On Infinite Words Determined by Stack Automata

Tim Smith

Northeastern University
Boston, MA, USA
smithtim@ccs.neu.edu

Abstract

We characterize the infinite words determined by one-way stack automata. An infinite language L determines an infinite word α if every string in L is a prefix of α. If L is regular or context-free, it is known that α must be ultimately periodic. We extend this result to the class of languages recognized by one-way nondeterministic checking stack automata (1-NCSA). We then consider stronger classes of stack automata and show that they determine a class of infinite words which we call multilinear. We show that every multilinear word can be written in a form which is amenable to parsing. Finally, we consider the class of one-way multihead deterministic finite automata (1:multi-DFA). We show that every multilinear word can be determined by some 1:multi-DFA, but that there exist infinite words determined by 1:multi-DFA which are not multilinear.

1998 ACM Subject Classification F.1.1 Models of Computation, F.4.3 Formal Languages

Keywords and phrases stack automaton, infinite word, pumping lemma, prefix language, multi-head finite automaton

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2013.413

1 Introduction

In this paper we study the complexity of infinite words in terms of what automata can determine them, focusing on the infinite words determined by one-way stack automata. Stack automata are a generalization of pushdown automata whose stack head, in addition to pushing and popping when at the top of the stack, can move up and down the stack in read-only mode. Stack automata were first considered by Ginsburg, Greibach, and Harrison [8, 7]; see [12] and [17] for more references and results. These automata can be restricted and generalized in a number of ways, yielding various language classes.

To associate these and other automata with infinite words, we follow Book [3] in using the concept of prefix languages. A prefix language is a language L such that for all $x, y \in L$, x is a prefix of y or y is a prefix of x. Every infinite prefix language determines an infinite word. Where C is a class of languages, we denote by $\omega(C)$ the class of infinite words determined by the prefix languages in C. Then for any class of automata, we can investigate the infinite words determined by the languages recognized by those automata. We give several results aimed at building up a classification of infinite words with respect to which classes of languges and automata can determine them.

We begin with the ultimately periodic words, those of the form $xyyy\cdots$, where x and y are strings and y is not empty. As observed in [3], every infinite regular prefix language determines an ultimately periodic word. Since the converse is also true, an infinite word is in $\omega(\mathrm{REG})$ iff it is ultimately periodic. It is further known that $\omega(\mathrm{CFL})$, the class of infinite words determined by context-free languages, equals $\omega(\mathrm{REG})$. This follows from a result of Book [3], who used the pumping lemma for context-free languages to show that every context-free prefix language is regular. Book showed the same for one-way deterministic

33rd Int'l Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2013).
Editors: Anil Seth and Nisheeth K. Vishnoi; pp. 413–424

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

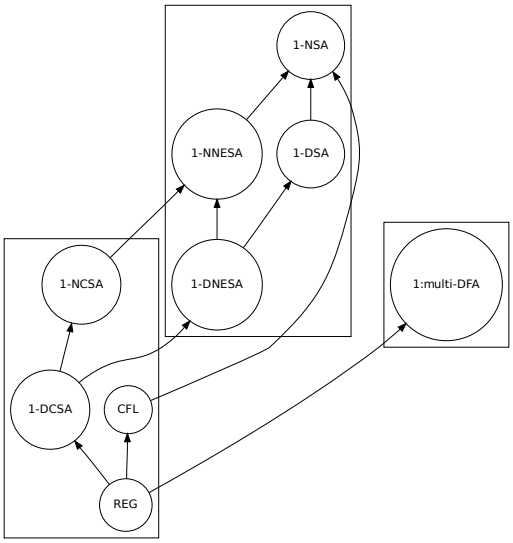

Figure 1 Relationships among the language classes considered in this paper. Arrows indicate inclusion of the lower class by the upper class. The three boxes correspond to the three types of infinite words determined by these languages.

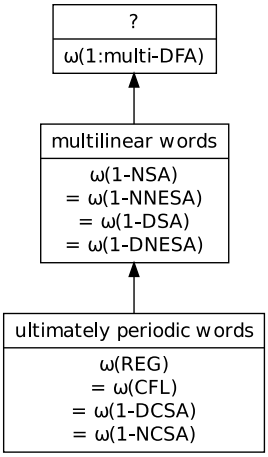

Figure 2 Three types of infinite words and the language classes which determine them. Arrows indicate proper inclusion.

checking stack automata (1-DCSA). We extend this result to the nondeterministic case (1-NCSA) using a weak pumping lemma for this class. That is, we show that every infinite word determined by a 1-NCSA is ultimately periodic.

Next, we consider a type of infinite word we call multilinear. A multilinear word consists of an initial segment q, followed by segments $r_1, \ldots, r_m$ which repeat in a way governed by linear polynomials. We show that these infinite words are determined by several classes of one-way stack automata. The most general of these is the class of one-way nondeterministic stack automata (1-NSA); various restrictions yield 1-DSA (deterministic stack automata), 1-NNESA (nondeterministic nonerasing stack automata), and 1-DNESA (deterministic nonerasing stack automata). We find with the help of a pumping lemma due to Ogden [15] that each of these classes determines exactly the multilinear infinite words. That is, $\omega(\text{1-NSA}) = \omega(\text{1-DSA}) = \omega(\text{1-NNESA}) = \omega(\text{1-DNESA})$.

