

Graph Reachability and Pebble Automata over Infinite Alphabets

Tony Tan

Hasselt University and Transnational University of Limburg

Let $\mathfrak{D}$ denote an infinite alphabet – a set that consists of infinitely many symbols. A word $w = a_0b_0a_1b_1 \cdots a_nb_n$ of even length over $\mathfrak{D}$ can be viewed as a directed graph G_w whose vertices are the symbols that appear in w , and the edges are $(a_0, b_0), (a_1, b_1), \dots, (a_n, b_n)$. For a positive integer m , define a language $\mathcal{R}_m$ such that a word $w = a_0b_0 \cdots a_nb_n \in \mathcal{R}_m$ if and only if there is a path in the graph G_w of length $\leq m$ from the vertex a_0 to the vertex b_n .

We establish the following hierarchy theorem for pebble automata over infinite alphabet. For every positive integer k , (i) there exists a k -pebble automaton that accepts the language $\mathcal{R}_{2k-1}$; (ii) there is no k -pebble automaton that accepts the language $\mathcal{R}_{2k+1-2}$. Using this fact, we establish the following main results in this paper: (a) a strict hierarchy of the pebble automata languages based on the number of pebbles; (b) the separation of monadic second order logic from the pebble automata languages; (c) the separation of one-way deterministic register automata languages from pebble automata languages.

Categories and Subject Descriptors: F.1.1 [Models of Computation]: Pebble automata; F.4.1 [Mathematical Logic]: Computational logic

General Terms: Languages

Additional Key Words and Phrases: Pebble automata, Graph reachability, Infinite alphabets

1. INTRODUCTION

Logic and automata for words over finite alphabets are relatively well understood and recently there is a broad research activity on logic and automata for words and trees over infinite alphabets. Partly, the study of infinite alphabets is motivated by the need for formal verification and synthesis of infinite-state systems and partly, by the search for automated reasoning techniques for XML. There has been a significant progress in this area, see [Björklund and Schwentick 2007; Bojanczyk et al. 2011; Demri and Lazić 2009; Kaminski and Francez 1994; Neven et al. 2004; Segoufin 2006] and this paper aims to contribute to the progress.

Roughly speaking, there are two approaches to studying languages over infinite alphabets: logic and automata. Below is a brief summary on both approaches. For a more comprehensive survey, we refer the reader to [Segoufin 2006]. The study of languages over infinite alphabets starts with the introduction of finite-memory automata (FMA) in [Kaminski and Francez 1994], also known as *register automata* (RA), that is, automata with a finite number of registers. From here on, we write RA_n to denote RA with n registers.

The study of RA was continued and extended in [Neven et al. 2004], in which *pebble automata* (PA) were also introduced. Each of these models has its own advantages and disadvantages. Languages accepted by RA are closed under standard language

The extended abstract of this article has been published in the proceedings of LICS 2009. The author is FWO Pegasus Marie Curie Fellow.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.

© 20YY ACM 1529-3785/20YY/01-ART1 \$15.00

DOI 10.1145/0000000.0000000 <http://doi.acm.org/10.1145/0000000.0000000>

operations: intersection, union, concatenation, and Kleene star. In addition, from the computational point of view, RA are a much easier model to handle. Their emptiness problem is decidable, whereas the same problem for PA is not. However, the PA languages possess a very nice logical property: closure under *all* boolean operations.

Recently there is a more general model of RA introduced in [Bojanczyk et al. 2011], that builds on the idea of nominal sets. In this model the structure for the symbols is richer. In addition to equality test, it allows for total order and partial order tests among the symbols.

In [Bouyer 2002] data words are introduced, which are an extension of words over infinite alphabet. Data words are words in which each position carries both a label from a finite alphabet, and a data value from an infinite alphabet. The paper [Bojanczyk et al. 2011] studies the logic for data words, and introduced the so-called *data automata*. It was shown that data automata define the logic $\exists\text{MSO}^2(\sim, <, +1)$, the fragment of existential monadic second order logic in which the first order part is restricted to two variables only, with the signatures: the data equality $\sim$, the order $<$ and the successor $+1$. An important feature of data automata is that their emptiness problem is decidable, even for infinite words, but is at least as hard as reachability for Petri nets. It was also shown that the satisfiability problem for the three-variable first order logic is undecidable.

Another logical approach is via the so called *linear temporal logic* with freeze quantifier, introduced in [Demri et al. 2005] and later also studied in [Demri and Lazić 2009]. Intuitively, these are LTL formula equipped with a finite number of registers to store the data values. We denote by $\text{LTL}_n^\downarrow[X, U]$, the LTL with freeze quantifier, where n denotes the number of registers and the only temporal operators allowed are the *neXt* operator X and the *Until* operator U . It was shown that alternating RA_n accept all $\text{LTL}_n^\downarrow[X, U]$ languages and the emptiness problem for alternating RA_1 is decidable. However, the complexity is non primitive recursive. Hence, the satisfiability problem for $\text{LTL}_1^\downarrow(X, U)$ is decidable as well. Adding one more register or past time operators, such as X^{-1} or U^{-1} , to $\text{LTL}_1^\downarrow(X, U)$ makes the satisfiability problem undecidable. In [Lazić 2011] a weaker version of alternating RA_1 , called *safety alternating* RA_1 , is considered, and the emptiness problem is shown to be EXPSpace-complete.

In this paper we continue the study of pebble automata (PA) for strings over infinite alphabets introduced in [Neven et al. 2004]. The original PA for strings over finite alphabet was first introduced and studied in [Globerman and Harel 1996]. Essentially PA are finite state automata equipped with a finite number of pebbles. The pebbles are placed on or lifted from the input word in the stack discipline – first in last out – and are intended to mark the positions in the input word. One pebble can only mark one position and the most recently placed pebble serves as the head of the automaton. The automaton moves from one state to another depending on the equality tests among data values in the positions currently marked by the pebbles, as well as the equality tests among the positions of the pebbles.

As mentioned earlier, PA languages possess a very nice logical property: closure under *all* boolean operations. Another desirable property of PA languages is, as shown in [Neven et al. 2004], that nondeterminism and two-way-ness do not increase the expressive power of PA [Neven et al. 2004, Theorem 4.6]. Moreover, the class of PA languages lies strictly in between $\text{FO}(\sim, <, +1)$ and $\text{MSO}(\sim, <, +1)$ [Neven et al. 2004, Theorems 4.1 and 4.2].

Moreover, looking at the stack discipline imposed on the placement of the pebbles, one can rightly view PA as a natural extension of $\text{FO}(\sim, <, +1)$. To simulate a first-order sentence of quantifier rank k , a pebble automaton with k pebbles suffices: one pebble for each quantifier depth. (See Proposition 2.5.)

In this paper we study PA as a model of computation for the directed graph reachability problem. To this end, we view a word of even length $w = a_0b_0a_1b_1 \cdots a_nb_n$ over an infinite alphabet as a directed graph $G_w = (V_w, E_w)$ with the symbols that appear in $a_0b_0a_1b_1 \cdots a_nb_n$ as the vertices in V_w and $(a_0, b_0), \dots, (a_n, b_n)$ as the edges in E_w . We say that the word w induces the graph G_w .

We prove that for any positive integer k , k pebbles are sufficient for recognizing the existence of a path of length $2^k - 1$ from the vertex a_0 to the vertex b_n , but are not sufficient for recognizing the existence of a path of length $2^{k+1} - 2$ from the vertex a_0 to the vertex b_n . Based on this result, we establish the following relations among the classes of languages over infinite alphabets which were previously unknown.

- (1) A strict hierarchy of the PA languages based on the number of pebbles.
- (2) The separation of monadic second order logic from the PA languages.
- (3) The separation of one-way deterministic RA languages from PA languages.

Some of these results settle questions left open in [Neven et al. 2004; Segoufin 2006].

Although, in general, the emptiness problem for PA is undecidable, we believe that our study may contribute to the technical aspect of reasoning on classes of languages with decidable properties. For example, in Section 4 a similar technique is used to obtain separation result for $\text{LTL}_1^\downarrow[X, U]$ languages, a class of languages with decidable satisfiability problem.

Related work. A weaker version of PA, called *top-view weak* PA was introduced and studied in [Tan 2010], where it was also shown that the emptiness problem is decidable. The results in this paper are not implied from that paper, as here the main concern is separation results. In fact, some of the separation results here also hold for the model in [Tan 2010].

There is also an analogy between our result with the classical first-order quantifier lower bounds for directed graph (s, t) -reachability which states the following: There is a first order sentence of quantifier rank k to express the existence of a path of length $\leq m$ from the source node s to the target node t if and only if $m \leq 2^k$. See, for example, [Turán 1984].

As far as we can see, our result is actually a tighter version of the classical result for first-order logic. It is tighter because PA is shown to be stronger than first-order logic (Proposition 2.5). In particular pebble automata do have states, thus, enjoy the usual benefits associated with automata, like counting the number of edges, or the number of neighbours up to $\leq m$, $\geq m$, or mod m , for an arbitrary but fixed positive integer m , without increasing the number of pebbles.

Other related results are those established in [Ajtai and Fagin 1990; Fagin et al. 1995; Schwentick 1996]. To the best of our knowledge, those results have no connection with the result in this paper. In [Ajtai and Fagin 1990] it is established that (s, t) -reachability in directed graph is not in monadic NP*, while in [Fagin et al. 1995; Schwentick 1996] it is established that undirected graph connectivity is not in monadic NP. However, no lower bound on first-order quantifier rank is established.

Organization. This paper is organized as follows. In Section 2 we review the monadic second-order logic $\text{MSO}(\sim, <, +1)$ and pebble automata (PA) for words over infinite alphabet. Section 3 is the core of the paper in which we present our main results. In Section 4 we discuss how to adjust our results and proofs presented in Section 3 to a weaker version of PA, called weak PA, whose relation to the logic $\text{LTL}_1^\downarrow(X, U)$ is presented in Section 5.

*Monadic NP is a complexity theoretic name for existential monadic second order logic.

2. MODELS OF COMPUTATIONS

In Section 2.1 we recall the definition of alternating pebble automata from [Neven et al. 2004], and in Section 2.2 a logic for languages over infinite alphabets.

We shall use the following notation: $\mathfrak{D}$ is a fixed infinite alphabet not containing the left-end marker $\triangleleft$ or the right-end marker $\triangleright$. The input word to an automaton is of the form $\triangleleft w \triangleright$, where $w \in \mathfrak{D}^*$. Symbols of $\mathfrak{D}$ are denoted by lower case letters a, b, c , etc., possibly indexed, and words over $\mathfrak{D}$ by lower case letters u, v, w , etc., possibly indexed.

2.1. Pebble automata

Definition 2.1. (See [Neven et al. 2004, Definition 2.3]) A *two-way alternating k -pebble automaton*, (in short k -PA) is a system $\mathcal{A} = \langle Q, q_0, F, \mu, U \rangle$ whose components are defined as follows.

- (1) $Q, q_0 \in Q$ and $F \subseteq Q$ are a finite set of *states*, the *initial state*, and the set of *final states*, respectively;
- (2) $U \subseteq Q - F$ is the set of *universal states*; and
- (3) μ is a finite set of transitions of the form $\alpha \rightarrow \beta$ such that
 - α is of the form (i, P, V, q) , where $i \in \{1, \dots, k\}$, $P, V \subseteq \{i+1, \dots, k\}$, $q \in Q$ and
 - β is of the form (q, act) , where $q \in Q$ and

$$\text{act} \in \{\text{left}, \text{right}, \text{stay}, \text{place-pebble}, \text{lift-pebble}\}.$$

The intuitive meaning of P and V in (i, P, V, q) is that P denotes the set of pebbles that occupy the same position as pebble i , while V the set of pebbles that read the same symbol as pebble i . A more precise explanation can be found below.

Given a word $w = a_1 \dots a_n \in \mathfrak{D}^*$, a *configuration of $\mathcal{A}$ on $\triangleleft w \triangleright$* is a triple $\gamma = [i, q, \theta]$, where $i \in \{1, \dots, k\}$, $q \in Q$ and $\theta : \{i, i+1, \dots, k\} \rightarrow \{0, 1, \dots, n, n+1\}$. The function θ defines the position of the pebbles and is called the *pebble assignment* of γ . The symbols in the positions 0 and $n+1$ are $\triangleleft$ and $\triangleright$, respectively. That is, we count the leftmost position in w as position 1.

The *initial configuration* of $\mathcal{A}$ on w is $\gamma_0 = [k, q_0, \theta_0]$, where $\theta_0(k) = 0$ is the *initial pebble assignment*. A configuration $[i, q, \theta]$ with $q \in F$ is called an *accepting configuration*. A transition $(i, P, V, p) \rightarrow \beta$ *applies to a configuration* $[j, q, \theta]$, if

- (1) $i = j$ and $p = q$,
- (2) $P = \{l > i \mid \theta(l) = \theta(i)\}$, and
- (3) $V = \{l > i \mid a_{\theta(l)} = a_{\theta(i)}\}$.

We define the transition relation $\vdash_{\mathcal{A}}$ on $\triangleleft w \triangleright$ as follows: $[i, q, \theta] \vdash_{\mathcal{A}, w} [i', q', \theta']$, if there is a transition $\alpha \rightarrow (p, \text{act}) \in \mu$ that applies to $[i, q, \theta]$ such that $q' = p$, for all $j > i$, $\theta'(j) = \theta(j)$, and

- if $\text{act} = \text{left}$, then $i' = i$ and $\theta'(i) = \theta(i) - 1$,
- if $\text{act} = \text{right}$, then $i' = i$ and $\theta'(i) = \theta(i) + 1$,
- if $\text{act} = \text{stay}$, then $i' = i$ and $\theta'(i) = \theta(i)$,
- if $\text{act} = \text{lift-pebble}$, then $i' = i + 1$,
- if $\text{act} = \text{place-pebble}$, then $i' = i - 1$, $\theta'(i - 1) = 0$ and $\theta'(i) = \theta(i)$.

As usual, we denote the reflexive, transitive closure of $\vdash_{\mathcal{A}, w}$ by $\vdash_{\mathcal{A}, w}^*$. When the automaton $\mathcal{A}$ and the word w are clear from the context, we shall omit the subscripts $\mathcal{A}$ and w . For $1 \leq i \leq k$, an *i -configuration* is a configuration of the form $[i, q, \theta]$, that is, when the head pebble is pebble i .

Remark 2.2. Here we define PA as a model of computation for languages over infinite alphabet. Another option is to define PA as a model of computation for data words.

A data word is a finite sequence of $\Sigma \times \mathcal{D}$, where Σ is a finite alphabet of labels. There is only a slight technical difference between the two models. Every data word can be viewed as a word over infinite alphabet in which every odd position contains a constant symbol. In the context of our paper, we ignore the finite labels, thus, Definition 2.1 is more convenient.

We now define how pebble automata accept words. Let $\gamma = [i, q, \theta]$ be a configuration of a PA $\mathcal{A}$ on a word w . We say that γ *leads to acceptance*, if and only if either $q \in F$, or the following conditions hold.

- if $q \in U$, then for all configurations γ' such that $\gamma \vdash \gamma'$, γ' leads to acceptance.
- if $q \notin F \cup U$, then there is at least one configuration γ' such that $\gamma \vdash \gamma'$ and γ' leads to acceptance.

A word $w \in \mathcal{D}^*$ is accepted by $\mathcal{A}$, if the initial configuration γ_0 leads to acceptance. The language $L(\mathcal{A})$ consists of all data words accepted by $\mathcal{A}$.

