Beyond ωBS-regular languages: The class of ωT-regular languages

Dario Della Monica[1], Angelo Montanari[2], and Pietro Sala[3]

[1] ICE-TCS, School of Computer Science
Reykjavik University, Reykjavik, Iceland dariodm@ru.is
[2] Department of Mathematics and Computer Science University of Udine, Udine, Italy
angelo.montanari@uniud.it
[3] Department of Computer Science
University of Verona, Verona, Italy pietro.sala@univr.it

Abstract. In the last years, various meaningful extensions of ω-regular languages have been proposed in the literature, including ωB-regular languages (ω-regular languages extended with boundedness), ωS-regular languages (ω-regular languages extended with strict unboundedness), and ωBS-regular languages, which are obtained from the combination of ωB- and ωS-regular languages. However, while its components satisfy a generalized closure property, namely, the complement of an ωB-regular (resp., ωS-regular) language is an ωS-regular (resp., ωB-regular) one, the class of ωBS-regular languages is not closed under complementation. The existence of non-ωBS-regular languages that are the complements of some ωBS-regular ones and express fairly natural properties of reactive systems motivates the search for larger, well-behaved classes of extended ω-regular languages. In this paper, we introduce the class of ωT-regular languages, that captures meaningful languages not belonging to the class of ωBS-regular languages. We provide an automaton-based encoding of this new class of languages and we prove the decidability of their emptiness problem.

1 Introduction

Regular languages of infinite words (*ω-regular languages*) have a fundamental role in computer science as they provide a natural setting for specification and verification of nonterminating finite-state systems. Since the seminal work by Büchi [5], McNaughton [8], and by Elgot and Rabin [6] in the sixties, a great research effort has been devoted to the development of the theory and the applications of ω-regular languages. In particular, equivalent characterizations of ω-regular languages have been given in terms of formal languages (*ω-regular expressions*), automata (Büchi, Rabin, and Muller automata), classical logic (weak/strong monadic second-order logic of one successor, WS1S/S1S for short), and temporal logic (quantified linear temporal logic, extended temporal logic).

Recent work by (among others) Bojańczyk and Colcombet has shown that ω-regular languages can be successfully extended in various ways, preserving their decidability and some of their closure properties [2,3,4]. As an example, extended ω-regular languages make it possible to constrain the distance between consecutive occurrences of a given symbol to be (un)bounded. Properties of this kind are interesting in the specification of reactive systems, as argued in [1], where the authors introduce and study *finitary fairness* as opposed to the classic notion of *fairness*, widely used in automated verification of concurrent systems. According to the latter, no individual process in a multi-process system is ignored for ever; finitary fairness imposes the stronger constraint that every enabled transition is executed within at most b time-units, where b is an unknown, constant bound. In [1] it is shown that such a notion enjoys some desirable mathematical properties that are violated by the weaker notion of fairness, and yet it captures all reasonable schedulers' implementations. An analogous property has been studied from a logical perspective in [7], where the logic PROMPT-LTL has been introduced. Roughly speaking, PROMPT-LTL extends LTL with the *prompt-eventually* operator, which states that an event will happen within the next b time-units, b being a constant bound.

From the point of view of formal languages, the proposed extensions pair the Kleene star $(.)^*$ with bounding/unbounding variants of it. Intuitively, the bounding exponent $(.)^B$ constrains parts of the input word to be of bounded size, while the unbounding exponent $(.)^S$ forces parts of the input word to be arbitrarily large. The two extensions have been studied both in isolation (ωB- and ωS-regular expressions) and in conjunction (ωBS-regular expressions). Equivalent characterizations of extended ω-regular languages have been given in terms of automata (ωBS-automata) and classical logic (extensions of S1S with an unbounding quantifier that allows one to express properties which are satisfied by arbitrarily large sets). In [4], the authors show that the complement of an ωB-regular language is an ωS-regular one and vice versa; moreover, they show that ωBS-regular languages, featuring both B- and S-constructors, strictly extend ωB- and ωS-regular languages and they are not closed under complementation.

In this paper, we focus our attention on those ω-languages which are complements of ωBS-regular ones, but do not belong to the class of ωBS-regular languages. Our ultimate goal is to provide a characterization of the class of these languages. We will start with an in-depth analysis of a paradigmatic example of the complement of an ωBS-regular language that lies outside the class of ωBS-regular languages [4]. It will allow us to identify a meaningful extension of ω-regular languages, which includes such a language and which is obtained by adding a new, fairly natural constructor $(.)^T$ to the standard constructors of ω-regular expressions. Decidability of the emptiness problem for this class of ω-languages, called ωT-regular languages, will be proved using an automata-theoretic argument: we introduce a new class of automata, called counter-queue automata, and we show that their emptiness problem is decidable; then, we provide an encoding of ωT-regular expressions into counter-queue automata, that allows us to reduce the emptiness problem for the former to the one for the latter.

The rest of the paper is organized as follows. In Section 2, we summarize existing extensions of ω-regular languages, with a special attention to ωBS-

regular ones, and we introduce the class of ωT-regular languages. In Section 3, we formally define counter-queue automata (CQ automata, for short) and we prove that their emptiness problem is decidable. Finally, in Section 4, we provide the encoding of ωT-regular languages into CQ automata. Conclusions give a short assessment of the work done and illustrate future research directions.

2 Extensions of ω-regular languages

In this section, we first provide a short account of the extensions of ω-regular languages proposed in the literature (details can be found in [2,3,4]) and then we outline a new meaningful one. To begin with, we observe that a word belonging to an ω-regular language (*ω-regular word*) can be seen as the concatenation of a finite prefix, belonging to a regular language, and an infinite sequence of finite words, which we refer to as *ω-iterations*, belonging to another regular language. ω-regular languages can be specified as ω-regular expressions. One interesting case is that of ω-iterations consisting of a finite sequence of words, generated by an occurrence of the Kleene star operator $(.)^*$, aka *$*$-constructor*, in the scope of the *ω-constructor* $(.)^\omega$. As an example, the ω-regular expression $(a^*b)^\omega$ generates the language of all and only those ω-words featuring an infinite sequence of ω-iterations consisting of a finite (possibly empty) sequence of a's followed by exactly one b. Given an ω-regular expression E featuring an occurrence of the $*$-constructor (sub-expression R^*) in the scope of the ω-constructor and an ω-word w belonging to the language of E, we refer to the sequence of the sizes of the (maximal) blocks of consecutive iterations of R in the different ω-iterations as the *(sequence of) exponents of R in (the ω-iterations of) w*. As an example, let us consider the ω-word $w = abaabaaabaaaab\ldots$, generated by the above ω-regular expression $(a^*b)^\omega$. The sequence of exponents of a in w is $1, 2, 3, 4, \ldots$. Sometimes, we will denote words in a compact way, by explicitly indicating the exponents of a sub-expression, e.g., we will write w as $a^1ba^2ba^3ba^4b\ldots$. Given an expression E, we will denote by $\mathcal{L}(E)$ the language defined by E. With a little abuse of notation, we will sometimes identify a language with the expression defining it, and vice versa, so, for instance, we will simply write "the ω-regular language $L = (a^*b)^\omega$" for $\mathcal{L}((a^*b)^\omega)$. It is worth pointing out that the Kleene star operator allows one to impose the existence of a finite sequence of words (described by its argument expression) within each ω-iteration, but it cannot be used to express properties on the sequence of exponents of its argument expression in the ω-iterations of an ω-word. Aiming at overcoming such a limitation, some meaningful extensions of ω-regular expressions have been investigated in the last years, that make it possible to constrain the behavior of the Kleene star operator in the limit.

2.1 Beyond ω-regularity

A first class of extended ω-regular languages is that of ωB-regular languages, which allow one to impose boundedness conditions. ωB-regular expressions are obtained from ω-regular ones by adding a variant of Kleene star $(.)^*$, called

B-*constructor* and denoted by $(.)^B$, to be used in the scope of the ω-constructor $(.)^\omega$. The bounded exponent B allows one to constrain the argument R of the expression R^B to be repeated in each ω-iteration a number of times less than a given bound fixed for the whole ω-word. As an example, the expression $(a^B b)^\omega$ denotes the language of ω-regular words in $(a^* b)^\omega$ for which there is an upper bound on the number of consecutive occurrences of a (the sequence of exponents of a is bounded). As the bound may vary from word to word, the language is not ω-regular. The class of ωS-regular languages extends that of ω-regular ones with strong unboundedness. By analogy with ωB-regular expressions, ωS-regular expressions are obtained from ω-regular ones by adding a variant of Kleene star $(.)^*$, called S-*constructor* and denoted by $(.)^S$, to be used in the scope of the ω-constructor $(.)^\omega$. For every ωS-regular expression containing the sub-expression R^S and for each natural number $k > 0$, the strictly unbounded exponent S constrains the number of ω-iterations in which the argument R is repeated exactly k times to be finite. Let us consider ω-regular words that feature an infinite number of instantiations of the expression R^S, that is, ω-regular words for which there exists an infinite number of ω-iterations including a sequence of consecutive R's generated by R^S. It can be easily checked that in these words the sequence of exponents of R tends towards infinity. As an example, the expression $(a^S b)^\omega$ denotes the language of ω-regular words w in $(a^* b)^\omega$ such that, for any natural number $k > 0$, there exists a suffix of w that only features maximal sequences of consecutive a's that are longer than k.

ωBS-regular expressions are built by making use of the operators of ω-regular expressions and of both the B- and the S-constructor. In [4], the authors show that the class of ωBS-regular languages strictly includes the classes of ωB- and ωS-regular languages, as witnessed by the ωBS-regular language $L = (a^B b + a^S b)^\omega$, which is neither ωB- nor ωS-regular[4]. Moreover, they prove that the class of ωBS-regular languages is not closed under complementation. A counter-example is given precisely by the ωBS-regular language L, whose complement is not ωBS-regular (notice that ωBS-regular languages whose complement is not an ωBS-regular language are neither ωB- nor ωS-regular languages, as the complement of an ωB-regular language is an ωS-regular one and vice versa).

In this paper, we investigate those ω-languages that do not belong to the class of ωBS-regular languages, but whose complement belongs to this class. To have some insights into these languages, let us consider the complement $\overline{L}$ of the language L above. On the one hand, it can be checked that any ω-word w in $\overline{L}$ that features an infinite number of occurrences of b must feature an infinite sequence of blocks of consecutive a's (between two consecutive b's) of unbounded size; otherwise, w would belong to L, as it would be captured by the sub-expression $a^B b$. On the other hand, for any such ω-word w, there must be a natural number $k > 0$ such that there exist infinitely many maximal blocks

[4] It must be noticed that the constructor $+$ occurring in L must not be thought of as performing the union of two languages, but rather as a "shuffling operator" that mixes ω-iterations belonging to the two different (sub-)languages. This will be made clear later on, when we will formally define the languages we deal with.

of consecutive a's whose size is exactly k; otherwise, w would belong to L, as it would be captured by the sub-expression $a^S b$. Thus, w is such that (i) for every natural number k, there exists $k' > k$ that occurs in the sequence of exponents of a in w, and (ii) there exists at least one natural number $k > 0$ that occurs infinitely often in the sequence of exponents of a in w. In fact, as an effect of the combined use of both B- and S-constructors, w is subject to an even stronger constraint: there exist infinitely many natural numbers that occur infinitely often in the sequence of exponents of a in w (notice that this latter constraint implies both the former ones). By way of contradiction, suppose that there are only finitely many natural numbers (exponents) that occur infinitely often. Let k be the largest one. Now, the ω-word w can be viewed as an infinite sequence of ω-iterations, each of them characterised by the corresponding exponent of a. If the exponent associated with an ω-iteration is greater than k, then it does not occur infinitely often, and thus the ω-iteration is captured by the sub-expression $a^S b$. Otherwise, if the exponent is not greater than k, then the corresponding ω-iteration is captured by the sub-expression $a^B b$. As an example, the word $a^1ba^2ba^1ba^3ba^1ba^4b\ldots$ does not belong to $\overline{L}$ as 1 is the only exponent occurring infinitely often. The word $a^1ba^2ba^1ba^2ba^3ba^1ba^2ba^3ba^4b\ldots$, on the other hand, *does* belong to $\overline{L}$ as infinitely many (actually all) natural numbers occur infinitely often in the sequence of exponents.

In the following, we focus our attention on ω-words featuring infinitely many exponents occurring infinitely often. More precisely, we introduce a new variant of the Kleene star operator $(.)^*$, called T-constructor and denoted by $(.)^T$, to be used in the scope of the ω-constructor $(.)^\omega$, and we define the corresponding class of extended ω-regular languages (ωT-regular languages). An expression R^T occurring in some ω-expression E forces two conditions on the ω-words belonging to E: (i) every exponent of R occurs infinitely often in the sequence, and (ii) the sequence features an infinite number of distinct exponents. As an example, it can be easily checked that the language $\overline{L}$ can be defined as $((a^*b)^*a^Tb)^\omega + (a^*b^*)^*a^\omega$, and thus it belongs to the class of ωT-regular languages. In the next two sections, we first provide a formal account of ωBS-regular languages [4] and then we define ωT-regular ones.

2.2 ωBS-regular languages

The class of ωBS*-regular languages* is the class of languages defined by ωBS-regular expressions. These latter are built on top of BS-regular expressions, just as ω-regular expressions are built on top of regular ones. Let Σ be a finite, nonempty alphabet. A BS*-regular expression* over Σ is defined by the following grammar [4]:

$$e \ ::= \ \varnothing \ | \ a \ | \ e \cdot e \ | \ e + e \ | \ e^* \ | \ e^B \ | \ e^S$$

where a belongs to Σ. We sometimes omit the concatenation operator, thus writing ee instead of $e \cdot e$.

Syntactically, BS-regular expressions differ from standard regular ones for the presence of the two constructors $(.)^B$ and $(.)^S$. Since the latter constrain the behaviour of the sequence of ω-iterations to the limit, it is not possible to simply define the semantics of BS-regular expressions in terms of languages of (finite) words, and then to obtain ωBS-regular languages through infinitely many, unrelated iterations of such words. In the following, we specify the semantics of BS-regular expressions in terms of languages of infinite sequences of words; suitable constraints are then imposed to force these sequences to satisfy some properties expressing the intended meaning of the B- and S-constructors.

Let $\boldsymbol{u}$ be an infinite sequence of words over Σ and let u_i be the i-th element of $\boldsymbol{u}$. Moreover, let $f : \mathbb{N} \to \mathbb{N}$ with $f(0) = 1$. The semantics of BS-regular expressions over Σ is defined as follows:

- $\mathcal{L}(\varnothing) = \varnothing$;
- for $a \in \Sigma$, $\mathcal{L}(a)$ is the infinite sequence of the one-letter word a $\{(a, a, a, \ldots)\}$;
- $\mathcal{L}(e_1 \cdot e_2) = \{\boldsymbol{w} \mid \forall i.w_i = u_i \cdot v_i,\ \boldsymbol{u} \in \mathcal{L}(e_1),\ \boldsymbol{v} \in \mathcal{L}(e_2)\}$;
- $\mathcal{L}(e_1 + e_2) = \{\boldsymbol{w} \mid \forall i.w_i \in \{u_i, v_i\},\ \boldsymbol{u}, \boldsymbol{v} \in \mathcal{L}(e_1) \cup \mathcal{L}(e_2)\}$[5];
- $\mathcal{L}(e^*) = \{(u_{f(0)}u_2 \ldots u_{f(1)-1}, u_{f(1)} \ldots u_{f(2)-1}, \ldots) \mid$
  $\boldsymbol{u} \in \mathcal{L}(e)$ and f is an unbounded and nondecreasing function$\}$;
- $\mathcal{L}(e^B) = \{(u_{f(0)}u_2 \ldots u_{f(1)-1}, u_{f(1)} \ldots u_{f(2)-1}, \ldots) \mid$
  $\boldsymbol{u} \in \mathcal{L}(e)$ and f is an unbounded and nondecreasing function such that $\exists n \in \mathbb{N}\ \forall i.(f(i+1) - f(i) < n)\}$;
- $\mathcal{L}(e^S) = \{(u_{f(0)}u_2 \ldots u_{f(1)-1}, u_{f(1)} \ldots u_{f(2)-1}, \ldots) \mid$
  $\boldsymbol{u} \in \mathcal{L}(e)$ and f is an unbounded and nondecreasing function such that $\forall n \in \mathbb{N}\ \exists k\ \forall i > k.(f(i+1) - f(i) > n)\}$.

Given a sequence $\boldsymbol{u} = (u_{f(0)}u_2 \ldots u_{f(1)-1}, u_{f(1)} \ldots u_{f(2)-1}, \ldots) \in e^{op}$, where $op \in \{*, B, T\}$, we define the *sequence of exponents of e in $\boldsymbol{u}$* as the sequence $\{f(i+1) - f(i)\}_{i \in \mathbb{N}}$. While the $*$-constructor does not impose any constraint on the sequence of exponents of its operand, the B-constructor forces the sequence of exponents to be bounded, while the S-constructor forces it to be strictly unbounded, that is, its limit tends towards infinity (equivalently, the S-constructor imposes that no exponent occurs infinitely many times in the sequence).

The *ω-constructor* defines languages of infinite words from languages of infinite sequences of words. Given a BS-regular expression e, the semantics of the ω-constructor is defined as follows:

- $\mathcal{L}(e^\omega) = \{w \mid w = u_1 u_2 u_3 \ldots \text{ for some } \boldsymbol{u} \in \mathcal{L}(e)\}$.

ωBS-expressions are defined by the following grammar (we denote languages of word sequences by lowercase letters, such as e, e_1, ..., and languages of words by uppercase ones, such as E, E_1, ..., R, R_1, ...):

$$E \ ::= \ E + E \ \mid \ R \cdot E \ \mid \ e^\omega$$

where R is a regular expression, e is a BS-regular expression, and the operators $+$ and $\cdot$ respectively denote union and concatenation of word languages (formally, $\mathcal{L}(E_1 + E_2) = \mathcal{L}(E_1) \cup \mathcal{L}(E_2)$ and $\mathcal{L}(E_1 \cdot E_2) = \{u \cdot v \mid u \in \mathcal{L}(E_1), v \in \mathcal{L}(E_2)\}$)[6].