Finally, we consider the class of one-way multihead deterministic finite automata (1:multi-DFA). We show that every multilinear word can be expressed in a form which is amenable to recognition by these automata. Then we show, using this form, that every such word can be determined by a 1:multi-DFA. We then give an example of an infinite word in $\omega(\text{1:multi-DFA})$ which is not multilinear. The problem of further characterizing the class of infinite words determined by 1:multi-DFA remains open.

1.1 Related work

The model used in this paper, in which infinite words are determined by languages of their prefixes, builds on Book's 1977 paper [3]. Book formulated the "prefix property" in order to allow languages to "approximate" infinite sequences, and showed that for certain classes of languages, if a language in the class has the prefix property, then it is regular. A follow-up

by Latteux [14] gives a necessary and sufficient condition for a prefix language to be regular. Languages whose complement is a prefix language, called "coprefix languages", have also been studied; see Berstel [2] for a survey of results on infinite words whose coprefix language is context-free. In Smith [16], prefix languages are used to categorize the infinite words determined by L systems, a type of parallel rewriting system.

Another approach is to consider sequence generators, devices which run indefinitely and output an infinite word piece by piece. This was the model of the seminal paper of Hartmanis and Stearns [10], as well as a 1970 follow-up by Fischer, Meyer, and Rosenberg [6], and several later papers beginning with Hromkovič, Karhumäki, and Lepistö [13], who investigated the computational complexity of infinite words generated by several kinds of iterated device. The question of what mechanisms and iterative devices suffice to generate particular infinite words has also been studied [5].

In another model, an automaton is associated with an infinite word α if when given a number n as input, it outputs the nth symbol of α. In 1972 Alan Cobham used this approach to associate finite automata with uniform tag sequences [4], leading to a literature on these "automatic sequences" [1].

1.2 Outline of paper

The paper is organized as follows. Section 2 gives preliminary definitions concerning prefix languages and automata. Section 3 gives results on ultimately periodic words and the languages and automata which determine them. Section 4 introduces multilinear infinite words and relates them to stack automata. Section 5 relates multilinear words to multihead finite automata. Section 6 gives our conclusions.

2 Preliminaries

An **alphabet** A is a finite set of symbols. A **word** is a concatenation of symbols from A. We denote the set of finite words by A^* and the set of infinite words by A^ω. A **string** x is an element of A^*. The length of x is denoted by $|x|$. We denote the empty string by λ. A **language** is a subset of A^*. A (symbolic) **sequence** S is an element of $A^* \cup A^\omega$. A **prefix** of S is a string x such that $S = xS'$ for some sequence S'. A **subword** (or factor) of S is a string x such that $S = wxS'$ for some string w and sequence S'. For $i \geq 1$, $S[i]$ denotes the ith symbol of S. For a string $x \neq \lambda$, x^ω denotes the infinite word $xxx\cdots$. Such a word is called **purely periodic**. An infinite word of the form xy^ω, where x and y are strings and $y \neq \lambda$, is called **ultimately periodic**.

2.1 Prefix languages

A **prefix language** is a language L such that for all $x, y \in L$, x is a prefix of y or y is a prefix of x. A language L **determines** an infinite word α iff L is infinite and every $x \in L$ is a prefix of α. For example, the infinite prefix language $\{\lambda, \texttt{ab}, \texttt{abab}, \texttt{ababab}, \ldots\}$ determines the infinite word $(\texttt{ab})^\omega$. The following propositions are basic consequences of the definitions.

▶ Remark. A language determines at most one infinite word.

▶ Remark. A language L determines an infinite word iff L is an infinite prefix language.

Notice that while a language determines at most one infinite word, an infinite word α may be determined by more than one language. Let $\text{Prefix}(\alpha) = \{x \mid x \text{ is a prefix of } \alpha\}$. We call $\text{Prefix}(\alpha)$ the **full** prefix language of α.

For a language class C, let $\omega(C) = \{\alpha \mid \alpha \text{ is an infinite word determined by some } L \in C\}$.

2.2 Automata

A **stack automaton** is a pushdown automaton with the extra ability to traverse its stack in read-only mode. In addition to moving its input head on the input tape, a stack automaton can move its stack head up and down to read symbols on the stack. Only when its stack head is at the top of the stack can it push or pop. A stack automaton is **nonerasing** if it never pops a symbol. A stack automaton is **checking** if it is nonerasing and if once it moves its stack head down from the top of the stack, it never again pushes a symbol. A stack automaton may be **deterministic** or **nondeterministic** and its input head may be **one-way** or **two-way**. See [12] and [17] for formal definitions and results.

Language Class	*Stack Automata*
1-NSA	one-way nondeterministic stack automata
1-DSA	one-way deterministic stack automata
1-NNESA	one-way nondeterministic nonerasing stack automata
1-DNESA	one-way deterministic nonerasing stack automata
1-NCSA	one-way nondeterministic checking stack automata
1-DCSA	one-way deterministic checking stack automata

From the definitions, we have

- 1-DCSA $\subseteq$ 1-NCSA, 1-DNESA $\subseteq$ 1-NNESA, 1-DSA $\subseteq$ 1-NSA,
- 1-DCSA $\subseteq$ 1-DNESA $\subseteq$ 1-DSA, and
- 1-NCSA $\subseteq$ 1-NNESA $\subseteq$ 1-NSA.

A **multihead finite automaton** is a finite automaton with one or more input heads. Here we are concerned only with 1:multi-DFA, the class of one-way multihead deterministic finite automata. This class is the union over all $i \geq 1$ of 1:i-DFA, the class of one-way i-head deterministic finite automata. Each such automaton begins with its input heads on the first symbol of the input. At each step, the automaton reads the input symbols under all of its heads and then changes state and moves any subset of its heads to the right. See [17] and [11] for formal definitions and results.