The automaton $\mathcal{A}$ is *nondeterministic*, if the set $U = \emptyset$, and it is *deterministic*, if for each configuration, there is exactly one transition that applies. If $\text{act} \in \{\text{right}, \text{lift-pebble}, \text{place-pebble}\}$ for all transitions, then the automaton is *one-way*. It turns out that PA languages are quite robust. Namely, alternation and two-wayness do not increase the expressive power to one-way deterministic PA, as stated in Theorem 2.4 below.

Remark 2.3. In [Neven et al. 2004] the model defined above is called *strong* PA. A weaker model in which the new pebble is placed at the position of the head pebble, is referred to as *weak* PA. Obviously for two-way PA, strong and weak PA are equivalent. However, for one-way PA, strong PA is indeed stronger than weak PA. We will postpone our discussion of weak PA until Section 4.

THEOREM 2.4. *For each $k \geq 1$, two-way alternating k -PA and one-way deterministic k -PA have the same recognition power.*

The proof of Theorem 2.4 is a straightforward adaption of the classical proof of the equivalence between the expressive power of alternating two-way finite state automata and deterministic one-way finite state automata [Ladner et al. 1984]. For this reason, we omit the proofs.

The main idea is that when pebble i is the head pebble, due to the stack discipline imposed on placing the pebbles, all the other pebbles (pebbles $i+1, \dots, k$) are fixed on their positions. Hence the transitions of pebble i , which are of the form $(i, P, V, q) \rightarrow (p, \text{act})$, can be viewed as transitions over the finite alphabet $(P, V) \in 2^{\{i+1, \dots, k\}} \times 2^{\{i+1, \dots, k\}}$. Thus, the idea in [Ladner et al. 1984] can be adapted to PA in a straightforward manner. The details are available as a technical report in [Tan 2009]. In view of this equivalence, we will always assume that the pebble automata under consideration are deterministic and one-way.

Next, we define the hierarchy of languages accepted by PA. For $k \geq 1$, PA_k is the set of all languages accepted by k -PA, and PA is the set of all languages accepted by pebble automata. That is,

$$\text{PA} = \bigcup_{k \geq 1} \text{PA}_k.$$

Normalisation of a k -PA. Before we end this subsection, we would like to remark that the behaviour of a k -PA can be normalised in the following way. Let $\mathcal{A}$ be a k -PA. By Theorem 2.4, we can assume that $\mathcal{A}$ is a one-way deterministic k -PA. By adding some extra states, we can normalise the behaviour of each pebble as follows. For each $i \in \{1, \dots, k\}$, pebble i behaves as follows.

- After pebble i moves right and $i > 1$, then pebble $(i - 1)$ is immediately placed (in position 0 reading the left end-marker $\triangleleft$).
If pebble $(i - 1)$ is not immediately placed, we can add some redundant states which force pebble $(i - 1)$ to be placed while remembering the original state in the redundant states.
- If $i < k$, pebble i is lifted only when it reaches the right-end marker $\triangleright$ of the input.
Again, if pebble i is lifted before it reaches the right-end marker $\triangleright$, then we can add some redundant states which force pebble $(i - 1)$ to move right while remembering the original state in the redundant states.
- Immediately after pebble i is lifted, pebble $(i + 1)$ moves right.
If pebble $(i + 1)$ does not move right, we can add some redundant states which force pebble $(i + 1)$ to move right while remembering the original state in the redundant states.

We also assume that in the automaton $\mathcal{A}$ only pebble k can enter a final state and it may do so only after it reads the right-end marker $\triangleright$ of the input. This also can be achieved by adding some redundant states. The normalisation described above will help us greatly in simplifying our proofs in Subsection 3.2 later on.

2.2. Logic

Formally, a word $w = a_1 \cdots a_n$ is represented by the logical structure with domain $\{1, \dots, n\}$; the natural ordering $<$ on the domain with its induced successor $+1$; and the equivalence relation $\sim$ on the domain $\{1, \dots, n\}$, where $i \sim j$ whenever $a_i = a_j$.

The atomic formulas in this logic are of the form $x < y$, $y = x + 1$, $x \sim y$. The first-order logic $\text{FO}(\sim, <, +1)$ is obtained by closing the atomic formulas under the propositional connectives and first-order quantification over $\{1, \dots, n\}$. The second-order logic $\text{MSO}(\sim, <, +1)$ is obtained by adding quantification over unary predicates on $\{1, \dots, n\}$. A sentence φ defines the set of words

$$L(\varphi) = \{w \mid w \models \varphi\}.$$

If $L = L(\varphi)$ for some sentence φ , then we say that the sentence φ *expresses* the language L .

We use the same notations $\text{FO}(\sim, <, +1)$ and $\text{MSO}(\sim, <, +1)$ to denote the languages expressible by sentences in $\text{FO}(\sim, <, +1)$ and $\text{MSO}(\sim, <, +1)$, respectively. That is,

$$\text{FO}(\sim, <, +1) = \{L(\varphi) \mid \varphi \text{ is an } \text{FO}(\sim, <, +1) \text{ sentence}\}$$

and

$$\text{MSO}(\sim, <, +1) = \{L(\varphi) \mid \varphi \text{ is an } \text{MSO}(\sim, <, +1) \text{ sentence}\}.$$

PROPOSITION 2.5. (See also [Neven et al. 2004, Theorem 4.1]) *If $\varphi \in \text{FO}(\sim, <, +1)$ is a sentence with quantifier rank k , then $L(\varphi) \in \text{PA}_k$.*

PROOF. (Sketch) First, it is straightforward that languages accepted by two-way alternating k -PA are closed under Boolean operations. By Theorem 2.4, two-way alternating and one-way deterministic k -PA are equivalent. Thus, the class PA_k is closed under Boolean operations. Therefore, it is sufficient to prove Proposition 2.5 when the formula φ is of the form $Qx_k \psi(x_k)$, where $Q \in \{\forall, \exists\}$ and $\psi(x_k)$ is a formula of quantifier rank $k - 1$.

The proof is by straightforward induction on k . A k -PA $\mathcal{A}$ iterates pebble k through all possible positions in the input word w . On each iteration, the automaton $\mathcal{A}$ recursively calls a $(k - 1)$ -PA $\mathcal{A}'$ that accepts the language $L(\psi(x_k))$, treating the position of pebble k as the assignment value for x_k .

The transition in the PA $\mathcal{A}'$ can test the atomic formula $x = y$ and $x \sim y$; while at the same time remembering in its states the order of the pebbles. The word w is accepted by $\mathcal{A}$, if the following holds.

- If Q is $\forall$, then $\mathcal{A}$ accepts w if and only if $\mathcal{A}'$ accepts on all iterations.
- If Q is $\exists$, then $\mathcal{A}$ accepts w if and only if $\mathcal{A}'$ accepts on at least one iteration.

This completes the sketch of our proof of Proposition 2.5. $\square$

We end this section with Theorem 2.6 below which states that a language accepted by pebble automaton can be expressed by an $\text{MSO}(\sim, <, +1)$ sentence.

THEOREM 2.6. ([Neven et al. 2004, Theorem 4.2]) *For every PA $\mathcal{A}$, there exists an $\text{MSO}(\sim, <, +1)$ sentence $\varphi_{\mathcal{A}}$ such that $L(\mathcal{A}) = L(\varphi_{\mathcal{A}})$.*

3. WORDS OF $\mathfrak{D}^*$ AS GRAPHS

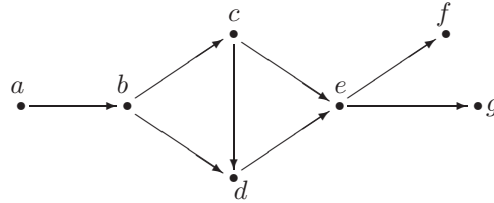
This section contains the main results in this paper:

- (1) The strict hierarchy of PA languages based on the number of pebbles.
- (2) The separation of $\text{MSO}(\sim, <, +1)$ from PA languages.
- (3) The separation of one-way deterministic RA languages from PA languages.

All three results share one common idea: We view a word of even length as a directed graph. Recall that $\mathfrak{D}$ is an infinite alphabet, and that we always denote the symbols in $\mathfrak{D}$ by the lower case letters $a, b, c, \dots$, possibly indexed.

We consider directed graphs in which the vertices come from $\mathfrak{D}$. A word $w = a_0 b_0 \dots a_n b_n \in \mathfrak{D}^*$ of even length induces a directed graph $G_w = (V_w, E_w)$, where V_w is the set of symbols that appear in w , that is, $V_w = \{a : a \text{ appears in } w\}$, and the set of edges is $E_w = \{(a_0, b_0), \dots, (a_n, b_n)\}$. We also write $s_w = a_0$ and $t_w = b_n$ to denote the first and the last symbol in w , respectively. For convenience, we consider only the words w in which s_w and t_w occur only once.

As an example, we take the following word $w = ab\ bc\ bd\ cd\ ce\ de\ ef\ eg$. Then $s_w = a$ and $t_w = g$. The graph induced by w is the $G_w = (V_w, E_w)$, where $V_w = \{a, b, c, d, e, f, g\}$ and $E_w = \{(a, b), (b, c), (b, d), (c, d), (c, e), (d, e), (e, f), (e, g)\}$, as illustrated in the picture below.



We need the following basic graph terminology. Let a and b be vertices in a graph G . A *path* of length m from a to b is a sequence of m edges $(a_{i_1}, b_{i_1}), \dots, (a_{i_m}, b_{i_m})$ in G such that $a_{i_1} = a$, $b_{i_m} = b$ and for each $j = 1, \dots, m-1$, $b_{i_j} = a_{i_{j+1}}$. The *distance* from a to b , denoted by $d_G(a, b)$, is the length of the shortest path from a to b in G . If there is no path from a to b in G , then we set $d_G(a, b) = \infty$.

We now define the following reachability languages. For $m \geq 1$,

$$\mathcal{R}_m = \{w \mid d_{G_w}(s_w, t_w) \leq m\}$$

and

$$\mathcal{R} = \bigcup_{m=1,2,\dots} \mathcal{R}_m.$$

Here we should remark that since we consider only the words w in which s_w and t_w occur only once, the language $\mathcal{R}_1$ consists of words of length 2 with different symbols.

PROPOSITION 3.1. *For each $k = 2, 3, \dots$, $\mathcal{R}_{2^k-1} \in PA_k$.*

The proof of this proposition is an implementation of Savitch's algorithm [Savitch 1970] for $(s-t)$ -reachability by pebble automata. It can be found in Subsection 3.1.

Lemma 3.2 below is the backbone of most of the results presented in this paper. For each $i = 0, 1, 2, \dots$, we define $n_i = 2^{i+1} - 2$. An equivalent recursive definition is $n_0 = 0$, and $n_{i+1} = 2n_i + 2$, for $i \geq 1$.

LEMMA 3.2. *For every k -pebble automaton $\mathcal{A}$, where $k \geq 1$, there exist a word $w \in \mathcal{R}_{n_k}$ and $\bar{w} \notin \mathcal{R}$ such that either $\mathcal{A}$ accepts both w and $\bar{w}$, or $\mathcal{A}$ rejects both w and $\bar{w}$.*

The proof of Lemma 3.2 is rather long and technical. We present it in Subsections 3.2 and 3.3. Meanwhile we discuss a number of consequences of this lemma. Corollary 3.3 below immediately follows from the lemma.

COROLLARY 3.3. $\mathcal{R}_{n_k} \notin PA_k$.

COROLLARY 3.4. $\mathcal{R} \notin PA$.

PROOF. Assume to the contrary that $\mathcal{R} = L(\mathcal{A})$ for a k -PA $\mathcal{A}$. Then, by Lemma 3.2, there exists a word $w \in \mathcal{R}_{n_k}$ and $\bar{w} \notin \mathcal{R}$ such that either $\mathcal{A}$ accepts both w and $\bar{w}$, or $\mathcal{A}$ rejects both w and $\bar{w}$. Both yield a contradiction to the assumption that $\mathcal{R} = L(\mathcal{A})$. $\square$

The following theorem establishes the proper hierarchy of the PA languages.

THEOREM 3.5. *For each $k = 2, \dots$, $PA_k \subsetneq PA_{k+1}$.*

PROOF. We contend that $\mathcal{R}_{2^{k+1}-1} \in PA_{k+1} - PA_k$, for each $k = 2, \dots, 3$. That $\mathcal{R}_{2^{k+1}-1} \in PA_{k+1}$ follows from Proposition 3.1. That $\mathcal{R}_{2^{k+1}-1} \notin PA_k$ follows from the fact that $n_k = 2^{k+1} - 2 < 2^{k+1} - 1$ and Lemma 3.2. $\square$

Another consequence of Corollary 3.4 is that the inclusion of PA in $MSO(\sim, <, +1)$ obtained in Theorem 2.6 is proper.

THEOREM 3.6. $PA \subsetneq MSO(\sim, <, +1)$.

PROOF. Without loss of generality, we may assume that $MSO(\sim, <, +1)$ contains two constant symbols, $\min$ and $\max$, which denote minimum and the maximum elements of the domain, respectively. For a word $w = a_1 \cdots a_n$, the minimum and the maximum elements are 1 and n , respectively, and not 0 and $n + 1$ which are reserved for the end-markers $\lhd$ and $\rhd$.

The language $\mathcal{R}$ can be expressed in $MSO(\sim, <, +1)$ as follows. There exist unary predicates S_{odd} and P such that either

— $\min + 1 = \max \wedge \min \approx \max$ (to capture $\mathcal{R}_1$),

or the following holds.

— For all x , if $x \neq \min$, then $x \approx \min$.

(This is to take care our assumption that the first symbol appears only once.)

— For all x , if $x \neq \max$, then $x \approx \max$.

(This is to take care our assumption that the last symbol appears only once.)

- S_{odd} is the set of all odd elements in the domain where $\min \in S_{\text{odd}}$ and $\max \notin S_{\text{odd}}$.
- The predicate P satisfies the conjunction of the following $\text{FO}(\sim, <, +1)$ sentences:
 - $P \subseteq S_{\text{odd}}$ and $\min \in P$ and $\max - 1 \in P$,
 - for all $x \in P - \{\max - 1\}$, there exists exactly one $y \in P$ such that $x + 1 \sim y$, and
 - for all $x \in P - \{\min\}$, there exists exactly one $y \in P$ such that $y + 1 \sim x$.

Now, the theorem follows from Corollary 3.4. $\square$

Remark 3.7. Combining Theorems 2.4 and 3.6, we obtain that $\text{MSO}(\sim, <, +1)$ is stronger than two-way alternating PA. This settles a question left open in [Neven et al. 2004] whether $\text{MSO}(\sim, <, +1)$ is strictly stronger than two-way alternating PA.

Next, we define a restricted version of the reachability languages. For a positive integer $m \geq 1$, the language $\mathcal{R}_m^+$ consists of all words of the form

$$c_0 c_1 \underbrace{\cdots}_{u_1} c_1 c_2 \underbrace{\cdots}_{u_2} c_2 c_3 \cdots \cdots \cdots c_{m-3} c_{m-2} \underbrace{\cdots}_{u_{m-2}} c_{m-2} c_{m-1} \underbrace{\cdots}_{u_{m-1}} c_{m-1} c_m$$

where for each $i \in \{0, \dots, m-1\}$, the symbol c_i does not appear in u_i and $c_i \neq c_{i+1}$. The language $\mathcal{R}^+$ is defined as

$$\mathcal{R}^+ = \bigcup_{m=1,2,\dots} \mathcal{R}_m^+.$$

Remark 3.8. Actually, in the proof of Lemma 3.2 we show that for every k -PA $\mathcal{A}$, there exist a word $w \in \mathcal{R}_{n_k}^+$ and $\bar{w} \notin \mathcal{R}^+$ such that either $\mathcal{A}$ accepts both w and $\bar{w}$, or $\mathcal{A}$ rejects both w and $\bar{w}$. Therefore, $\mathcal{R}^+ \not\subseteq \text{PA}$.