---

[5] Unlike the case of word languages, when applied to languages of word sequences, the operator $+$ does not return the union of the two argument languages. As an example, $\mathcal{L}(a) \cup \mathcal{L}(b) \subsetneq \mathcal{L}(a + b)$, as witnessed by the word sequence $(a, b, a, b, a, b, \ldots)$.

[6] Notice the abuse of notation with the previous definition of the operators $+$ and $\cdot$ over languages of infinite word sequences.

Similarly to what we did with the concatenation of languages of word sequences, we will sometimes omit the concatenation operator between word languages.

2.3 ωT-regular languages

As we have already recalled, the class of ωBS-regular languages is not closed under complementation, that is, there are ω-languages, that are the complements of ωBS-regular ones, which are not ωBS-regular. This is the case, for instance, with the language $\bar{L}$, which is the complement of the ωBS-regular language $L = (a^B b + a^S b)^\omega$ (see Subsection 2.1).

In Subsection 2.1, we studied in some detail the distinctive features of the language $\bar{L}$ and we showed that ω-words belonging to it are, to a certain extent, characterised by sequences of exponents where infinitely many exponents occur infinitely often. In order to capture extended ω-regular languages that satisfy such a property, we now introduce a new class of ω-regular languages, called *ωT-regular languages*, that includes all and only those languages that can be expressed by *ωT-regular expressions*, which are defined by the following grammar:

$$T ::= T + T \mid R \cdot T \mid t^\omega$$
$$t ::= \varnothing \mid a \mid t \cdot t \mid t + t \mid t^* \mid t^T$$

where R is a regular expression and $a \in \Sigma$.

The sub-grammar rooted in the non-terminal t generates the *T-regular expressions*. The only new ingredient in the above definition is the T-constructor $(.)^T$, that, given a language of word sequences t, defines the following language:

– $\mathcal{L}(t^T) = \{(u_{f(0)} u_2 \dots u_{f(1)-1}, u_{f(1)} \dots u_{f(2)-1}, \dots) \mid$
 $\boldsymbol{u} \in \mathcal{L}(t)$ and f is an unbounded and nondecreasing function such that
 (i) $\forall n \exists i . f(i+1) - f(i) > n$
 (ii) $\forall n . [\text{if } \exists i . f(i+1) - f(i) = n, \text{ then } \forall k \exists j > k . f(j+1) - f(j) = n]\}$.

It is not difficult to convince oneself that such a formal definition of the semantics of the T-constructor conforms with the intuitive one we provided in Subsection 2.1: item (i) guarantees the existence of infinitely many exponents in the sequence and item (ii) forces each exponent (occurring at least once) to occur infinitely many times in the sequence of exponents (of words).

3 Counter-queue automata

In this section, we introduce a new class of automata, called counter-queue automata (CQ automata), and we show that their emptiness problem is decidable.

3.1 The class of CQ automata

To start with, we introduce the notion of a queue (of natural numbers) devoid of repetitions: a *queue* q is a finite word over $\mathbb{N}$ such that all its elements are different. We denote the empty queue by $\varnothing$. Given a queue q, we denote by $q[i]$ the i-th number in q. Moreover, we denote the set of the elements of q and the maximum among them by $Set(q)$ and $\max(q)$, respectively. Formally,

$Set(q) = \{n \in \mathbb{N} : \exists i.q[i] = n\}$ and $\max(q) = \max(Set(q))$ if $Set(q) \neq \emptyset$, -1 otherwise. The first and the last element of q can be selected by means of the usual *front* and *back* operations: $front(q) = q[1]$ and $back(q) = q[|q|]$. The *enqueue* operation satisfies the uniqueness constraint on the elements of q: for every $n \in \mathbb{N}$, $enqueue(q, n) = q \cdot n$ if $n \notin Set(q)$, q otherwise. The *dequeue* operation is defined as usual: $dequeue(q) = q[2] \ldots q[|q|]$. We denote by $\mathcal{Q}$ the set of all queues.

A *counter-queue automaton* (*CQ* automaton) is a quintuple $\mathcal{A} = (S, \Sigma, s_0, N, \Delta)$, where S is a finite set of states, Σ is a finite alphabet, $s_0 \in S$ is the initial state, N is a natural number, and $\Delta \subseteq S \times (\Sigma \cup \{\epsilon\}) \times S \times (\{1, \ldots, N\} \times \{no\_op, inc, check\})$ is a transition relation such that for every $(s, \sigma, s', (k, no\_op)) \in \Delta$, it holds $k = 1$ (see Figure 1). Given a *CQ* automaton $\mathcal{A} = (S, \Sigma, s_0, N, \Delta)$, a *configuration* of $\mathcal{A}$ is a pair $c = (s, C)$, where $s \in S$ and $C \in (\mathbb{N} \times \mathcal{Q})^N$ is a *counter-queue* configuration. For $1 \leqslant i \leqslant N$, we denote by $C[i] = (n_i, q_i)$ the i-th component of a counter-queue configuration C, where n_i and q_i are its counter and queue components, respectively. In the following, we will often refer to n_i as $counter(C[i])$ and to q_i as $queue(C[i])$.

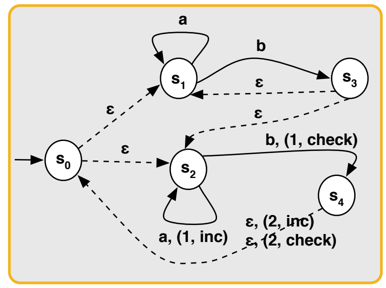

Fig. 1. A *CQ* automaton for the language $((a^*b)^*a^Tb)^\omega$ $(N = 2)$.

Let $\mathcal{A} = (S, \Sigma, s_0, N, \Delta)$. We define a ternary relation $\rightarrow_{\mathcal{A}}$ over pairs of configurations and symbols in $\Sigma \cup \{\epsilon\}$ such that for all pairs of configurations $(s, C), (s', C')$ and $\sigma \in \Sigma \cup \{\epsilon\}$, $(s, C) \rightarrow^{\sigma}_{\mathcal{A}} (s', C')$ iff there exists $\delta = (s, \sigma, s', (k, op)) \in \Delta$ such that $C[k'] = C'[k']$ for all $k' \neq k$, and

- if $op = no\_op$, then $C[k] = C'[k]$;
- if $op = inc$, then $counter(C'[k]) = counter(C[k]) + 1$ and $queue(C'[k]) = queue(C[k])$;
- if $op = check$, then $counter(C'[k]) = 0$; moreover,
  - if $counter(C[k]) = front(queue(C[k]))$, then $queue(C'[k]) = enqueue(dequeue(queue(C[k])), counter(C[k]))$;

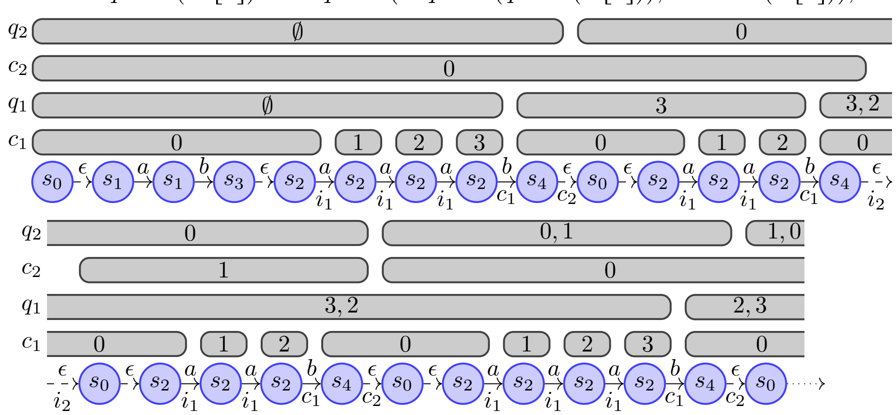

Fig. 2. A prefix of a computation of the automaton in Figure 1. A configuration is characterised by a circle (state) and the rounded-corner rectangles above it (counter-queue configuration). c_i (resp., q_i) is its counter (resp., queue) component.

- if $counter(C[k]) \neq front(queue(C[k]))$, then $queue(C'[k]) = enqueue(queue(C[k]), counter(C[k]))$.

In such a case, we say that $(s, C) \rightarrow^{\sigma}_{\mathcal{A}} (s', C')$ via δ. Let $\rightarrow^{*}_{\mathcal{A}}$ be the reflexive and transitive closure of $\rightarrow^{\sigma}_{\mathcal{A}}$ (where we abstract away symbols in $\Sigma \cup \{\epsilon\}$). The *initial configuration* of $\mathcal{A}$ is the pair (s_0, C_0), where for every $1 \leqslant k \leqslant N$ we have $C_0[k] = (0, \emptyset)$. A *computation* of $\mathcal{A}$ is an infinite sequence of configurations $\mathcal{C} = (s_0, C_0)(s_1, C_1)\ldots$, where $(s_i, C_i) \rightarrow^{\sigma}_{\mathcal{A}} (s_{i+1}, C_{i+1})$, for some $\sigma \in \Sigma \cup \{\epsilon\}$, for all $i \in \mathbb{N}$ (see Figure 2). Given two configurations (s_i, C_i) and (s_j, C_j) in $\mathcal{C}$, with $i \leqslant j$, we say that (s_j, C_j) is *ϵ-reachable* from (s_i, C_i), written $(s_i, C_i) \rightarrow^{*\epsilon}_{\mathcal{A}} (s_j, C_j)$, if for all $i < j' \leqslant j$, $(s_{j'-1}, C_{j'-1}) \rightarrow^{\epsilon}_{\mathcal{A}} (s_{j'}, C_{j'})$. Given a computation $\mathcal{C}$ of $\mathcal{A}$ and an ω-word $w \in \Sigma^{\omega}$, we say that w is a *$\mathcal{C}$-induced word* if there exists an increasing function $f : \mathbb{N} \rightarrow \mathbb{N}$ such that:

– $(s_0, C_0) \rightarrow^{*\epsilon}_{\mathcal{A}} (s_{f(1)}, C_{f(1)})$, and
– for all $i \geqslant 1$, $(s_{f(i)}, C_{f(i)}) \rightarrow^{w[i]}_{\mathcal{A}} (s_{f(i)+1}, C_{f(i)+1}) \rightarrow^{*\epsilon}_{\mathcal{A}} (s_{f(i+1)}, C_{f(i+1)})$.

A computation $\mathcal{C}$ of $\mathcal{A}$ is *accepting* if and only if:

(i) there exists an ω-word w induced by $\mathcal{C}$;
(ii) for all $1 \leqslant k \leqslant N$, $\lim_{i \rightarrow +\infty} |queue(C_i[k])| = +\infty$;
(iii) for all $1 \leqslant k \leqslant N$, $i \geqslant 0$, and $n \in Set(queue(C_i[k]))$, it holds that $|\{i' : back(queue(C_{i'}[k])) = n\}| = +\infty$.

In such a case, we say that w is *accepted* by $\mathcal{A}$. We denote by $\mathcal{L}(\mathcal{A})$ the set of all and only the ω-words $w \in \Sigma^{\omega}$ that are accepted by $\mathcal{A}$, and we say that $\mathcal{A}$ *accepts* the language $\mathcal{L}(\mathcal{A})$. As an example, Figure 1 depicts a CQ automaton with two counters ($N = 2$) for the language $((a^*b)^*a^Tb)^{\omega}$. (Notice that an automaton for the same language with one counter only can be devised.)

3.2 Decidability of the emptiness problem for CQ automata

In this section, we prove that the emptiness problem for CQ automata is decidable by a game-theoretic argument.

W.l.o.g., from now on, we restrict our attention to simple CQ automata. A CQ automaton $\mathcal{A} = (S, \Sigma, s_0, N, \Delta)$ is *simple* iff for each $s \in S$, either $|\{(s, \sigma, s', (k, op)) \in \Delta\}| = 1$ or $op = no\_op$, $k = 1$, and $\sigma = \epsilon$ for all $(s, \sigma, s', (k, op)) \in \Delta$. Basically, a simple CQ automaton has two kinds of state: those in which it can fire exactly one action and those in which it makes a nondeterministic choice. Moreover for every pair of configurations $(s, C), (s', C')$ such that $(s, C) \rightarrow^{\sigma}_{\mathcal{A}} (s', C')$, the transition $\delta \in \Delta$ that has been fired in (s, C) is uniquely determined by s and s'. By exploiting ϵ-transitions and by adding a suitable number of states, every CQ automaton $\mathcal{A}$ may be turned into a simple one $\mathcal{A}'$ such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$.

The set of states of a simple CQ automaton can be partitioned in four subsets: (i) the set of states s from which only one transition of the form $(s, \sigma, s', (k, check))$ can be fired ($check_k$ states); (ii) the set of states s from which only one transition of the form $(s, \sigma, s', (k, inc))$ can be fired (inc_k states); (iii) the set of states s from which only one transition of the form $(s, \sigma, s', (1, no\_op))$, with $\sigma \neq \epsilon$, can be fired (sym states); (iv) the set of states s from which possibly many transitions of the form $(s, \epsilon, s', (1, no\_op))$ can be fired ($choice$ states).

Let $\mathcal{A} = (S, \Sigma, s_0, N, \Delta)$ be a CQ automaton. A *partial computation* of $\mathcal{A}$ is a *finite sequence* $\mathcal{P} = (s_0, C_0) \dots (s_n, C_n)$ such that, for all $0 \leqslant i < n$, $(s_i, C_i) \rightarrow^{\sigma}_{\mathcal{A}} (s_{i+1}, C_{i+1})$, for some $\sigma \in \Sigma \cup \{\epsilon\}$. If (s_0, C_0) is the initial configuration of $\mathcal{A}$, then $\mathcal{P}$ is a *prefix computation* of $\mathcal{A}$. We denote by $Partial_{\mathcal{A}}$ and $Prefixes_{\mathcal{A}}$ the sets of all partial computations of $\mathcal{A}$ and of all prefix computations of $\mathcal{A}$, respectively. Clearly, $Prefixes_{\mathcal{A}} \subseteq Partial_{\mathcal{A}}$ holds. Given a prefix computation $\mathcal{P} = (s_0, C_0) \dots (s_n, C_n)$ and a partial computation $\mathcal{P}' = (s'_0, C'_0) \dots (s'_m, C'_m)$, we say that $\mathcal{P}$ can be extended with $\mathcal{P}'$ iff $\mathcal{P}'' = \mathcal{P} \cdot \mathcal{P}' = (s_0, C_0) \dots (s_n, C_n)(s'_0, C'_0) \dots (s'_m, C'_m)$ is a prefix computation. In such a case, we say that $\mathcal{P}''$ is an extension of $\mathcal{P}$.

Let $\mathcal{A} = (S, \Sigma, s_0, N, \Delta)$ be a CQ automaton and $\mathcal{P} = (s_0, C_0) \dots (s_n, C_n) \in Partial_{\mathcal{A}}$. For all $s \in S$, it holds that if $(s_n, C_n) \rightarrow^{\sigma}_{\mathcal{A}} (s, C)$, for some counter-queue configuration C, then C is uniquely determined by s_n, s, and C_n, that is, there is no $C' \neq C$ such that $(s_n, C_n) \rightarrow^{\sigma}_{\mathcal{A}} (s, C')$. We define the *extension of* $\mathcal{P}$ *with* s, denoted by $\mathcal{P} << s$, as $(s_0, C_0) \dots (s_n, C_n)(s, C)$.

We can associate a two-player game, called *CQ game*, with each CQ automaton $\mathcal{A}$. The configurations of the game are the prefix computations of $\mathcal{A}$. The initial configuration is the shortest prefix computation of $\mathcal{A}$, namely, $\mathcal{P}_0 = (s_0, C_0)$. Let $i \geqslant 0$ be the current turn and $\mathcal{P}_i = (s_0, C_0) \dots (s_n, C_n)$ be the current game configuration. The first player (*Spoiler*) moves by choosing a priority $p_i \in \{check_k, max_k | 1 \leqslant k \leqslant N\} \cup \{sym\}$; the second player (*Duplicator*) replies with a partial computation $\mathcal{Q}_i = (s'_0, C'_0) \dots (s'_m, C'_m)$ such that: (*i*) $\mathcal{P}_i$ can be extended with $\mathcal{Q}_i$; (*ii*) if $p_i = check_k$, for some $1 \leqslant k \leqslant N$, then there exists $0 \leqslant j < m$ such that $front(queue(C'_j[k])) = back(queue(C'_{j+1}[k]))$; (*iii*) if $p_i = max_k$, for some $1 \leqslant k \leqslant N$, then there exists $0 \leqslant j < m$ such that $back(queue(C'_{j+1}[k])) > \max(queue(C'_j[k]))$; (*iv*) if $p_i = sym$, then there exists $0 \leqslant j < m$ such that $(s'_j, \sigma, s'_{j+1}, (k, op)) \in \Delta$, for some pair (k, op) and some $\sigma \neq \epsilon$. A play of a CQ game is a sequence of pairs $\mathcal{P}\ell = (\mathcal{P}_0, p_0)(\mathcal{P}_1, p_1) \dots$, where, at each round $i \geqslant 0$, p_i is Spoiler's move and $\mathcal{P}_{i+1}$ is the result of the extension of $\mathcal{P}_i$ with Duplicator's move $\mathcal{Q}_i$. Let $\mathcal{P}\ell(n)$ be the finite prefix of $\mathcal{P}\ell$ of length n; moreover, let $\mathcal{P}lay_{\mathcal{A}}$ be the set of all possible finite prefixes of all possible plays of the CQ game on $\mathcal{A}$. Duplicator wins a play of the CQ game iff the play is infinite, that is, she is able to reply to Spoiler's move at every round. A *strategy* for Duplicator in the CQ game on $\mathcal{A}$ is a function $str : \mathcal{P}lay_{\mathcal{A}} \rightarrow Partial_{\mathcal{A}}$. In a play $\mathcal{P}\ell = (\mathcal{P}_0, p_0)(\mathcal{P}_1, p_1) \dots$, Duplicator acts according to str if for all $i \geqslant 0$, $\mathcal{P}_{i+1} = \mathcal{P}_i \cdot str(\mathcal{P}\ell(i))$, that is, $\mathcal{P}_{i+1}$ is the result of the extension of $\mathcal{P}_i$ with $str(\mathcal{P}\ell(i))$. A strategy str for Duplicator is winning iff Duplicator wins every play in which she acts according to str. The proof of the following lemma is straightforward and thus omitted.