3 Ultimately periodic words

Recall that an infinite word is ultimately periodic if it has the form xy^ω, where x and y are strings and $y \neq \lambda$. Clearly every ultimately periodic word is determined by some regular language.

▶ **Theorem 1.** *Every ultimately periodic word α is in $\omega(REG)$.*

Proof. The infinite word α has the form xy^ω for some strings x and y where $y \neq \lambda$. Then the regular language xy^* determines α. So α is in ω(REG). ◀

As observed by Book [3], the converse, that every infinite regular prefix language determines an ultimately periodic word, is also true. In fact, as Book showed, the same holds for context-free languages.

▶ **Theorem 2** (Book). *Suppose α is in $\omega(CFL)$. Then α is ultimately periodic.*

Proof. Since α is in ω(CFL), some $L \in$ CFL determines α. Take any such L. Then L is infinite and every $s \in L$ is a prefix of α. By the pumping lemma for context-free languages, there is a string $uvxyz$ such that $|vy| \geq 1$ and for all $n \geq 0$, $uv^n xy^n z$ is in L. Suppose $|v| \geq 1$. Then because every string in L is a prefix of α, and every prefix of such a string is also a prefix of α, uv is a prefix of α, as are uvv, $uvvv$, and so on. Consequently $\alpha = uv^\omega$, so α is ultimately periodic. So say $|v| = 0$. Then $|y| \geq 1$ and $\alpha = uxy^\omega$, again making α ultimately periodic. ◀

Checking stack automata

The class of languages recognized by one-way checking stack automata is incomparable with the context-free languages. Nonetheless, this class too determines just the infinite words determined by regular languages. Book [3] proved the deterministic case (1-DCSA); our result holds in the nondeterministic case (1-NCSA) also. (Book showed that 1-NCSA contains non-regular prefix languages, but this does not imply that ω(1-NCSA) $\neq \omega$(REG).) We employ a weak pumping lemma for 1-NCSA which we obtain using results from Greibach [9].

For $k \geq 1$, a language L is **k-iterative** if there is an $n \geq 0$ such that for all $s \in L$ where $|s| \geq n$, there are strings x_1, y_1, x_2, y_2, …, x_k, y_k, x_{k+1} such that

- $s = x_1 y_1 x_2 y_2 \cdots x_k y_k x_{k+1}$,
- $|y_1 \cdots y_k| \geq 1$, and
- for all $i \geq 0$, $x_1 y_1^i x_2 y_2^i \cdots x_k y_k^i x_{k+1}$ is in L.

L is **weakly k-iterative** if it is either finite or contains an infinite k-iterative subset. Notice that every regular language is 1-iterative and every context-free language is 2-iterative, due to the pumping lemmas for these classes.

In proving the following lemma we use results from Greibach [9] formulated for a type of device called a one-way preset Turing machine. Greibach observes that a certain subclass of these devices, called nonwriting regular-based, can be regarded as checking stack automata. In particular, a one-way checking stack automaton can be simulated by a one-way nonwriting regular-based preset Turing machine, and vice versa, without changing the number of stack visits, crosses, or reversals by more than 1. Hence results for this subclass translate into facts about checking stack automata.

▶ **Lemma 3.** *Suppose L is in 1-NCSA. Then L is weakly k-iterative for some $k \geq 1$.*

Proof. A checking stack automaton M is **finite visit** if there is a $k \geq 1$ such that for every string s accepted by M, there is an accepting computation of M for s in which no stack position is visited more than k times. Suppose L is accepted by a finite visit 1-NCSA M. Then there is a $k \geq 1$ such that L is in the class k-VISIT(REGL) of [9]. Then by Lemma 4.22 of [9], L is weakly k-iterative. So say there is no such M. Then L is not in the class FINITEVISIT(REGL) of [9]. Then by Lemma 4.25 of [9], L is weakly 1-iterative. ◀

▶ **Theorem 4.** *Suppose α is in ω(1-NCSA). Then α is ultimately periodic.*

Proof. Since α is in ω(1-NCSA), some $L \in$ 1-NCSA determines α. Take any such L. Then L is infinite and every $s \in L$ is a prefix of α. By Lemma 3, L is weakly k-iterative for some $k \geq 1$. Then there is a string $x_1 y_1 x_2 y_2 \cdots x_k y_k x_{k+1}$ such that $|y_1 \cdots y_k| \geq 1$ and for all $i \geq 0$, $x_1 y_1^i x_2 y_2^i \cdots x_k y_k^i x_{k+1}$ is in L. Let j be the lowest number such that y_j is non-empty. Then $x_1 x_2 \cdots x_j y_j$ is a prefix of α, as are $x_1 x_2 \cdots x_j y_j y_j$, $x_1 x_2 \cdots x_j y_j y_j y_j$, and so on. Therefore $\alpha = x_1 x_2 \cdots x_j y_j^\omega$, so α is ultimately periodic. ◀

Summarizing, we have the following.

