subrun $\pi_i \equiv_c \rho_i$ that is not c-captured, then we replace ρ_i with π_i. (We want the local transformation to keep the $\equiv_{\circ\circ}$-equivalence class, so we do not modify subruns before the first or after the last reset of 1 in ρ.) The local transformation consists of applying the replacement steps as long as possible. This process terminates, since the replacement steps for different c's work on different parts of the subrun and each step decreases the number of captured subruns.

In the second stage, which is called the *global transformation*, we add increments on counter 1 to some subruns. The idea is that at the end of the global transformation, a run does not satisfy condition 2(a). Just as the local transformation, it consists of applying a replacement step as long as possible. The replacement step works as follows. We try to find a subrun $\rho_m \ldots \rho_{m'}$ as in item 2(a). By assumption 2(a) and since all the graphs G_i's are equivalent to G_1, we can find new subruns $\pi_m, \ldots, \pi_{m'}$ in $G_m, \ldots, G_{m'}$ respectively, which increment counter 1 without resetting it (that is, of type -) and go from q to q. We use these runs instead of $\rho_m \ldots \rho_{m'}$. The iteration of this replacement step terminates, since each time we add new subruns with increments.

Neither the local nor the global transformation change the $\equiv_{\circ\circ}$-equivalence class of the run.

Let π be a run obtained from ρ by applying first the local and then the global transformation. (Since the global transformation is nondeterministic, there may be more than one such run.) This run cannot satisfy 2(a), since the global transformation could still be applied.

Since π is $\equiv_{\infty}$-equivalent to ρ, it must be ∞-captured by assumption on ρ. There are two possible reasons: either because of 1, or because of 2(b). We will now do a case analysis on the reasons why this happens. In each case we will conclude that the original run ρ satisfies (5.1). As before, we decompose π into $(\pi_0 | \cdots | \pi_{n+1})$.

Assume now item 1 holds for π, i.e., some subrun π_i is ∞-captured for some i. We also know that any run $\pi'_i \equiv_{\infty} \pi_i$ would also be ∞-captured, since otherwise π_i would be replaced in the local transformation process (the global transformation does not touch subruns with resets on counter 1). In particular, the induction hypothesis gives

$$val(\pi_i, 1, \infty) \leq K .$$

Moreover, the runs ρ_i and π_i agree on the part between the first and last reset of counter 1. This is because neither of the transformation processes touched this part. Indeed, the first local transformation process never modified ρ_i for $c = \infty$ (since otherwise it would cease being captured), while the global process only modifies subruns without resets of counter 1. This gives the desired (5.1), since

$$val(\rho, 1, \infty) \leq val(\rho_i, 1, \infty) = val(\pi_i, 1, \infty) \leq f'(K) \leq f(K) .$$

Otherwise, π satisfies item 2(b). Let us fix i, j as in 2(b). Let $I \subseteq \{i+1, \ldots, j-1\}$ be the set of those indexes l where π_l does at least one increment on counter 1. A maximal contiguous (i.e. containing consecutive numbers) subset of I is called an *inc-segment*. According to case 2(b), an inc-segment cannot contain two distinct hinted positions, its size is therefore at most twice the maximal width K of h. Assume now that there are more than $|M|$ inc-segments. Then two inc-segments contain runs with the same source state, say state q, at respective positions l and l' with $l < l'$. This means that there is a run that goes from q to q in some graph $G_{l+1} \cdots G_{l'}$ and does at least one increment but no resets on counter 1. Using idempotency and the equivalence of all the graphs $G_1, \ldots, G_n$, this implies that the graph $G_{l'}$ admits such a run. Since $\rho_{l'}$ contains no increments of 1 by definition of an inc-segment, this means that 2(a) holds; a contradiction. Consequently there are at most $|M|$ inc-segments, each of size at most $2K$. It follows that at most $2K|M|$ subruns among $\pi_i, \ldots, \pi_j$ contain an increment of the counter 1.