Lemma 1. *Let $\mathcal{A}$ be a CQ automaton. We have that $\mathcal{L}(\mathcal{A}) \neq \varnothing$ iff there exists a winning strategy for Duplicator in the CQ game on $\mathcal{A}$.*

We now show that the problem of deciding whether there exists a winning strategy for Duplicator in the CQ game on a given CQ automaton $\mathcal{A}$ is decidable. To this end, we introduce the concept of a winning witness. An $\mathbb{N}$-*word* α is a finite word over $\mathbb{N}^+$ such that, for all $1 \leqslant i < |\alpha|$, $\alpha[i] < \alpha[i+1]$ holds. Given an

$\mathbb{N}$-word α, we say that n belongs to α, written $n \in \alpha$, iff there exists i for which $\alpha[i] = n$, and we denote by $Set(\alpha)$ the set $\{n \in \mathbb{N} : n \in \alpha\}$. Clearly, for any given set $S \subset \mathbb{N}^+$, there is exactly one $\mathbb{N}$-word α such that $Set(\alpha) = S$; we denote such a word by α_S. Given two $\mathbb{N}$-words α_1 and α_2, we define $\alpha_1 \cup \alpha_2$ and $\alpha_1 \cap \alpha_2$ as the $\mathbb{N}$-words $\alpha_{Set(\alpha_1) \cup Set(\alpha_2)}$ and $\alpha_{Set(\alpha_1) \cap Set(\alpha_2)}$, respectively. Moreover, we denote by $[b, e]$ the $\mathbb{N}$-word $\alpha_{\{b, b+1, \ldots, e\}}$. Finally, for each $1 \leqslant k \leqslant N$, let $\beta_k^{\mathcal{P}}$ be the $\mathbb{N}$-word $i_1 \ldots i_m$ such that $\{s_{i_1}, \ldots s_{i_m}\}$ is the set of all and only the $check_k$ state in $\mathcal{P}$ and let $\gamma_k^{\mathcal{P}}$ be the $\mathbb{N}$-word $i_1 \ldots i_m$ such that $\{s_{i_1}, \ldots s_{i_m}\}$ is the set of all and only the inc_k state in $\mathcal{P}$. We let $\beta^{\mathcal{P}} = \bigcup_{1 \leqslant k \leqslant N} \beta_k^{\mathcal{P}}$ and $\gamma^{\mathcal{P}} = \bigcup_{1 \leqslant k \leqslant N} \gamma_k^{\mathcal{P}}$.

Definition 1 (winning witness). *Let $\mathcal{P} = (s_0, C_0) \ldots (s_n, C_n) \in Prefixes_{\mathcal{A}}$. $\mathcal{P}$ is a* winning witness *iff there exist $2N + 3$ indexes $0 \leqslant begin < b_1 < e_1 < \ldots < b_N < e_N < limit < end \leqslant n$ such that the following conditions hold:*

- *there is j such that $begin \leqslant j \leqslant end$ and s_j is a sym state;*
- *$s_{begin} = s_{end}$ and, for each $1 \leqslant k \leqslant N$, $s_{b_k} = s_{e_k}$, s_{b_k} is an inc_k state, and, for any $b_k \leqslant j \leqslant e_k$, s_j is not a $check_k$ state;*
- *for each $1 \leqslant k \leqslant N$, there is $e_N < j < limit$ such that s_j is a $check_k$ state;*
- *let $\beta^{\mathcal{P}} \cap [0, limit] = \bar{j}_1 \ldots \bar{j}_M$; then, there are $2M$ indexes $\bar{b}_1 < \bar{e}_1 < \ldots < \bar{b}_M < \bar{e}_M$, with $limit < \bar{b}_1$ and $\bar{e}_M < end$, such that, for each $1 \leqslant i \leqslant M$, there is $1 \leqslant k \leqslant N$ for which $\bar{j}_i \in \beta_k^{\mathcal{P}}$, $[\bar{b}_i, \bar{e}_i] \cap \beta_k^{\mathcal{P}} = \bar{b}_i \bar{e}_i$ (that is, $s_{\bar{b}_i}$ and $s_{\bar{e}_i}$ are $check_k$ states and there are no $check_k$ states in between), and $counter(C_{\bar{j}_i}[k]) = |[\bar{b}_i, \bar{e}_i] \cap \gamma_k^{\mathcal{P}}|$.*

A winning witness can be seen as a finite representation of a winning strategy, as stated by the following lemma, which links the existence of a winning strategy for Duplicator in the CQ game to the existence of a winning witness.

Lemma 2. *Let $\mathcal{A}$ be a CQ automaton. Then, Duplicator has a winning strategy in the CQ game on $\mathcal{A}$ iff $Prefixes_{\mathcal{A}}$ contains a winning witness.*

Proof (sketch—details in the appendix). As for the left-to-right direction, let us assume that there exists a winning strategy for Duplicator. By Lemma 1, it follows that there is an accepting computation $\mathcal{C}$ of $\mathcal{A}$. It is not difficult to show that one can choose the index *end* in $\mathcal{C}$ large enough to guarantee the existence of a sequence of indexes $0 \leqslant begin < b_1 < e_1 < \ldots < b_N < e_N < limit < end \leqslant n$ that satisfies the conditions of Definition 1.

As for the converse implication, let us assume that $Prefixes_{\mathcal{A}}$ contains a winning witness $\mathcal{P} = (s_0, C_0) \ldots (s_n, C_n)$. Let $0 \leqslant begin < b_1 < e_1 < \ldots < b_N < e_N < limit < end \leqslant n$ be the indexes satisfying the conditions of Definition 1. We show how to devise a winning strategy $str_{\mathcal{P}}$ for Duplicator in the CQ game on $\mathcal{A}$. Since the strategy we define is memoryless, i.e., it only depends on the last pair of a finite sequence (play prefix) $\mathcal{P}\ell(m)$, with $m \in \mathbb{N}$, it is enough to define it for a generic configuration $\mathcal{P}$ and Spoiler's move p. Strategy $str_{\mathcal{P}}$ is defined inductively as follows.

(Base case) Let $\mathcal{P}_0$ be the initial game configuration. To all possible moves by Spoiler, Duplicator replies with the partial computation $\mathcal{Q}_0 =$

$(s_1, C_1) \ldots (s_{end}, C_{end})$. It can be easily checked that, independently from Spoiler's move, $\mathcal{Q}_0$ is a correct move for Duplicator.

(Inductive step) Let $\mathcal{P}_i = (s_0, C_0)(s_1, C_1) \ldots (s_{end}, C_{end}) \ldots (s_{n_i}, C_{n_i})$, with $i > 0$, be a generic game configuration. The next move by Duplicator depends on Spoilers's one, p_i (notice that $(s_0, C_0)(s_1, C_1) \ldots (s_{end}, C_{end})$ is a prefix of the winning witness $\mathcal{P}$ as well as of every game configuration $\mathcal{P}_i$ of a play in which Duplicator applies $str_{\mathcal{P}}$): (i) if $p_i = sym$, Duplicator replies with the partial computation $(s_{begin}, C_{begin}) \ldots (s_{end}, C_{end})$; (ii) if $p_i = check_k$, Duplicator replies with the partial computation $(s_{begin}, C_{begin}) \ldots (s_{n_i}, C_{n_i})$; (iii) if $p_i = max_k$, Duplicator replies with the partial computation obtained from $(s_{begin}, C_{begin}) \ldots (s_{end}, C_{end})$, by "pumping" its fragment $(s_{b_k}, C_{b_k}) \ldots (s_{e_k}, C_{e_k})$, that is, by looping over it a suitable number of times so that the k-th counter of the last configuration of the last loop iteration is greater than every element inserted so far in the k-th queue. Notice that the last element (s_{n_i}, C_{n_i}) of every resulting game configuration $\mathcal{P}_i$, with $i > 0$, is such that $s_{n_i} = s_{end}$. It is possible to show that all moves returned by the strategy are valid moves for Duplicator. □

Theorem 1. *The emptiness problem for CQ automata is decidable.*

Proof (sketch—details in the appendix). Thanks to Lemma 2, given a CQ automaton $\mathcal{A}$, it suffices to provide an algorithm that searches $Prefixes_{\mathcal{A}}$ for winning witnesses. Starting from the initial configuration, the algorithm nondeterministically extends prefix computations by guessing, at each step, the next configuration. When a configuration (s_i, C_i) is generated, thus building the prefix computation $(s_0, C_0) \ldots (s_i, C_i)$, the algorithm guesses whether or not i is one of the indexes in $\mathcal{I} = \{begin, b_1, e_1, \ldots, b_N, e_N, limit, end\}$ (see Definition 1). If all those indexes are located, the algorithm returns *true* iff all the conditions of Definition 1 are fulfilled. (The problem of checking the fulfillment of the conditions of Definition 1 with respect to a prefix computation is clearly decidable.) In principle, if one of the indexes in $\mathcal{I}$ has not been reached yet, the search should go on. However, termination is guaranteed as it is possible to show that if there is a winning witness with two consecutive indexes belonging to $\mathcal{I}$ being located too far away from each other (according to some computable bound), then there is a winning witness where the distance between those indexes is shorter (and previously located indexes are unchanged). This gives the following termination condition: if the search for an index in $\mathcal{I}$ fails too many times (according to the aforementioned bound), the algorithm returns *false.* □

4 From ωT-regular languages to CQ automata

In this section, we show how to map an ωT*-regular expression* T into a corresponding CQ automaton $\mathcal{A}$ such that $\mathcal{L}(T) = \mathcal{L}(\mathcal{A})$. We build the automaton $\mathcal{A}$ in a compositional way: for each sub-expression T' of T, starting from the atomic ones, we introduce a set $\mathcal{S}_{T'}$ of CQ automata and then we show how to produce the set of automata for complex sub-expressions by suitably combining automata in the sets associated with their sub-expressions. Eventually, we obtain a set of automata for the ωT-regular expression T. The automaton $\mathcal{A}$ results from the merge of the automata in such a set. W.l.o.g., we assume the sets of states of all automata generated in the construction to be pairwise disjoint, i.e., if $\mathcal{A}' \in \mathcal{S}_{T'}$ and $\mathcal{A}'' \in \mathcal{S}_{T''}$, where T' and T'' are two (not necessarily distinct) sub-expressions of T, then the set of states of $\mathcal{A}'$ and the one of $\mathcal{A}''$ are disjoint.

We proceed by structural induction on ωT-regular expressions, that is, when building the set $\mathcal{S}_{T'}$ of CQ automata for a sub-expression T' of T, we assume the sets of CQ automata for the sub-expressions of T' to be available. In addition, by construction, we force all generated CQ automata $\mathcal{A} = (S, \Sigma, s_0, N, \Delta)$ to feature a distinguished *final state* s_f such that $(s_f, \sigma, s', (k, op)) \in \Delta$ implies $\sigma = \epsilon$, $s' = s_f$, $k = 1$, and $op = inc$.

We first deal with T-regular expressions (sub-grammar rooted in t in Section 2.3). Since a T-regular expression produces a language of word sequences and our automata accept ω-words, we must find a way to extract sequences from ω-words. Let $\mathcal{C} = (s_0, C_0)(s_1, C_1)\ldots$ be an accepting computation of $\mathcal{A}$ such that $(s_i, C_i) \rightarrow^{\sigma}_{\mathcal{A}} (s_{i+1}, C_{i+1})$ via δ_i, for each $i \geqslant 0$, and let w be $\mathcal{C}$-induced via a function f. Moreover, let $g : \mathbb{N} \rightarrow \mathbb{N}$ be an increasing function such that, for every $i \in \mathbb{N}$, $i \in img(g)$ iff δ_i has the form $(s_i, \sigma, s_{i+1}, (1, check)) \in \Delta$. We denote by $\boldsymbol{u}_{w,f}$ the word sequence whose i-th element is $w[j] \ldots w[j + n]$, where $f(j-1) < g(i) \leqslant f(j) < f(j+n) < g(i+1) \leqslant f(j+n+1)$. We define the language of sequences accepted by $\mathcal{A}$ as $\mathcal{L}_s(\mathcal{A}) = \{\boldsymbol{u}_{w,f} : w \text{ is } \mathcal{C}\text{-induced via } f, \text{ for some accepting computation } \mathcal{C} \text{ of } \mathcal{A}\}$.

Automata for T-regular expressions are built as follows. For each expression t, we build a set $\mathcal{S}_t = \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$, with $\mathcal{A}_i = (S_i, \Sigma, s^i_0, N_i, \Delta_i)$, with final state s^i_f, for $1 \leqslant i \leqslant n$, such that $\mathcal{L}(t) = \bigcup_{1 \leqslant i \leqslant n} \mathcal{L}_s((S_i, \Sigma, s^i_0, N_i, \Delta_i \cup \{(s^i_f, \epsilon, s^i_0, (1, check))\}))$. Moreover, for any CQ automaton $\mathcal{A} = (S, \Sigma, s_0, N, \Delta)$ and natural number $N' > 1$, we define the *N'-shifted version* of $\mathcal{A}$ as the automaton $\mathcal{A}' = (S, \Sigma, s_0, N + N', \{(s, \sigma, s, (k + N', op)) : (s, \sigma, s, (k, op)) \in \Delta\})$.

Base cases. If $t = \varnothing$, then $\mathcal{S}_t = \{(\{s_0, s_f\}, \Sigma, s_0, 1, \{\})\}$; if $t = a$, then $\mathcal{S}_t = \{(\{s_0, s_f\}, \Sigma, s_0, 1, \{(s_0, a, s_f, (1, no\_op)), (s_f, \epsilon, s_f, (1, inc))\})\}$.

Inductive step. Let $t = t_1 \cdot t_2$, $\mathcal{A} = (S, \Sigma, s_0, N, \Delta) \in \mathcal{S}_{t_1}$, and $\mathcal{A}' = (S', \Sigma, s'_0, N', \Delta') \in \mathcal{S}_{t_2}$. Moreover, let $\mathcal{A}'' = (S, \Sigma, s_0, N + 1, \Delta'')$ and $\mathcal{A}''' = (S', \Sigma, s'_0, N' + N + 1, \Delta''')$ be the 1-shifted version of $\mathcal{A}$ and the $N + 1$-shifted version of $\mathcal{A}'$, respectively. We define $\mathcal{A} \cdot \mathcal{A}' = (S \cup S' \cup \{s''_f\}, \Sigma, s_0, N + N' + 1, \Delta'' \cup \Delta''' \cup \{(s_f, \epsilon, s'_0, (2, check)), (s'_f, \epsilon, s''_f, (N+2, check)), (s''_f, \epsilon, s''_f, (1, inc))\})$, with s''_f as the final state of $\mathcal{A} \cdot \mathcal{A}'$. $\mathcal{S}_{t_1 \cdot t_2}$ is the set $\{\mathcal{A} \cdot \mathcal{A}' : \mathcal{A} \in \mathcal{S}_{t_1}, \mathcal{A}' \in \mathcal{S}_{t_2}\}$.

Let $t = t_1 + t_2$, $\mathcal{A} = (S, \Sigma, s_0, N, \Delta) \in \mathcal{S}_{t_1}$, and $\mathcal{A}' = (S', \Sigma, s'_0, N', \Delta') \in \mathcal{S}_{t_2}$. Moreover, let $\mathcal{A}''$ and $\mathcal{A}'''$ be defined as in the previous case. We define $\mathcal{A}+\mathcal{A}'$ as the set $\{\mathcal{A}_{+_1}, \mathcal{A}_{+_2}, \mathcal{A}_{+_3}\}$, where $\mathcal{A}_{+_1} = (S \cup S' \cup \{\overline{s}_{01}, \overline{s}_{f1}\}, \Sigma, N' + N + 1, \Delta'' \cup \Delta''' \cup \{(\overline{s}_{01}, \epsilon, s_0, (1, no\_op)), (\overline{s}_{01}, \epsilon, s'_0, (1, no\_op)), (s_f, \epsilon, \overline{s}_{f1}, (2, check)), (s'_f, \epsilon, \overline{s}_{f1}, (N+2, check)), (\overline{s}_{f1}, \epsilon, \overline{s}_{f1}, (1, inc))\} \cup \{(s_f, \epsilon, s_f, (k, *)) : * \in \{inc, check\}, N + 2 \leqslant k \leqslant N + N' + 1\})$, $\mathcal{A}_{+_2} = (S \cup S' \cup \{\overline{s}_{02}, \overline{s}_{f2}\}, \Sigma, N' + N + 1, \Delta'' \cup \Delta''' \cup \{(\overline{s}_{02}, \epsilon, s_0, (1, no\_op)), (\overline{s}_{02}, \epsilon, s'_0, (1, no\_op)), (s_f, \epsilon, \overline{s}_{f2}, (2, check)), (s'_f, \epsilon, \overline{s}_{f2}, (N+2, check)), (\overline{s}_{f2}, \epsilon, \overline{s}_{f2}, (1, inc))\} \cup \{(s'_f, \epsilon, s'_f, (k, *)) : * \in \{inc, check\}, 2 \leqslant k \leqslant N + 1\})$, and $\mathcal{A}_{+_3} = (S \cup S' \cup \{\overline{s}_{03}, \overline{s}_{f3}\}, \Sigma, N' + N + 1, \Delta'' \cup \Delta''' \cup$

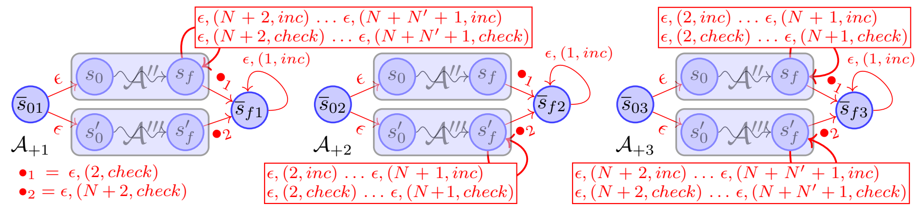

Fig. 3. The automata $\mathcal{A}_{+_1}$, $\mathcal{A}_{+_2}$, and $\mathcal{A}_{+_3}$ (inductive step $t_1 + t_2$).