▶ **Theorem 5.** *$\omega(REG) = \omega(CFL) = \omega(1\text{-}DCSA) = \omega(1\text{-}NCSA)$, and α is in this class of infinite words iff α is ultimately periodic.*

Proof. Immediate from Theorems 1, 2, and 4 and the inclusions REG $\subseteq$ CFL and REG $\subseteq$ 1-DCSA $\subseteq$ 1-NCSA. ◀

4 Multilinear words

We now introduce the multilinear infinite words, a class which properly includes the ultimately periodic words. To our knowledge this type of infinite word has not previously been discussed. An infinite word is **multilinear** if it has the form

$$q \prod_{n\geq 0} r_1^{a_1 n+b_1} r_2^{a_2 n+b_2} \cdots r_m^{a_m n+b_m},$$

where $\prod$ denotes concatenation, q is a string, each r_i is a non-empty string, and m and each a_i and b_i are nonnegative integers such that $a_i + b_i > 0$. Examples:

- $\mathtt{ab} \prod_{n\geq 0} \mathtt{cd} = \mathtt{abcdcdcd}\cdots$
- $\prod_{n\geq 0} \mathtt{a}^{n+1}\mathtt{b} = \mathtt{abaabaaab}\cdots$
- $\prod_{n\geq 0} \mathtt{10}^{2n} = \mathtt{11001000010000001}\cdots$ (characteristic sequence of the perfect squares)

With the next few theorems we relate multilinear words to one-way stack automata.

▶ **Theorem 6.** *Suppose α is in $\omega(1\text{-}NSA)$. Then α is multilinear.*

Proof. Since α is in ω(1-NSA), some $L \in$ 1-NSA determines α. Take any such L. Then L is infinite and every $s \in L$ is a prefix of α. By Ogden's pumping lemma for one-way stack automata [15], there are

- strings μ and ν,
- strings ρ_i, σ_i, and τ_i for each $i \geq 0$,
- strings α_j, β_j, ϕ_j, χ_j and ψ_j for bounds on j implicit below, and
- positive integers m and p

such that, among other conditions,

(i) for each $j \geq 0$, $\mu\rho_0\rho_1\cdots\rho_j\sigma_j\tau_j\tau_{j-1}\cdots\tau_0\nu$ is in L,
(ii) for each $i \geq 1$, $\rho_i = \alpha_0\beta_1^{i-1}\phi_1\beta_2^{i-1}\alpha_1\beta_3^{i-1}\phi_2\beta_4^{i-1}\alpha_2\cdots\phi_{m-1}\beta_{2m-2}^{i-1}\alpha_{m-1}$,
(iii) $|\rho_0| = 0$ iff for all $i > 0$, $|\rho_i| = 0$,
(iv) for each $i \geq 0$, $\sigma_i = \chi_0\psi_1^i\chi_1\psi_2^i\chi_2\cdots\psi_{p-1}^i\chi_{p-1}$, and
(v) there is a j such that $|\psi_j| > 0$.

Suppose $|\rho_0| = 0$. Then by (iii), every ρ_i is empty, so by (i), we have that for each $j \geq 0$, $\mu\sigma_j\tau_j\tau_{j-1}\cdots\tau_0\nu$ is in L. Then for each $j \geq 0$, $\mu\sigma_j$ is a prefix of α. Then by (iv), for each $i \geq 0$, $\mu\chi_0\psi_1^i\chi_1\psi_2^i\chi_2\cdots\psi_{p-1}^i\chi_{p-1}$ is a prefix of α. Let j be the lowest number such that $|\psi_j| > 0$; by (v), there is such a j. Then for each $i \geq 0$, $\mu\chi_0\chi_1\cdots\chi_{j-1}\psi_j^i\chi_j\cdots\psi_{p-1}^i\chi_{p-1}$ is a prefix of α. Then $\mu\chi_0\chi_1\cdots\chi_{j-1}\psi_j$ is a prefix of α, as are $\mu\chi_0\chi_1\cdots\chi_{j-1}\psi_j\psi_j$, $\mu\chi_0\chi_1\cdots\chi_{j-1}\psi_j\psi_j\psi_j$, and so on. Therefore $\alpha = \mu\chi_0\chi_1\cdots\chi_{j-1}\psi_j^\omega$, which is ultimately periodic and hence multilinear.

So say $|\rho_0| > 0$. By (iii), some ρ_i other than ρ_0 is non-empty. By (ii), if ρ_1 is non-empty, then so are all subsequent ρ_is, and if ρ_1 is empty, then some β_j must be non-empty, and all subsequent ρ_is are again non-empty. So for all $i \geq 2$, $|\rho_i| > 0$. By (i), for each $j \geq 0$, $\mu\rho_0\rho_1 \cdots \rho_j$ is a prefix of α. Therefore $\alpha = \mu \prod_{n\geq 0} \rho_n = \mu\rho_0 \prod_{n\geq 0} \rho_{n+1} = \mu\rho_0 \prod_{n\geq 0} \alpha_0\beta_1^n\phi_1\beta_2^n\alpha_1\beta_3^n\phi_2\beta_4^n\alpha_2 \cdots \phi_{m-1}\beta_{2m-2}^n\alpha_{m-1}$, which is multilinear. ◀

▶ **Theorem 7.** *Suppose α is multilinear. Then α is in ω(1-DNESA).*

Proof. The infinite word α has the form $q \prod_{n\geq 0} r_1^{a_1 n+b_1} r_2^{a_2 n+b_2} \cdots r_m^{a_m n+b_m}$. Let A be a one-way deterministic nonerasing stack automaton, operating as follows. First, A checks that the input begins with q. In what follows, A will push counter symbols onto its stack; so far, the stack is empty. Next, for each i between 1 and m, A first checks the input for b_i occurrences of r_i. It then reads its stack, for each counter symbol checking the input for a_i occurrences of r_i. After checking r_m, A pushes a counter symbol onto its stack and proceeds as before. If any input symbol causes a check to fail, A rejects; otherwise, when A reaches end of input, it accepts. Now A recognizes Prefix(α), the full prefix language of α. Since Prefix(α) determines α, α is in ω(1-DNESA). ◀