Let us come back to the original run ρ. As in the case of item 1, we use the induction hypothesis to show that

$$val(\rho_i, 1, \circ) = val(\pi_i, 1, \circ) \leq f'(K),$$
$$\text{and} \quad val(\rho_j, 1, \multimap) = val(\pi_j, 1, \multimap) \leq f'(K).$$

Since the global transformation only adds increments on counter 1, we use the remarks from the previous paragraph to conclude that there are at most $2K|M|$ subruns among $\rho_{i+1}, \ldots, \rho_{j-1}$ that increment counter 1. Furthermore, all runs $\equiv_{-}$-equivalent to one of the $\rho_{i+1}, \cdots, \rho_{j-1}$ are --captured, since otherwise the local transformation would be applied, and then one of the runs $\pi_{i+1}, \cdots, \pi_{j-1}$ would not be captured. Therefore, we can use the induction hypothesis to show that

$$val(\rho_{i+1}, 1, -), \cdots, val(\rho_{j-1}, 1, -) \leq f'(K) .$$

All this together witnesses the expected

$$val(\rho, 1, \infty) \leq (2K|M| + 2)f'(K) . \qquad \square$$

5.5.3. *Case of complementing an ωB-automaton.* As when complementing an ωS-automaton, we restate the lemma with $\preceq$ expanded to its definition, which is $\geq$ in this case.

Lemma 5.24. *Let c be one of $\circ\text{-}, \circ\circ, \circ\text{-}$ or -. There are two strongly unbounded functions f and g, and a regular language $L_c \subseteq (\mathcal{L}_S)^*$ such that for every natural number K, every run ρ over every M-transition graph labeled by $\mathcal{L}$ whose type contains the event $(1,c)$ satisfies:*
- *(correctness) if a hint h is $\geq K$ in the graph and every run $\pi \equiv_c \rho$ satisfies $(\pi, h) \in L_c$, then $val(\rho, 1, c) \geq f(K)$, and;*
- *(completeness) if a hint h in the graph is $\leq g(K)$ and $val(\rho, 1, c) \geq K$ then (ρ, h) belongs to L_c.*

We remark here that f will be the identity function, while g will be more or less a k-fold iteration of the square root, with k the maximal Simon level of M-transition graphs, taken from Theorem 5.21.

Proof. The structure of the proof follows the one in Lemma 5.23. As in that lemma, the language L_c is defined via an induction on the Simon level of the transition graph. We also only treat the case of $c = \infty$. The other cases can be treated with the same technique.

Consider a decomposed transition graph

$$G = (H|G_1|G_2|\ldots|G_n|H')$$

of Simon level $k+1$, along with a corresponding hint h, decomposed as $(h_0|\ldots|h_{n+1})$. Let ρ be a run over G, decomposed as $(\rho_0|\ldots|\rho_{n+1})$. As in Lemma 5.23, instead of saying that (ρ, h) belongs to the language L_c^{k+1}, we say that ρ is c-captured; likewise for the runs $\rho_0, \ldots, \rho_{n+1}$.

We define ρ to be ∞-captured if either:
(1) some ρ_i is ∞-captured or there is a subrun of the form $\rho_i \ldots \rho_j$ such that counter 1 is reset in both ρ_i and ρ_j but not in $\rho_{i+1}, \ldots, \rho_{j-1}$, and either: ρ_i is $\circ\text{-}$-captured, one of $\rho_{i+1}, \ldots, \rho_{j-1}$ is --captured, or ρ_j is $\text{-}\circ$-captured;
(2) there is a subrun of the form $\rho_i \ldots \rho_j$ such that counter 1 is reset in both ρ_i and ρ_j but not in $\rho_{i+1}, \ldots, \rho_{j-1}$, and either
   (a) there are $m \leq m'$ in $\{i \ldots j-1\}$ such that there is an increment of 1 in one of $\rho_m, \ldots, \rho_{m'}$, but G_1 contains a path from the source state of ρ_m to the target state of $\rho_{m'}$ without any increment nor reset, or;
   (b) there are two hinted positions $m < m'$ in $\{i+1 \ldots j-1\}$ such that all the subruns $\rho_m, \ldots, \rho_{m'}$ increment counter 1.

As in the case of $T = S$, this definition corresponds to a regular property L_∞^{k+1} of hinted runs.