The following theorem answers a question left open in [Neven et al. 2004; Segoufin 2006]: Can one-way deterministic FMA be simulated by pebble automata? (FMA stands for *finite-memory automata*, the original name of register automata as introduced in [Kaminski and Francez 1994]. We refer the reader to [Kaminski and Francez 1994, Definition 1] for the formal definition of FMA.)

THEOREM 3.9. *The language $\mathcal{R}^+$ is accepted by one-way deterministic FMA, but is not accepted by pebble automata.*

PROOF. Note that $\mathcal{R}^+$ is accepted by a one-way deterministic FMA with two registers.[†] On input word $w = c_0 c_1 \cdots c_{n-1} c_n$, the automaton stores c_1 in the first register and then moves right (using the second register to scan the input symbols) until it finds a symbol $c_i = c_1$. If it finds one, then it stores c_{i+1} in the first register and moves right again until it finds another symbol $c_{i'} = c_{i+1}$. It repeats the process until either of the following holds.

- The symbol in the second last position c_{n-1} is the same as the content of the first register, or,
- it cannot find a symbol currently stored in the first register.

In the former case, the automaton accepts the input word w , and in the latter case it rejects. By Remark 3.8, the language $\mathcal{R}^+$ is not a PA language. This proves Theorem 3.9. $\square$

[†]Here we use the original definition of FMA as introduced in [Kaminski and Francez 1994]. If we use the definition of register automata (RA) as in [Segoufin 2006; Demri and Lazić 2009], then one register is sufficient to accept $\mathcal{R}^+$.

3.1. Proof of Proposition 3.1

In this subsection we prove Proposition 3.1. Before we proceed with the proof, we remark that when processing an input word w , an automaton $\mathcal{A}$ can remember in its state whether a pebble is currently at an odd- or even-numbered position in w . Moreover, we always denote the input word w by $a_0b_0 \cdots a_nb_n$ – that is, we denote the symbols on the odd positions by a_i 's and the symbols on the even position by b_i 's. We can also assume that the automaton always rejects words of odd length.

We are going to construct a k -PA $\mathcal{A}$ that accepts $\mathcal{R}_{2^k-1}$. Essentially the automaton $\mathcal{A}$ consists of the following subautomata.

- An i -PA $\mathcal{A}_i^{j,j'}$, for each $i \in \{1, \dots, k-1\}$ and $j, j' \in \{i+1, \dots, k\}$.
The purpose of each automaton $\mathcal{A}_i^{j,j'}$ is to detect the existence of a path $\leq 2^i - 1$ from the vertex seen by pebble j to the vertex seen by pebble j' .
- An i -PA $\mathcal{A}_i^{*,j}$, for each $i \in \{1, \dots, k-1\}$ and $j \in \{i+1, \dots, k\}$.
The purpose of each automaton $\mathcal{A}_i^{*,j}$ is to detect the existence of a path $\leq 2^i - 1$ from the vertex s_w to the vertex seen by pebble j .
- An i -PA $\mathcal{A}_i^{j,*}$, for each $i \in \{1, \dots, k-1\}$ and $j \in \{i+1, \dots, k\}$.
The purpose of the automaton $\mathcal{A}_i^{j,*}$ is to detect the existence of a path $\leq 2^i - 1$ from vertex seen by pebble j to the vertex t_w .

We are going to show how to construct those subautomata $\mathcal{A}_i^{j,j'}$, $\mathcal{A}_i^{j,*}$ and $\mathcal{A}_i^{*,j}$ by induction on i .

The basis is $i = 1$. The construction of $\mathcal{A}_1^{j,j'}$, $\mathcal{A}_1^{j,*}$ and $\mathcal{A}_1^{*,j}$ is as follows.

- The automaton $\mathcal{A}_1^{j,j'}$ performs the following.
 - (1) It checks whether the symbols seen by pebbles j and j' are the same, which means that there is a path of length 0 from the vertex seen by pebble j to the vertex seen by pebble j' .
 - (2) Otherwise, it iterates pebble 1 on every odd position in w checking whether there exists an index l such that a_l is the same symbol seen by pebble j . If there is, it moves to the right one step to read b_l and checks whether it is the same symbol seen by pebble j' . This means that there is a path of length 1 from the vertex seen by pebble j to the vertex seen by pebble j' .
- The automaton $\mathcal{A}_1^{*,j}$ simply puts pebble 1 on the second position of w to read b_0 and checks whether it is the same symbol seen by pebble j . (Here we use the assumption that s_w occurs only once in w , which implies that there cannot be a path of length 0 in this case.)
- The automaton $\mathcal{A}_1^{j,*}$ simply puts pebble 1 on the second last position of w to read a_n and checks whether it is the same symbol seen by pebble j . (Here we use the assumption that t_w occurs only once in w , which implies that there cannot be a path of length 0 in this case.)

For the induction step, we describe the construction of the automata $\mathcal{A}_i^{j,j'}$, $\mathcal{A}_i^{j,*}$ and $\mathcal{A}_i^{*,j}$ as follows.

- The automaton $\mathcal{A}_i^{j,j'}$ performs the following. It iterates pebble i on each position in the input word w .
 - (1) When pebble i is on the odd position reading the symbol a_l , it invokes the automaton $\mathcal{A}_{i-1}^{j,i}$ to check whether there exists a path of length $\leq 2^{i-1} - 1$ from the vertex seen by pebble j to the vertex a_l .

- (2) If there is such a path, it moves pebble i one step to the right reading the symbol b_l . It then invokes the automaton $\mathcal{A}_{i-1}^{i,j'}$ to check whether there exists a path of length $\leq 2^{i-1} - 1$ from the vertex b_l to the vertex seen by pebble j' .
 Now there exists a path of length $\leq 2^i - 1$ from the vertex seen by pebble j to the vertex seen by pebble j' if and only if there exists an index l such that (i) there exists a path of length $\leq 2^{i-1} - 1$ from the vertex seen by pebble j to the vertex a_l , and (ii) there exists a path of length $\leq 2^{i-1} - 1$ from the vertex b_l to the vertex seen by pebble j' . This implies the correctness of our construction of $\mathcal{A}_i^{j,j'}$.
- The automaton $\mathcal{A}_i^{*,j}$ performs the following. It iterates pebble i on each position in the input word w .
- (1) When pebble i is on the odd position reading the symbol a_l , it invokes the automaton $\mathcal{A}_{i-1}^{*,i}$ to check whether there exists a path of length $\leq 2^{i-1} - 1$ from the vertex s_w to the vertex a_l .
- (2) If there is such a path, it moves pebble i one step to the right reading the symbol b_l . It then invokes the automaton $\mathcal{A}_{i-1}^{i,j}$ to check whether there exists a path of length $\leq 2^{i-1} - 1$ from the vertex b_l to the vertex seen by pebble j .
- It follows immediately that $\mathcal{A}_i^{*,j}$ checks the existence of a path $\leq 2^i - 1$ from the vertex s_w to the vertex seen by pebble j .
- The automaton $\mathcal{A}_i^{j,*}$ performs the following. It iterates pebble i on each position in the input word w .
- (1) When pebble i is on the odd position reading the symbol a_l , it invokes the automaton $\mathcal{A}_{i-1}^{j,i}$ to check whether there exists a path of length $\leq 2^{i-1} - 1$ from the vertex seen by pebble j to the vertex a_l .
- (2) If there is such a path, it moves pebble i one step to the right reading the symbol b_l . It then invokes the automaton $\mathcal{A}_{i-1}^{i,*}$ to check whether there exists a path of length $\leq 2^{i-1} - 1$ from the vertex b_l to the vertex t_w .
- It follows immediately that $\mathcal{A}_i^{j,*}$ checks the existence of a path $\leq 2^i - 1$ from the vertex seen by pebble j to the vertex t_w .

Now the automaton $\mathcal{A}$ performs the following. It iterates pebble k on each position in the input word w .

- (1) When pebble k is on the odd position reading the symbol a_l , it invokes the automaton $\mathcal{A}_{k-1}^{*,k}$ to check whether there exists a path of length $\leq 2^{k-1} - 1$ from the vertex s_w to the vertex a_l .
- (2) If there is such a path, it moves pebble k one step to the right reading the symbol b_l . It then invokes the automaton $\mathcal{A}_{k-1}^{k,*}$ to check whether there exists a path of length $\leq 2^{k-1} - 1$ from the vertex b_l to the vertex t_w .

Hence, $\mathcal{A}$ is the desired automaton for $\mathcal{R}_{2^k-1}$ and this completes the proof of Proposition 3.1.

3.2. Proof of Lemma 3.2

The proof of Lemma 3.2 is rather long and technical. This subsection and the next are devoted to it.

Recall that for each $i \in \{0, 1, 2, \dots\}$, we define $n_i = 2^{i+1} - 2$. An equivalent recursive definition is $n_0 = 0$, and $n_i = 2n_{i-1} + 2$, when $i \geq 1$.

Let $\mathcal{A} = \langle Q, q_0, \mu, F \rangle$ be a strong k -PA. By Theorem 2.4, we can assume that $\mathcal{A}$ is a one-way deterministic k -PA. Moreover, we can normalised the behaviour of $\mathcal{A}$ as explained at the end of Subsection 2.1.

We define the following integers: $\beta_0 = 1$, $\beta_1 = |Q|$, and for $i \geq 2$,[‡]

$$\beta_i = |Q|! \times \beta_{i-1}!$$

For the rest of this subsection and the next, we fix the integers k and m , where k is the number of pebbles of $\mathcal{A}$ and $m = \beta_{k+1}$.

We define the following graph $G_{n_k, m} = (V_{n_k, m}, E_{n_k, m})$. The set $V_{n_k, m}$ consists of the following vertices.

- $a_0, a_1, \dots, a_{n_k}$;
- $b_0, b_1, \dots, b_{n_k-1}$;
- $c_{1,i}, \dots, c_{n_k-1,i}$, for each $i = 1, \dots, m-1$; and
- $d_{1,i}, \dots, d_{n_k-1,i}$, for each $i = 1, \dots, m-1$,

where $a_0, \dots, a_{n_k}, b_0, \dots, b_{n_k-1}, c_{1,1}, \dots, c_{n_k-1,m-1}, d_{1,1}, \dots, d_{n_k-1,m-1}$ are all different. The set $E_{n_k, m}$ consists of the following edges.

- $(a_0, a_1), (a_1, a_2), \dots, (a_{n_k-1}, a_{n_k})$;
- $(b_0, b_1), (b_1, b_2), \dots, (b_{n_k-2}, b_{n_k-1})$;
- $(c_{1,i}, c_{2,i}), (c_{2,i}, c_{3,i}), \dots, (c_{n_k-2,i}, c_{n_k-1,i})$, for each $i = 1, \dots, m-1$; and
- $(d_{1,i}, d_{2,i}), (d_{2,i}, d_{3,i}), \dots, (d_{n_k-2,i}, d_{n_k-1,i})$, for each $i = 1, \dots, m-1$.

Figure 1 below illustrates the graph $G_{n_k, m}$.

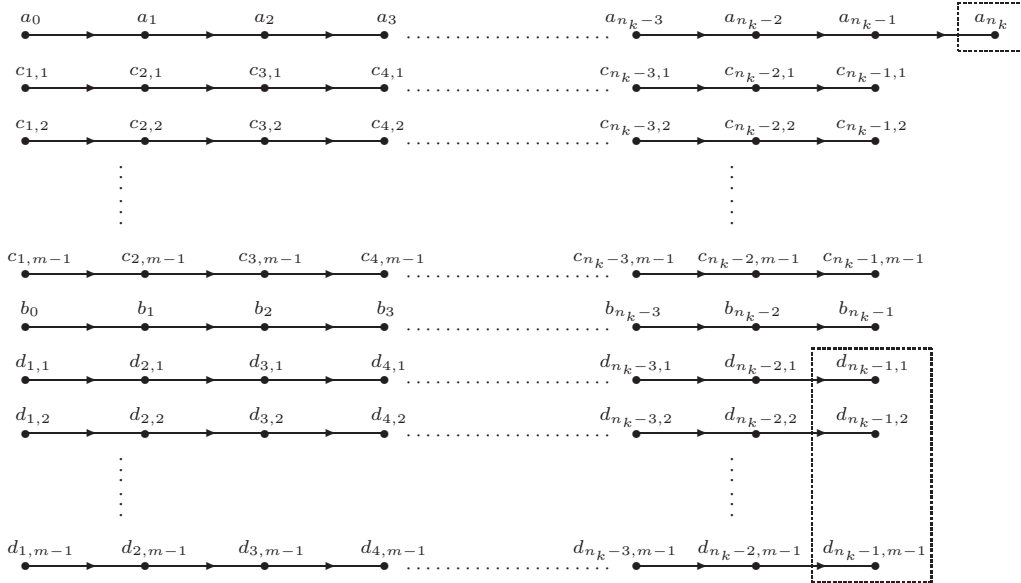


Fig. 1. The full graph is the graph $G_{n_k, m}$. The graph depicted by $\overline{w}(n_k, m)$ is also the above graph but without the nodes inside the dashed box and the edges adjacent to them.

[‡]! denotes factorial.

Now consider the following word $w(n_k, m)$:

$$w(n_k, m) = a_0 a_1 C_1 b_0 b_1 D_1 \cdots a_{n_k-2} a_{n_k-1} C_{n_k-1} b_{n_k-2} b_{n_k-1} D_{n_k-1} a_{n_k-1} a_{n_k} \quad (1)$$

where for each $i = 0, 1, \dots, n_k - 2$,

$$\begin{aligned} - C_i &= c_{i,1} c_{i+1,1} \cdots c_{i,m-1} c_{i+1,m-1}; \\ - D_i &= d_{i,1} d_{i+1,1} \cdots d_{i,m-1} d_{i+1,m-1}. \end{aligned}$$

This word $w(n_k, m)$ induces the graph $G_{n_k, m}$, that is, $G_{w(n_k, m)} = G_{n_k, m}$ and $s_{w(n_k, m)} = a_0$ and $t_{w(n_k, m)} = a_{n_k}$.

Now let

$$\bar{w}(n_k, m) = a_0 a_1 C_1 b_0 b_1 D_1 \cdots a_{n_k-2} a_{n_k-1} C_{n_k-1} b_{n_k-2} b_{n_k-1}. \quad (2)$$

That is, the word $\bar{w}(n_k, m)$ is obtained by deleting the suffix $D_{n_k-1} a_{n_k-1} a_{n_k}$ from $w(n_k, m)$.

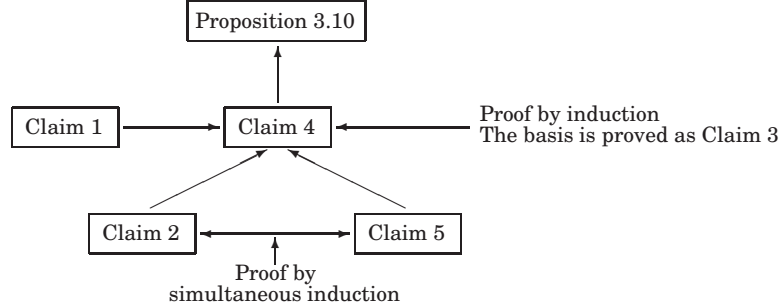
The graph $G_{\bar{w}(n_k, m)}$ is also illustrated in the graph in Figure 1, the graph $G_{\bar{w}(n_k, m)}$ is without the nodes inside the dashed box and the edges adjacent to them.

Note that $s_{\bar{w}(n_k, m)} = a_0$ and $t_{\bar{w}(n_k, m)} = b_{n_k-1}$. Obviously, $w(n_k, m) \in \mathcal{R}_{n_k}$, while $\bar{w}(n_k, m) \notin \mathcal{R}$.