▶ **Theorem 8.** *ω(1-NSA) = ω(1-DSA) = ω(1-NNESA) = ω(1-DNESA), and α is in this class of infinite words iff α is multilinear.*

Proof. Immediate from Theorems 6 and 7 and the inclusions 1-NSA $\supseteq$ 1-DSA $\supseteq$ 1-DNESA and 1-NSA $\supseteq$ 1-NNESA $\supseteq$ 1-DNESA. ◀

5 Multihead finite automata

In this section we relate multilinear infinite words to multihead finite automata. First, we show that every multilinear infinite word can be expressed in a certain form which is amenable to recognition by these automata. Then we show, using this form, that every multilinear infinite word can be determined by a one-way multihead deterministic finite automaton (1:multi-DFA).

Following the definition in the previous section, a multilinear infinite word can be viewed as a pair $[q, t]$, where t is a **term list** of m triples $[r_i, a_i, b_i]$. We say that two such pairs (or two term lists) are **equivalent** if they express the same multilinear word. Notice that if two term lists t_1, t_2 are equivalent, then the first term of t_1 begins with the same symbol as the first term of t_2. We say that t_1 and t_2 are **strongly equivalent** if they are equivalent and if the last term of t_1 begins with the same symbol as the last term of t_2. Any term i with $a_i > 0$ we call a **growth term**. Any pair $[q, t]$ can be **rotated**, yielding the equivalent pair $[q\ r_1^{b_1}, [t[2], \ldots, t[m], [r_1, a_1, b_1 + a_1]]]$. In the proofs below, we allow a growth term to temporarily have a negative b_i if it can later be "rotated away" (made nonnegative by repeated rotations). To this end, when $b_1 < 0$, we define rotation of the pair $[q\ r_1^{-b_1}, t]$ to yield the equivalent pair $[q, [t[2], \ldots, t[m], [r_1, a_1, b_1 + a_1]]]$. For use below, we give several conditions which a pair or term list may or may not satisfy.

- Condition 1. For every i from 1 to m, $b_i \geq 1$.
- Condition 2a. For every i from 1 to $m-1$, $r_i[1] \neq r_{i+1}[1]$.
- Condition 2b. If $m \geq 2$, $r_1[1] \neq r_m[1]$.

For example, take the multilinear infinite word

$$\alpha = \prod_{n\geq 0} (\mathtt{abc})^n \mathtt{aba}^n = \mathtt{ababcabaabcabcabaa}\cdots$$

The multilinear pair [λ, [[abc, 1, 0], [ab, 0, 1], [a, 1, 0]]] expresses α but does not meet any of the three conditions. However, the equivalent pair [ab, [[a, 1, 1], [b, 0, 1], [cab, 1, 1]]] meets all three conditions, giving

$$\alpha = \mathtt{ab} \prod_{n\geq 0} \mathtt{a}^{n+1}\mathtt{b}(\mathtt{cab})^{n+1}$$

We will show that every multilinear infinite word can be expressed as a pair satisfying conditions 1, 2a, and 2b. The proof outline is first to show that every multilinear term list has an equivalent term list satisfying condition 2a (Lemma 12), and next to show that every multilinear pair satisfying condition 2a is equivalent to a pair satisfying conditions 1, 2a, and 2b (Theorem 15). The notion of strong equivalence is used in the proof of Theorem 15, where we take a term list satisfying condition 2a, rotate it so that it satisfies 2b but now has a portion which does not satisfy 2a, and then replace that portion with a strongly equivalent one satisfying 2a, so that the whole then satisfies 2a and 2b.

▶ **Lemma 9.** *Given a multilinear term list* $[[r_1, a_1, b_1], [r_2, a_2, b_2]]$ *such that* $a_1 = 0$ *and* $a_2 > 0$*, there is a strongly equivalent term list meeting condition 2a.*

Proof. Let $s = r_1^{b_1}$. Suppose there is an i such that $s[i] \neq r_2[1]$. Take the first such i. If $i = 1$, we can just return $[[s, 0, 1], [r_2, a_2, b_2]]$. Otherwise, we split s at i and return $[[s[1]\cdots s[i-1]], 0, 1], [s[i]\cdots s[|s|]], 0, 1], [r_2, a_2, b_2]]$. So say there is no such i. Suppose there is an i such that $r_2[i] \neq r_2[1]$. Take the first such i. Since $[[s\ r_2, 0, 1], [r_2, a_2, b_2 - 1]]$ is equivalent to the original term list, we can return $[[s\ r_2[1]\cdots r_2[i-1], 0, 1], [r_2[i]\cdots r_2[|r_2|], 0, 1], [r_2, a_2, b_2 - 1]]$. So say there is no such i. Then every symbol in s and r_2 equals $r_2[1]$. So return $[[r_2[1], |r_2| \cdot a_2, |s| + |r_2| \cdot b_2]]$. ◀

▶ **Lemma 10.** *Given a multilinear term list* $[[r_1, a_1, b_1], [r_2, a_2, b_2]]$ *such that* $a_1 > 0$ *and* $a_2 > 0$*, there is a strongly equivalent term list meeting condition 2a.*