The intuition is that a ∞-captured run does a lot of increments on counter 1, at least relative to the size of the hint. The first reason for doing a lot of increments is that there are a lot of increments in a subrun inside one of the component transition graphs, which corresponds to item 1. The clause 2(b) is also self-explanatory: the number of increments is at least as big as the size of the hint. The first clause 2(a) is more involved. It says that the run is suboptimal in a sense, i.e., it can be converted into one that does fewer increments. We will then show—using 1 and 2(b)—that the run with fewer increments also does a lot of increments.

We now proceed to show that the above definition satisfies the completeness and correctness conditions from the statement of the lemma.

Completeness. Let g' be the strictly increasing function obtained by induction hypothesis for Simon level k. We set $g(K)$ to be

$$\min\left(g'(\sqrt{K}), \frac{\sqrt{K}-2}{|M|+1}\right) \ .$$

Assume now that the hint h is $\leq g(K)$ and that the run ρ satisfies $val(\rho, 1, \infty) \geq K$. We want to show that ρ is ∞-captured.

Let $\rho_i \ldots \rho_j$ be a minimal part of ρ for which

$$val(\rho_i \ldots \rho_j, 1, \infty) \geq K \ .$$

The general idea is very simple. There are two possible cases: either one of the sub-runs $\rho_i, \ldots, \rho_j$ does more than $\sqrt{K}$ increments on counter 1, or there are at least $\sqrt{K}$ subruns among $\rho_i, \ldots, \rho_j$ that increment counter 1. In either case the run ρ will be ∞-captured.

In the first case, we use the induction hypothesis and

$$g'(\sqrt{K}) \geq g(K)$$

to obtain item 1 in the definition of being ∞-captured.

The more difficult case is the second one. We will study the subruns $\rho_{i+1}, \ldots, \rho_{j-1}$ that do not reset counter 1. As in the previous lemma, we consider the set $I \subseteq \{i+1, \ldots, j-1\}$ of those indexes l where ρ_l does at least one increment on counter 1. By our assumption, I contains at least $\sqrt{K} - 2$ indexes (we may have lost 2 because of ρ_i and ρ_j).

A maximal contiguous subset of I is called an *inc-segment*. There are two possible cases. Either there are few (at most $|M|$) inc-segments, in which case one of the inc-segments must perform many increments and the run ρ is ∞-captured by item 2(b), or there are many inc-segments, in which case the run ρ is ∞-captured by item 2(a). The details are spelled out below.

Consider first the case when there are at least $|M| + 1$ inc-segments. In this case, we can find two inc-segments that begin with indexes, respectively, $m' > m > 1$, such that the states q_{m-1} and $q_{m'-1}$ are the same. Since inc-segments are maximal, the index $m-1$ does not belong to I and hence the run ρ_{m-1} does not increment counter 1 and goes from state q_{m-2} to state q_{m-1}. Since the transition graphs G_{m-1} and G_1 are equivalent, there is such a run in G_1, too. Since the run $\rho_{m-1} \cdots \rho_{m'-1}$ goes from q_{m-2} to $q_{m'-1} = q_{m-1}$, we obtain item 2(a) and thus ρ is ∞-captured.

We are left with the case when there are at most $|M| + 1$ inc-segments. Recall that I contains at least $\sqrt{K} - 2$ elements. Since there are at most $|M| + 1$ inc-segments, at least one of the inc-segments must have size at least:

$$\frac{\sqrt{K}-2}{|M|+1} \ .$$

But by our assumption on the hint h, this inc-segment must contain two hinted positions, and thus ρ is ∞-captured by item 2(b).

Correctness. We set $f(K) = K$. Assume that the hint h is $\geq K$ and consider a run ρ such that every run $\pi \equiv_\infty \rho$ is ∞-captured. We need to show that

$$val(\rho, 1, \infty) \geq f(K) = K \ . \tag{5.2}$$

We proceed as in the previous lemma: we first transform the run ρ into an $\equiv_c$-equivalent run π which is more likely to have fewer increments of counter 1. We then show that the run π satisfies property (5.2), which can then also be transferred back to ρ.

As in the previous lemma, there are two stages of the transformation: a local one, and a global one.