To prove Lemma 3.2, we are going to prove the following proposition.

PROPOSITION 3.10. *The automaton $\mathcal{A}$ either accepts both $w(n_k, m)$ and $\bar{w}(n_k, m)$, or rejects both $w(n_k, m)$ and $\bar{w}(n_k, m)$.*

The proof is rather complicated. It consist of five claims and their interdependence is illustrated below.[§]

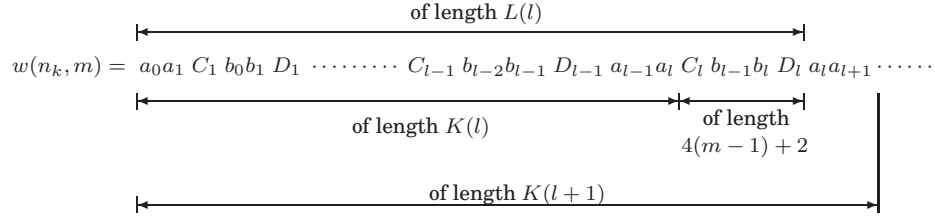


In the proof we will need quite a number of notions which, for the sake of readability, are listed below one-by-one before we define them properly.

- The notions of $K(l)$ and $L(l)$.
- The notion of successor of a pebble assignment.
- The notion of compatibility between two pebble assignments.

The notions of $K(l)$ and $L(l)$. For $l \in \{0, 1, \dots, n_k - 1\}$, we define the integers $K(l)$ and $L(l)$ which are illustrated as follows.

[§]We are going to prove Claims 2 and 5 by induction simultaneously. This will be made precise in Subsection 3.3.



Formally, for $l \in \{0, 1, \dots, n_k\}$,

$$K(l) = \begin{cases} 0, & \text{if } l = 0 \\ 4m(l-1) + 2, & \text{if } l \geq 1 \end{cases}$$

and for $l \in \{0, 1, \dots, n_k\}$,

$$L(l) = \begin{cases} K(l+1) - 2, & \text{if } l \leq n_k - 1 \\ K(n_k), & \text{otherwise.} \end{cases}$$

In particular, $K(n_k)$ is precisely the length of the word $w(n_k, m)$ and $L(0) = 0$.

The notion of successor of a pebble assignment. Let θ be an assignment of pebbles $i, i+1, \dots, k$ of $\mathcal{A}$ on a word w . That is, θ is a function from $\{i, i+1, \dots, k\}$ to $\{0, 1, \dots, |w|+1\}$. (Recall that positions 0 and $|w|+1$ contain the left- and right-end markers $\triangleleft$ and $\triangleright$, respectively.) If $0 \leq \theta(i) \leq |w|$, we define $\text{Succ}_i(\theta) = \theta'$, where for each $j \in \{i, i+1, \dots, k\}$,

$$\theta'(j) = \begin{cases} \theta(j) & \text{if } j \geq i+1 \\ \theta(i) + 1 & \text{if } j = i \end{cases}$$

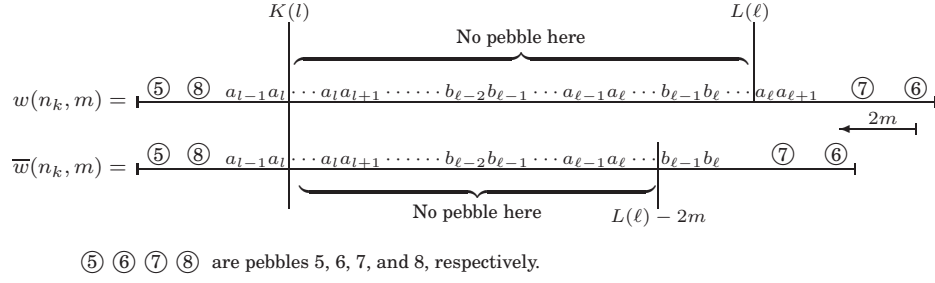
The notion of compatibility between two configurations. Let $i \geq 1$ and $[i, q, \theta]$ and $[i, \bar{q}, \bar{\theta}]$ be configurations of $\mathcal{A}$ on $w(n_k, m)$ and $\bar{w}(n_k, m)$, respectively, when pebble i is the head pebble. For an integer $l \in \{0, 1, \dots, n_k\}$, we say that the configurations $[i, q, \theta]$ and $[i, \bar{q}, \bar{\theta}]$ are *compatible* with respect to l , if

— $q = \bar{q}$;

and for each $j \in \{i, \dots, k\}$,

- either $\theta(j) \leq K(l)$ or $\theta(j) \geq L(l + n_i)$;
- either $\bar{\theta}(j) \leq K(l)$ or $\bar{\theta}(j) \geq L(l + n_i) - 2m$;
- if $\theta(j) \leq K(l)$, then $\bar{\theta}(j) \leq K(l)$ and $\theta(j) = \bar{\theta}(j)$;
- if $\bar{\theta}(j) \leq K(l)$, then $\theta(j) \leq K(l)$ and $\theta(j) = \bar{\theta}(j)$;
- if $\theta(j) \geq L(l + n_i)$, then $\bar{\theta}(j) \geq L(l + n_i) - 2m$ and $\theta(j) = \bar{\theta}(j) + 2m$;
- if $\bar{\theta}(j) \geq L(l + n_i) - 2m$, then $\theta(j) \geq L(l + n_i)$ and $\theta(j) = \bar{\theta}(j) + 2m$.

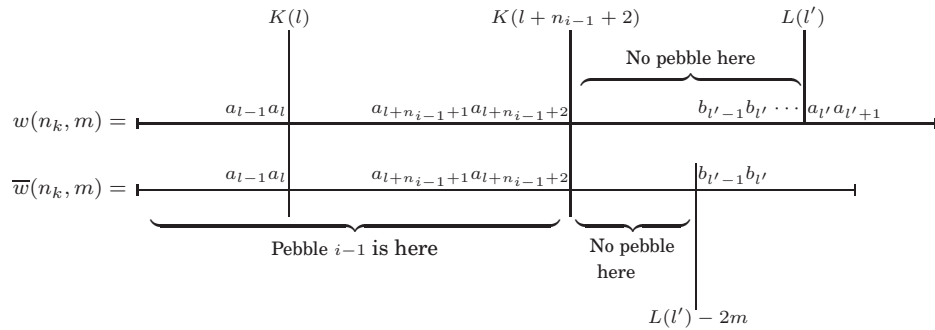
Below we give an illustration of the compatibility of two configurations of an 8-PA on $w(n_8, m)$ and $\bar{w}(n_8, m)$, respectively, with respect to l . The index ℓ is $l + n_5$.



CLAIM 1. *Suppose that $[i, q, \theta]$ and $[i, q, \bar{\theta}]$ are configurations of $\mathcal{A}$ on $w(n_k, m)$ and $\bar{w}(n_k, m)$, respectively. If $[i, q, \theta]$ and $[i, q, \bar{\theta}]$ are compatible with respect to some $l \in \{0, \dots, n_k\}$, then*

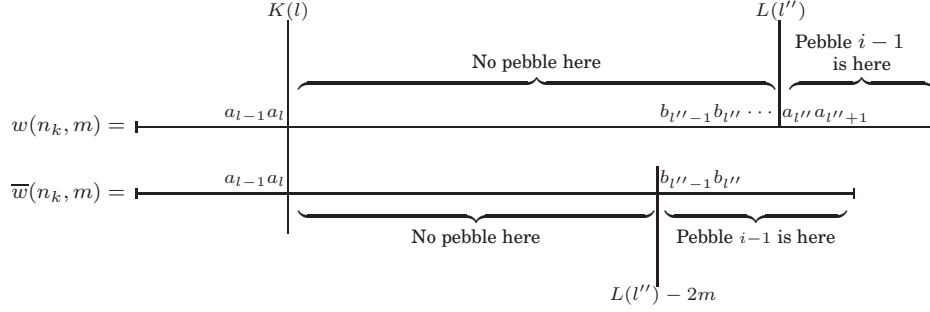
- (1) *for all $h \in \{0, \dots, K(l + n_{i-1} + 2)\}$ and for all $p \in Q$, the configuration $[i - 1, p, \theta \cup \{(i - 1, h)\}]$ (on $w(n_k, m)$) and the configuration $[i - 1, p, \bar{\theta} \cup \{(i - 1, h)\}]$ (on $\bar{w}(n_k, m)$) are compatible with respect to $l + n_{i-1} + 2$;*
- (2) *for all $h \in \{L(l + n_{i-1}), \dots, K(n_k)\}$ and for all $p \in Q$, the configuration $[i - 1, p, \theta \cup \{(i - 1, h)\}]$ (on $w(n_k, m)$) and the configuration $[i - 1, p, \bar{\theta} \cup \{(i - 1, h - 2m)\}]$ (on $\bar{w}(n_k, m)$) are compatible with respect to l .*

PROOF. It follows from the fact that $n_i = 2n_{i-1} + 2$. We prove it by picture here. For case (1), the proof is as follows. Let $l' = l + n_i$.



There is no pebble on the positions between $K(l + n_{i-1} + 2)$ and $L(l')$ in the word $w(n_k, m)$ as well as on the positions between $K(l + n_{i-1} + 2)$ and $L(l') - 2m$ in the word $\bar{w}(n_k, m)$ due to the assumption that $[i, q, \theta]$ and $[i, q, \bar{\theta}]$ are compatible with respect to l . Since $l' - (l + n_{i-1} + 2) = n_{i-1}$, the configuration $[i - 1, p, \theta \cup \{(i - 1, h)\}]$ (on $w(n_k, m)$) and the configuration $[i - 1, p, \bar{\theta} \cup \{(i - 1, h)\}]$ (on $\bar{w}(n_k, m)$) are compatible with respect to $l + n_{i-1} + 2$, for all $h \in \{0, \dots, K(l + n_{i-1} + 2)\}$ and for all $p \in Q$.

For case (2), the proof is as follows. We let $l'' = l + n_{i-1}$.



There is no pebble on the positions between $K(l)$ and $L(l'')$ in the word $w(n_k, m)$ as well as on the positions between $K(l)$ and $L(l'') - 2m$ in the word $\bar{w}(n_k, m)$ due to the assumption that $[i, q, \theta]$ and $[i, q, \bar{\theta}]$ are compatible with respect to l . Hence, case (2) follows immediately. This completes the proof of Claim 1. $\square$

Remark 3.11. Let $[i, q, \theta]$ and $[i, q, \bar{\theta}]$ be configurations of $\mathcal{A}$ on $w(n_k, m)$ and $\bar{w}(n_k, m)$, respectively and assume that they are compatible with respect to an integer l . Let $j, j' \in \{i, i+1, \dots, k\}$ and let

- x and y denote the symbols seen by pebbles j and j' , respectively, on $w(n_k, m)$ according to the configuration θ , and
- $\bar{x}$ and $\bar{y}$ denote the symbols seen by pebbles j and j' , respectively, on $\bar{w}(n_k, m)$ according to the configuration $\bar{\theta}$.

Then $x = y$ if and only if $\bar{x} = \bar{y}$.

The reason is as follows. Since $[i, q, \theta]$ and $[i, q, \bar{\theta}]$ are compatible with respect to l , we have the following four cases.

- (a) $\theta(j) \leq K(l)$ and $\theta(j') \leq K(l)$.
In this case, $\bar{\theta}(j) = \theta(j)$ and $\bar{\theta}(j') = \theta(j')$ and we immediately have $x = y$ if and only if $\bar{x} = \bar{y}$.
- (b) $\theta(j) \leq K(l)$ and $\theta(j') \geq L(l + n_i)$.
In this case, $\bar{\theta}(j) = \theta(j)$ and $\bar{\theta}(j') = \theta(j') - 2m$. Now in $w(n_k, m)$ and $\bar{w}(n_k, m)$ each symbol appears at most twice and they are of distance $4m - 2$ apart. Since $L(l + n_i) - K(l) > 4m - 2$, we have $x \neq y$. Similarly, $L(l + n_i) - 2m - K(l) > 4m - 2$, hence $\bar{x} = \bar{y}$.
- (c) $\theta(j) \geq L(l + n_i)$ and $\theta(j') \leq K(l)$.
The proof is similar to case (b) above.
- (d) $\theta(j) \geq L(l + n_i)$ and $\theta(j') \geq L(l + n_i)$.
In this case, $\bar{\theta}(j) = \theta(j) - 2m$ and $\bar{\theta}(j') = \theta(j') - 2m$ and we immediately have $x = y$ if and only if $\bar{x} = \bar{y}$.

Now this immediately implies that for every transition $\alpha \rightarrow \beta$ of the automaton $\mathcal{A}$, it applies to $[i, q, \theta]$ if and only if it applies to $[i, q, \bar{\theta}]$.

The following claim is important. However, due to the complexity of its proof, we postpone it until Subsection 3.3.

CLAIM 2. For each $i \in \{1, \dots, k\}$, and for every run of $\mathcal{A}$ on $w(n_k, m)$:

$$[i, p_0, \theta_0] \vdash_{\mathcal{A}, w(n_k, m)}^* [i, p_1, \theta_1] \vdash_{\mathcal{A}, w(n_k, m)}^* \dots \vdash_{\mathcal{A}, w(n_k, m)}^* [i, p_{N+1}, \theta_{N+1}] \quad (3)$$

where

- $N = K(n_k) = \text{length of } w(n_k, m)$;
- $\theta_0(i) = 0$;
- $\theta_{N+1}(i) = N + 1$;
- $\theta_{h+1} = \text{Succ}_i(\theta_h)$, for each $h \in \{0, \dots, N\}$ – that is, for each $j \in \{i + 1, \dots, k\}$, $\theta_0(j) = \dots = \theta_{N+1}(j)$ and $\theta_h(i) = h$, for each $h \in \{0, \dots, N + 1\}$;

if l is an integer such that

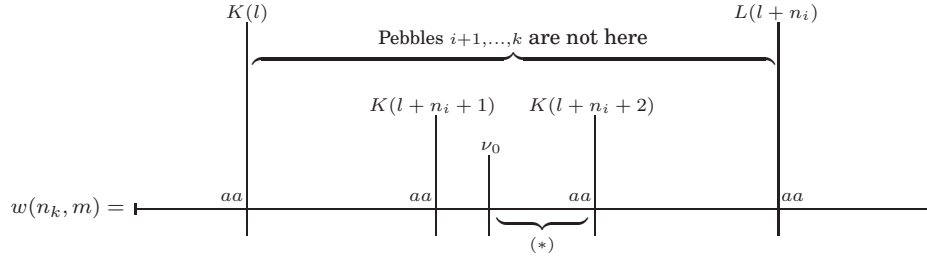
- (1) if $i = k$, then $l = 0$; and
- (2) if $i \neq k$, then l is an integer such that for each $j \in \{i + 1, \dots, k\}$, either $\theta_0(j) \leq K(l)$, or $\theta_0(j) \geq L(l + n_i) + 1$,

then there exist two positive integers ν_0 and ν such that

- $\nu = \pi \beta_{i-1}!$, where $1 \leq \pi \leq |Q|$;
- $K(l + n_{i-1} + 1) + 1 \leq \nu_0 \leq K(l + n_{i-1} + 1) + \beta_i$;
- for each h where $\nu_0 \leq h \leq K(l + n_{i-1} + 2) - \nu$, we have $p_h = p_{h+\nu}$.

In particular, since $\beta_{i+1} = |Q|! \times \beta_i!$ and $m = \beta_{k+1}$, we have ν divides β_{i+1} , and thus ν also divides m . Therefore, $p_{K(l+n_{i-1}+2)-2-2m} = p_{K(l+n_{i-1}+2)-2}$.