Proof. Suppose $r_1^\omega = r_2^\omega$. Then $r_1^{|r_2|} = r_2^{|r_1|}$, so by Theorem 1.5.3 of [1], there are $k, l > 0$ such that $r_1 = z^k$ and $r_2 = z^l$ for some string z. Then $[[z, \frac{|r_1|}{|z|} \cdot a_1 + \frac{|r_2|}{|z|} \cdot a_2, \frac{|r_1|}{|z|} \cdot b_1 + \frac{|r_2|}{|z|} \cdot b_2]]$ meets the condition. So say $r_1^\omega \neq r_2^\omega$. Let p be the longest common prefix of r_1^ω and r_2^ω. If $p = \lambda$, then $[[r_1, a_1, b_1], [r_2, a_2, b_2]]$ already meets the condition. Otherwise, let $c = (|p| \bmod |r_1|) + 1$, let $d = \frac{|p|}{|r_1|}$ (rounded down), let e $= (|p| \bmod |r_2|) + 1$, and let $f = \frac{|p|}{|r_2|}$ (rounded down). Then $r_1[c] \neq r_2[e]$. Suppose $c = 1$. Let u be the term list $[[r1, a1, b1 + d], [r_2[e]\cdots r_2[|r_2|], 0, 1], [r_2, a_2, b_2 - f - 1]]$. Then u is equivalent to the original term list. Its first term already starts with a different symbol than its second term, while by Lemma 9, its last two terms can be replaced with an equivalent term list meeting condition 2a. So say $c \neq 0$. Let u be the term list $[[r_1[1]\cdots r_1[c-1]], 0, 1], [r_1[c]\cdots r_1[|r_1|]r_1[1]\cdots r_1[c-1]], a1, b1 + d], [r_2[e]\cdots r_2[|r_2|], 0, 1], [r_2, a_2, b_2 - f - 1]]$. Then u is equivalent to the original term list. Its second term already starts with a different symbol than its third term, while by Lemma 9, its first two terms and last two terms can be replaced with equivalent term lists meeting condition 2a. ◀

▶ **Lemma 11.** *Given a multilinear term list* $[[r_1, a_1, b_1], [r_2, a_2, b_2]]$ *such that* $a_2 = 0$*, there is an equivalent term list meeting condition 2a.*

Proof. If $a_1 = 0$, then we can just join the terms, returning $[[r_1^{b_1} r_2^{b_2}, 0, 1]]$. So say $a_1 > 0$. Let $s = r_2^{b_2}$. Then $s = r_1^i\ x$ for some $i \geq 0$ and string x such that r_1 is not a prefix of x. If $x = \lambda$, return $[[r_1, a_1, b_1 + i]]$. Let p be the longest common prefix of r_1 and x. If $p = \lambda$, return $[[r_1, a_1, b_1 + i], [x, 0, 1]]$. Otherwise, $r_1 = py$ and $x = pz$ for some strings y, z. Let u be the term list $[[p, 0, 1], [yp, a_1, b_1]]$. By Lemma 9, there is an equivalent term list u' meeting condition 2a whose last term begins with the same symbol as y. If $z = \lambda$, then u' is equivalent to the original term list, and we are finished. Otherwise, append to u' the term $[z, 0, 1]$. Now u' is equivalent to the original term list. Further, since z begins with a different symbol than does y, u' meets condition 2a, and we are finished. ◀

▶ **Lemma 12.** *Given a multilinear term list t, there is an equivalent term list meeting condition 2a.*

Proof. We proceed by induction on $|t|$. If $|t| \leq 1$, then condition 2a is already met, so just return t. If $|t| = 2$, then by Lemmas 9, 10, and 11, the result holds. So say $|t| > 2$. Suppose for induction that the result holds for any term list of size less than $|t|$. Then by the induction hypothesis, there is a term list u equivalent to $[t[1], \ldots, t[m-1]]$ and meeting condition 2a. Let $x = [u[1], \ldots, u[|u|-1]]$ and let $y = u[|u|]$. Then again by the induction hypothesis, there is a term list v equivalent to $[y, t[m]]$ and meeting condition 2a. So $x + v$ is equivalent to t. Now, x and v each meet condition 2a. Further, since v is equivalent to $[y, t[m]]$, the first term of v begins with the same symbol as y. Since u met condition 2a, the last term of x begins with a different symbol than does y. Hence $x + v$ meets condition 2a. ◀

▶ **Lemma 13.** *Given a multilinear term list t whose last term is a growth term, there is a strongly equivalent term list meeting condition 2a.*

Proof. If $|t| = 1$, then the condition is already met, so just return t. Otherwise, by Lemma 12, there is a term list u equivalent to $[t[1], \ldots, t[m-1]]$ and meeting condition 2a. Let $x = [u[1], \ldots, u[|u|-1]]$ and let $y = u[|u|]$. By Lemmas 9 and 10, there is a term list v equivalent to $[y, t[m]]$, meeting condition 2a, and whose last term starts with the same symbol as does $t[m]$. So $x + v$ is strongly equivalent to t. Now, x and v each meet condition 2a. Further, since v is equivalent to $[y, t[m]]$, the first term of v begins with the same symbol as y. Since u met condition 2a, the last term of x begins with a different symbol than does y. Hence $x + v$ meets condition 2a. ◀

▶ **Lemma 14.** *Given a multilinear pair $[q, t]$ such that t contains exactly one growth term, there is an equivalent pair meeting conditions 2a and 2b.*