The local transformation works just the same as the local transformation in the previous lemma: if we can replace some subrun ρ_i by an equivalent one that is not captured, then we do so.

The global transformation makes sure that 2(a) is no longer satisfied. It works as follows. Assume that item 2(a) is satisfied, and let m and m' be defined accordingly. Since G_1 is idempotent and all the G's are equivalent, the transition graph $G_{m+1} \ldots G_{m'}$ is equivalent to G_1. It follows that we can find a run with neither increments nor resets that can be plugged in place of $\rho_{m+1} \ldots \rho_{m'}$. The new run obtained is $\equiv_c$-equivalent to the original one and the value $val(\rho, 1, \infty)$ is diminished during this process. We iterate this transformation until no more such replacements can be applied (this obviously terminates in fewer iterations than the number of increments of counter 1 in the original run).

Consider now a run π obtained from ρ by first applying the local and then the global transformation. Since $\pi \equiv_c \rho$ holds, our hypothesis says that π is ∞-captured. Since π does not satisfy item 2(a) by construction, it must satisfy either item 1 or item 2(b). In case of item 1, we do the same reasoning as in the previous lemma and use the induction hypothesis to conclude that (5.2) holds. The remaining case is item 2(b), when there are two distinct positions $m < m'$ where each of the runs $\pi_m, \ldots, \pi_{m'}$ all increment counter 1. Since the global transformation process only removed subruns that increment counter 1, this means that all the runs $\rho_m, \ldots, \rho_{m'}$ also increment counter 1. But since the hint was $\geq K$, we have $m' - m \geq K$ and we conclude with the desired (5.2). □

5.6. **Verifying a specification.** In this section, we conclude the proof of the "Moreover.." part of Proposition 5.2 (and therefore also the proof of Theorem 5.1). Recall that in the previous section, we did the proof only for atomic specifications, which talk about a single event. The purpose of this section is to generalize those results to all specifications, where positive boolean combinations of atomic specifications are involved.

Given a specification τ, we want to construct a sequence $\bar{T}$-automaton $\mathcal{A}_\tau$ that verifies if a sequence of words $v_1, v_2, \ldots$ satisfies:

(*) For some $\bar{T}$-function f, in every word v_j every run satisfies τ under $f(j)$.

We will actually prove the above result in a slightly more general form, where the specification τ can be a *generalized specification*. The generalization is twofold.

First, a generalized specification can describe runs in arbitrary transition graphs, and not just those that describe runs of a counter automaton. This is a generalization since transitions in a transition graph contain not only the counter actions, but also some additional labels (recall that labels in a transition graph are of the form $\text{Act} \times \mathcal{L}$).

Second, in a generalized specification, atomic specifications of the form "the run has type t" can be replaced by more powerful atomic specifications of the form "the run, when treated as a sequence of labels in $\text{Act} \times \mathcal{L}$, belongs to a regular language $L \subseteq (\text{Act} \times \mathcal{L})^*$".

This more general form will be convenient in the induction proof.

In our construction we will speak quantitatively of runs of counter automata. Consider a sequence counter automaton (we do not specify its acceptance condition) and a *run* ρ of

this automaton over a finite word u. The only requirement on this run is that it starts in an initial state, ends in a final state, and is consistent with the transition function. Given such a run ρ and a natural number K, we say that it is $(\geq K)$-*accepting* (respectively, $(\leq K)$-*accepting*) if for each counter, any two distinct resets of this counter are separated by at least K increments of it (respectively, at most K). The link with the acceptance condition of B and S-sequence automata is obvious: a run sequence $\rho_1, \rho_2, \ldots$ is accepting for a B-automaton if there exists a natural number K such that every run ρ_i is $(\leq K)$-accepting. Similarly, this run sequence is accepting for an S-automaton if there exists a strongly unbounded function f such that every run ρ_i is $(\geq f(i))$-accepting.

The heart of this section is the following lemma, which is established by repeated use of Lemma 5.22.