$\{(\overline{s}_{03}, \epsilon, s_0, (1, no\_op)), (\overline{s}_{03}, \epsilon, s'_0, (1, no\_op)), (s_f, \epsilon, \overline{s}_{f3}, (2, check)), (s'_f, \epsilon, \overline{s}_{f3}, (N+2, check)), (\overline{s}_{f3}, \epsilon, \overline{s}_{f3}, (1, inc))\} \cup \{(s_f, \epsilon, s_f, (k, *)) : * \in \{inc, check\}, 2 \leqslant k \leqslant N+1\} \cup \{(s'_f, \epsilon, s'_f, (k, *)) : * \in \{inc, check\}, N+2 \leqslant k \leqslant N+N'+1\})$. The final state of $\mathcal{A}_{+_i}$ is $\overline{s}_{fi}$, for $1 \leqslant i \leqslant 3$. $\mathcal{S}_{t_1+t_2}$ is the set $\bigcup_{\mathcal{A} \in \mathcal{S}_{t_1}, \mathcal{A}' \in \mathcal{S}_{t_2}} \mathcal{A} + \mathcal{A}'$.

Let $t = t_1^*$ and $\mathcal{A} = (S, \Sigma, s_0, N, \Delta) \in \mathcal{S}_{t_1}$. Moreover, let $\mathcal{A}''$ be defined as in the previous cases. We let $\mathcal{A}_* = (S \cup \{s''_f\}, \Sigma, s_0, N+1, \Delta'' \cup \{(s''_f, \epsilon, s''_f, (1, inc)), (s_f, \epsilon, s''_f, (2, check)), (s_f, \epsilon, s_0, (\epsilon, no\_op))\})$, with s''_f as the final state. $\mathcal{S}_{t_1^*}$ is the set $\{\mathcal{A}_* : \mathcal{A} \in \mathcal{S}_{t_1}\}$.

Let $t = t_1^T$ and $\mathcal{A} = (S, \Sigma, s_0, N, \Delta) \in \mathcal{S}_{t_1}$. Moreover, let $\mathcal{A}'' = (S, \Sigma, s_0, N+2, \Delta'')$ be the 2-shifted version of $\mathcal{A}$. We let $\mathcal{A}_T = (S \cup \{s''_f\}, s_0, N+2, \Delta'' \cup \{(s_f, \epsilon, s_f, (3, check)), (s_f, \epsilon, s_0, (2, inc)), (s_f, \epsilon, s''_f, (2, check)), (s''_f, \epsilon, s''_f, (1, inc))\})$, with s''_f as the final state. $\mathcal{S}_{t_1^T}$ is the set $\{\mathcal{A}_T : \mathcal{A} \in \mathcal{S}_{t_1}\}$.

The following lemma states the correctness of the proposed encoding.

Lemma 3. *Let t be a T-regular expression and $\mathcal{S}_t$ be the corresponding set of automata. It holds that* $\mathcal{L}(t) = \bigcup_{\mathcal{A}_i = (S_i, \Sigma, s_0^i, N_i, \Delta_i) \in \mathcal{S}_t} \mathcal{L}_s((S_i, \Sigma, s_0^i, N_i, \Delta_i \cup \{(s_f^i, \epsilon, s_0^i, (1, check))\}))$.

Proof (sketch—details in the appendix). The proof is by induction on the structure of T-regular expressions. We only consider the case in which $t = t_1 + t_2$, which is definitely the most complex one. A sequence $\boldsymbol{w}$ belonging to $\mathcal{L}_s(t_1 + t_2)$ features words belonging to either $\mathcal{L}_s(t_1)$ or $\mathcal{L}_s(t_2)$. Hence, for $\boldsymbol{w} \in \mathcal{L}_s(t_1 + t_2)$, there are $\boldsymbol{u_i} \in \mathcal{L}_s(t_i)$ $(i \in \{1, 2\})$ and $f : \mathbb{N}^+ \rightarrow \{1, 2\}$ such that $\boldsymbol{w}[i] = \boldsymbol{u_{f(i)}}[i]$, for all $i \in \mathbb{N}^+$. Three cases may arise.

- If there is an index i such that $\boldsymbol{w}[j] = \boldsymbol{u_1}[j]$ for each $j \geqslant i$, then $\boldsymbol{w}$ is accepted by $\mathcal{A}_{+1}$. The computation will eventually end up visiting, besides states $\overline{s}_{01}$ and $\overline{s}_{f1}$, only states of the fragment $\mathcal{A}''$ of $\mathcal{A}_{+1}$ (see Figure 3). Since states of the fragment $\mathcal{A}'''$ are visited a finite number of times only, the problem arises of fulfilling automaton's acceptance conditions relative to the counter-queue configuration components corresponding to $\mathcal{A}'''$ (see accepting conditions ii and iii in Section 3.2). More precisely, for each $j \in \{N+2, \ldots, N+N'+1\}$, the queue associated with the j-th component must never stop growing up during the computation and every element in the queue must eventually be checked. Both conditions are handled by the loop transitions on state s_f, which permit free increment and check. (Acceptance conditions relative to the counter-queue configuration components corresponding to $\mathcal{A}''$ are handled by $\mathcal{A}''$ itself, as its states are visited infinitely often.)
- The case in which there is an index i such that $\boldsymbol{w}[j] = \boldsymbol{u_2}[j]$ for each $j \geqslant i$ is symmetric ($\boldsymbol{w}$ is accepted by $\mathcal{A}_{+2}$).

– If there are infinitely many indexes i and i' such that $\boldsymbol{w}[i] = \boldsymbol{u_1}[i]$ and $\boldsymbol{w}[i'] = \boldsymbol{u_2}[i']$, then $\boldsymbol{w}$ is accepted by $\mathcal{A}_{+3}$. The computation will visit infinitely many times both the states of $\mathcal{A}''$ and those of $\mathcal{A}'''$. Therefore, all acceptance conditions are fulfilled, each fragment of the automaton taking care of the corresponding components.

For each $\mathcal{A}_i = (S_i, \Sigma, s_0^i, N_i, \Delta_i) \in \mathcal{S}_t$, we define $\mathcal{A}'_i$ as $(S_i, \Sigma, s_0^i, N_i, \Delta_i \cup \{(s_f^i, \epsilon, s_0^i, (1, \mathit{check}))\})$. It is not difficult to show that $\bigcup_{\mathcal{A}_i \in \mathcal{S}_t} \mathcal{L}_s(\mathcal{A}'_i) = \mathcal{L}(t)$, by making use of the invariant of the inductive construction. □

We are now ready to deal with ωT-regular expressions (see Section 2.3). We must distinguish three cases. If $T = T_1 + T_2$, then $\mathcal{S}_{T_1+T_2}$ is equal to $\mathcal{S}_{T_1} \cup \mathcal{S}_{T_2}$. Let $T = R \cdot T'$, $A_R = (S_R, F_R, \Sigma, s_0^R, \Delta_R)$ be the NFA that recognises the regular language $\mathcal{L}(R)$, and $\mathcal{A} = (S, \Sigma, s_0, N, \Delta) \in S_{T'}$. We let $A_R \cdot \mathcal{A} = (S \cup S_R, \Sigma, s_0^R, N, \Delta \cup \{(s, \sigma, s', (1, \mathit{no\_op})) : (s, \sigma, s') \in \Delta_R\} \cup \{(s, \epsilon, s_0, (1, \mathit{no\_op})) : s \in F_R\})$, with final state s_f. $\mathcal{S}_{R \cdot T'}$ is the set $\{A_R \cdot \mathcal{A} : \mathcal{A} \in \mathcal{S}_{T'}\}$. Finally, let $T = t^\omega$. We define $\mathcal{S}_t$ as $\{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$, where $\mathcal{A}_i = (S_i, \Sigma, s_0^i, N_i, \Delta_i)$ and s_f^i is the final state of $\mathcal{A}_i$, for every $1 \leqslant i \leqslant n$. $\mathcal{S}_{t^\omega}$ is the set $\{(S_i, \Sigma, s_0^i, N_i, \Delta_i \cup \{(s_f^i, \epsilon, s_0^i, (1, \mathit{check}))\}) : 1 \leqslant i \leqslant n\}$. As in the case of T-regular expressions, it can be easily checked that $\bigcup_{\mathcal{A} \in \mathcal{S}_T} \mathcal{L}(\mathcal{A}) = \mathcal{L}(T)$ for all ωT-regular expressions T.

To complete the reduction, we only need to show how to merge the automata in $\mathcal{S}_T$ into a single one $\mathcal{A}_T$ accepting the language $\mathcal{L}(T)$. Let $\mathcal{S}_T = \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$, with $\mathcal{A}_i = (S_i, \Sigma, s_0^i, N_i, \Delta_i)$, for $1 \leqslant i \leqslant n$, and let $N_{\max} = \max\{N_i : 1 \leqslant i \leqslant n\}$. For each $1 \leqslant i \leqslant n$, let $\overline{\Delta}_i = \Delta_i \cup \{(s, \epsilon, s, (N_j, *)) : * \in \{\mathit{inc}, \mathit{check}\}, s \in S_i, N_i < N_j \leqslant N_{\max}\})$ and let s_0 be a fresh state. We define $\mathcal{A}_T$ as the automaton $(\bigcup_{1 \leqslant i \leqslant n} S_i \cup \{s_0\}, \Sigma, s_0, N_{\max}, \bigcup_{1 \leqslant i \leqslant n} (\overline{\Delta}_i \cup \{(s_0, \epsilon, s_0^i, (1, \mathit{no\_op}))\}))$.

Theorem 2. *For every ωT-regular expression T, there is a CQ automaton $\mathcal{A}$ such that $\mathcal{L}(T) = \mathcal{L}(\mathcal{A})$.*

Corollary 1. *The emptiness problem for ωT-regular languages is decidable.*

5 Conclusions

In this paper, we investigated a new class of extended ω-regular languages, called ωT-regular languages, that captures meaningful languages not belonging to the class of ωBS-regular languages. We proved the decidability of its emptiness problem by exploiting of a new class of automata, called counter-queue automata.

As for future work, we would like to study the class of ωBST-regular languages, which is obtained from the combination of ωT- and ωBS-regular languages. In particular, we are interested in the problem of establishing whether or not it is closed under complementation. In addition, we would like to investigate the logical side of the problem. At the best of our knowledge, no (classical) temporal logic counterparts of extended ω-regular languages were provided in the literature. Recently, we started to work to fill in such a gap [9,10].

References

1. Alur, R., Henzinger, T.A.: Finitary fairness. ACM Trans. Program. Lang. Syst. 20(6), 1171–1194 (1998), `http://doi.acm.org/10.1145/295656.295659`
2. Bojańczyk, M.: A bounding quantifier. In: CSL. LNCS, vol. 3210, pp. 41–55. Springer (2004)
3. Bojańczyk, M.: Weak MSO with the unbounding quantifier. Theory of Computing Systems 48(3), 554–576 (2011)
4. Bojanczyk, M., Colcombet, T.: Bounds in ω-regularity. In: LICS. pp. 285–296. IEEE Computer Society (2006)
5. Büchi, J.R.: On a decision method in restricted second order arithmetic. In: Proc. of the 1960 International Congress on Logic, Methodology and Philosophy of Science. pp. 1–11. Stanford Univ. Press (1962)
6. Elgot, C.C., Rabin, M.O.: Decidability and undecidability of extensions of second (first) order theory of (generalized) successor. J. Symb. Log. 31(2), 169–181 (1966), `http://dx.doi.org/10.2307/2269808`
7. Kupferman, O., Piterman, N., Vardi, M.Y.: From liveness to promptness. Formal Methods in System Design 34(2), 83–103 (2009), `http://dx.doi.org/10.1007/s10703-009-0067-z`
8. McNaughton, R.: Testing and generating infinite sequences by a finite automaton. Information and Control 9(5), 521–530 (1966)
9. Montanari, A., Sala, P.: Adding an equivalence relation to the interval logic $AB\overline{B}$: complexity and expressiveness. In: LICS. pp. 193–202. IEEE Computer Society (2013)
10. Montanari, A., Sala, P.: Interval logics and ωB-regular languages. In: LATA. LNCS, vol. 7810, pp. 431–443. Springer (2013)

A Proofs for Section 3.2

A.1 Proof of Lemma 2

Lemma 2. Let $\mathcal{A}$ be a CQ automaton. Then, Duplicator has a winning strategy in the CQ game on $\mathcal{A}$ iff $Prefixes_{\mathcal{A}}$ contains a winning witness.

Proof. For the left-to-right direction, since there exists a winning strategy for Duplicator in the CQ game on $\mathcal{A}$ we have from Lemma 1 that $\mathcal{L}(\mathcal{A}) \neq \emptyset$. Let $\mathcal{C} = (s_0, CQ_0)\dots$ an accepting computation for $\mathcal{A}$. Let *begin* be the minimum index for which the state s_{begin} is repeated infinitely often in $\mathcal{C}$. Since $\mathcal{C}$ is accepting we have that for every $1 \leqslant k \leqslant C$ there exists an infinite sequence of indexes $S_k = b_k^1 < e_k^1 < b_k^2 < e_k^2 < \dots$ for which for every $i \in \mathbb{N}$ we have $s_{b_k^i} = s_{e_k^i}$, $s_{b_k^i}$ is an inc_k state, for every $b_k^i < j < e_k^i$ we have that s_j is not a $check_k$. Then, there exists $i_1, \dots, i_C$ indexes that satisfy $begin < b_1^{i_1} < e_1^{i_1} < \dots < b_C^{i_C} < e_C^{i_C}$. Since $\mathcal{C}$ is accepting we have that for every $1 \leqslant k \leqslant C$ a $check_k$ state is visited infinitely many times, thus we can choose an index *limit* for which for every $1 \leqslant k \leqslant C$ there exists $e_C < j < limit$ for which s_j is $check_k$ state. For every $1 \leqslant k \leqslant C$ and for every $m \in Set(CQ_{limit}[k])$ we take an index $j_{m,k} > limit$ such that $m = counter(CQ_{j_{m,k}}[k])$ and $s_{j_{m,k}}$ is a $check_k$ state, the existence of such index is guaranteed by the fact that $\mathcal{C}$ is accepting. Finally it suffices to take an index $end > \max(\{j_{m,k} : 1 \leqslant k \leqslant C, m \in Set(CQ_{limit}[k]),\})$ with $s_{end} = s_{begin}$. We have that the prefix $\mathcal{P} = (s_0, CQ_0)\dots(s_{end}, CQ_{end})$ is a winning witness for the CQ game on $\mathcal{A}$.

For the right-to-left direction let us suppose that there exists a winning witness $\mathcal{P} \in Prefixes_{\mathcal{A}}$ then we build a winning strategy *str* for Duplicator in the CQ game on $\mathcal{A}$. Let us observe that for every prefix in $Prefixes_{\mathcal{A}}$ $(s_0, CQ_0)\dots(s_n, CQ_n)$ we have that Duplicator may extend it by simply introducing a sequence of states $s_{n+1}, \dots, s_{n+m}$ such that $(s_n, \sigma, s_{n+1}, (k, op)) \in \Delta$ the resulting prefix $(s_0, CQ_0)\dots(s_n, CQ_n)(s_{n+1}, CQ_{n+1})\dots(s_{n+m}, CQ_{n+m})$ is uniquely determined since the automata is simple. Then we will describe the answers of Duplicator according to *str* as finite word of states in S since the resulting prefix is uniquely determined. Let $\mathcal{P} = (\overline{s}_0, CQ_0)\dots(\overline{s}_n, CQ_n)$ be our winning witness. We build the strategy *str* inductively. We denote with $\mathcal{P}\ell[i]$ the i-th element in the play $\mathcal{P}\ell$. We begin by putting $str(\mathcal{P}l(0)) = \overline{s}_0 \dots \overline{s}_1$ no matter what is the priority of $\mathcal{P}l[0]$. It is easy to see that the winning witness is a successful response for every priority in $\mathcal{P}l[0]$. at each step $i > 0$ we guarantee the following invariant conditions on the play $\mathcal{P}l$:

- $str(\mathcal{P}l(i)) = \overline{s}_{begin+1} \dots \overline{s}_{b_1-1}(\overline{s}_{b_1} \dots \overline{s}_{e_1-1})^{k_1^i}\overline{s}_{e_1} \dots (\overline{s}_{b_C} \dots \overline{s}_{e_C-1})^{k_C^i}\overline{s}_{e_C} \dots \overline{s}_{limit} \dots \overline{s}_{end}$ for some $k_j^i \geqslant 1$ for every $1 \leqslant j \leqslant C$. Basically is the suffix starting at position $begin + 1$ in which for every $1 \leqslant j \leqslant C$ we have repeated every sub-word $\overline{s}_{b_j} \dots \overline{s}_{e_j-1}$ k_j^i number of times;
- let $str(\mathcal{P}l[i]) = ((s_0, CQ_0)\dots(s_{n'}, CQ_{n'}), p)$ then we have $counter(CQ_{n'}[j]) = counter(CQ_n[j])$ for every $1 \leqslant j \leqslant C$. After every answer the counters on the top of the prefix computation are the same of the ones on the top of the winning witness.