Below we give an illustration of the intuitive meaning of the indexes l, ν_0, ν in Claim 2 for $i \neq k$. Let l be the integer assumed in the hypothesis of Claim 2. (For simplicity, we do not put the indexes on the a 's.)



The meaning of Claim 2 is that in region (*) pebble i enters the same state every ν steps.

CLAIM 3. Let

$$[1, p_0, \theta_0] \vdash_{\mathcal{A}, w(n_k, m)} \dots [1, p_N, \theta_N] \vdash_{\mathcal{A}, w(n_k, m)} [1, p_{N+1}, \theta_{N+1}]$$

be a run of $\mathcal{A}$ on $w(n_k, m)$, where N is the length of $w(n_k, m)$ and $\theta_0(1) = 0$, and $\theta_{j+1} = \text{Succ}_1(\theta_j)$, for each $j \in \{0, \dots, N\}$; and let

$$[1, r_0, \bar{\theta}_0] \vdash_{\mathcal{A}, \bar{w}(n_k, m)} \dots [1, r_M, \bar{\theta}_M] \vdash_{\mathcal{A}, \bar{w}(n_k, m)} [1, r_{M+1}, \bar{\theta}_{M+1}]$$

be a run of $\mathcal{A}$ on $\bar{w}(n_k, m)$, where M is the length of $\bar{w}(n_k, m)$ and $\bar{\theta}_0(1) = 0$, and $\bar{\theta}_{j+1} = \text{Succ}_1(\bar{\theta}_j)$, for each $j \in \{0, \dots, M\}$.

If $[1, p_0, \theta_0]$ and $[1, r_0, \bar{\theta}_0]$ are compatible with respect to an $l \in \{0, \dots, n_k - n_1\}$, then $p_{N+1} = r_{M+1}$.

PROOF. Consider the run

$$[1, p_0, \theta_0] \vdash_{\mathcal{A}, w(n_k, m)} \dots [1, p_N, \theta_N] \vdash_{\mathcal{A}, w(n_k, m)} [1, p_{N+1}, \theta_{N+1}],$$

where $\theta_0(1) = 0$, and $\theta_{j+1} = \text{Succ}_1(\theta_j)$, for each $j \in \{0, \dots, N\}$; and the run

$$[1, r_0, \bar{\theta}_0] \vdash_{\mathcal{A}, \bar{w}(n_k, m)} \dots [1, r_M, \bar{\theta}_M] \vdash_{\mathcal{A}, \bar{w}(n_k, m)} [1, r_{M+1}, \bar{\theta}_{M+1}],$$

where $\bar{\theta}_0(1) = 0$, and $\bar{\theta}_{j+1} = \text{Succ}_1(\bar{\theta}_j)$, for each $j \in \{0, \dots, M\}$.

Suppose that $[1, p_0, \theta_0]$ and $[1, r_0, \theta_0]$ are compatible with respect to an integer l . This means that $p_0 = r_0$. We are going to show that $p_{N+1} = r_{M+1}$ in three stages. (In the following let $l' = l + 2$.)

Stage 1. $p_{K(l')} = r_{K(l')}$.

To prove this, we show that $p_h = r_h$, for each $h \in \{0, \dots, K(l')\}$. The proof is by induction on h . The proof for the base case, $h = 0$, follows from compatibility of $[1, p_0, \theta_0]$ and $[1, r_0, \bar{\theta}_0]$.

For the induction step, suppose that $p_h = r_h$. By Remark 3.11, a transition $\alpha \rightarrow \beta$ applies to $[1, p_h, \theta_h]$ if and only if it applies to $[1, r_h, \bar{\theta}_h]$. Hence, $p_{h+1} = r_{h+1}$.

Stage 2. $p_{K(l')-2} = p_{K(l')-2m-2} = r_{K(l')-2m-2}$.

In Stage 1, we already show that $p_{K(l')-2m-2} = r_{K(l')-2m-2}$. That $p_{K(l')-2} = p_{K(l')-2m-2}$ follows from Claim 2.

Stage 3. $p_{N+1} = r_{M+1}$.

We are going to prove that $p_h = r_{h-2m}$, for each $h \in \{K(l') - 2, \dots, N + 1\}$.

The proof is by induction on h . The proof for the base case, $h = K(l') - 2$, is already shown in Step 2.

For the induction step, suppose that $p_h = r_{h-2m}$. By Remark 3.11, a transition $\alpha \rightarrow \beta$ applies to $[1, p_h, \theta_h]$ if and only if it applies to $[1, r_{h-2m}, \bar{\theta}_{h-2m}]$. Thus, $p_{h+1} = r_{h+1}$.

This completes the proof of Claim 3. $\square$

The following claim is the generalisation of Claim 3 which implies Proposition 3.10.

CLAIM 4. *For each $i \in \{1, \dots, k\}$, the following holds. Let*

$$[i, p_0, \theta_0] \vdash_{\mathcal{A}, w(n_k, m)}^* \dots [i, p_N, \theta_N] \vdash_{\mathcal{A}, w(n_k, m)}^* [i, p_{N+1}, \theta_{N+1}]$$

be a run of $\mathcal{A}$ on $w(n_k, m)$, where N is the length of $w(n_k, m)$ and $\theta_0(i) = 0$, and $\theta_{j+1} = \text{Succ}_i(\theta_j)$, for each $j \in \{0, \dots, N\}$; and let

$$[i, r_0, \bar{\theta}_0] \vdash_{\mathcal{A}, \bar{w}(n_k, m)}^* \dots [i, r_M, \bar{\theta}_M] \vdash_{\mathcal{A}, \bar{w}(n_k, m)}^* [i, r_{M+1}, \bar{\theta}_{M+1}]$$

be a run of $\mathcal{A}$ on $\bar{w}(n_k, m)$, where M is the length of $\bar{w}(n_k, m)$ and $\bar{\theta}_0(i) = 0$, and $\bar{\theta}_{j+1} = \text{Succ}_i(\bar{\theta}_j)$, for each $j \in \{0, \dots, M\}$.

If $[i, p_0, \theta_0]$ and $[i, r_0, \bar{\theta}_0]$ are compatible with respect to an $l \in \{0, \dots, n_k - n_i\}$, then $p_{N+1} = r_{M+1}$.

PROOF. The proof is by induction on i . The basis is $i = 1$, which we have already proved in Claim 3.

For the induction hypothesis, we assume that Claim 4 holds for the case of $i - 1$. We are going to show that it holds for the case of i . The line of reasoning is almost the same as Claim 3. For completeness, we present it here.

Consider the following run

$$[i, p_0, \theta_0] \vdash_{\mathcal{A}, w(n_k, m)}^* \dots [i, p_N, \theta_N] \vdash_{\mathcal{A}, w(n_k, m)}^* [i, p_{N+1}, \theta_{N+1}]$$

and

$$[i, r_0, \bar{\theta}_0] \vdash_{\mathcal{A}, \bar{w}(n_k, m)}^* \dots \vdash_{\mathcal{A}, \bar{w}(n_k, m)}^* [i, r_{M+1}, \bar{\theta}_{M+1}].$$

By the assumption that $[i, p_0, \theta_0]$ and $[i, r_0, \theta_0]$ are compatible, we have $p_0 = r_0$. We are going to prove that $p_{N+1} = r_{M+1}$ in three stages. Let $l' = l + n_{i-1} + 2$.

Stage 1. $p_{K(l')-2} = r_{K(l')-2}$.

To prove this subclaim, we show that $p_h = r_h$, for each $h \in \{0, \dots, K(l')\}$. The proof is by induction on h . The proof for the base case $p_0 = r_0$ follows from the fact that $[i, p_0, \theta_0]$ and $[i, r_0, \theta_0]$ are compatible.

For the induction step, suppose that $p_h = r_h$. By the normalisation of the automaton $\mathcal{A}$, the run is of the form:

$$[i, p_h, \theta_h] \vdash_{\mathcal{A}, w(n_k, m)} [i-1, p'_0, \theta'_0] \vdash_{\mathcal{A}, w(n_k, m)}^* \dots \vdash_{\mathcal{A}, w(n_k, m)}^* [i-1, p'_{N+1}, \theta'_{N+1}]$$

and

$$[i, r_h, \bar{\theta}_h] \vdash_{\mathcal{A}, \bar{w}(n_k, m)} [i-1, r'_0, \bar{\theta}'_0] \vdash_{\mathcal{A}, \bar{w}(n_k, m)}^* \dots \vdash_{\mathcal{A}, \bar{w}(n_k, m)}^* [i, r'_{M+1}, \bar{\theta}'_{M+1}],$$

where $\theta'_h(i-1) = h$ for each $h \in \{1, \dots, N+1\}$ and $\bar{\theta}'_h(i-1) = h$ for each $h \in \{1, \dots, M+1\}$.

By determinism of $\mathcal{A}$, we have $p'_0 = r'_0$. Then, by Claim 1, since $0 \leq h \leq K(l')$, we have $[i-1, p'_0, \theta'_0]$ and $[i-1, r'_0, \bar{\theta}'_0]$ compatible with respect to l' . By the induction hypothesis of Claim 4, we have $p'_{N+1} = r'_{M+1}$. Then, by determinism of $\mathcal{A}$, we have $p_{h+1} = r_{h+1}$.

Stage 2. $p_{K(l')-2} = p_{K(l')-2m-2} = r_{K(l')-2m-2}$.

In Stage 1 we already have $p_{K(l')-2m-2} = r_{K(l')-2m-2}$. Claim 2 implies that

$$p_{K(l')-2} = p_{K(l')-2m-2}.$$

Stage 3. $p_{N+1} = r_{M+1}$.

By Stage 2, we have $p_{K(l')-2} = r_{K(l')-2m-2}$. We are going to prove that $p_h = r_{h-2m}$, for each $h \in \{K(l')-2, \dots, N+1\}$.

The proof is by induction on h . The proof for the base case, $h = K(l')$, follows from Stage 2.

For the induction step, suppose that $p_h = r_{h-2m}$. By the normalisation of the automaton $\mathcal{A}$, we assume that the run is of the form:

$$[i, p_h, \theta_h] \vdash_{\mathcal{A}, w(n_k, m)} [i-1, p'_0, \theta'_0] \vdash_{\mathcal{A}, w(n_k, m)}^* \dots \vdash_{\mathcal{A}, w(n_k, m)}^* [i-1, p'_{N+1}, \theta'_{N+1}]$$

and

$$[i, r_{h-2m}, \bar{\theta}_{h-2m}] \vdash_{\mathcal{A}, \bar{w}(n_k, m)} [i-1, r'_0, \bar{\theta}'_0] \vdash_{\mathcal{A}, \bar{w}(n_k, m)}^* \dots \vdash_{\mathcal{A}, \bar{w}(n_k, m)}^* [i, r'_{M+1}, \bar{\theta}'_{M+1}],$$

where $\theta'_h(i-1) = h$ for each $h \in \{1, \dots, N+1\}$ and $\bar{\theta}'_h(i-1) = h$ for each $h \in \{1, \dots, M+1\}$.

That we have $[i, p_h, \theta_h] \vdash_{\mathcal{A}, w(n_k, m)} [i-1, p'_0, \theta'_0]$ and $[i, r_{h-2m}, \bar{\theta}_{h-2m}] \vdash_{\mathcal{A}, \bar{w}(n_k, m)} [i-1, r'_0, \bar{\theta}'_0]$ is due to the normalisation of the automaton $\mathcal{A}$ described in the beginning of Subsection 3.2.

By determinism of $\mathcal{A}$, we have $p'_0 = r'_0$. Then, by Claim 1, since $h \geq K(l')$, we have $[i-1, p'_0, \theta'_0]$ and $[i-1, r'_0, \bar{\theta}'_0]$ compatible with respect to l . By the induction hypothesis of Claim 4, we have $p'_{N+1} = r'_{M+1}$. Then, by determinism of $\mathcal{A}$, we have $p_{h+1} = r_{h+1-2m}$.

This completes the proof of Claim 4. $\square$

PROOF. (of Proposition 3.10) We simply apply Claim 4, in which $i = k$, and both p_0, r_0 are the initial state q_0 of $\mathcal{A}$. Note that the initial configurations of $\mathcal{A}$ on $w(n_k, m)$ and $\bar{w}(n_k, m)$ are the same, thus, they are compatible. $\square$

3.3. Proof of Claim 2

In this subsection we are going to prove Claim 2. The proof is also rather long and technical. We need the following definition.

Definition 3.12. In the following, let $i \in \{1, \dots, k\}$.

- (1) An assignment $\theta : \{i, \dots, k\} \mapsto \{0, 1, \dots, K(n_k) + 1\}$ of pebbles $i, i + 1, \dots, k$ on $w(n_k, m)$ is called a *pebble- i assignment*.
- (2) For two pebble- i assignments θ_1 and θ_2 , we say that they have the same *pebble ordering*, if for each $j, j' \in \{i, i + 1, \dots, k\}$, $\theta_1(j) \leq \theta_1(j')$ if and only if $\theta_2(j) \leq \theta_2(j')$.

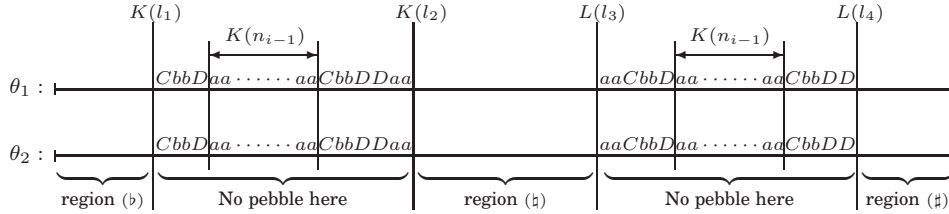
In this subsection we are going to prove Claim 2 together with Claim 5 below. In fact, we are going to prove *both* claims simultaneously. (We will give the structure of the proofs later on.)

CLAIM 5. Let $[i, q, \theta_1]$ and $[i, q, \theta_2]$ be configurations of $\mathcal{A}$ on $w(n_k, m)$ such that

- (1) θ_1 and θ_2 have the same pebble ordering;
- (2) for each $j \in \{i, \dots, k\}$, $\theta_1(j) \leq \theta_2(j)$;
- (3) there exist integers l_1, l_2, l_3, l_4 and π such that $l_1 \leq l_2 \leq l_3 \leq l_4$ and $1 \leq \pi < \frac{m}{\beta_{i-1}!}$ and for each $j \in \{i, \dots, k\}$,
 - (a) if $\theta_1(j) \leq K(l_1)$ or $\theta_1(j) \geq L(l_4) + 1$, then $\theta_1(j) = \theta_2(j)$;
 - (b) if $\theta_2(j) \leq K(l_1)$ or $\theta_2(j) \geq L(l_4) + 1$, then $\theta_1(j) = \theta_2(j)$;
 - (c) $l_2 - l_1 \geq n_{i-1} + 1$;
 - (d) $l_4 - l_3 \geq n_{i-1} + 1$;
 - (e) $\text{Image}(\theta_1) \cap (\{K(l_1) + 1, \dots, K(l_2)\} \cup \{L(l_3) + 1, \dots, L(l_4)\}) = \emptyset$;
 - (f) $\text{Image}(\theta_2) \cap (\{K(l_1) + 1, \dots, K(l_2)\} \cup \{L(l_3) + 1, \dots, L(l_4)\}) = \emptyset$;
 - (g) if $\theta_1(j) \in \{K(l_2) + 1, \dots, L(l_3)\}$, then $\theta_2(j) \in \{K(l_2) + 1, \dots, L(l_3)\}$ and $\theta_2(j) - \theta_1(j) = \pi \beta_{i-1}!$;
 - (h) if $\theta_2(j) \in \{K(l_2) + 1, \dots, L(l_3)\}$, then $\theta_1(j) \in \{K(l_2) + 1, \dots, L(l_3)\}$ and $\theta_2(j) - \theta_1(j) = \pi \beta_{i-1}!$.