Lemma 5.25. *Given a set of states M and a generalized specification τ over M, there exist two strongly unbounded functions f, g, and a counter automaton $\mathcal{A}_\tau$ that reads finite M-transition graphs, such that for any M-transition graph G,*

- *(correctness) if there is a $(\preceq K)$-accepting run ρ of $\mathcal{A}_\tau$ over G then all runs over G satisfy τ under $f(K)$, and;*
- *(completeness) if all runs in G satisfy τ under K then there is a $(\preceq g(K))$-accepting run ρ of $\mathcal{A}_\tau$ over G.*

In other words, acceptance by $\mathcal{A}_\tau$ and being captured by the specification are asymptotically the same thing. Before establishing this lemma, we show how it completes the proof. We only do the case of $T = S$, the other case is done in a similar manner.

We want to verify if a sequence of words $v_1, v_2, \ldots$ satisfies the property (*). Since a B-function is essentially a constant K and $\preceq$ is $\leq$, this boils down to verifying that there is some K such that all runs in $v_1, v_2, \ldots$ satisfy τ under $\leq K$. We take the automaton $\mathcal{A}_\tau$ from Lemma 5.25 and set the acceptance condition so that all of its counters are bounded. The automaton $\mathcal{A}_\tau$ works over transition graphs, while property (*) talks about input words for the complemented automaton $\mathcal{A}$, but this is not a problem: the automaton $\mathcal{A}_\tau$ can be modified so that it treats an input letter as the appropriate transition graph, taken from the transition function of $\mathcal{A}$. We claim that this is the desired automaton for property (*). Indeed, if the automaton $\mathcal{A}_\tau$ accepts a sequence $v_1, v_2 \ldots$ then its counters never exceed some value K. But then by Lemma 5.25, all runs of $\mathcal{A}$ in all words v_j satisfy τ under $\leq g(K)$. Conversely, if all runs of $\mathcal{A}$ in all words v_j satisfy τ under $\leq K$, then by Lemma 5.25, the automaton $\mathcal{A}_\tau$ has an accepting run where the counters never exceed the value $f(K)$.

Proof of Lemma 5.25. Let us fix a generalized specification τ, for which we want to construct the automaton $\mathcal{A}_\tau$ of Lemma 5.25. The proof is by induction on the number of pairs (α, c) such that the atomic specification $val(\rho, 1, c) \preceq K$ appears in τ. If there are no such pairs, satisfying τ under K does not depend anymore on K and can be checked by a standard finite automaton, without counters. Up to a renumbering of counters, we assume that the minimum counter appearing in the generalized specification is 1, and we set c to be such that $val(\rho, 1, c) \preceq K$ appears is τ. Our objective is to get rid of all occurrences of $val(\rho, 1, c) \preceq K$ in τ.

Given as input the M-transition graph G, the automaton $\mathcal{A}_\tau$ we are constructing works as follows. To aid reading, we present $\mathcal{A}_\tau$ as a cascade of nondeterministic automata, which is implemented using the standard Cartesian product construction.

(1) First, $\mathcal{A}_\tau$ guesses a Simon decomposition of the transition graph G, as well as a hint h over this decomposition (the intention is that the hint h is $\preceq K$, this will be verified in the next step). Note that neither the composition nor the hint need to be unique. The automaton $\mathcal{A}_\tau$ produces the relabeled transition graph (G, h), which is used as input of the next steps.

(2) In this step, the automaton $\mathcal{A}_\tau$ checks that the hint is $\preceq K$. This step requires (hierarchical) counters—the automaton follows the structure of the decomposition and uses a counter for each level to verify that the number of hinted positions is consistent with $\preceq K$.

(3) The automaton $\mathcal{A}_\tau$ accepts the graph G if the M-transition graph (G, h) is accepted by $\mathcal{A}_{\tau\uparrow}$ in which $\tau\uparrow$ is a new generalized specification constructed from τ as follows:

   (a) The regular languages in the atomic specifications are adapted to the new larger transition alphabet (which is $\mathcal{L}_S$ instead of $\mathcal{L}$). In other words, every atomic specification of the kind: "the labeling of the run belongs to a regular language L" is replaced by: "if the additional coordinates from $\mathcal{L}_S$ are removed, the labeling of the run belongs to L".