Suppose that we are at the step $i > 0$ in a play $\mathcal{P}l$ that Duplicator is playing according to str and let p_i the priority in $\mathcal{P}l[i]$ (i.e., the Spoiler's move) three cases may arise:

- $p_i = sym$, then Duplicator put $str(\mathcal{P}l(i)) = \overline{s}_{begin+1} \ldots \overline{s}_{end}$, since there exists $begin \leqslant j \leqslant end$ which is a sym state we have that a transition $(s, \sigma, s', (k, op))$ is fired in the extension of the i-th prefix;
- $p_i = max_k$ for some $1 \leqslant k \leqslant C$. Let $\mathcal{P}l[i] = ((s_0, CQ_0) \ldots (s_{n'}, CQ_{n'}), p_i)$ then we put $str(\mathcal{P}l(i)) = \overline{s}_{begin+1} \ldots \overline{s}_{b_k - 1}(\overline{s}_{b_k} \ldots \overline{s}_{e_k - 1})^{max(CQ_{n'}[k])+1}\overline{s}_{e_k} \ldots \overline{s}_{end}$. Since by definition of winning witness there is no $check_k$ state in the incremented loop $(\overline{s}_{b_k} \ldots \overline{s}_{e_k - 1})^{max(CQ_{n'}[k])+1}$ and there is at least one $check_k$ state afterwards we have that a number $m > max(CQ_{n'}[k])$ will be introduced in the queue $CQ_{n'+(e_k - b_k)*max(CQ_{n'}[k])+(end-begin)}[k]$;
- $p_i = max_k$ for some $1 \leqslant k \leqslant C$. Let $\mathcal{P}l[i] = ((s_0, CQ_0) \ldots (s_{n'}, CQ_{n'}), p_i)$ and $m = front(CQ_{n'}[k])$ (i.e., the number to be checked for the counter k). Two cases may arise: (i) $m \in Set(CQ_n[k])$ (i.e. m has been introduced in $str(\mathcal{P}l(0))$), then Duplicator puts $str(\mathcal{P}l(i)) = \overline{s}_{begin+1} \ldots \overline{s}_{end}$. Let $last_k$ be the maximum index or which $\overline{s}_{last_k}$ is a $check_k$ state and $last_k \leqslant limit$, such an index always exists for the third condition in the definition of winning witness. We have from the combination of the third and the fourth condition in the definition of winning witness that there exist an index $last_k < j' < end$ for which $m = counter(CQ_{j'}[k])$ and j' is a $check_k$ state, thus m is checked at least one time in the extension of the prefix; (ii) $m \in Counter(CQ_{n''}[k])$ and $s_{n''}$ is a $check_k$ state with $n < n'' \leqslant n'$, then let $0 < i' \leqslant i$ the iteration for which the index n'' has been introduced in the prefix. Then we put $str(i) = str(i')$, from the second invariant condition we have that the value m is checked.

The above infinite procedure generate a winning strategy str for winning the CQ game on $\mathcal{A}$. □

A.2 More definitions on $\mathbb{N}$ words

An $\mathbb{N}$-*word* is a finite word over the set $\mathbb{N}^+$ of positive natural numbers such that, for every $1 \leqslant i < |\alpha|$, $\alpha[i] < \alpha[i+1]$ holds. Given an $\mathbb{N}$-word α, we say that n belongs to α, written $n \in \alpha$, if and only if there exists i for which $\alpha[i] = n$, and we denote by $Set(\alpha)$ the set $\{n \in \mathbb{N} : n \in \alpha\}$. Clearly, for any given set $S \subset \mathbb{N}^+$ there is exactly one $\mathbb{N}$-word α such that $Set(\alpha) = S$; we denote such a word by α_S. Given two $\mathbb{N}$-words α_1 and α_2, we define the union $\alpha_1 \cup \alpha_2$ as the $\mathbb{N}$-word $\alpha_{Set(\alpha_1) \cup Set(\alpha_2)}$, the intersection $\alpha_1 \cap \alpha_2$ as $\alpha_{Set(\alpha_1) \cap Set(\alpha_2)}$, and the difference $\alpha_1 \backslash \alpha_2$ as $\alpha_{Set(\alpha_1) \backslash Set(\alpha_2)}$.

We denote by $[b, e]$ the $\mathbb{N}$-word $\alpha_{\{b, b+1, \ldots, e\}}$, and by (b, e) the $\mathbb{N}$-word $[b+1, e-1]$. Moreover, for a word w on some alphabet and an $\mathbb{N}$-word α we define the *projection* of w on α, denoted by $\pi_\alpha(w)$, as the word $\pi_\alpha(w) = w[\alpha[1]] \ldots w[\alpha[|\alpha|]]$.

Let $\mathcal{P} = (s_0, C_0) \ldots (s_n, C_n) \in Partial_{\mathcal{A}}$ for some CQ automaton $\mathcal{A}$. We define the word $\mathcal{P}_S$ as $\mathcal{P}_S = s_0 \ldots s_n$. Moreover, for each $1 \leqslant k \leqslant N$, we define the

$\mathbb{N}$-word $\beta_k^{\mathcal{P}}$ as $\beta_k^{\mathcal{P}} = i_1 \dots i_m$ such that $\{s_{i_1}, \dots s_{i_m}\}$ is the set of all and only the $check_k$ state in $\mathcal{P}$, and the $\mathbb{N}$-word $\gamma_k^{\mathcal{P}}$ as $\gamma_k^{\mathcal{P}} = i_1 \dots i_m$ such that $\{s_{i_1}, \dots s_{i_m}\}$ is the set of all and only the inc_k state in $\mathcal{P}$. We let $\beta^{\mathcal{P}} = \bigcup_{1 \leqslant k \leqslant N} \beta_k^{\mathcal{P}}$ and $\gamma^{\mathcal{P}} = \bigcup_{1 \leqslant k \leqslant N} \gamma_k^{\mathcal{P}}$.

A.3 Introducing decorations

Let $\mathcal{P} = (s_0, C_0) \dots (s_n, C_n) \in Partial_{\mathcal{A}}$ for some CQ automaton $\mathcal{A}$. A *decoration* $\mathcal{P}_D = \mathcal{S}_1 \dots \mathcal{S}_n$ is a sequence of elements in $(S^*)^N$ (N-dimensional vectors on finite words over S) such that for each $0 \leqslant i \leqslant n$ and each $1 \leqslant k \leqslant N$ we have $|\mathcal{S}_i[k]| = counter(C_i[k])$.

Given $\mathcal{P} = (s_0, CQ_0) \dots (s_n, CQ_n) \in Partial_{\mathcal{A}}$ and a decoration $\mathcal{P}_D = \mathcal{S}_1 \dots \mathcal{S}_n$ for it, we have that $\mathcal{P}'$ and $\mathcal{P}'_D$ is an extension of $\mathcal{P}$ an $\mathcal{D}$ iff $\mathcal{P}'$ and $\mathcal{P}'_D$ form a decorated partial and both $\mathcal{P}$ is a prefix of $\mathcal{P}'$ and $\mathcal{D}$ is a prefix of $\mathcal{D}'$. We have a partial order $\leqslant$ on decorated-partials given by the extension property.

Given a partial $\mathcal{P} = (s_0, CQ_0) \dots (s_n, CQ_n) \in Partial_{\mathcal{A}}$ and a decoration $\mathcal{P}_D = \mathcal{S}_1 \dots \mathcal{S}_n$ for it let $C_{chk} \subseteq \{1, \dots, C\}$ the set of indexes k such that there exists $0 \leqslant i \leqslant n$ for which s_i is a $check_k$ state, $C_{chk} = \{k : \exists i \; s_i$ is a $check_k$ state $\}$. We say that $\mathcal{P}$ is *contractible* if and only if there exists $M \geqslant 1$ indexes $b_1 < e_1 < \dots < b_M < e_M$ and for every $k \in \{1, \dots, k\} \backslash C_{chk}$ there exists N_k indexes $\overline{b}_1^k < \overline{e}_1^k < \dots < \overline{b}_{N_k}^k < \overline{e}_{N_k}^k$ for which:(i) for every $1 \leqslant i \leqslant M$ we have $s_{b_i} = s_{e_i}$ and $counter(CQ_{b_i}[k]) = counter(CQ_{e_i}[k])$ for every $k \in C_{chk}$; (ii) for every $k \in \{1, \dots, k\} \backslash C_{chk}$ and every $1 \leqslant i \leqslant N_k$ we have $\mathcal{S}_n[k][\overline{b}_i^k] = \mathcal{S}_n[k][\overline{e}_i^k]$; (iii) for every $k \in \{1, \dots, k\} \backslash C_{chk}$ we have $\Sigma_{1 \leqslant i \leqslant M} counter(CQ_{e_i}[k]) - counter(CQ_{b_i}[k]) = \Sigma_{1 \leqslant i \leqslant N_k} \overline{e}_i^k - \overline{b}_i^k$. We say that a decorated-partial admits a contraction if and only if it admits a sub-decorated-partial that is contractible. If a decorated-partial does not admit a contraction we say that is *contraction-safe*

A.4 Lemmas instrumental in proving Theorem 1

Lemma 4. *The partial order $\leqslant$, restricted to the set of contraction-safe decorated-partials, does not admit infinite chains.*

Proof. Suppose by contraddiction that there exists an infinite chain $(\mathcal{P}_0, \mathcal{P}_{D0}) < (\mathcal{P}_1, \mathcal{P}_{D1}) < \dots$ an infinite chain in the set of all contraction-free decorated-partials. Let $\mathcal{P}_\omega = (s_0, CQ_0) \dots (s_1, CQ_n) \dots$ and $\mathcal{P}_{D\omega} = \mathcal{S}_0 \mathcal{S}_1 \dots$ the limit of such chain. Let $b_0 < e_0 < b_1 < e_1 < \dots$ be a sequence of indexes such that: (i) there exists $1 \leqslant h \leqslant C$ for which for every $i \in \mathbb{N}$ s_{b_i} and s_{e_i} for every $b_i \leqslant j \leqslant e_i$ s_j is not a $check_k$ state, $s_{e_i} = s_{e_{i+1}}$ and $counter(CQ_{e_i}[k]) \leqslant counter(CQ_{e_{i+1}}[k])$ for each $1 \leqslant k \leqslant C$; (ii) Let $C_{chk}^i = \{k : 1 \leqslant k \leqslant C, \exists b_i < j < e_i \; s_j$ is a $check_k$ state $\}$ then $C_{chk}^i = C_{chk}^{i+1}$ for every $i \in \mathbb{N}$ and exists $B \in \mathbb{N}$ for which for every $k \in C_{chk}^i$ we have $counter(CQ_{e_i}[k]) \leqslant B$, since such sets are all the same we will denote them with C_{chk}; (iii) for every $i \in \mathbb{N}$ either (a) $counter(CQ_{e_i}[h]) - counter(CQ_{b_i}[h]) < counter(CQ_{e_{i+1}}[h]) - counter(CQ_{b_{i+1}}[h])$ or (b) $counter(CQ_{e_i}[k]) = counter(CQ_{e_{i+1}}[k])$ for each $1 \leqslant k \leqslant C$. We assume w.l.o.g that $C_{chk} = \{k, \ldots, C\}$ for some $1 \leqslant k \leqslant C$ Notice that if condition (iii-b) holds we have that the prefix ending in s_1 admits a contraction (contradiction). Let us assume $counter(CQ_{e_i}[h]) - counter(CQ_{b_i}[h]) < counter(CQ_{e_{i+1}}[h]) - counter(CQ_{b_{i+1}}[h])$ for every $i \in \mathbb{N}$. Since increments happen one at a time we may assume w.l.o.g. that $e_i - b_i < e_{i+1} - b_{i+1}$ (we may always take an infinite subsequence that satisfy this property). Let $M = B^{|C_{chk}|} \cdot |S| + 1$. Let us observe that we have that for every $i \in \mathbb{N}$ and for every $b_i < j < e_i - M$ that there exists at least two indexes $j \leqslant j < j' < j'' < e_i - M$ for which $s_{j'} = s_{j''}$ and $counter(CQ_{j'}[h]) = counter(CQ_{j''}[h])$ for each $h \in C_{chk}$ and $j'' - j'` \leqslant M$. For every j for which there exists $j \leqslant j < j' < e_i$ which is the minimum index for which $j' - j \leqslant M$ $s_j = s_{j'}$ and $counter(CQ_j[h]) = counter(CQ_{j'}[h])$ for each $h \in C_{chk}$ we define z_j as the $C' = C - |C_{chk}|$ dimensional vector such that $z_j[h] = counter(CQ_j[h]) = counter(CQ_{j'}[h])$ for each $1 \leqslant h \leqslant C'$. Let us observe that for each j that admits the above property we have $|z_j| = \sum_{1 \leqslant h \leqslant C'} z_j[h] \leqslant M$ then we have C'^M possible different vectors. Then since $e_i - b_i < e_{i+1} - b_{i+1}$ for every $i \in \mathbb{N}$ there exists z for which for every $i \in \mathbb{N}$ there exists $i < i'$ such that $|\{j : e_i < j < b_i, z_j = z\}| < |\{j : e_{i'} < j < b_{i'}, z_j = z\}|$. Let $NZ = \{k : z[k] > 0\}$ then for every $i \in \mathbb{N}$ there exists $i < i'$ we have that there exists $i < i'$ such that $counter(CQ_i[k]) < counter(CQ_{i'}[k])$ for every $k \in NZ$ (recall that counter $k \in NZ$ is not checked between b_i and e_i and the number of vectors z is increasing). Then we have for every $i \in \mathbb{N}$ there exists $i < i'$ that $|\mathcal{P}_{Di}[k]| < |\mathcal{P}_{Di'}[k]|$ for each $k \in NZ$. Given a word $w \in S^*$ we have that there exist at least $o = \lfloor \frac{|w|}{|S|} \rfloor$ indexes $i_1, \ldots, i_o$ such that for every $1 \leqslant j \leqslant o$ there exists $i_j < i' \leqslant i' + |S|$ for which $w[i_j] = w[i']$. For every $k \in NZ$ and every $i \in \mathbb{N}$ we define the set of pairs $O_i^k = \{(j, m) : \mathcal{P}_{Di}[k][j] = \mathcal{P}_{Di}[k][j + m], 1 \leqslant m \leqslant |S|\}$. Since for every $i \in \mathbb{N}$ there exists $i < i'$ we have that there exists $i < i'$ such that $counter(CQ_i[k]) < counter(CQ_{i'}[k])$ for every $k \in NZ$ then there for every $i \in \mathbb{N}$ there exists $i < i'$ we have that there exists $i < i'$ such that $|O_i^k| < |O_{i'}^k|$. Then for every $k \in NZ$ there exists m_k for which for every $i \in \mathbb{N}$ there exists $i < i'$ with $|\{(j, m_k) \in O_i^k\}| < |\{(j, m_k) \in O_{i'}^k\}|$. Summing up since the second sub-sequence is built on the first one there exists a subsequence $\overline{b}_0 < \overline{e}_0 < \overline{b}_1 < \overline{e}_1 < \ldots$ of $b_0 < e_0 < b_1 < e_1 < \ldots$ such that:

- for every $i \in \mathbb{N}$ we have $|\{j : \overline{b}_i < j < \overline{e}_i, z_j = z\}| < |\{j : \overline{b}_{i+1} < j < \overline{e}_{i+1}, z_j = z\}|$;
- for every $i \in \mathbb{N}$ and every $k \in NZ$ we have $|\{(j, m_k) \in O_i^k\}| < |\{(j, m_k) \in O_{i+1}^k\}|$.

Then there exists an index $\overline{i} \in \mathbb{N}$ for which:

- there exist $P = \prod_{k \in NZ} m_k$ points $j_1 < \ldots < j_P$ in $\{j : e_{\overline{i}} < j < b_{\overline{i}}, z_j = z\}$ such that $j_{i+1} - j_i > M$ for every $1 \leqslant i < P$;
- for every $k \in NZ$ there exists $P_k = z[k] \cdot \prod_{k' \in NZ, k' \neq k} m_{k'}$ indexes $j_1 < \ldots < j_{P_k}$ in $O_{\overline{i}}^k$ such that $j_{i+1} - j_i > |S|$ for every $1 \leqslant i < P_k$.

Then it is easy to see that the decorated-partial $(\mathcal{P}_0, \mathcal{P}_{D0}) < (\mathcal{P}_{\overline{e}_i}, \mathcal{P}_{D\overline{e}_i})$ admits a contraction (contradiction). $\square$

Lemma 5. *Given a decorated winning witness* $\mathcal{P} = (s_0, CQ_0) \ldots (s_n, CQ_n) \in Partial_{\mathcal{A}}$ *with decoration* $\mathcal{P}_D = \mathcal{S}_1 \ldots \mathcal{S}_n$ *with indexes* $0 \leqslant begin < b_1 < e_1 < \ldots < b_C < e_C < limit < \overline{b}_1 < \overline{e}_1 < \ldots < \overline{b}_M < \overline{e}_M < end$ *if one of the following conditions holds:*

- *there exists one interval* (b, e) *among* $(0, begin), (begin, b_1), \{(b_k, e_k) : 1 \leqslant k \leqslant C\}, \{(e_k,\, b_{k+1}) : 1 \leqslant k < C\}, (e_C, limit)$ *for which* $(\mathcal{P}, \mathcal{P}_D)|_{(b,e)}$ *is not decoration safe;*
- *there exists one interval* (b, e) *among* $(limit, \overline{b}_1), \{(\overline{e}_i, \overline{b}_{i+1}) : 1 \leqslant i < M\}$ *for which* $(\mathcal{P}, \mathcal{P}_D)|_{(b,e)}$ *is not state-contraction-free;*
- *there exists one interval* (b, e) *among* $\{(\overline{b}_i, \overline{e}_i) : 1 \leqslant i \leqslant M\}$ *for which* b *is a* $check_k$ *state for some* $1 \leqslant k \leqslant C$ *and* $\mathcal{P}|_{(b,e)}$ *is not* inc_k*-contraction-free;*
- *the partial* $\mathcal{P}|_{(\overline{e}_M, end)}$ *is not sym-contraction-free;*

then there exists a decorated winning witness $\mathcal{P}' = (s'_0, CQ'_0) \ldots (s'_{n'}, CQ'_{n'}) \in Partial_{\mathcal{A}}$ *with decoration* $\mathcal{P}'_D = \mathcal{S}'_1 \ldots \mathcal{S}'_{n'}$ *where* $n' < n$