If $[i, q, \theta_1] \vdash^* [i, p, \text{Succ}_i(\theta_1)]$ and $[i, q, \theta_2] \vdash^* [i, r, \text{Succ}_i(\theta_2)]$, then $p = r$.

Below we give an intuitive meaning of Claim 5. Consider the following illustration, where θ_1 and θ_2 are configurations on $w(n_k, m)$ with the same pebble ordering.



The meanings of l_1, l_2, l_3, l_4 and π are such that for each $j \in \{i, \dots, k\}$,

- if pebble j are found in region (b) on both configurations θ_1 and θ_2 , then $\theta_1(j) = \theta_2(j)$;
- if pebble j are found in region (b) on both configurations θ_1 and θ_2 , then $\theta_2(j) - \theta_1(j) = \pi \beta_{i-1}!$;
- if pebble j are found in region (b) on both configurations θ_1 and θ_2 , then $\theta_2(j) = \theta_1(j)$.

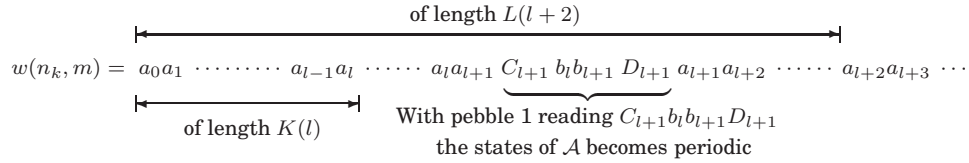
On both configurations θ_1 and θ_2 no pebbles are found in the region between $K(l_1) + 1$ and $K(l_2)$ as well as in between $L(l_3) + 1$ and $L(l_4)$. Claim 5 states that both configurations $[i, q, \theta_1]$ and $[i, q, \theta_2]$ are essentially the “same.” In the sense that if $[i, q, \theta_1] \vdash^* [i, p, \text{Succ}_i(\theta_1)]$ and $[i, q, \theta_2] \vdash^* [i, r, \text{Succ}_i(\theta_2)]$, then $p = r$.

The proofs of both Claims 2 and 5 use a rather involved inductive argument. In fact, we are going to prove *both* claims simultaneously by induction. The induction step on the proof of each claim uses the induction hypothesis of both claims. The overall structure of the proofs of both Claims 2 and 5 is as follows.

- (1) We prove the base case $i = 1$ of Claim 2.
- (2) We prove the base case $i = 1$ of Claim 5.
- (3) For the induction hypothesis, we assume that *both* Claims 2 and 5 hold for the case i .
- (4) For the induction step, we prove Claim 2 for the case $i + 1$.
This step uses the hypothesis that both Claims 2 and 5 hold for case i .
- (5) For the other induction step, we prove Claim 5 for the case $i + 1$.
As in Step 4, this step uses the hypothesis that both Claims 2 and 5 hold for case i .

Proof of the base case $i = 1$ for Claim 2. Let l be an integer such that for each $j \in \{2, \dots, k\}$, either $\theta(j) \leq K(l)$ or $\theta(j) \geq L(l + 2)$, where the number 2 comes from $n_1 = 2$.

The symbols in $C_{l+1}b_l b_{l+1}D_{l+1}$ are different from all the symbols seen by pebbles $2, \dots, k$. We are going to show that when reading $C_{l+1}b_l b_{l+1}D_{l+1}$, pebble 1 enters into a loop of states. See the illustration below.



On reading the segment $C_{l+1}b_l b_{l+1}D_{l+1}$, the transitions used are of the form $(1, \emptyset, \emptyset, s) \rightarrow (s', \text{right})$. Due to the determinism of the automaton $\mathcal{A}$, there exist integers ν_0 and ν such that $\nu_0, \nu \leq |Q|$ and for each h where $\nu_0 \leq h \leq K(l + n_{i-1} + 2) - \nu$, we have $p_h = p_{h+\nu}$. In particular, since $\beta_2 = |Q|! \times \beta_1!$, we have ν divides β_2 . Furthermore, β_2 also divides $m = \beta_{k+1}$, thus, ν divides m , therefore, $p_{K(l+n_{i-1}+2)-2-2m} = p_{K(l+n_{i-1}+2)-2}$.

Proof of the base case $i = 1$ for Claim 5. Suppose $[1, q, \theta_1]$ and $[1, q, \theta_2]$ are configurations of $\mathcal{A}$ on $w(n_k, m)$ and l_1, l_2, l_3, l_4, π are integers such that the conditions (1), (2), (3.a)-(3.h) above hold. Moreover, suppose also that

$$[1, q, \theta_1] \vdash [1, p, \text{Succ}_1(\theta_1)] \quad \text{and} \quad [1, q, \theta_2] \vdash [1, r, \text{Succ}_1(\theta_2)].$$

We are going to show that $p = r$.

By conditions (3.e) and (3.f), there can only be three cases: $\theta_1(1) \leq K(l_1)$, $K(l_2) + 1 \leq \theta_1(1) \leq L(l_3)$, and $\theta_1(1) \geq L(l_4) + 1$.

Case 1. $\theta_1(1) \leq K(l_1)$.

By condition (3.a), we have $\theta_1(1) = \theta_2(1)$. By conditions (1), (2), (3.a) and (3.b), for any $j \in \{2, \dots, k\}$, we have

$$\theta_1(j) = \theta_1(1) \quad \text{if and only if} \quad \theta_2(j) = \theta_2(1). \quad (4)$$

By condition (3.c), $l_2 - l_1 \geq 1$. Moreover, no symbol in $C_{l_2}b_{l_2-1}b_{l_2}D_{l_2} \dots a_{n_k-1}a_{n_k}$ appears in $a_0a_1 \dots a_{l_1-1}a_{l_1}$, and by conditions (3.e) and (3.f), no pebbles are placed

on $C_{l_1} \cdots a_{l_2-1} a_{l_2}$. Therefore, for any $j \in \{2, \dots, k\}$,

$$\begin{aligned} &\text{pebbles } j \text{ and } 1 \text{ read the same symbol in the configuration } [1, q, \theta_1] \\ &\quad \text{if and only if} \\ &\text{pebbles } j \text{ and } 1 \text{ read the same symbol in the configuration } [1, q, \theta_2] \end{aligned} \quad (5)$$

Thus, by Equalities 4 and 5, the same transition applies to both $[1, q, \theta_1]$ and $[1, q, \theta_2]$. Since $\mathcal{A}$ is deterministic, we have $p = r$.

Case 2. $K(l_2) + 1 \leq \theta_1(1) \leq L(l_3)$.

That is, $\theta_2(1) = \theta_1(1) + \pi\beta_{i-1}!$, where $1 \leq \pi\beta_{i-1}! < m$. By the same conditions (3.g) and (3.h), for any $j \in \{2, \dots, k\}$,

$$\theta_1(j) = \theta_1(1) \quad \text{if and only if} \quad \theta_2(j) = \theta_2(1). \quad (6)$$

By condition (3.c), $l_2 - l_1 \geq 1$. Moreover, any symbol in $C_{l_2} b_{l_2-1} b_{l_2} D_{l_2} \cdots a_{n_k-1} a_{n_k}$ does not appear in $a_0 a_1 \cdots a_{l_1-1} a_{l_1}$. Therefore, for any $j \in \{2, \dots, k\}$, if pebbles j and 1 read the same symbol in the configuration $[1, q, \theta_1]$, then $K(l_2) + 1 \leq \theta_1(j) \leq L(l_3)$; and similarly, if pebbles j and 1 read the same symbol in the configuration $[1, q, \theta_2]$, then $K(l_2) + 1 \leq \theta_2(j) \leq L(l_3)$. By conditions (3.g) and (3.h), $\theta_2(j) = \theta_1(j) + \pi\beta_{i-1}!$. Due to the definition of $w(n_k, m)$, we have

$$\begin{aligned} &\text{pebbles } j \text{ and } 1 \text{ read the same symbol in the configuration } [1, q, \theta_1] \\ &\quad \text{if and only if} \\ &\text{pebbles } j \text{ and } 1 \text{ read the same symbol in the configuration } [1, q, \theta_2] \end{aligned} \quad (7)$$

Thus, by Equalities 6 and 7, the same transition applies to both $[1, q, \theta_1]$ and $[1, q, \theta_2]$. Since $\mathcal{A}$ is deterministic, we have $p = r$.

Case 3. $\theta_1(1) \geq L(l_4) + 1$.

The proof is similar to the one for Case 1 above, thus, omitted.

This completes the proof of the base case $i = 1$ for Claim 5.

The induction hypothesis. Both Claims 2 and 5 hold for case i .

The induction step for Claim 2. We are going to show that Claim 2 holds for the case $i + 1$.

Suppose we have the following run:

$$[i + 1, p_0, \theta_0] \vdash_{\mathcal{A}, w(n_k, m)}^* [i + 1, p_1, \theta_1] \vdash_{\mathcal{A}, w(n_k, m)}^* \cdots \vdash_{\mathcal{A}, w(n_k, m)}^* [i + 1, p_{N+1}, \theta_{N+1}]$$

Let l be the integer as stated in Claim 2. Since $m > |Q|\beta_i!$, there exists a pair (η, η') of indexes such that

- $K(l + n_i + 1) + 1 \leq \eta < \eta' \leq K(l + n_i + 2) - 2$;
- $\eta' - \eta = \pi\beta_i!$, where $1 \leq \pi \leq |Q|$;
- $p_\eta = p_{\eta'}$.

We pick such pair (η, η') in which η is the smallest. We claim that $\nu_0 = \eta$ and $\nu = \eta' - \eta$ are the desired two integers in Claim 2.

We are going to show that for each $h \in \{\nu_0, \dots, K(l + n_i + 2) - 2 - \nu\}$,

$$\text{if } p_h = p_{h+\nu}, \text{ then } p_{h+1} = p_{h+\nu+1}. \quad (8)$$

Since by definition of ν_0 and ν , we already have $p_{\nu_0} = p_{\nu_0+\nu}$, this immediately implies that for each $h \in \{\nu_0, \dots, K(l + n_i + 2) - 2 - \nu\}$, $p_h = p_{h+\nu}$.

To prove Equality 8, suppose $p_h = p_{h+\nu}$. Consider the following run:

- $[i + 1, p_h, \theta_h] \vdash [i, s_0, \theta_h \cup \{(i, 0)\}]$;
- $[i, s_0, \theta_h \cup \{(i, 0)\}] \vdash^* \cdots \vdash^* [i, s_{N+1}, \theta_h \cup \{(i, N + 1)\}]$

— $[i, s_{N+1}, \theta_h \cup \{(i, N+1)\}] \vdash [i+1, s', \theta_h] \vdash [i+1, p_{h+1}, \theta_{h+1}]$.

and the following run:

— $[i+1, p_{h+\nu}, \theta_{h+\nu}] \vdash [i, t_0, \theta_{h+\nu} \cup \{(i, 0)\}]$;
 — $[i, t_0, \theta_{h+\nu} \cup \{(i, 0)\}] \vdash^* \dots \vdash^* [i, t_{N+1}, \theta_{h+\nu} \cup \{(i, N+1)\}]$
 — $[i, t_{N+1}, \theta_{h+\nu} \cup \{(i, N+1)\}] \vdash [i+1, t', \theta_h] \vdash [i+1, p_{h+\nu+1}, \theta_{h+\nu+1}]$.

Since $p_h = p_{h+\nu}$ and $\mathcal{A}$ is deterministic, we have $s_0 = t_0$. Our aim is to prove that $s_{N+1} = t_{N+1}$. To this end, there are a few steps.

Step 1 (Application of the hypothesis that Claim 5 holds for the case i). For each $j \in \{0, \dots, K(l + n_{i-1} + 2)\}$, we claim that $s_j = t_j$.

To apply the induction hypothesis that Claim 5 for the case i , we take the integers

$$\begin{aligned} l_1 &= l + n_{i-1} + 2 \\ l_2 &= l + n_i + 1 \\ l_3 &= l_2 \\ l_4 &= l + n_{i+1} \end{aligned}$$

Recall that l is the integer such that every pebble, except pebbles i and $(i+1)$, are located either $\leq K(l)$, or $\geq L(l + n_{i+1})$. Recall also that $\nu = \pi\beta_i!$.

It is straightforward to show that $l_2 - l_1 \geq n_{i-1} + 1$ and $l_4 - l_3 \geq n_{i-1} + 1$, and all the conditions (1), (2) and (3.a)–(3.h) hold. Since $s_0 = t_0$, applying the hypothesis for each $j \in \{0, \dots, K(l + n_{i-1} + 2)\}$ – that Claim 5 hold for the case i – we have $s_j = t_j$.

Step 2 (Application of the hypothesis that Claim 2 holds for the case i). For each $j \in \{K(l + n_{i-1} + 1) + 1, \dots, K(l + n_{i-1} + 2) - 2\}$, in the configuration $[i, s_j, \theta_h \cup \{(i, j)\}]$ the integer l satisfies the condition that each pebbles $i+1, \dots, k$ are located either $\leq K(l)$, or $L(l + n_i)$.

Applying the induction hypothesis that Claim 2 holds for the case i , there exist two integers ν'_0 and ν' such that

— $K(l + n_{i-1} + 1) + 1 \leq \nu'_0 \leq K(l + n_{i-1} + 1) + \beta_i$;
 — $1 \leq \nu' \leq \beta_i$;

— $s_j = s_{j+\nu'}$, for each $j \in \{K(l + n_{i-1} + 1) + 1, \dots, K(l + n_{i-1} + 2) - \nu' - 2\}$.

In particular, ν' divides β_{i+1} , by definition of β_{i+1} , thus, $s_j = s_{j+\nu}$, for each $j \in \{K(l + n_{i-1} + 1) + 1, \dots, K(l + n_{i-1} + 2) - \nu - 2\}$.

Similarly, we can show that $t_j = t_{j+\nu}$, for each $j \in \{K(l + n_{i-1} + 1) + 1, \dots, K(l + n_{i-1} + 2) - \nu - 2\}$.

Step 3 (Application of the hypothesis that Claim 5 holds for the case i). For each $j \in \{K(l + n_{i-1} + 1) + \nu_0, \dots, L(n_i + 2 + n_{i-1} + 1)\}$, we claim that $s_j = t_{j+\nu}$.

To apply the induction hypothesis that Claim 5 for the case i , we take the following integers.

$$\begin{aligned} l_1 &= l \\ l_2 &= l + n_{i-1} + 1 \\ l_3 &= l + n_i + 2 + n_{i-1} + 1 \\ l_4 &= l + n_{i+1} \end{aligned}$$

It is straightforward to show that $l_2 - l_1 \geq n_{i-1} + 1$ and $l_4 - l_3 \geq n_{i-1} + 1$, and all the conditions (1), (2) and (3.a)–(3.h) hold.

From Steps 1 and 2, we already have

$$s_{K(l+n_{i-1}+1)+\nu_0} = t_{K(l+n_{i-1}+1)+\nu_0}$$

$$s_{K(l+n_{i-1}+1)+\nu_0+\nu} = t_{K(l+n_{i-1}+1)+\nu_0+\nu}$$

$$s_{K(l+n_{i-1}+1)+\nu_0} = s_{K(l+n_{i-1}+1)+\nu_0+\nu}$$

Applying the hypothesis for each $j \in \{K(l+n_{i-1}+1)+\nu_0, \dots, L(n_i+2+n_{i-1}+1)-\nu\}$ – that Claim 5 hold for the case i – on the configurations $[i, s_j, \theta_h \cup \{(i, j)\}]$ and $[i, t_{j+\nu}, \theta_h \cup \{(i, j+\nu)\}]$, we have $s_j = t_{j+\nu}$.