   (b) Every atomic specification $val(\rho, 1, c) \preceq K$ is replaced by an atomic specification "the labeling of the run belongs to L_c", with L_c the regular language from Lemma 5.22. Note that this way we remove all atomic specifications that involve $val(\rho, 1, c)$, and therefore the induction assumption can be applied to the generalized specification $\tau\uparrow$.

We now proceed to show that this automaton $\mathcal{A}_\tau$ satisfies the statement of Lemma 5.25. This proof is by an induction parallel to the induction used in constructing $\mathcal{A}_\tau$.

Completeness. Let g_1 be the strongly unbounded function obtained from the completeness clause in Lemma 5.22, as applied to the event $(1, c)$ that we are eliminating. By applying the induction assumption to the smaller specification $\tau\uparrow$, but a larger set of transition labels, we know there is some strongly unbounded function g_2 such that if all runs in a graph (G, h) satisfy $\tau\uparrow$ under K then there is a $(\preceq g_2(K))$-accepting run of $\mathcal{A}_{\tau\uparrow}$ over (G, h).

Let g be the coordinate-wise maximum (with respect to $\preceq$) of the functions g_1, g_2; this function is clearly strongly unbounded. We claim that the completeness clause of Lemma 5.25 holds for the function g. Indeed, let G be a graph where all runs satisfy τ under K. We need to show that there is a run of $\mathcal{A}_\tau$ that is $(\preceq g(K))$-accepting. The hint h guessed in step (2) of the construction is chosen so that every two hinted positions are separated by exactly $g_1(K)$ non-hinted positions at the same level. Since g is greater than g_1 under $\succeq$, step (2) can be done in a run that is $(\preceq g(K))$-accepting. Thanks to the assumption on g_1 and the definition of $\tau\uparrow$, if a run ρ over a transition graph G satisfies τ under K, then the run (ρ, h) satisfies the specification $\rho_{\tau\uparrow}$ under K. In particular, by the assumption that every run ρ over G satisfies τ under K, we can use the completeness for $\mathcal{A}_{\tau\uparrow}$ to infer that $\mathcal{A}_{\tau\uparrow}$ has a run over (G, h) that is $(\preceq g_2(K))$-accepting, which gives an accepting run of $\mathcal{A}_\tau$.

Correctness. The correctness proof essentially follows the same scheme, but requires more care, because of the stronger assumptions in the correctness clause of Lemma 5.22. Let G be an M-transition graph and let K be a natural number such that there is a $(\preceq K)$-accepting run of $\mathcal{A}_\tau$ over G. This means that the hint h from step (2) is $\preceq K$, and that (G, h) is $(\preceq K)$-accepted by $\mathcal{A}_{\tau\uparrow}$.

By induction hypothesis, every run in (G, h) satisfies $\tau\uparrow$ under $f(K)$, for some strongly unbounded function f. In particular, for every run ρ over G, (ρ, h) satisfies $\tau\uparrow$ under $f(K)$.

Let ρ be a run over G. We will prove that ρ satisfies τ under $f(K)$. Two cases can happen.

- For all $\pi \equiv_c \rho$, the run (π, h) belongs to L_c. Then by Lemma 5.22, we get $val(\rho, 1, c) \preceq f(K)$. Since the boolean combination in τ is positive, this means that ρ satisfies τ if it satisfies the specification τ' obtained from τ by replacing each occurrence of $val(\rho, 1, c) \preceq f(K)$ with *true*. However, by our assumption, the run (ρ, h) satisfies the specification $\tau\uparrow$ under $f(K)$, so also ρ satisfies τ' under $f(K)$.

- Otherwise, for some $\pi \equiv_c \rho$, the run (π, h) does not belong to L_c. Since all runs (ρ, h) in (G, h) satisfy $\tau\uparrow$ under $f(K)$, this run π must satisfy the generalized specification τ' obtained from τ by replacing every occurrence of $val(\rho, 1, c) \preceq K$ with *false*. But a property of the $\equiv_c$-equivalence is that no atomic specification different from $val(\rho, 1, c) \preceq K$ can see the difference between two $\equiv_c$-equivalent runs. In particular, ρ also satisfies τ' under $f(K)$. By consequence, ρ satisfies τ under $f(K)$. □

6. Future work

We conclude the paper with some open questions.