Proof. If $|t| = 1$, then $[q, t]$ already meets the conditions. So say $|t| > 1$. Rotate $[q, t]$ until $t[1]$ is the growth term. Let $s = r_2^{b_2} \cdots r_m^{b_m}$. Suppose $r_1^\omega = s\ r_1^\omega$. Then $r_1^\omega = s^\omega$, so by Theorem 1.5.3 of [1], there are $k, l > 0$ such that $r_1 = z^k$ and $s = z^l$ for some string z. Then $[q, t]$ is ultimately periodic, so $[q, [z, 0, 1]]$ is an equivalent pair which meets the conditions. So say $r_1^\omega \neq s\ r_1^\omega$. Let p be the longest common prefix of r_1^ω and $s\ r_1^\omega$. Then p is a prefix of $s\ r_1$. If $p = \lambda$, then $[q, t]$ already meets the conditions. Otherwise, let u be the term list $[[r_1, 0, 1], [r_1, a_1, b_1 - 1], [s, 0, 1]]$. The pair $[q, u]$ is equivalent to $[q, t]$. Now rotate $[q, u]$ and combine terms to give $[[r_1, a_1, b_1 - 1], [s\ r_1, 0, 1]]$. The string $s\ r_1$ has the form px for some $x \neq \lambda$ and p has the form $r_1^i\ y$ for some $i \geq 0$ and string y such that $r_1 = yz$ for some $z \neq \lambda$. Notice that $x[1] \neq z[1]$. If $y = \lambda$ then we have $[[r_1, a_1, b_1 - 1 + i], [x, 0, 1]]$ and we are finished. Otherwise, we have $[[r_1, a_1, b_1 - 1 + i], [y, 0, 1], [x, 0, 1]]$, which is equivalent to $[[y, 0, 1], [zy, a_1, b_1 - 1 + i], [x, 0, 1]]$. Rotating twice, we get $[[x, 0, 1], [y, 0, 1], [zy, a_1, b_1 - 1 + i + a_1]]$. By Lemma 13, there is an equivalent term list whose last term starts with the same symbol as does z. Then q paired with this term list meets the conditions. ◀

▶ **Theorem 15.** *Let α be a multilinear infinite word. Then α has the form*

$$q \prod_{n \geq 0} r_1^{a_1 n + b_1} r_2^{a_2 n + b_2} \cdots r_m^{a_m n + b_m}$$

for some $m \geq 0$, string q, non-empty strings r_i, and nonnegative integers a_i, b_i where $a_i + b_i > 0$, such that

- *for every i from 1 to m, $b_i \geq 1$,*
- *for every i from 1 to $m-1$, $r_i[1] \neq r_{i+1}[1]$, and*
- *if $m \geq 2$, $r_1[1] \neq r_m[1]$.*

Proof. α can be viewed as a pair $[q, t]$, where t is a term list of m triples $[r_i, a_i, b_i]$. We will give an equivalent pair meeting conditions 1, 2a, and 2b. First, by Lemma 12, there is a term list u which is equivalent to t and which meets condition 2a. We will give a pair $[q', u']$ which is equivalent to $[q, u]$ and meets conditions 2a and 2b. If u is empty, just set $q' = q$ and $u' = u$. Otherwise, suppose u contains no growth terms. Then u can be contracted into a single term, so set $q' = q$ and $u' = [r_1^{b_1} \cdots r_m^{b_m}, 0, 1]$. So suppose u contains exactly one growth term. Then by Lemma 14, there is a multilinear pair $[q', u']$ which is equivalent to $[q, u]$ and meets conditions 2a and 2b. Finally, suppose u contains more than one growth term. Let $u[i]$ be the first growth term in u. Let $[q', u']$ be the result of rotating $[q, u]$ by i terms. Now u' ends with the growth term $u[i]$. The first symbol of $u'[1]$ ($u[i+1]$) differs from the first symbol of $u'[m]$ ($u[i]$), so u' meets condition 2b. Further, u' meets condition 2a, except that the first symbol of $u'[m-i]$ may equal the first symbol of $u'[m-i+1]$. Using Lemma 13, replace the terms from $m-i$ to m with a strongly equivalent term list meeting condition 2a. Now u' meets conditions 2a and 2b. Finally, now that we have an equivalent pair which meets conditions 2a and 2b, we can rotate it until condition 1 is met. For each term i, if $a_i = 0$, then already $b_i > 0$, whereas if $a_i > 0$, then each rotation increases b_i to $b_i + a_i$. So repeatedly rotating $[q', u']$ will eventually cause it to meet condition 1. ◀

▶ **Theorem 16.** *Every multilinear infinite word is in ω(1:multi-DFA).*

Proof. Take any multilinear infinite word α. By Theorem 15, α can be expressed in a form $q \prod_{n \geq 0} r_1^{a_1 n + b_1} r_2^{a_2 n + b_2} \cdots r_m^{a_m n + b_m}$ meeting the conditions of that theorem. If $m = 1$, then α is ultimately periodic, so α is in $\omega(\text{REG}) \subseteq \omega$(1:multi-DFA). So say $m \geq 2$. Let A be a one-way 2-head deterministic finite automaton, operating as follows. First, A checks that the input begins with q, moving both heads to the right after each symbol. Next, A keeps one head stationary while using the other head to check the $n = 0$ subword, verifying that each r_i occurs b_i times. Now one head is at the beginning of the $n = 0$ subword and the other head is at the beginning of the $n = 1$ subword. For each $j \geq 1$, A checks the $n = j$ subword as follows. For each i from 1 to m, A first uses its right head to checks for a_i occurrences of r_i, keeping its left head stationary. Then A moves both heads to the right, checking for occurrences of r_i under the left head until no more are found, and rejecting if the symbol under the right head ever differs from the symbol under the left head. After $a_i(j-1) + b_i$ occurrences of r_i, the left head will encounter the first symbol of r_{i+1} (or r_1 if $i = m$). Since, by the conditions of Theorem 15, this symbol is different from the first symbol of r_i, A is now at the start of term $i+1$, and can proceed to check this term, and so on until it has checked all m terms, at which point it moves on to subword $n = j+1$. If any input symbol causes a check to fail, A rejects; otherwise, when A reaches end of input, it accepts. Now A recognizes Prefix(α), the full prefix language of α. Since Prefix(α) determines α, α is in ω(1:multi-DFA). ◀

Finally, we give a simple example of an infinite word in ω(1:multi-DFA) which is not multilinear.