Proof. We prove that in each of then four cases we can contract $\mathcal{P}$ into a shorter winning witness. Let us suppose that the first case holds. Without loss of generality we can assume that $\pi_{(0,begin)}(\mathcal{P}, \mathcal{P}_D)$ is not decoration safe (the case for the other intervals is analogous). By definition we have that there exists two indexes $0 < b < e < begin$ for witch $\pi_{(0,begin)}(\mathcal{P}, \mathcal{P}_D)$ is contractible. Let $C_{chk} \subseteq \{1, \ldots, C\}$ the set of indexes k such that there exists $b \leqslant i \leqslant e$ for which s_i is a $check_k$ state and let $\overline{C}_{chk} = \{1, \ldots, C\} \backslash C_{chk}$. For the sake of simplicity we assume w.l.o.g. that the counters in $\overline{C}_{chk}$ are the counters $\{1, \ldots, |\overline{C}_{chk}|\}$ and we put $\overline{C} = |\overline{C}_{chk}|$. Then we have that there exists $N \geqslant 1$ indexes $\hat{b}_1 < \hat{e}_1 < \ldots < \hat{b}_N < \hat{e}_N$ such that for every $k \in \overline{C}_{chk}$ there exists N_k indexes $\breve{b}^k_1 < \breve{e}^k_1 < \ldots < \breve{b}^k_{N_k} < \breve{e}^k_{N_k}$ for which: (i) for every $1 \leqslant i \leqslant N$ we have $s_{\hat{b}_i} = s_{\hat{e}_i}$ and $counter(CQ_{\hat{b}_i}[k]) = counter(CQ_{\hat{e}_i}[k])$ for every $k \in C_{chk}$; (ii) for every $k \in \overline{C}_{chk}$ and every $1 \leqslant i \leqslant N_k$ we have $\pi_{\breve{b}^k_i}(\mathcal{S}_n[k]) = \pi_{\breve{e}^k_i}(\mathcal{S}_n[k])$; (iii) for every $k \in \overline{C}_{chk}$ we have:

$$\sum_{1 \leqslant i \leqslant M} counter(CQ_{\hat{e}_i}[k]) - counter(CQ_{\hat{b}_i}[k]) = \sum_{1 \leqslant i \leqslant N_k} \breve{e}^k_i - \breve{b}^k_i.$$

Since $\mathcal{P}$ is simple witness there exists a unique strictly increasing function $f : Set(\beta^{\mathcal{P}} \cap [0, limit]) \to \{\overline{b}_1, \ldots \overline{b}_M\}$ such that for each $i \in \beta^{\mathcal{P}} \cap [0, limit]$ we have $f(i) = |\beta^{\mathcal{P}} \cap [0, i]|$. For each $k \leqslant \overline{C}$ let $l_k = \max Set(\beta^{\mathcal{P}}_k \cap [0, b])$, by definition of decorated witness we have for each $k \leqslant \overline{C}$:

$$\pi_{[1,counter(CQ[k])]}(S_e[k]) = \pi_{[\overline{b}_{f(l_k)}, \overline{e}_{f(l_k)}] \cap \gamma^{\mathcal{P}}_k \cap [1,counter(CQ[k])]}(\mathcal{P}_S)$$

Thus, for all $k \leqslant \overline{C}$ and all $j \in [i, N_k]$, it holds that:

$$\pi_{\breve{b}^k_j}(\mathcal{S}_e[k]) (= \pi_{\breve{e}^k_j}(\mathcal{S}_e[k])) = \pi_{\breve{b}^k_j}(\pi_{[\overline{b}_{f(l_k)}, \overline{e}_{f(l_k)}] \cap \gamma^{\mathcal{P}}_k}(\mathcal{P}_S))$$

which, in its turn, is equal to

$$\pi_{\check{e}^k_j}(\pi_{[\overline{b}_{f(l_k)},\overline{e}_{f(l_k)}]\cap\gamma^{\mathcal{P}}_k}(\mathcal{P}_S)))$$

Again, w.l.o.., we assume that $l_1 < \ldots < l_{\overline{C}}$ and for each $k \leqslant \overline{C}$ and each $j \in [1, N_k]$ we define $\check{pb}_{k,j} = f(l_k) + \pi_{\check{b}^k_j}([\overline{b}_{f(l_k)}, \overline{e}_{f(l_k)}] \cap \gamma^{\mathcal{P}}_k)$ and $\check{pe}_{k,j} = f(l_k) + \pi_{\check{e}^k_j}([\overline{b}_{f(l_k)}, \overline{e}_{f(l_k)}] \cap \gamma^{\mathcal{P}}_k)$ we define the following word α over the naturals:

$$\alpha = \begin{array}{c}[0, \hat{b}_1] \cup \bigcup_{i=2}^{N}(\hat{e}_{i-1}, \hat{b}_i] \cup (\hat{e}_N, \check{pb}_{1,1}] \cup \bigcup_{k=1}^{\overline{C}}\bigcup_{i=2}^{N_k}(\check{pe}_{k,i-1}, \check{pb}_{k,i}] \cup \\ \cup \bigcup_{k=1}^{\overline{C}-1}(\check{pe}_{k,N_k}, \check{pb}_{k+1,1}] \cup (\check{pe}_{\overline{C},N_{\overline{C}}}, end].\end{array}$$

Since $\mathcal{A}$ is simple the computation is uniquely determined by the sequence of states then we define $\mathcal{P}'$ as the element of $Prefixes_{\mathcal{A}}$ such that $\mathcal{P}'_S = \pi_\alpha \mathcal{P}_S$ and its decoration $\mathcal{P}'_D = \pi_\alpha(\mathcal{P}_D)$. Now we have to define the indexes for the new witness. Let $shift = \sum 1 \leqslant j \leqslant N|(\hat{b}_j, \hat{e}_j]|$ and for every $1 \leqslant i \leqslant N$ let $shift^i_{chk} = |\{i : \exists 1 \leqslant j \leqslant N, i \in (\hat{b}_j, \hat{e}_j] \cap \beta^{\mathcal{P}}\}|$. We put $begin' = begin - shift$, for every $1 \leqslant k \leqslant C$ we put $b'_k = b_k - shift$ and $e'_k = e_k - shift$, $limit' = limit - shift$ and $M' = M - shift^N_{chk}$. For every $1 \leqslant i \leqslant M'$ let $i' = i + \sum_{\{j : \exists i''(i'' \in (\hat{b}_j, \hat{e}_j] \cap \beta^{\mathcal{P}}, f(i'') < i)\}} shift^j_{chk}$ we have $\overline{b}'_i = \overline{b}_{i'} - shift$ and $\overline{e}'_i = \overline{e}_{i'} - shift$. Finally we put $end' = end - shift$. It is easy to see that $(\mathcal{P}', \mathcal{P}'_D)$ with indexes $0 \leqslant begin' < b'_1 < e'_1 < \ldots < b'_C < e'_C < limit' < \overline{b}'_1 < \overline{e}'_1 < \ldots < \overline{b}'_{M'} < \overline{e}'_{M'} < end'$ is a decorated winning witness for $\mathcal{A}$.

For the second case let us suppose that there exists one interval (b, e) among $(limit, \overline{b}_1), \{(\overline{e}_i, \overline{b}_{i+1}) : 1 \leqslant i < M\}$ for which $(\mathcal{P}, \mathcal{P}_D)|_{(b,e)}$ is not state-contraction-free. Suppose that such interval is $(limit, \overline{b}_1)$ (the other cases are analogous). Then by definition there exist two indexes $limit \leqslant b < e \leqslant \overline{b}_1$ for which $s_b = s_e$. Then we put $\alpha = [0, b] \cup (e, end]$ we define $\mathcal{P}'$ as the element of $Prefixes_{\mathcal{A}}$ such that $\mathcal{P}'_S = \pi_\alpha \mathcal{P}_S$ and its decoration $\mathcal{P}'_D = \pi_\alpha(\mathcal{P}_D)$. Now we have to define the indexes $0 \leqslant begin' < b'_1 < e'_1 < \ldots < b'_C < e'_C < limit' < \overline{b}'_1 < \overline{e}'_1 < \ldots < \overline{b}'_{M'} < \overline{e}'_{M'} < end'$ for the new witness. We have $M' = M$, $*' = *$ for every $* \in \{begin, b_1, e_1, b_C, e_C, limit\}$, for each $1 \leqslant j \leqslant M$ $\overline{b}'_j = \overline{b}_j - (e - b)$ and for each $1 \leqslant j \leqslant M$ $\overline{e}'_j = \overline{e}_j - (e - b)$. Finally we put $end' = end - (e - b)$. It is easy to see that $(\mathcal{P}', \mathcal{P}'_D)$ with indexes $0 \leqslant begin' < b'_1 < e'_1 < \ldots < b'_C < e'_C < limit' < \overline{b}'_1 < \overline{e}'_1 < \ldots < \overline{b}'_{M'} < \overline{e}'_{M'} < end'$ is a decorated winning witness for $\mathcal{A}$.

For the third case let us suppose that there exists i for which for which $\overline{b}_i$ is a $check_k$ state for some $1 \leqslant k \leqslant C$ and $\mathcal{P}|_{(\overline{b}_i, \overline{e}_i)}$ is not inc_k-contraction-free. Then by definition there exist two indexes $\overline{b}_i \leqslant b < e \leqslant \overline{e}_i$ for which $s_b = s_e$ and for every $b \leqslant i \leqslant e$ we have that $s_i \notin \gamma^{\mathcal{P}}_k$. Then we put $\alpha = [0, b] \cup (e, end]$ we define $\mathcal{P}'$ as the element of $Prefixes_{\mathcal{A}}$ such that $\mathcal{P}'_S = \pi_\alpha \mathcal{P}_S$ and its decoration $\mathcal{P}'_D = \pi_\alpha(\mathcal{P}_D)$. Now we have to define the indexes $0 \leqslant begin' < b'_1 < e'_1 < \ldots < b'_C < e'_C < limit' < \overline{b}'_1 < \overline{e}'_1 < \ldots < \overline{b}'_{M'} < \overline{e}'_{M'} < end'$ for the new witness.

We have $M' = M$, $*' = *$ for every $* \in \{begin, b_1, e_1, b_C, e_C, limit\}$, for each $1 \leqslant j \leqslant i$ $\overline{b}'_j = \overline{b}_j$, for each $1 \leqslant j < i$ $\overline{e}'_j = \overline{e}_j$, for each $i < j \leqslant M$ $\overline{b}'_j = \overline{b}_j - (e-b)$ and for each $i \leqslant j < M$ $\overline{e}'_j = \overline{e}_j - (e-b)$. Finally we put $end' = end - (e-b)$. It is easy to see that $(\mathcal{P}', \mathcal{P}'_D)$ with indexes $0 \leqslant begin' < b'_1 < e'_1 < \ldots < b'_C < e'_C < limit' < \overline{b}'_1 < \overline{e}'_1 < \ldots < \overline{b}'_{M'} < \overline{e}'_{M'} < end'$ is a decorated winning witness for $\mathcal{A}$.

For the fourth case let us suppose that the partial $\mathcal{P}|_{(\overline{e}_M, end)}$ is not *sym*-contraction-free. Then by definition there exist two indexes $\overline{b}_i \leqslant b < e \leqslant \overline{e}_i$ for which $s_b = s_e$ and for every $b \leqslant i \leqslant e$ we have that s_i is not a *sym* state. Then we put $\alpha = [0, b] \cup (e, end]$ we define $\mathcal{P}'$ as the element of $Prefixes_{\mathcal{A}}$ such that $\mathcal{P}'_S = \pi_\alpha \mathcal{P}_S$ and its decoration $\mathcal{P}'_D = \pi_\alpha(\mathcal{P}_D)$. Now we have to define the indexes $0 \leqslant begin' < b'_1 < e'_1 < \ldots < b'_C < e'_C < limit' < \overline{b}'_1 < \overline{e}'_1 < \ldots < \overline{b}'_{M'} < \overline{e}'_{M'} < end'$ for the new witness. We have $M' = M$, $*' = *$ for every $* \in \{begin, b_1, e_1, b_C, e_C, limit\}$, for each $1 \leqslant j \leqslant M$ $\overline{b}'_j = \overline{b}_j$, for each $1 \leqslant j < M$ $\overline{e}'_j = \overline{e}_j$. Finally we put $end' = end - (e-b)$. It is easy to see that $(\mathcal{P}', \mathcal{P}'_D)$ with indexes $0 \leqslant begin' < b'_1 < e'_1 < \ldots < b'_C < e'_C < limit' < \overline{b}'_1 < \overline{e}'_1 < \ldots < \overline{b}'_{M'} < \overline{e}'_{M'} < end'$ is a decorated winning witness for $\mathcal{A}$. □

For a graphical account of how the contraction described in Lemma 5 works take a look to Figure 4 (before contraction) and Figure 5 (after contraction).

A.5 Proof of Theorem 1

Theorem 1. The emptiness problem for *CQ* automata is decidable.

Proof. The algorithm given in Figures 6 and 7 decides whether or not there exists a winning witness for $\mathcal{A}$. Its soundness and completeness are guaranteed by the results proved in section A.4.

B Proofs for Section 4

B.1 Additional results on sequences

Given a sequence $\boldsymbol{u} = (u_1, u_2, \ldots)$ of finite words in Σ^* and two indexes $i, i' \in \mathbb{N}^+$. We define the i', i-*scrambled* version of $\boldsymbol{u}$ as the sequence $\boldsymbol{u}_{i,i'} = (u_1, \ldots, u_i, u_{i'}, u_{i+1}, \ldots)$. Basically we can choose every words in the sequence and put everywhere we want. Now we prove that ωT sequences are closed for the scrambling operation.

Lemma 6. *Given a T-regular expression t we have that for every $\boldsymbol{u} \in \mathcal{L}_s(t)$ and every pair of indexes $i, i' \in \mathbb{N}^+$ we have $\boldsymbol{u}_{i,i'} \in \mathcal{L}_s(t)$.*

Proof. By structural induction on the T-regular expression t.

If $t = \varnothing$ the result is trivial since there are no sequences in $\mathcal{L}(t)$.

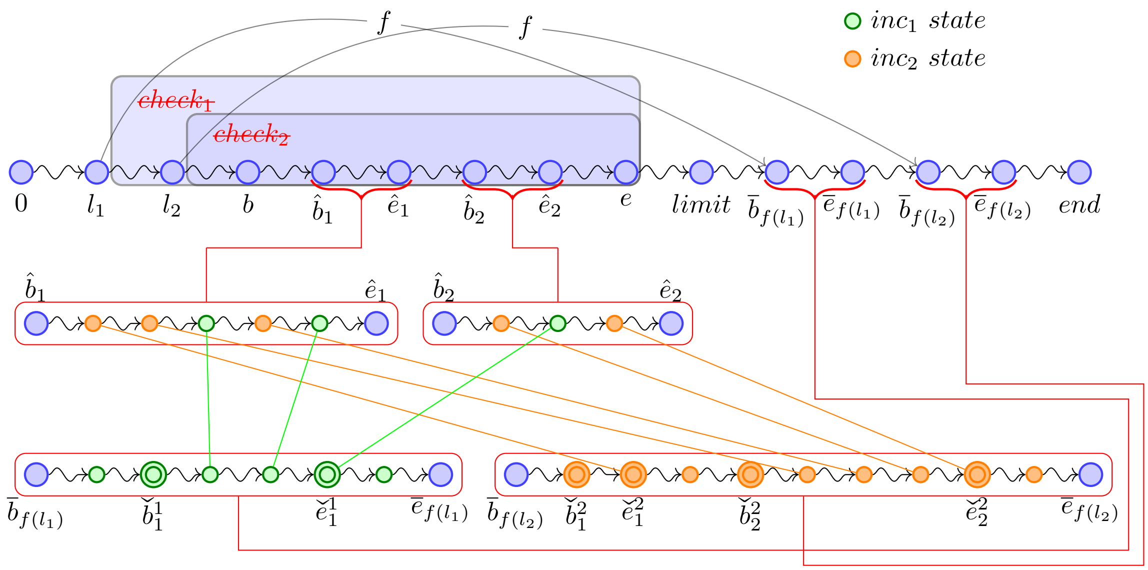

Fig. 4. A winning witness in which the contraction operation described in Lemma 5 may be applied.

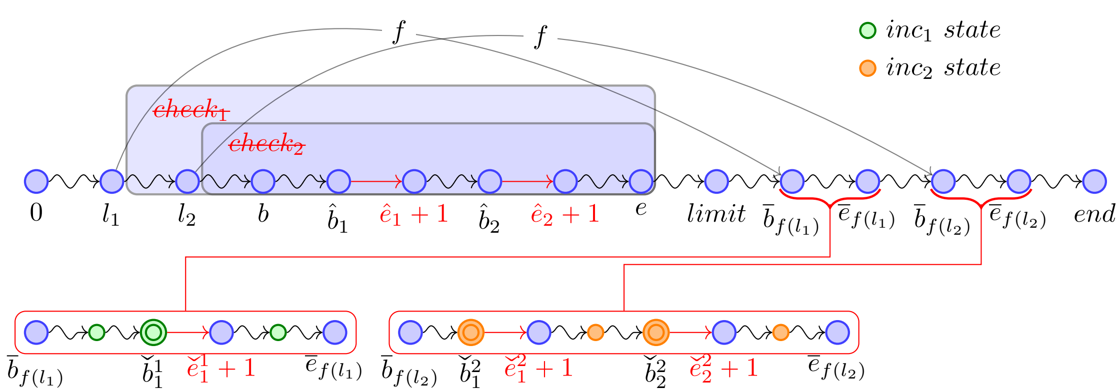

Fig. 5. The winning witness resulting from the application of the contraction operation to the witness Figure 4.

nextINC$(s_0 \dots s_n \in S^*, k)$
$n \leftarrow |\sigma|$
guess $s_{n+1} \in S$
if $\exists\ \mathcal{P} = (s_0, CQ_0) \dots (s_n, CQ_n)(s_{n+1}, CQ_{n+1}) \in Partial_{\mathcal{A}}$
then $\begin{cases} \textbf{if } (s_{n+1} \textit{ is a check}_k \textit{ state}) \vee \exists i (i \leqslant n \wedge s_i = s_{n+1} \wedge \\ \quad \forall i \leqslant j \leqslant n (s_j \textit{ is not an inc}_k \textit{ state})) \\ \quad \textbf{then fail} \end{cases}$
else fail
return (s_n)

nextSYM$(s_0 \dots s_n \in S^*)$
$n \leftarrow |\sigma|$
guess $s_{n+1} \in S$
if $\exists\ \mathcal{P} = (s_0, CQ_0) \dots (s_n, CQ_n)(s_{n+1}, CQ_{n+1}) \in Partial_{\mathcal{A}}$
then $\begin{cases} \textbf{if } \exists 0 \leqslant i \leqslant n\ (s_i = s_{n+1}) \wedge \\ \quad \forall i \leqslant j \leqslant n (s_j \textit{ is not a sym state}) \\ \quad \textbf{then fail} \end{cases}$
return (s_n)

nextCS$(s_0 \dots s_n \in S^*)$
$n \leftarrow |\sigma|$
guess $s_{n+1} \in S$
if $\exists\ \mathcal{P} = (s_0, CQ_0) \dots (s_n, CQ_n)(s_{n+1}, CQ_{n+1}) \in Partial_{\mathcal{A}}$
then $\begin{cases} \textbf{guess } \textit{a decoration } \mathcal{P}_D = \mathcal{S}_0 \dots \mathcal{S}_{n+1} \\ \textbf{if } (\mathcal{P}, \mathcal{P}_D) \textit{ is not contraction-safe} \\ \quad \textbf{then fail} \end{cases}$
else fail
return (s_n)

nextSF$(s_0 \dots s_n \in S^*)$
$n \leftarrow |\sigma|$
guess $s_{n+1} \in S$
if $\exists\ \mathcal{P} = (s_0, CQ_0) \dots (s_n, CQ_n)(s_{n+1}, CQ_{n+1}) \in Partial_{\mathcal{A}}$
then $\begin{cases} \textbf{if } \exists 0 \leqslant i \leqslant n\ (s_i = s_{n+1}) \\ \quad \textbf{then fail} \end{cases}$
return (s_n)

Fig. 6. The auxiliary procedures for Algorithm A.1

If $t = a$ then we have that $\mathcal{L}(t) = \{(a, a, \dots)\}$ and $(\boldsymbol{a, a, \dots})_{i,i'} = (a, a, \dots)$ for every $i, i' \in \mathbb{N}^+$.