The first set of questions concerns our proofs. In our proof of Theorem 3.3, the translation from non-hierarchical automata to hierarchical ones is very costly, in particular it uses ωBS-regular expressions as an intermediate step. Is there a better and more direct construction? Second, our complementation proof is very complicated. In particular, our construction is non-elementary (it is elementary if the number of counters is fixed). It seems that a more efficient construction is possible since the, admittedly simpler, but certainly related, limitedness problem for nested distance desert automata is in PSPACE [10].

The second set of questions concerns the model presented in this paper. We provide here a raw list of such questions. Are the ωBS-automata (resp. ωB, resp. ωS) equivalent to their deterministic form? (We expect a negative answer.) Is there a natural form of deterministic automata capturing ωBS-regularity (in the same way deterministic parity automata describe all ω-regular languages)? Are ωBS-automata equivalent to their alternating form? (We expect a positive answer, at least for the class of ωB and ωS-automata.) Does the number of counters induce a strict hierarchy of languages? (We expect a positive answer.) Similarly, does the nesting of B-exponents (resp. S-exponents) induce a strict hierarchy of languages? (We conjecture a positive answer.) Is there an algebraic model for ωBS-regular languages (resp. ωB, ωS-regular languages), maybe extending ω-semigroups? Other questions concern decidability. Is it decidable if an ωBS-regular language is ω-regular (resp. ωB-regular, resp. ωS-regular)? (We think that, at least, it is possible, given an ωB or ωS-regular language, to decide whether it is ω-regular or not.) Are the hierarchies concerning the number of counters, and the number of nesting of exponents decidable?

Other paths of research concern the possible extensions of the model. As we have defined them, ωBS-regular languages are not closed under complementation. Can we find a larger class that is? What are the appropriate automata? Such an extension would lead to the decidability of the full logic MSOLB. Last, but not least, is it possible to extend our results to trees?

References

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

[2] A. Bouquet, O. Serre, and I. Walukiewicz. Pushdown games with unboundedness and regular conditions. In *Foundations of Software Technology and Theoretical Computer Science*, volume 2914 of *Lecture Notes in Computer Science*, pages 88–99, 2003.

[3] J. R. Büchi. Weak second-order arithmetic and finite automata. *Z. Math. Logik Grundl. Math.*, 6:66–92, 1960.

[4] J. R. Büchi. On a decision method in restricted second-order arithmetic. In *Proc. 1960 Int. Congr. for Logic, Methodology and Philosophy of Science*, pages 1–11, 1962.

[5] T. Colcombet. Factorisation forests for infinite words, application to countable scattered linear orderings. In *FCT*, pages 226–237, 2007.

[6] T. Colcombet and C. Löding. Transforming structures by set interpretations. *Logical Methods in Computer Science*, 3(2), 2007.

[7] L.C. Eggan. Transition graphs and the star height of regular events. *Michigan Math. J.*, 10:385–397, 1963.

[8] K. Hashiguchi. Algorithms for determining relative star height and star height. *Inf. Comput.*, 78(2):124–169, 1988.

[9] B. R. Hodgson. Décidabilité par automate fini. *Ann. Sci. Math. Québec*, 7(3):39–57, 1983.

[10] D. Kirsten. Distance desert automata and the star height problem. *RAIRO*, 3(39):455–509, 2005.

[11] F. Klaedtke and H. Ruess. Parikh automata and monadic second–order logics with linear cardinality constraints. Technical Report 177, Institute of Computer Science at Freiburg University, 2002.

[12] M. O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the AMS*, 141:1–23, 1969.

[13] I. Simon. Factorization forests of finite height. *Theoretical Computer Science*, 72:65 – 94, 1990.

[14] W. Thomas. Languages, automata, and logic. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Language Theory*, volume III, pages 389–455. Springer, 1997.

[15] W. Thomas. Complementation of Büchi automata revisited. In *Jewels are forever. Contributions on Theoretical Computer Science in Honor of Arto Saalomaa*, pages 109 – 120. Springer, 1999.