▶ **Theorem 17.** *Not every infinite word in ω(1:multi-DFA) is multilinear.*

Proof. Let α be the infinite word $\prod_{n\geq 0} \mathtt{a}^{2^n}\mathtt{b} = \mathtt{a}^1\mathtt{b}\mathtt{a}^2\mathtt{b}\mathtt{a}^4\mathtt{b}\mathtt{a}^8\mathtt{b}\cdots$. Clearly α is not multilinear. Let A be a one-way 2-head deterministic finite automaton, operating as follows. Initially, A keeps one head stationary while using the other head to check that the input begins with `ab`. Subsequently, A moves both heads to the right. For each `a` under the left head, A checks for two occurrences of `a` under the right head, while for each `b` under the left head, A checks for one occurrence of `b` under the right head. If any input symbol causes a check to fail, A rejects; otherwise, when A reaches end of input, it accepts. Now A recognizes Prefix(α), the full prefix language of α. Since Prefix(α) determines α, α is in ω(1:multi-DFA). ◀

6 Conclusion

In this paper we have given several results aimed at building up a classification of infinite words with respect to which classes of automata can determine them. To associate automata with infinite words, we used the concept of prefix languages. This concept can be applied not just to automata, but to arbitrary language classes, offering many opportunities for further research. For a given language class, we can ask what class of infinite words it determines. From the other direction, for a given infinite word, we can ask in what language classes it can be determined. It is hoped that work in this area will help to establish a theory of the complexity of infinite words as determined by their prefix languages. One specific task would be to further characterize the infinite words determined by one-way multihead deterministic finite automata (1:multi-DFA), beyond the result established in this paper, that the multilinear infinite words are properly contained by this class.

Acknowledgements. I want to thank my advisor, Rajmohan Rajaraman, for supporting this work, encouraging me, and offering many helpful comments and suggestions.

References

1 Jean-Paul Allouche and Jeffrey Shallit. *Automatic Sequences: Theory, Applications, Generalizations*. Cambridge University Press, New York, NY, USA, 2003.

2 J. Berstel. Properties of infinite words : Recent results. In B. Monien and R. Cori, editors, *STACS 89*, volume 349 of *Lecture Notes in Computer Science*, pages 36–46. Springer Berlin Heidelberg, 1989.

3 Ronald V. Book. On languages with a certain prefix property. *Mathematical Systems Theory*, 10:229–237, 1977.

4 Alan Cobham. Uniform tag sequences. *Theory of Computing Systems*, 6:164–192, 1972. 10.1007/BF01706087.

5 Karel Culik and Juhani Karhumäki. Iterative devices generating infinite words. *Int. J. Found. Comput. Sci.*, 5(1):69–97, 1994.

6 Patrick C. Fischer, Albert R. Meyer, and Arnold L. Rosenberg. Time-restricted sequence generation. *J. Comput. Syst. Sci.*, 4:50–73, February 1970.

7 Seymour Ginsburg, Sheila A. Greibach, and Michael A. Harrison. One-way stack automata. *J. ACM*, 14(2):389–418, April 1967.

8 Seymour Ginsburg, Sheila A. Greibach, and Michael A. Harrison. Stack automata and compiling. *J. ACM*, 14(1):172–201, January 1967.

9 S.A. Greibach. One way finite visit automata. *Theoretical Computer Science*, 6(2):175 – 221, 1978.

10 J. Hartmanis and R. E. Stearns. On the computational complexity of algorithms. *Transactions of The American Mathematical Society*, 117:285–306, 1965.

11 Markus Holzer, Martin Kutrib, and Andreas Malcher. Complexity of multi-head finite automata: Origins and directions. *Theor. Comput. Sci.*, 412(1-2):83–96, January 2011.

12 J.E. Hopcroft and J.D. Ullman. *Introduction to automata theory, languages, and computation*. Addison-Wesley series in computer science. Addison-Wesley, 1979.

13 Juraj Hromkovič, Juhani Karhumäki, and Arto Lepistö. Comparing descriptional and computational complexity of infinite words. In *Proceedings of the Colloquium in Honor of Arto Salomaa on Results and Trends in Theoretical Computer Science*, pages 169–182, London, UK, 1994. Springer-Verlag.

14 Michel Latteux. Une note sur la propriété de prefixe. *Mathematical Systems Theory*, 11:235–238, 1978.

15 William F. Ogden. Intercalation theorems for stack languages. In *Proceedings of the first annual ACM symposium on Theory of computing*, STOC '69, pages 31–42, New York, NY, USA, 1969. ACM.

16 Tim Smith. On infinite words determined by L systems. In Juhani Karhumäki, Arto Lepistö, and Luca Zamboni, editors, *Combinatorics on Words*, volume 8079 of *Lecture Notes in Computer Science*, pages 238–249. Springer Berlin Heidelberg, 2013.

17 K. Wagner and G. Wechsung. *Computational Complexity*. Mathematics and its Applications. Springer, 1986.