If $t = t_1 \cdot t_2$ given two indexes $i, i' \in \mathbb{N}^+$ and two sequences $u \in \mathcal{L}(t_1)$ and $w \in \mathcal{L}(t_2)$. By definition we have $(\boldsymbol{u} \cdot \boldsymbol{w}) = (u_1 \cdot w_1, u_2 \cdot w_2, \dots)$ and $(\boldsymbol{u} \cdot \boldsymbol{w})_{i,i'} = (u_1 \cdot w_1, u_2 \cdot w_2, \dots, u_i \cdot w_i, u_{i'} \cdot w_{i'}, u_{i+1} \cdot w_{i+1})$. Let us observe that $(\boldsymbol{u} \cdot \boldsymbol{w})_{i,i'} = u_{i,i'} \cdot w_{i,i'}$. By inductive hypothesis we have $\boldsymbol{u}_{i,i'} \in \mathcal{L}(t_1)$ and $\boldsymbol{w}_{i,i'} \in \mathcal{L}(t_2)$ and thus $(\boldsymbol{u} \cdot \boldsymbol{w})_{i,i'} \in \mathcal{L}(t_1 \cdot t_2)$.

If $t = t_1 + t_2$ given two indexes $i, i' \in \mathbb{N}^+$ two sequences $u \in \mathcal{L}(t_1)$ and $w \in \mathcal{L}(t_2)$. By definition we have $(\boldsymbol{u} + \boldsymbol{w}) = (v_1, v_2, \dots)$ where $v_j \in \{u_j, w_j\}$ and $(\boldsymbol{u} + \boldsymbol{w})_{i,j} = (v_1, \dots, v_i, v_{i'}, v_{i+1})$. Let us observe that $(\boldsymbol{u} + \boldsymbol{w})_{i,i'} = u_{i,i'} + w_{i,i'}$. By inductive hypothesis we have $\boldsymbol{u}_{i,i'} \in \mathcal{L}(t_1)$ and $\boldsymbol{w}_{i,i'} \in \mathcal{L}(t_2)$ and thus $(\boldsymbol{u} + \boldsymbol{w})_{i,i'} \in \mathcal{L}(t_1 + t_2)$.

If $t = t_1^*$ given a sequence $u \in \mathcal{L}(t_1^*)$. For every unbounded non-decreasing function $g : \mathbb{N} \to \mathbb{N}$ such that we have $u^* = (u_{g(0)} \dots u_{g(1)-1}, u_{g(1)} \dots u_{g(2)-1}, \dots)$ and $u^*_{i,i'} = (u_{g(0)} \dots u_{g(1)-1}, \dots, u_{g(i)} \dots u_{g(i+1)-1}, u_{g(i')} \dots u_{g(i'+1)-1}, u_{g(i+1)} \dots u_{g(i+2)-1}, \dots)$. Let $\Delta = g(i' + 1) - g(i')$ Let us observe that the sequence $u' = (u_{g(0)}, \dots, u_{g(1)-1}, \dots, u_{g(i)} \dots, u_{g(i+1)-1}, u_{g(i')}, \dots u_{g(i'+1)-1}, u_{g(i+1)}, \dots, u_{g(i+2)-1}, \dots)$ satisfies $u' = (((u_{g(i+1)-1, g(i')})_{g(i+1), g(i')+1}) \dots)_{g(i+1)+\Delta-1, g(i'+2)-1}$ and by inductive hypothesis (we repeat the scrambling operation a finitely number of times on a sequence in $\mathcal{L}_s(t)$) we have $u' \in \mathcal{L}_s(t)$ and thus for $g'(n) = \begin{cases} g(n) & n \leqslant i \\ g(n) + \Delta & \text{otherwise} \end{cases}$ we have $u' \in \mathcal{L}(t^*)$.

If $t = t_1^T$ given a sequence $u \in \mathcal{L}(t_1^*)$. For every unbounded and nondecreasing function $g : \mathbb{N} \to \mathbb{N}$ such that:
 (i) $\forall n \exists i. g(i+1) - g(i) > n$
 (ii) $\forall n.$[if $\exists i. g(i+1) - g(i) = n$, then $\forall k \exists j > k. g(j+1) - g(j) = n$]$\}$.

Algorithm A.1: FindWitness($\mathcal{A} = (S, \Sigma, s_0, C, \Delta)$)

```
i ← 1
begin_flag ← ⊥
let 𝒫 = (s_0, CQ_0) where (s_0, CQ_0)
    the initial configuration of 𝒜
while ¬begin_flag
  do { guess begin_flag ∈ {⊥, ⊤}
       if begin_flag
         then { begin ← i − 1
         else { s_i ← nextCS(𝒫)
                𝒫 ← 𝒫 · s_i
                i ← i + 1
q_begin ← nextCS(s_1 . . . s_{begin−1})
i ← i + 1
for j ← 1 to C
  do { if j = 1
         then pre ← begin
         else pre ← e_{j−1}
       b_flag ← ⊥
       while ¬b_flag
         do { guess b_flag ∈ {⊥, ⊤}
              if b_flag
                then { if s_{i−1} is an inc_j state
                         then b_j ← i − 1
                         else fail
                else { s_i ← nextCS(s_{pre+1} . . . s_{i−1})
                       i ← i + 1
       e_flag ← ⊥
       while ¬e_flag
         do { guess e_flag ∈ {⊥, ⊤}
              if e_flag
                then { if s_{i−1} is an inc_j state
                         then e_j ← i − 1
                         else fail
                else { s_i ← nextCS(s_{pre+1} . . . s_{i−1})
                       if s_i is a check_j state
                         then fail
                       i ← i + 1
       if j = C
         then { limit_flag ← ⊥
                while ¬limit_flag
                  do { guess limit_flag ∈ {⊥, ⊤}
                       if limit_flag
                         then { limit ← i − 1
                         else { s_i ← nextCS(s_{C+1} . . . s_{i−1})
                                i ← i + 1
. . .
let c_1 . . . c_M be the sequence
    of indexes c_j ⩽ limit s.t. c_j is a check state
for j ← 1 to M
  do { let 𝒫 = (s_0, CQ_0) . . . (s_{i−1}, CQ_{i−1}) ∈ Prefixes_𝒜
       let k s.t. c_j is a check_k state
       if j = 1
         then pre ← limit
         else pre ← j − 1
       b̄_flag ← ⊥
       while ¬b̄_flag
         do { guess b̄_flag ∈ {⊥, ⊤}
              if b̄_flag
                then { b̄_j ← i − 1
                else { s_i ← nextSF(s_{pre+1} . . . s_{i−1})
                       i ← i + 1
       count ← counter(CQ_{c_j}[k])
       ē_flag ← ⊥
       while ¬ē_flag
         do { guess ē_flag ∈ {⊥, ⊤}
              if ē_flag
                then { if s_{i−1} is a check_k state ∧
                          count = 0 ∧ s_{b̄_j} = s_{i−1}
                         then ē_j ← i − 1
                         else fail
                else { s_i ← nextINC(s_{b̄_j+1} . . . s_{i−1}, k)
                       if s_i is an inc_k state
                         then count ← count − 1
                       i ← i + 1
       if j = M
         then { end_flag ← ⊥
                sym_flag ← ⊥
                while ¬end_flag
                  do { guess end_flag ∈ {⊥, ⊤}
                       if end_flag
                         then { end ← i − 1
                         else { s_i ← nextSYM(s_{ē_M+1} . . . s_{i−1})
                                if s_i is a sym state
                                  then sym_flag ← ⊤
                                i ← i + 1
if s_end = s_limit ∧ sym_flag
  then success
  else fail
```

Fig. 7. The algorithm that decides the existence of a winning witness for $\mathcal{A}$

we have $u^T = (u_{g(0)} \dots u_{g(1)-1}, u_{g(1)} \dots u_{g(2)-1}, \dots)$ and $u^*_{i,i'} = (u_{g(0)} \dots u_{g(1)-1}, \dots, u_{g(i)} \dots u_{g(i+1)-1}, u_{g(i')} \dots u_{g(i'+1)-1}, u_{g(i+1)} \dots u_{g(i+2)-1}, \dots)$. Let $\Delta = g(i'+1) - g(i')$ Let us observe that the sequence $u' = (u_{g(0)}, \dots, u_{g(1)-1}, \dots, u_{g(i)} \dots, u_{g(i+1)-1}, u_{g(i')}, \dots u_{g(i'+1)-1}, u_{g(i+1)}, \dots, \quad u_{g(i+2)-1}, \dots)$ satisfies $u' = (((u_{g(i+1)-1,g(i')})_{g(i+1),g(i')+1}) \dots)_{g(i+1)+\Delta-1,g(i'+2)-1}$ and by inductive hypothesis (we repeat the scrambling operation a finitely number of times on a sequence in $\mathcal{L}_s(t)$) we have $u' \in \mathcal{L}_s(t)$ and thus for $g'(n) = \begin{cases} g(n) & n \leqslant i \\ g(n) + \Delta & \text{otherwise} \end{cases}$ (g' satisfies the same properties of g) we have $u' \in \mathcal{L}(t^T)$. $\square$

Moreover, given a language of word sequences $\mathcal{L}$, we define its *subsequence language* $\Pi(\mathcal{L})$ as the language $\Pi(\mathcal{L}) = \{\boldsymbol{u} : \exists \boldsymbol{u'} \in \mathcal{L}, \exists \pi : \mathbb{N}^+ \to \mathbb{N}^+$ increasing s.t. $\boldsymbol{u'}_\pi = \boldsymbol{u}\}$. Finally, for any T-regular expression e, we define $e[T/*]$ as the regular expression obtained from e by replacing each occurrence of the T operator by the $*$ one

Lemma 7. *For every T-regular t expression we have $\Pi(\mathcal{L}_s(t)) = \mathcal{L}_s(t[T/*])$.*

Proof. Both $\Pi(\mathcal{L}_s(t)) \subseteq \mathcal{L}_s(t[T/*])$ and $\Pi(\mathcal{L}_s(t)) \supseteq \mathcal{L}_s(t[T/*])$ are proved by structural induction on the T-regular expression t.

We begin with $\Pi(\mathcal{L}_s(t)) \subseteq \mathcal{L}_s(t[T/*])$.

Base cases are trivial since $t[T/*] = T$ for $t \in \{a, \varnothing\}$.

If $t = a$ then we have that $\mathcal{L}(t) = \{(a, a, \dots)\}$ and $(a, a, \dots)_\pi = (a, a, \dots)$ for every increasing function π.

If $t = t_1 \cdot t_2$ given two sequences two sequences $u \in \mathcal{L}(t_1)$ and $w \in \mathcal{L}(t_2)$ we have $u \cdot w = (u_1 \cdot w_1, u_2 \cdot w_2, \dots)$. Given an increasing function π we have $(u \cdot w)_\pi = (u_{\pi(1)} \cdot w_{\pi(1)}, u_{\pi(2)} \cdot w_{\pi(2)}, \dots)$ by inductive hypothesis we have $u_\pi \in \Pi(\mathcal{L}_s(t_1)) = \mathcal{L}_s(t_1[T/*])$ and $w_\pi \in \Pi(\mathcal{L}_s(t_2)) = \mathcal{L}_s(t_2[T/*])$ and thus $(u \cdot w)_\pi = u_\pi \cdot w_\pi$.

If $t = t_1 + t_2$ given two sequences $u \in \mathcal{L}(t_1)$ and $w \in \mathcal{L}(t_2)$ we have $u + w = (v_1, v_2, \dots)$ with $v_i \in \{u_i, w_i\}$ for every $i \in \mathbb{N}^+$. Given an increasing function π we have $(u \cdot w)_\pi = (v_{\pi(1)}, v_{\pi(2)}, \dots)$ by inductive hypothesis we have $u_\pi \in \Pi(\mathcal{L}_s(t_1)) = \mathcal{L}_s(t_1[T/*])$ and $w_\pi \in \Pi(\mathcal{L}_s(t_2)) = \mathcal{L}_s(t_2[T/*])$ and thus $(u+w)_\pi = u_\pi + w_\pi$.

If $t = t_1^*$ given a sequence $u \in \mathcal{L}(t_1^*)$, an unbounded non-decreasing function $g : \mathbb{N} \to \mathbb{N}$ and an increasing function π. We have $u^* = (u_{g(0)} \dots u_{g(1)-1}, u_{g(1)} \dots u_{g(2)-1}, \dots)$ and $u^*_\pi = (u_{g(i_1)} \dots u_{g(i_1+1)-1}, u_{g(i_2)} \dots u_{g(i_2+1)-1}, \dots)$ such that for every $j \in \mathbb{N}^+$ we have $i_j = \pi(j)$. Consider now the sequence $u' = (u_{g(i_1)}, \dots, u_{g(i_1+1)-1}, u_{g(i_2)}, \dots, u_{g(i_2+1)-1}, \dots)$ this is the sequence u'_π of $\pi(\mathcal{L}(t_1))$ for the increasing function π' with $img(\pi') = [g(i_1), \dots g(i_1+1)-1] \cup [g(i_2), g(i_2+1)-1] \cup \dots$. Let us define $\Delta_j = (g(i_j+1)-1) - g(i_j)$. By inductive hypothesis we have $u'_\pi \in t[T/*]$. Thus $u^*_\pi \in \mathcal{L}_s((t[T/*])^*))$ using the function $g'(n) = \Sigma_{0\dots n} \Delta_j$.

If $t = t_1^T$ given a sequence $u \in \mathcal{L}(t_1^T)$, an unbounded non-decreasing function $g : \mathbb{N} \to \mathbb{N}$ that satisfies

$$\begin{array}{l}(i)\ \ \forall n \exists i. g(i+1) - g(i) > n \\ (ii)\ \forall n.[\text{if } \exists i. g(i+1) - g(i) = n, \text{ then } \forall k \exists j > k. g(j+1) - g(j) = n]\}. \end{array} \text{ and}$$

an increasing function π. We have $u^T = (u_{g(0)} \ldots u_{g(1)-1}, u_{g(1)} \ldots u_{g(2)-1}, \ldots)$ and $u_\pi^T = (u_{g(i_1)} \ldots u_{g(i_1+1)-1}, u_{g(i_2)} \ldots u_{g(i_2+1)-1}, \ldots)$ such that for every $j \in \mathbb{N}^+$ we have $i_j = \pi(j)$. Consider now the sequence $u' = (u_{g(i_1)}, \ldots, u_{g(i_1+1)-1}, u_{g(i_2)}, \ldots, u_{g(i_2+1)-1}, \ldots)$ this is the sequence u'_π of $\pi(\mathcal{L}(t_1))$ for the increasing function π' with $img(\pi') = [g(i_1), \ldots g(i_1 + 1) - 1] \cup [g(i_2), g(i_2 + 1) - 1] \cup \ldots$. Let us define $\Delta_j = (g(i_j + 1) - 1) - g(i_j)$. By inductive hypothesis we have $u'_\pi \in t[T/*]$. For $g'(n) = \Sigma_{0 \ldots n} \Delta_j$ let us observe that g' is unbounded non decreasing but it does not necessarily satisfies

$$\begin{array}{l}(i)\ \ \forall n \exists i. g(i+1) - g(i) > n \\ (ii)\ \forall n.[\text{if } \exists i. g(i+1) - g(i) = n, \text{ then } \forall k \exists j > k. g(j+1) - g(j) = n]\}. \end{array} \text{ and}$$

thus $u_\pi^* \in \mathcal{L}_s((t[T/*])^*)$.

Now we prove $\Pi(\mathcal{L}_s(t)) \supseteq \mathcal{L}_s(t[T/*])$.

Base cases are trivial since $t[T/*] = T$ for $t \in \{a, \varnothing\}$.

If $t = t_1 \cdot t_2$ given two sequences two sequences $u \in \mathcal{L}(t_1[T/*])$ and $w \in \mathcal{L}(t_2[T/*])$. By inductive hypothesis there exist two sequences $u' \in \mathcal{L}(t_1)$ and $w' \in \mathcal{L}(t_2)$ and two increasing functions π and π' for which $u'_\pi = u$and $w'_{\pi'} = w$. We build a word w'' as follows:

1. $w'' = w'$;
2. if for every j we have $w'_{\pi'(j)} = w''_{\pi(j)}$ then exit;
3. let j be the minimum index such that $w'_{\pi'(j)} \neq w''_{\pi(j)}$;
4. let j' be an index such that $w''_{j'} = w'_{\pi'(j)}$;
5. we put $w'' = w''_{\pi(j)-1,j'}$;
6. return to 2.