Step 4 (Application of the hypothesis that Claim 2 holds for the case i). For each $j \in \{K(l+n_i+2+n_{i-1}+1)+1, \dots, K(l+n_i+2+n_{i-1}+2)-2\}$, in the configuration $[i, s_j, \theta_h \cup \{(i, j)\}]$ the integer $l+n_i+2$ satisfies the condition that each pebbles $i+1, \dots, k$ are located either $\leq K(l+n_i+2)$, or $\geq L(l+n_i+2+n_i) = L(l+n_{i+1})$. Applying the induction hypothesis that Claim 2 holds for the case i , there exist two integers ν''_0 and ν'' such that

$$-K(l+n_i+2+n_{i-1}+1)+1 \leq \nu''_0 \leq K(l+n_i+2+n_{i-1}+1)+\beta_i;$$

$$-1 \leq \nu'' \leq \beta_i;$$

$$-s_j = s_{j+\nu''}, \text{ for each } j \in \{K(l+n_i+2+n_{i-1}+1)+1, \dots, K(l+n_i+2+n_{i-1}+1)-\nu''-2\}.$$

In particular, ν'' divides β_{i+1} , and by definition of β_{i+1} , thus, $s_j = s_{j+\nu}$, for each $j \in \{K(l+n_i+2+n_{i-1}+1)+1, \dots, K(l+n_i+2+n_{i-1}+2)-\nu-2\}$.

Similarly, we can show that $t_j = t_{j+\nu}$, for each $j \in \{K(l+n_i+2+n_{i-1}+1)+1, \dots, K(l+n_i+2+n_{i-1}+2)-\nu-2\}$. In particular, we have

$$s_{K(l+n_i+2+n_{i-1}+2)-2} = t_{K(l+n_i+2+n_{i-1}+2)-2}.$$

By definition of $L(\cdot)$ and $K(\cdot)$, this is equivalent to stating that

$$s_{L(l+n_i+2+n_{i-1}+1)} = t_{L(l+n_i+2+n_{i-1}+1)}.$$

Step 5 (Application of the hypothesis that Claim 5 holds for the case i). For each $j \in \{L(l+n_{i-1}+1), \dots, N+1\}$, we claim that $s_j = t_j$.

To apply the induction hypothesis that Claim 5 for the case i , we take the integers

$$\begin{aligned} l_1 &= l \\ l_2 &= l+n_{i-1}+1 \\ l_3 &= l_2 \\ l_4 &= l_3+n_{i-1}+1 \end{aligned}$$

It is straightforward to show that $l_2 - l_1 \geq n_{i-1} + 1$ and $l_4 - l_3 \geq n_{i-1} + 1$, and all the conditions (1), (2) and (3.a)–(3.h) hold.

By Step 4, we already have $s_{L(l+n_i+2+n_{i-1}+1)} = t_{L(l+n_i+2+n_{i-1}+1)}$. Applying the hypothesis for each $j \in \{L(l+n_{i-1}+1), \dots, N+1\}$ – that Claim 5 hold for the case i – on the configurations $[i, s_j, \theta_h \cup \{(i, j)\}]$ and $[i, t_j, \theta_h \cup \{(i, j)\}]$, we have $s_j = t_j$.

From here, as $s_{N+1} = t_{N+1}$ and $\mathcal{A}$ is deterministic, we have $s' = t'$. And again, by the determinism of $\mathcal{A}$, this implies $p_{h+1} = p_{h+1+\nu}$. This completes the induction step for Claim 2.

The induction step for Claim 5. We are going to show that Claim 5 holds for the case $i+1$.

Suppose $[i+1, q, \theta_1]$ and $[i+1, q, \theta_2]$ are configurations of $\mathcal{A}$ on $w(n_k, m)$ such that the conditions (1), (2), (3.a)–(3.g) above hold.

Consider the following run:

$$\begin{aligned} &-[i+1, q, \theta_1] \vdash [i, s_0, \theta_h \cup \{(i, 0)\}]; \\ &-[i, s_0, \theta_h \cup \{(i, 0)\}] \vdash^* \dots \vdash^* [i, s_{N+1}, \theta_1 \cup \{(i, N+1)\}] \end{aligned}$$

— $[i, s_{N+1}, \theta_1 \cup \{(i, N+1)\}] \vdash [i+1, s', \theta_1] \vdash [i+1, p, \text{Succ}_{i+1}(\theta_1)]$.

and the following run:

— $[i+1, q, \theta_2] \vdash [i, t_0, \theta_2 \cup \{(i, 0)\}]$;
 — $[i, t_0, \theta_2 \cup \{(i, 0)\}] \vdash^* \dots \vdash^* [i, t_{N+1}, \theta_2 \cup \{(i, N+1)\}]$
 — $[i, t_{N+1}, \theta_2 \cup \{(i, N+1)\}] \vdash [i+1, t', \theta_2] \vdash [i+1, r, \text{Succ}_{i+1}(\theta_2)]$.

We are going to show that $p = r$. It can be proved in a similar manner as in the proof of the induction step of Claim 2, thus, omitted.

Briefly, the proof is divided into the same Steps 1–5 above. The reasoning on each step still applies in this induction step, and at the end we obtain $s_{N+1} = t_{N+1}$, thus, $s' = t'$ and $p = r$.

4. WEAK PA

There is an analogue of our results from the previous section to another, but weaker, version of pebble automata. In the model defined in Section 2, the new pebble is placed in the beginning of the input word. This model is called *strong* PA in [Neven et al. 2004]. An alternative would be to place the new pebble at the position of the most recent one. The model defined this way is usually referred as *weak* PA. Formally, it is defined by setting $\theta'(i-1) = \theta(i)$ (and keeping $\theta'(i) = \theta(i)$) in the case of $\text{act} = \text{place-pebble}$ in the definition of the transition relation in Definition 2.1.

We give the formal definition below.

Definition 4.1. A *one-way alternating weak k -pebble automaton*, (in short weak k -PA) is a system $\mathcal{A} = \langle Q, q_0, F, \mu, U \rangle$ whose components are defined as follows.

- (1) $Q, q_0 \in Q$ and $F \subseteq Q$ are a finite set of *states*, the *initial state*, and the set of *final states*, respectively;
- (2) $U \subseteq Q - F$ is the set of *universal states*; and
- (3) μ is a finite set of transitions of the form $\alpha \rightarrow \beta$ such that
 - α is of the form (i, P, V, q) , where $i \in \{1, \dots, k\}$, $P, V \subseteq \{i+1, \dots, k\}$, $q \in Q$ and
 - β is of the form (q, act) , where $q \in Q$ and
$$\text{act} \in \{\text{right}, \text{place-pebble}, \text{lift-pebble}\}.$$

The definitions of pebble assignment, configurations, initial and final configurations as well as application of a transition on configurations are the same as defined in the case of strong PA in Subsection 2.1.

We define the transition relation $\vdash_{\mathcal{A}}$ on $\langle w \rangle$ as follows: $[i, q, \theta] \vdash_{\mathcal{A}, w} [i', q', \theta']$, if there is a transition $\alpha \rightarrow (p, \text{act}) \in \mu$ that applies to $[i, q, \theta]$ such that $q' = p$, for all $j > i$, $\theta'(j) = \theta(j)$, and

- if $\text{act} = \text{right}$, then $i' = i$ and $\theta'(i) = \theta(i) + 1$,
- if $\text{act} = \text{lift-pebble}$, then $i' = i + 1$,
- if $\text{act} = \text{place-pebble}$, then $i' = i - 1$, $\theta'(i-1) = \theta(i)$ and $\theta'(i) = \theta(i)$.

Note the difference on the definition of θ' for the case of $\text{act} = \text{place-pebble}$ from the one in the case of strong PA in Subsection 2.1.

THEOREM 4.2. [Tan 2010, Theorem 3] *For each $k \geq 1$, one-way alternating, non-deterministic and deterministic weak k -PA have the same recognition power.*

However, weak k -PA is weaker than strong k -PA. For example, $\mathcal{R}_{2^{k-1}}$ is not a weak k -PA language, see Lemma 4.3 below.

Let

$$\text{wPA}_k = \{L \mid L \text{ is accepted by a weak } k\text{-PA}\}$$

and

$$\text{wPA} = \bigcup_{k \geq 1} \text{wPA}_k$$

The following lemma is the weak PA version of Proposition 3.1 and Corollary 3.3.

LEMMA 4.3. *For each $k = 1, 2, \dots$, $\mathcal{R}_k^+ \in \text{wPA}_k$, but $\mathcal{R}_{k+1}^+ \notin \text{wPA}_k$.*

PROOF. First, we prove that $\mathcal{R}_k^+ \in \text{wPA}_k$. The weak k -PA $\mathcal{A}$ that accepts $\mathcal{R}_k^+$ works as follows. On an input word $w = a_0 b_0 \dots a_n b_n$, it works as follows.

- (1) It places pebble k on the second position to read the symbol b_0 .
- (2) For each $i = k - 1, \dots, 1$, it does the following.
 - (a) Place pebble i , and non-deterministically moves it right until it finds an odd position that contains the same symbol read by pebble $i + 1$.
 - (b) If it finds such position, it moves pebble i one step to the right.
 - (c) If it cannot find such position, it rejects the input word.
- (3) If at the end, pebble 1 is on the last position, then the automaton accepts the input word.

It is quite straightforward to show that the automaton $\mathcal{A}_k$ accepts $\mathcal{R}_k^+$.

Now we prove that $\mathcal{R}_{k+1}^+ \notin \text{wPA}_k$. Suppose to the contrary that there is a weak k -PA $\mathcal{A}$ that accepts $\mathcal{R}_{k+1}^+$. By adding some extra states, we can normalise the behaviour of each pebble as follows. For each $i \in \{1, \dots, k\}$, pebble i behaves as follows.

- After pebble i moves right, then pebble $(i - 1)$ (when $i > 1$) is immediately placed (in position 0 reading the left end-marker $\triangleleft$).
- If $i < k$, pebble i is lifted only when it reaches the right-end marker $\triangleright$ of the input.
- Immediately after pebble i is lifted, pebble $(i + 1)$ moves right.

We also assume that in the automaton $\mathcal{A}$ only pebble k can enter a final state and it may do so only after it reads the right-end marker $\triangleright$ of the input.

We let $m = \beta_{k+1}$, as defined in Subsection 3.2, where $\beta_0 = 1$, $\beta_1 = |Q|$, and for $i \geq 2$,

$$\beta_i = |Q|! \times \beta_{i-1}!$$

Also recall that the words $w(k + 1, m)$ and $\bar{w}(k + 1, m)$ are defined as follows.

$$\begin{aligned} w(k + 1, m) &= a_0 a_1 C_1 b_0 b_1 D_1 \dots a_{k-1} a_k C_k b_{k-1} b_k D_k a_k a_{k+1} \\ \bar{w}(k + 1, m) &= a_0 a_1 C_1 b_0 b_1 D_1 \dots a_{k-1} a_k C_k b_{k-1} b_k, \end{aligned}$$

where for each $i = 1, \dots, k$,

- $C_i = c_{i,1} c_{i+1,1} \dots c_{i,m-1} c_{i+1,m-1}$;
- $D_i = d_{i,1} d_{i+1,1} \dots d_{i,m-1} d_{i+1,m-1}$.

Obviously $w(k + 1, m) \in \mathcal{R}_{k+1}^+$, while $\bar{w}(k + 1, m) \notin \mathcal{R}^+$. We establish the following claim that immediately implies $\mathcal{R}_{k+1}^+ \notin \text{wPA}_k$.

CLAIM 6. *The automaton $\mathcal{A}$ either accepts both $w(k + 1, m)$ and $\bar{w}(k + 1, m)$, or rejects both $w(k + 1, m)$ and $\bar{w}(k + 1, m)$.*

PROOF. The proof is similar to the proof of Proposition 3.10. So we simply sketch it here. Let

$$[k, p_0, \bar{\theta}_0] \vdash_{\mathcal{A}, w(n_k, m)}^* \dots \vdash_{\mathcal{A}, w(k, m)}^* [k, p_{N+1}, \bar{\theta}_{N+1}]$$

be a run of $\mathcal{A}$ on $w(k + 1, m)$, where N is the length of $w(k + 1, m)$ and $\theta_j(k) = j$, for each $j \in \{0, \dots, N + 1\}$.

Let

$$[k, r_0, \bar{\theta}_0] \vdash_{\mathcal{A}, \bar{w}(k+1, m)}^* \cdots \vdash_{\mathcal{A}, \bar{w}(k+1, m)}^* [k, r_{M+1}, \bar{\theta}_{M+1}]$$

be a run of $\mathcal{A}$ on $\bar{w}(k+1, m)$, where M is the length of $\bar{w}(k+1, m)$ and $\theta_j(k) = j$, for each $j \in \{0, \dots, M+1\}$.

Now $p_0 = r_0$, as both are the initial state of $\mathcal{A}$. We are going to show that $p_{N+1} = r_{M+1}$. It consists of three steps.

Step 1. $p_m = r_m$.

This step is similar to Claim 4 proved in Subsection 3.2. That is, suppose $[k, q, \theta]$ and $[k, q, \bar{\theta}]$ are configurations on $w(k+1, m)$ and $\bar{w}(k+1, m)$, respectively, and $0 \leq \theta(k) = \bar{\theta}(k) \leq m$. If

$$\begin{aligned} [k, q, \theta] &\vdash_{\mathcal{A}, w(k+1, m)}^* [k, p, Succ_k(\theta)] \\ [k, q, \bar{\theta}] &\vdash_{\mathcal{A}, \bar{w}(k+1, m)}^* [k, r, Succ_k(\theta)] \end{aligned}$$

then $p = r$.[¶]

Step 2. $r_m = p_m = p_{2m}$.

This step is similar to Claim 2 stated in Subsection 3.2. That is, there exist two integers ν_0 and ν such that for every $h \in \{m + \nu_0, \dots, 2m - \nu\}$, we have $p_h = p_{h+\nu}$. The main idea is that since the integer m is big enough, there exists an integer ν such that on every ν steps, pebble k will enter into the same state. The integer m is defined so that it is divisible by every possible such ν , thus, implies $p_m = p_{2m}$. That $r_m = p_m$ is deduced from the previous step.

Step 3. $p_{N+1} = r_{M+1}$.

Here we make use of the fact that $\mathcal{A}$ is a weak PA. From previous step we have $p_{2m} = r_m$. On the configuration $[k, p_{2m}, \theta_{2m}]$ of $\mathcal{A}$ on $w(k+1, m)$, pebble k only “sees” $a_1 a_2 C_2 b_1 b_2 D_2 \cdots a_{k-1} a_k C_k b_{k-1} b_k D_k a_k a_{k+1}$; while on the configuration $[k, r_m, \theta_m]$ of $\mathcal{A}$ on $\bar{w}(k+1, m)$, pebble k only “sees” $b_0 b_1 D_1 \cdots a_{k-1} a_k C_k b_{k-1} b_k$.

Since $a_1 a_2 C_2 b_1 b_2 D_2 \cdots a_{k-1} a_k C_k b_{k-1} b_k D_k a_k a_{k+1}$ and $b_0 b_1 D_1 \cdots a_{k-1} a_k C_k b_{k-1} b_k$ are essentially the same, we have $p_{2m+1} = r_{m+1}$. Similarly, from $p_{2m+1} = r_{m+1}$, we also can conclude that $p_{2m+2} = r_{m+2}$ and then $p_{2m+3} = r_{m+3}$ and so on until we get $p_{N+1} = r_{M+1}$.

□

This completes the proof of Lemma 4.3. □

Lemma 4.3 immediately implies the strict hierarchy for wPA languages.