The existence of the index j' in step four is guaranteed 4 by the fact that w'' is build by taking only the words of the sequence w' and thus of the sequence $w'_{\pi'} = w$. The limit of this (possibly infinite) procedure is a sequence w'' such that:

- $w'' \in \mathcal{L}(t_2)$ since $w' \in \mathcal{L}(t_2)$ and w'' is built via scrambling operations only (we apply Lemma 6 here);
- $w''_\pi = w'_{\pi'}$.

Finally we have $(u' \cdot w'')_\pi = u \cdot w$ and $(u' \cdot w'')_\pi \in \Pi(\mathcal{L}_s(t))$. thus $u \cdot w \in \Pi(\mathcal{L}_s(t))$.

If $t = t_1 + t_2$ given two sequences two sequences $u \in \mathcal{L}(t_1[T/*])$ and $w \in \mathcal{L}(t_2[T/*])$. By inductive hypothesis there exist two sequences $u' \in \mathcal{L}(t_1)$ and $w' \in \mathcal{L}(t_2)$ and two increasing functions π and π' for which $u'_\pi = u$and $w'_{\pi'} = w$. We build a word w'' as in the previous step. Finally we have $(u' + w'')_\pi = u + w$ and $(u' + w'')_\pi \in \Pi(\mathcal{L}_s(t))$. thus $u + w \in \Pi(\mathcal{L}_s(t))$.

If $t = t_1^*$ given a sequence $u \in \mathcal{L}(t_1[T/*])$ by inductive hypothesis there exist a sequence $u' \in \mathcal{L}(t_1)$ and an increasing functions π such that $u'_\pi = u$. Let f an unbounded non decreasing function we build a sequence u'' as follows:

1. $u'' = u'$;

2. if for every $i \in \mathbb{N}^+$ and every $0 \leqslant j < f(i+1) - f(i)$ we have $u''_{\pi(i)+j} = u_{f(i)+j}$ then exit;
3. let i be the t minimum index such there exists $0 \leqslant j < f(i+1) - f(i)$ for which we have $u''_{\pi(i)+j} \neq u_{f(i)+j}$, let us assume j be the minimum among such indexes;
4. let j' be an index such that $u''_{j'} = u_{f(i)+j}$;
5. we put $w'' = w''_{\pi(j)-1,j'}$;
6. return to 2.

For the very same arguments explained in the previous cases we have that $u'' \in \mathcal{L}(t_1)$. Moreover for every $i \in \mathbb{N}^+$ and every $0 \leqslant j < f(i+1) - f(i)$ we have $u''_{\pi(i)+j} = u_{f(i)+j}$. Let us define the sequence $(u'')^* = (u''_1 \dots u''_{\pi(1)+f(1)-1}, u''_{\pi(1)+f(1)} \dots u''_{\pi(2)-1}, u''_{\pi(2)} \dots u''_{\pi(2)+f(2)-1}, \dots)$ we have $(u'')^* \in \mathcal{L}_s(t)$ and let π' be the increasing function with $img(\pi') = \{2n+1 : n \in \mathbb{N}\}$ it is easy to see that $(u'')^*_{\pi'} \in \mathcal{L}_s(t) = (u_1 \dots u_{f(1)-1}, u_{f(1)} \dots u_{f(2)-1}, \dots)$ and thus $u \in \Pi(\mathcal{L}_s(t))$.

If $t = t_1^T$ given a sequence $u \in \mathcal{L}(t_1[T/*])$ by inductive hypothesis there exist a sequence $u' \in \mathcal{L}(t_1)$ and an increasing functions π such that $u'_\pi = u$. Let f an unbounded non decreasing function we build a sequence u'' as in the previous case and we have that for every $i \in \mathbb{N}^+$ and every $0 \leqslant j < f(i+1) - f(i)$ we have $u''_{\pi(i)+j} = u_{f(i)+j}$.

We define the sequence u''', the functions f' and π' as the result of the following procedure:

1. $u''' = u''$, $\overline{\pi} = \pi$, π' is the increasing function with $img(\pi') = \{2n+1 : n \in \mathbb{N}\}$, g is the increasing function with $img(g) = \{0, \pi(1), \pi(1) + f(1), \dots\}$ and we put $i = 1$;
2. let $M_i = \sum_{j=1}^{k}(f(j) - f(j-1)) + \sum_{j=1}^{i} j (= i(i+1)/2)$ for some $k > i$ such that $M_i > \overline{\pi}(i+1) - \overline{\pi}(i) + (f(i))$;
3. let $\Delta_i = M_i - \overline{\pi}(i+1) - \overline{\pi}(i) + (f(i))$ we apply $u''' = u'''_{\pi'(i)+f(i),\pi'(i)+f(i)}$ for Δ_i times;
4. let $\overline{\pi}'$ be the function $\overline{\pi}'(n) = \overline{\pi}(n)$ for $n \leqslant i$ and $\overline{\pi}'(n) = \overline{\pi}(n) + \Delta_i$ otherwise we put $\overline{\pi} = \overline{\pi}'$;
5. let π'' be the increasing function with $\pi''(j) = \pi'(j)$ for each $j \leqslant i$ $\pi''(j) = \pi'(j) + k + (i(i+1)/2)$ otherwise, we put $\pi' = \pi''$;
6. let g' the increasing function with $img(g') = \{i \in img(g) : i \leqslant \overline{\pi}(i) + f(i)\} \cup \bigcup_{k'=1,\dots k'}\{\overline{\pi}(i) + f(i) + \sum_{j=1}^{k'}(f(j) - f(j-1)) \cup \bigcup_{j'=1}^{i}\{\overline{\pi}(i) + f(i) + \sum_{j=1}^{k}(f(j) - f(j-1)) + j'\}$, we put $g = g'$;
7. $i = i + 1$;
8. go back to step 2.

Let us define the sequence $(u''')^T = (u'''_1 \dots u'''_{g(1)-1}, u'''_{g(1)} \dots u'''_{g(2)-1}, \dots)$ where u''' and g are the limit of the sequence and function generated by the above procedure. Clearly, $(u''')^T$ belongs to $\mathcal{L}_s(t^T)$. Finally, we have that $(u''')^T_{\pi'} = u$ where π' is the limit of the function π' by the above procedure belongs.

B.2 Proof of Lemma 3

Given a CQ automaton $\mathcal{A} = (S, \Sigma, s_0, N, \Delta)$ we define the relation $\to^{*\epsilon}_{\mathcal{A}}$ as the reflexive and transitive closure of the relation $\to^{\epsilon}_{\mathcal{A}}$. Given a CQ automaton $\mathcal{A} = (S, \Sigma, s_0, N, \Delta)$ and a word $w \in \Sigma^*$ We define the relation $\to^{w}_{\mathcal{A}}$ between the configurations of $\mathcal{A}$ where $(s, CQ) \to^{w}_{\mathcal{A}} (s', CQ')$ if and only if $(s, CQ) \to^{*\epsilon}_{\mathcal{A}} (s_1, CQ_1) \to^{w[1]}_{\mathcal{A}} (s_2, CQ_2) \to^{*\epsilon}_{\mathcal{A}} \ldots \to^{*\epsilon}_{\mathcal{A}} (s_2, CQ_2) \to^{w[2]}_{\mathcal{A}} (s_3, CQ_3) \to^{*\epsilon}_{\mathcal{A}} \ldots \to^{*\epsilon}_{\mathcal{A}} (s_{2|w|-1}, CQ_{2|w|-1}) \to^{w[|w|]}_{\mathcal{A}} (s_{2|w|}, CQ_{2|w|}) \to^{*\epsilon}_{\mathcal{A}} (s', CQ')$.

Lemma 3.

Let t be a T-regular expression and $\mathcal{S}_t$ be the corresponding set of automata. It holds that $\mathcal{L}(t) = \bigcup_{\mathcal{A}_i=(S_i,\Sigma,s_0^i,N_i,\Delta_i)\in\mathcal{S}_t} \mathcal{L}_s((S_i, \Sigma, s_0^i, N_i, \Delta_i \cup \{(s_f^i, \epsilon, s_0^i, (1, \mathit{check}))\}))$.

Proof. Given a T-regular expression we prove that the set of automata $\mathcal{S}_t$ build inductively as described in Section 4 satisfies $\mathcal{L}(t) = \bigcup_{\mathcal{A}_i=(S_i,\Sigma,s_0^i,N_i,\Delta_i)\in\mathcal{S}_t} \mathcal{L}_s((S_i, \Sigma, s_0^i, N_i, \Delta_i \cup \{(s_f^i, \epsilon, s_0^i, (1, \mathit{check}))\}))$.

If $t = \emptyset$ we have the automaton $\mathcal{S}_t = \{(\{s_0, s_f\}, \Sigma, s_0, 1, \{\})\}$ as the unique element of $\mathcal{S}_t$. It is easy to see that $(\{s_0, s_f\}, \Sigma, s_0, 1, \{(s_f, \epsilon, s_0, (1, \mathit{check}))\})$ does not recognize any sequence thus $\bigcup_{\mathcal{A}_i=(S_i,\Sigma,s_0^i,N_i,\Delta_i)\in\mathcal{S}_t} \mathcal{L}_s((S_i, \Sigma, s_0^i, N_i, \Delta_i \cup \{(s_f^i, \epsilon, s_0^i, (1, \mathit{check}))\})) = \emptyset$.

If $t = a$ we have the set of automata consisting of $\mathcal{S}_t = \{(\{s_0, s_f\}, \Sigma, s_0, 1, \{(s_0, a, s_f, (1, \mathit{no\_op})), (s_f, \epsilon, s_f, (1, \mathit{inc}))\})\}$ and we consider the automaton $\mathcal{A}' = (s_0, a, s_f, (1, \mathit{no\_op})), (s_f, \epsilon, s_f, (1, \mathit{inc})), (s_f, \epsilon, s_0, (1, \mathit{check}))\})$. We have to prove that $\mathcal{L}_s(\mathcal{A}') = \{(a, a, \ldots)\}$. For $\mathcal{L}_s(\mathcal{A}') \subseteq \{(a, a, \ldots)\}$ consider a successful computation $\mathcal{P}$ of $\mathcal{A}'$ we have by the structure of the automata that it is of the form $\mathcal{P} = (s_0, \emptyset) \to^{a}_{\mathcal{A}'} (s_f, CQ_1)^{k_1} \to^{\epsilon}_{\mathcal{A}'} (s_0, CQ_2) \to^{a}_{\mathcal{A}'} (s_f, CQ_3)^{k_2} \to^{\epsilon}_{\mathcal{A}'} (s_0, CQ_4) \ldots$ where $(s_f, CQ_j)^{k_i}$ is a shorthand for the iteration $(s_f, CQ_j) \to^{\epsilon}_{\mathcal{A}'} (s_f, CQ_j)$ k_i times. It is easy to see that exactly one a symbol appear in between two consecutive s_0 where the counter one has been checked exactly once. Since we have that $\{(a, a, \ldots)\}$ is a singleton set and we already proved that $\mathcal{L}_s(\mathcal{A}') \subseteq \{(a, a, \ldots)\}$ and $\mathcal{L}_s(\mathcal{A}') \neq \emptyset$ we may conclude that $\mathcal{L}_s(\mathcal{A}') = \{(a, a, \ldots)\}$. If $t = t_1 \cdot t_2$ by inductive hypothesis we have that $\mathcal{L}(t_1) = \bigcup_{\mathcal{A}_i=(S_i,\Sigma,s_0^i,N_i,\Delta_i)\in\mathcal{S}_{t_1}} \mathcal{L}_s((S_i, \Sigma, s_0^i, N_i, \Delta_i \cup \{(s_f^i, \epsilon, s_0^i, (1, \mathit{check}))\}))$ and $\mathcal{L}(t_2) = \bigcup_{\mathcal{A}_i=(S_i,\Sigma,s_0^i,N_i,\Delta_i)\in\mathcal{S}_{t_2}} \mathcal{L}_s((S_i, \Sigma, s_0^i, N_i, \Delta_i \cup \{(s_f^i, \epsilon, s_0^i, (1, \mathit{check}))\}))$.
For every pair of automata $\mathcal{A} \in \mathcal{S}_{t_1}$ and $\mathcal{A}' \in \mathcal{S}_{t_2}$ let $\mathcal{A}'' = (S, \Sigma, s_0, N+1, \Delta'')$ and $\mathcal{A}''' = (S', \Sigma, s_0', N' + N + 1, \Delta''')$ be the 1-shifted version of $\mathcal{A}$ and the $N+1$-shifted version of $\mathcal{A}'$, respectively. For every such pair we have that the automata $\mathcal{A} \cdot \mathcal{A}' = (S \cup S' \cup \{s_f''\}, \Sigma, s_0, N + N' + 1, \Delta'' \cup \Delta''' \cup \{(s_f, \epsilon, s_0', (2, \mathit{check})), (s_f', \epsilon, s_f'', (N+2, \mathit{check})), (s_f'', \epsilon, s_f'', (1, \mathit{inc}))\})$ belongs to $\mathcal{S}_{t_1 \cdot t_2}$. An accepting computation $\mathcal{P}$ of the automata $(S \cup S' \cup \{s_f''\}, \Sigma, s_0, N+N'+1, \Delta'' \cup \Delta''' \cup \{(s_f, \epsilon, s_0', (2, \mathit{check})), (s_f', \epsilon, s_f'', (N+2, \mathit{check})), (s_f'', \epsilon, s_f'', (1, \mathit{inc}))\} \cup \{(s_f'', \epsilon, s_0, (1, \mathit{check}))\})$ has the following form $\mathcal{P} = (s_0, CQ_0) \ldots (s_f, CQ_{n_1})(s_0', CQ_{n_1+1}) \ldots (s_f', CQ_{n_2})(s_f'', CQ_{n_2+1})^{k_1}(s_0, CQ_{n_3}) \ldots$

$(s_f, CQ_{n_4})(s'_0, CQ_{n_4+1})\dots(s'_f, CQ_{n_5})(s''_f, CQ_{n_5+1})^{k_2}\dots$. This means that we have exactly one *check* operation of the counters 2 and $N+2$ in this order in between two consecutive *check* operations of the counter 1. Since $\mathcal{A}''$ and $\mathcal{A}'''$ are only shifted version of the automata $\mathcal{A}$ and $\mathcal{A}'$ we have, by inductive hypothesis that the sequences u and w induced by the traces $s_{n_i}, CQ_{n_i})\dots(s_f, CQ_{n_i+1})$ and $(s'_0, CQ_{n_{i+1}+1})\dots(s'_f, CQ_{n_{i+2}})$ (for $i \in \mathbb{N}$) respectively, satisfy $u \in \mathcal{L}_s(t_1)$ and $w \in \mathcal{L}(t_2)$ (inductive hypothesis) and thus $u \cdot w \in \mathcal{L}_s(t_1 \cdot t_2)$. On the other hand let us consider a sequence $(v_1, v_2, \dots) \in \mathcal{L}(t_1 \cdot t_2)$ by definition such a sequence may be seen as a sequence $(u_1 \cdot w_1, u_2 \cdot w_2, \dots)$ where $v_i = u_i \cdot w_i$ then $u = (u_1, u_2, \dots) \in \mathcal{L}_s(t_1)$ and $w = (w_1, w_2, \dots) \in \mathcal{L}_s(t_2)$. By inductive hypothesis we have that there exists an automaton $\mathcal{A} = (S, \Sigma, s_0, N, \Delta) \in \mathcal{S}_{t_1}$ and an automaton $\mathcal{A}' = (S', \Sigma, s'_0, N', \Delta') \in \mathcal{S}_{t_2}$ such that $u \in \mathcal{L}_s(\mathcal{A})$ and $w \in \mathcal{L}_s(\mathcal{A}')$. Then we have two accepting computations $\mathcal{P} = (s_0, CQ_0) \rightarrow^{u_1}_{\mathcal{A}} (s_f, CQ_{n_1})^{k_1}(s_0, CQ_{n_1+k_1+1}) \rightarrow^{u_2}_{\mathcal{A}} (s_f, CQ_{n_2})^{k_2}\dots$, $\mathcal{P}' = (s'_0, CQ'_0) \rightarrow^{w_1}_{\mathcal{A}} (s'_f, CQ'_{m_1})^{h_1}(s'_0, CQ'_{m_1+h_1+1}) \rightarrow^{u_2}_{\mathcal{A}} (s_f, CQ'_{m_2})^{h_2}\dots$ of $(S, \Sigma, s_0, N, \Delta \cup \{(s_f, \epsilon, s_0, (1, \mathit{check}))\})$ and $(S', \Sigma, s'_0, N', \Delta' \cup \{(s'_f, \epsilon, s'_0, (1, \mathit{check}))\})$ respectively. Let us consider the computation $\mathcal{P}'' = (s_0, CQ''_0) \rightarrow^{u_1}_{\mathcal{A}} (s_f, CQ''_{n_1})^{k_1}(s'_0, CQ'''_0) \rightarrow^{w_1}_{\mathcal{A}} (s'_f, CQ'''_{m_1})^{h_1}(s''_f, \overline{CQ}_1)^{l_1}(s_0, CQ_{n_1+k_1+1}) \rightarrow^{u_2}_{\mathcal{A}} (s_f, CQ_{n_2})^{k_2}(s'_0, CQ'_{m_1+h_1+1}) \rightarrow^{u_2}_{\mathcal{A}} (s_f, CQ'_{m_2})^{h_2}(s''_f, \overline{CQ}_2)^{l_2}\dots$ where CQ''_i is the 1-shifted version of the queue CQ_i,CQ'''_i is the $N+1$-shifted version of the queue CQ'_i and $|\{h_i : i \in \mathbb{N}\}| = \omega$ and for every $j \in \mathbb{N}$ $|\{i : h_i = h_j\}| = \omega$.

By construction we have that $\mathcal{P}''$ is an accepting computation of the automata $(S \cup S' \cup \{s''_f\}, \Sigma, s_0, N+N'+1, \Delta'' \cup \Delta''' \cup \{(s_f, \epsilon, s'_0, (2, \mathit{check})), (s'_f, \epsilon, s''_f, (N+2, \mathit{check})), (s''_f, \epsilon, s''_f, (1, \mathit{inc}))\} \cup \{(s''_f, \epsilon, s_0, (1, \mathit{check}))\})$ and thus $u \cdot w \in \mathcal{L}_s(\mathcal{A} \cdot \mathcal{A}')$.