THEOREM 4.4. *For each $k = 1, 2, \dots$, $wPA_k \subsetneq wPA_{k+1}$.*

5. LINEAR TEMPORAL LOGIC WITH ONE REGISTER FREEZE QUANTIFIER

In this section we recall the definition of Linear Temporal Logic (LTL) augmented with one register freeze quantifier [Demri and Lazić 2009]. We consider only one-way temporal operators “next” X and “until” U , and do not consider their past time counterparts. Moreover, in [Demri and Lazić 2009] the LTL model is defined over data words. Since in this paper we essentially ignore the finite labels, the LTL model presented here also ignores the finite labels. However, the result here can be adopted in a straightforward manner for the data word model.

[¶]The only difference between this proof and the proof of Claim 4 is that here the induction hypothesis is that for each $1 \leq i \leq k-1$, weak i -PA cannot differentiate between $w(i+1, m)$ and $\bar{w}(i+1, m)$; while in Claim 4 the induction hypothesis is strong i -PA cannot differentiate between $w(n_i, m)$ and $\bar{w}(n_i, m)$.

Roughly, the logic $LTL_1^\downarrow(X, U)$ is the standard LTL augmented with a register to store a symbol from the infinite alphabet. Formally, the formulas are defined as follows.

- Both True and False belong to $LTL_1^\downarrow(X, U)$.
- $\uparrow$ is in $LTL_1^\downarrow(X, U)$.
- If φ, ψ are in $LTL_1^\downarrow(X, U)$, then so are $\neg\varphi$, $\varphi \vee \psi$ and $\varphi \wedge \psi$.
- If φ is in $LTL_1^\downarrow(X, U)$, then so is $X\varphi$.
- If φ is in $LTL_1^\downarrow(X, U)$, then so is $\downarrow\varphi$.
- If φ, ψ are in $LTL_1^\downarrow(X, U)$, then so is $\varphi U \psi$.

Intuitively, the predicate $\uparrow$ is intended to mean that the current symbol is the same as the symbol in the register, while $\downarrow\varphi$ is intended to mean that the formula φ holds when the register contains the current symbol. This will be made precise in the definition of the semantics of $LTL_1^\downarrow(X, U)$ below.

An occurrence of $\uparrow$ within the scope of some freeze quantification $\downarrow$ is bounded by it; otherwise, it is free. A sentence is a formula with no free occurrence of $\uparrow$.

Next, we define the *freeze quantifier rank* of a sentence φ , denoted by $fqr(\varphi)$.

- $fqr(\text{True}) = fqr(\text{False}) = fqr(\uparrow) = 0$.
- $fqr(X\varphi) = fqr(\neg\varphi) = fqr(\varphi)$, for every φ in $LTL_1^\downarrow(X, U)$.
- $fqr(\varphi \vee \psi) = fqr(\varphi \wedge \psi) = fqr(\varphi U \psi) = \max(fqr(\varphi), fqr(\psi))$, for every φ and ψ in $LTL_1^\downarrow(X, U)$.
- $fqr(\downarrow\varphi) = fqr(\varphi) + 1$, for every φ in $LTL_1^\downarrow(X, U)$.

Finally, we define the semantics of $LTL_1^\downarrow(X, U)$. Let $w = a_1 \cdots a_n$ be a word. For a position $l = 1, \dots, n$, a symbol a and a formula φ in $LTL_1^\downarrow(X, U)$, $w, l \models_a \varphi$ means that φ is satisfied by w at position l when the content of the register is a . As usual, $w, l \not\models_a \varphi$ means the opposite. The satisfaction relation is defined inductively as follows.

- $w, l \models_a \text{True}$ and $w, l \not\models_a \text{False}$, for all $l = 1, 2, 3, \dots$ and $a \in \mathcal{D}$.
- $w, l \models_a \varphi \vee \psi$ if and only if $w, l \models_a \varphi$ or $w, l \models_a \psi$.
- $w, l \models_a \varphi \wedge \psi$ if and only if $w, l \models_a \varphi$ and $w, l \models_a \psi$.
- $w, l \models_a \neg\varphi$ if and only if $w, l \not\models_a \varphi$.
- $w, l \models_a X\varphi$ if and only if $1 \leq l < n$ and $w, l+1 \models_a \varphi$.
- $w, l \models_a \varphi U \psi$ if and only if there exists $l' \geq l$ such that $w, l' \models_a \psi$ and $w, l'' \models_a \varphi$, for all $l'' = l, \dots, l' - 1$.
- $w, l \models_a \downarrow\varphi$ if and only if $w, l \models_{a_l} \varphi$
- $w, l \models_a \uparrow$ if and only if $a = a_l$.

For a sentence φ in $LTL_1^\downarrow(X, U)$, we write $w, 1 \models \varphi$, if $w, 1 \models_a \varphi$ for some $a \in \mathcal{D}$. Note that since φ is a sentence, all occurrences of $\uparrow$ in φ are bounded. Thus, it makes no difference which data value a is used in the statement $w, 1 \models_a \varphi$ of the definition of $w, 1 \models \varphi$. We define the language $L(\varphi)$ by $L(\varphi) = \{w \mid w, 1 \models \varphi\}$.

THEOREM 5.1. *For every sentence $\psi \in LTL^\downarrow(X, U)$, there exists a weak k -PA $\mathcal{A}_\psi$, where $k = fqr(\psi) + 1$, such that $L(\mathcal{A}_\psi) = L(\psi)$.*

PROOF. Let ψ be an $LTL_1^\downarrow(X, U)$ sentence. We construct an alternating weak k -PA $\mathcal{A}_\psi$, where $k = fqr(\psi) + 1$ such that given a word w , the automaton $\mathcal{A}_\psi$ “checks” whether $w, 1 \models \psi$. $\mathcal{A}_\psi$ accepts w if it is so. Otherwise, it rejects.

Intuitively, the computation of $w, 1 \models \psi$ is done recursively as follows. The automaton $\mathcal{A}_\psi$ “consists of” the automata $\mathcal{A}_\varphi$ for all sub-formula φ of ψ .

- If $\psi = \varphi \vee \varphi'$, then $\mathcal{A}_\psi$ nondeterministically chooses one of $\mathcal{A}_\varphi$ or $\mathcal{A}_{\varphi'}$ and proceeds to run one of them.
- If $\psi = \varphi \wedge \varphi'$, then $\mathcal{A}_\psi$ splits its computation (by conjunctive branching) into two and proceeds to run both $\mathcal{A}_\varphi$ and $\mathcal{A}_{\varphi'}$.
- If $\psi = \mathbf{X}\varphi$, $\mathcal{A}_\psi$ moves to the right one step. If it reads the right-end marker $\triangleright$, then it rejects immediately. Otherwise, it proceeds to run $\mathcal{A}_\varphi$.
- If $\psi = \uparrow$, then $\mathcal{A}_\psi$ checks whether the symbol seen by its head pebble is the same as the one seen by the second last placed pebble. If it is not the same, then it rejects immediately.
- If $\psi = \downarrow \varphi$, then $\mathcal{A}_\psi$ places a new pebble and proceeds to run $\mathcal{A}_\varphi$.
- If $\psi = \varphi \mathbf{U} \varphi'$, then $\mathcal{A}_\psi$ repeatedly does the following.
 - (1) It splits its computation (by conjunctive branching) into two.
 - (2) In one branch it runs $\mathcal{A}_\varphi$.
 - (3) In the other it moves one step to the right and starts on Step 1 again.
 It repeatedly performs (1)–(3) until it nondeterministically decides to run $\mathcal{A}_{\varphi'}$.
- If $\psi = \neg \varphi$, then $\mathcal{A}_\psi$ runs the complement of $\mathcal{A}_\varphi$. The complement of $\mathcal{A}_\varphi$ can be constructed by switching the accepting states into non-accepting states and the non-accepting states into accepting states, as well as, switching the universal states into non-universal states and the non-universal states into universal states.

Note that since $\text{fqr}(\varphi) = k$, on each computation path the automaton $\mathcal{A}_\psi$ only needs to place the pebble k times, thus, $\mathcal{A}_\psi$ requires only $(k + 1)$ pebbles.

Now it is a straightforward induction on the length of φ to show that

$$w, l \models_a \varphi \text{ if and only if the configuration } [i, q, \theta] \text{ leads to acceptance,}$$

where

- $i = \text{fqr}(\varphi) + 1$;
- q is the initial state of $\mathcal{A}_\varphi$;
- θ is a pebble assignment where $\theta(i) = l$ and $\theta(j) \leq l$, for each $j \in \{i + 1, \dots, k + 1\}$;
- a is the symbol seen by pebble $(i + 1)$, if $i \neq k + 1$. (If $i = k + 1$, then a can be an arbitrary symbol.)

From here, it immediately follows that $L(\mathcal{A}_\psi) = L(\psi)$. $\square$

Our next results deal with the expressive power of $\text{LTL}_1^\downarrow(\mathbf{X}, \mathbf{U})$ based on the freeze quantifier rank. It is an analog of the classical hierarchy of first order logic based on the ordinary quantifier rank. We start by defining an $\text{LTL}_1^\downarrow(\mathbf{X}, \mathbf{U})$ sentence for the language $\mathcal{R}_m^+$ defined in Section 3.

LEMMA 5.2. *For each $k = 1, 2, 3, \dots$, there exists a sentence ψ_k in $\text{LTL}_1^\downarrow(\mathbf{X}, \mathbf{U})$ such that $L(\psi_k) = \mathcal{R}_k^+$ and $\text{fqr}(\psi_1) = 1$; and $\text{fqr}(\psi_k) = k - 1$, when $k \geq 2$.*

PROOF. First, we define a formula φ_k such that $\text{fqr}(\varphi_k) = k - 1$ and for every word $w = d_1 \dots d_n$, for every $i = 1, \dots, n$,

$$w, i \models_{d_i} \varphi_k \text{ if and only if } d_i \dots d_n \in \mathcal{R}_k^+. \quad (9)$$

We construct φ_k inductively as follows.

- $\varphi_1 = \mathbf{X}(\neg \uparrow) \wedge \neg(\mathbf{X}(\mathbf{X} \text{ True}))$.
- For each $k = 1, 2, 3, \dots$,

$$\varphi_{k+1} = \mathbf{X}(\neg \uparrow) \wedge \mathbf{X}\left(\downarrow \mathbf{X}\left((\neg \uparrow) \mathbf{U}(\uparrow \wedge \varphi_k)\right)\right)$$

Note that since $\text{fqr}(\varphi_1) = 0$, then for each $k = 1, 2, \dots$, $\text{fqr}(\varphi_k) = k - 1$.

It is straightforward to show that φ_k satisfies Equation (9). The desired sentence ψ_k is defined as follows.

- $\psi_1 = \downarrow (\mathbf{x}(\neg \uparrow) \wedge \neg (\mathbf{x}(\mathbf{x} \text{ True})))$.
- For each $k = 2, 3, \dots$,

$$\psi_k = \downarrow (\mathbf{x}(\neg \uparrow)) \wedge \mathbf{x} \left(\downarrow \mathbf{x} \left((\neg \uparrow) \mathbf{U} (\uparrow \wedge \varphi_{k-1}) \right) \right)$$

Obviously, $\text{fqr}(\psi_1) = 1$. For $k \geq 2$, $\text{fqr}(\varphi_{k-1}) = k - 2$, thus, $\text{fqr}(\psi_k) = k - 1$. $\square$

LEMMA 5.3. *For each $k = 1, 2, \dots$, the language $\mathcal{R}_{k+1}^+$ is not expressible by a sentence in $\text{LTL}_1^\downarrow(X, U)$ of freeze quantifier rank $(k - 1)$.*

PROOF. By Lemma 4.3, $\mathcal{R}_{k+1}^+$ is not accepted by weak k -PA. Then, by Theorem 5.1, $\mathcal{R}_{k+1}^+$ is not expressible by $\text{LTL}_1^\downarrow(X, U)$ sentence of freeze quantifier rank $(k - 1)$. $\square$

Combining both Lemmas 5.2 and 5.3, we obtain that for each $k = 1, 2, \dots$, the language $\mathcal{R}_{k+1}$ separates the class of $\text{LTL}_1^\downarrow(X, U)$ sentences of freeze quantifier rank k from the class of $\text{LTL}_1^\downarrow(X, U)$ sentences of freeze quantifier rank $(k - 1)$. Formally, we state it as follows.

THEOREM 5.4. *For each $k = 1, 2, \dots$, the class of sentences in $\text{LTL}_1^\downarrow(X, U)$ of freeze quantifier rank k is strictly more expressive than those of freeze quantifier rank $(k - 1)$.*

Acknowledgement. The author would like to thank the anonymous referees, both the conference and the journal versions, for their careful reading and comments which greatly improve the paper. The author also would like to thank Michael Kaminski for his support and guidance when this work was done.

REFERENCES

- AJTAI, M. AND FAGIN, R. 1990. Reachability is harder for directed than for undirected finite graphs. *Journal of Symbolic Logic* 55, 1, 113–150.
- BJÖRKLUND, H. AND SCHWENTICK, T. 2007. On notions of regularity for data languages. In *FCT*. 88–99.
- BOJANCZYK, M., DAVID, C., MUSCHOLL, A., SCHWENTICK, T., AND SEGOUFIN, L. 2011. Two-variable logic on data words. *ACM Transactions on Computational Logic* 12, 4, 27.
- BOJANCZYK, M., KLIN, B., AND LASOTA, S. 2011. Automata with group actions. In *LICS*. 355–364.
- BOUYER, P. 2002. A logical characterization of data languages. *Information Processing Letters* 84, 2, 75–85.
- DEMRI, S. AND LAZIĆ, R. 2009. LTL with the freeze quantifier and register automata. *ACM Transactions of Computational Logic* 10, 3.
- DEMRI, S., LAZIĆ, R., AND NOWAK, D. 2005. On the freeze quantifier in constraint LTL: Decidability and complexity. In *TIME*. 113–121.
- FAGIN, R., STOCKMEYER, L. J., AND VARDI, M. Y. 1995. On monadic NP vs. monadic co-NP. *Information and Computation* 120, 1, 78–92.
- GLOBERMAN, N. AND HAREL, D. 1996. Complexity results for multi-pebble automata and their logics. *Theoretical Computer Science* 169, 161–184.
- KAMINSKI, M. AND FRANCEZ, N. 1994. Finite-memory automata. *Theoretical Computer Science* 134, 2, 329–363.
- LADNER, R. E., LIPTON, R. J., AND STOCKMEYER, L. J. 1984. Alternating pushdown and stack automata. *SIAM Journal of Computing* 13, 1, 135–155.
- LAZIĆ, R. 2011. Safety alternating automata on data words. *ACM Transaction of Computational Logic* 12, 2, 10.

- NEVEN, F., SCHWENTICK, T., AND VIANU, V. 2004. Finite state machines for strings over infinite alphabets. *ACM Transactions on Computational Logic* 5, 3, 403–435.
- SAVITCH, W. J. 1970. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences* 4, 2, 177–192.
- SCHWENTICK, T. 1996. On winning Ehrenfeucht games and monadic NP. *Annals of Pure and Applied Logic* 79, 1, 61–92.
- SEGOUFIN, L. 2006. Automata and logics for words and trees over an infinite alphabet. In *CSL*. 41–57.
- TAN, T. 2009. Determinizing two-way alternating pebble automata over infinite alphabets. Technical report, Department of Computer Science, Technion – Israel Institute of Technology. Can be found in <http://www.cs.technion.ac.il/users/wwwb/cgi-bin/tr-list.cgi/2009/CS>.
- TAN, T. 2010. On pebble automata for data languages with decidable emptiness problem. *Journal of Computer and Systems Sciences* 76, 8, 778–791.
- TURÁN, G. 1984. On the definability of properties of finite graphs. *Discrete Mathematics* 49, 3, 291–302.